

---

# A LEARNING SYSTEM FOR MOTION PLANNING OF FREE-FLOAT DUAL-ARM SPACE MANIPULATOR TOWARDS NON-COOPERATIVE OBJECT

---

**Shengjie Wang**

Department of Automation  
Tsinghua University

Beijing

wangsj19@mails.tsinghua.edu.cn

**Yuxue Cao**

Beijing Institute of Control Engineering  
Beijing

sg18801138270@163.com

**Xiang Zheng**

Department of Computer Science  
City University of Hong Kong  
Hong Kong

xzheng235-c@my.cityu.edu.hk

**Tao Zhang\***

Department of Automation  
Tsinghua University  
Beijing

taozhang@tsinghua.edu.cn

\*

## ABSTRACT

Recent years have seen the emergence of non-cooperative objects in space, like failed satellites and space junk. These objects are usually operated or collected by free-float dual-arm space manipulators. Thanks to eliminating the difficulties of modeling and manual parameter-tuning, reinforcement learning (RL) methods have shown a more promising sign in the trajectory planning of space manipulators. Although previous studies demonstrate their effectiveness, they cannot be applied in tracking dynamic targets with unknown rotation (non-cooperative objects). In this paper, we proposed a learning system for motion planning of free-float dual-arm space manipulator (FFDASM) towards non-cooperative objects. Specifically, our method consists of two modules. Module I realizes the multi-target trajectory planning for two end-effectors within a large target space. Next, Module II takes as input the point clouds of the non-cooperative object to estimate the motional property, and then can predict the position of target points on a non-cooperative object. We leveraged the combination of Module I and Module II to track target points on a spinning object with unknown regularity successfully. Furthermore, the experiments also demonstrate the scalability and generalization of our learning system.

**Keywords** Space robotics · Motion planning · Reinforcement learning

## 1 Introduction

With the continuous increase of non-cooperative space objects like faulty satellites and space junk, space manipulators become indispensable in many space maintenance tasks. Free-float dual-arm space manipulator (FFDASM) is a common configuration for space manipulators, of which the free-float advantage helps to save fuel consumed in adjusting the pose and the dual-arm structure enlarges the workspace efficiently [1]. However, the characteristic of FFDASM has some negative impacts on model-based algorithms in trajectory planning tasks. Due to the coupling effects between the base and two manipulators, there are many time-varying and dynamic parameters in the Jacobian matrix, which are hard to be identified [2, 3]. Furthermore, trajectory planning aiming for a single target should take the joint trajectories

---

\*Corresponding author: Tao Zhang(taozhang@tsinghua.edu.cn)

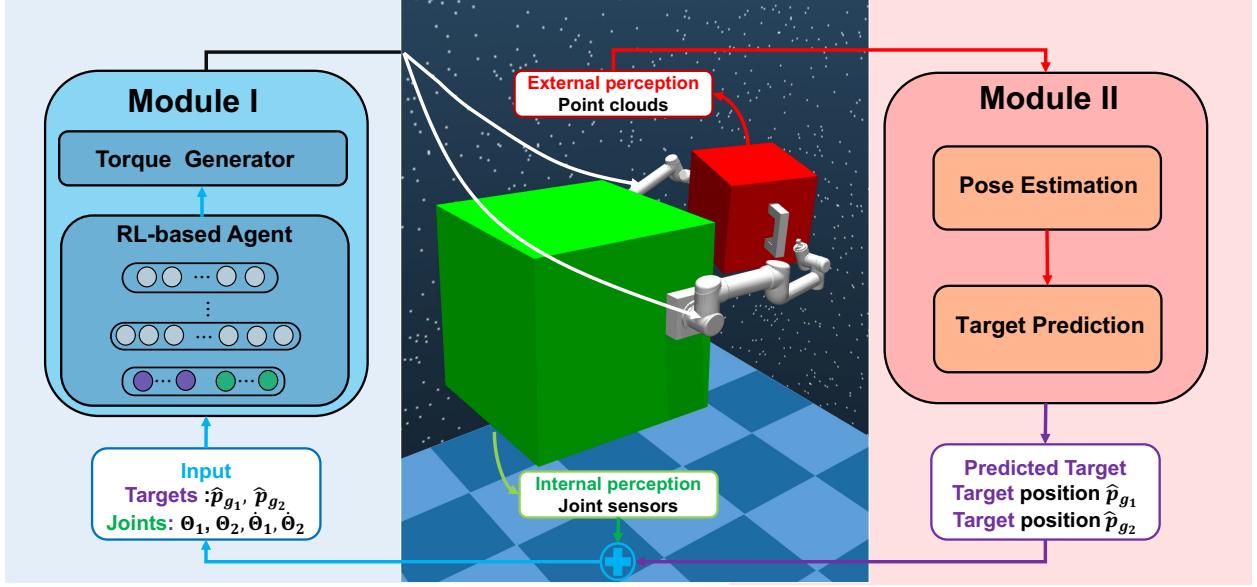


Figure 1: Framework of our two-module learning system for motion planning of free-float dual-arm space manipulator. Module I estimates the pose of spinning object, and predict the target points. Module II receives the prediction, and then allows the space manipulator to track the targets by a RL-based agent.

into account, introducing more biases in controllers [4, 5]. What's more, it will take a long time to manually adjust parameters for experts in order to achieve the full potential of these model-based controllers [6, 7].

To eliminate the mentioned problems above, model-free methods attracts more attentions from researchers. Model-free methods doesn't make use of approximate kinematic and dynamic models of space manipulators. In the meantime, the controller is built by a polynomial function, bessel curve function, or neural network with learnable parameters [8]. Intuitively, these functions only take some part of feedback from sensors as input, without concern of modeling. To reach the target, we can view the trajectory planning as an optimization problem naturally. Then, the optimization method updates the parameters of functions at each iteration. After the convergence, the optimal controller is obtained to perform the task. To recap, the essential components are function of controller and optimization algorithm. Trajectory planning for fixed start and end is able to be solved by the combination of polynomial controllers and some non-linear optimization methods, such as PSO and GA methods [8–10]. In comparison, reinforcement learning methods lead to a dramatic improvement in robustness and generalization, especially for the field of robotics [11–16]. Interestingly, concurrent studies have also revealed the promising results on the intersection of reinforcement learning and space robots.

However, to realize a real-world planning controller based on reinforcement learning, we need to overcome a number of major challenges. The first problem is that most previous studies pay more attention to the single fixed target during the training, which means it is unable to handle the sim-to-real gap, e.g., static target in another position or dynamic one with unknown kinematics [17–19]. Therefore, multi-target trajectory planning remains an existing task to be done. Nevertheless, standard algorithms encounter sample inefficiency towards multiple targets. Furthermore, although the previous study realizes the task in which two end-effectors track moving targets in the learning phase, it still needs to re-train policies for each rotating speed [19] when deployed in the test environment. The underlying reason is that the lack of perception module restricts the practical application in different environments. Thus, we need to provide enough measure information for the real-world practice, for example a non-cooperative object's angular velocity. To sum up, there remains an open question on developing a motion planning system for moving target points on a non-cooperative object.

To address the above problems, the main contribution of this paper is a learning system for motion planning of the free-float dual-arm space manipulator towards non-cooperative objects. The essential function of the system is to realize the trajectory planning for target points on the surface of a spinning object by Module I and Module II, which is shown in Fig. 1. We further made the following contributions:

- We proposed the Constrained Hindsight Experience Replay (C-HER) algorithm for multiple targets within a large space, improving the sample efficiency of standard actor-critic framework.
- We developed the target's pose prediction module utilizing the point clouds of objects in space, which firstly estimates the rotating axis and angular speed of object, and then predicts the pose of target points.
- From the perspective of space robotics, this work deploys learning-based perception and planning modules in tracking target points on a non-cooperative object successfully.

The combination of representation learning and reinforcement learning makes our system possible to track a non-cooperative object without the prior knowledge of the spinning speed, so as to easily generalize to diverse environments in the future.

## 2 Related Work

In contrast to model-based methods, model-free motion planning methods avoid potential modeling errors of the complex system dynamics. Also, they didn't need experts to manually adjust the parameters of modular controllers. The development of model-free methods can be roughly divided into two periods. In the early time, researchers applied various searching approaches to solving the optimization problem of trajectory planning. For instance, population-based methods including Differential Evolution (DE) [8], Particle Swarm Optimization (PSO) [9] and Genetic Algorithm (GA) [10] can be leveraged to obtain the optimal joint trajectory parameterized by polynomial functions, B-Spline or Bézier curve. The results demonstrate the effectiveness when the start and end are fixed, but the uncertainty of model and disturbances have a huge impact on performance.

Hereafter, due to the significant generalization, deep reinforcement learning methods have become popular in the field of robotics, especially for the exploration of quadruped robots in unstructured environments [11, 20], in-hand object reorientation of a multi-finger robotic hand [12, 21], and multi-robot path planning [22–24]. For the free-floating space manipulator, Yan *et al.* proposed a trajectory planning method based on Soft Q-learning for a 3-DoF free-floating space robot [17]. To reach multiple targets within a large space, Wang *et al.* developed an improved version of Proximal Policy Optimization (PPO) for a 6-DoF space robot [25]. However, in our experiments, the method can not work well in a 12-DoF dual-arm environment. The subsequent work mainly focus on the position and attitude decoupling control for the setting of a single robotic arm [26]. Recently, there are some preliminary applications for free-float dual-arm space manipulator. Wu *et al.* applied Deep Deterministic Policy Gradient (DDPG) algorithm with the dense reward function, to realize the task for 8-DoF FFDASM [18]. Then, Li *et al.* considered the final velocity of end-effector into reward function to improve the previous method, but the results aren't evaluated by comparison [19]. In their later work, DDPG was combined with APF to improve the convergence of the algorithm, using potential energy difference for the reward calculation [27]. Furthermore, instead of introducing the constraints in the reward function, advanced methods usually take the combination of constraints and objectives into account during optimization, like Penalty method and Larangian method. Interestingly, their method trained a policy that can help the end-effectors reach the targets on the object, the spinning velocity of which is known constant. In this work, we solved trajectory planning for the spinning objects under unknown kinematic law, because we provided practical approaches to estimate the rotation of the object, and predict the position of targets.

## 3 Problem Statement

### 3.1 Modeling of Free-Float Dual-Arm Space Manipulator

The core components of FFDASM are a base satellite and two robotic manipulators, Arm-1 and Arm-2 respectively. Two manipulators are fixed rigidly on the base, and the gravity is ignored in the simulation. Concretely, we chose two 6-DoF UR-5 robots as the manipulators, and the cubic structure as the base. Due to its slight impact, we removed the solar array for simplicity. Furthermore, the default kinematic and dynamic parameters are similar to the real space robot.

We first defined some notions. Arm-1 and Arm-2 represent two mission arms, the end-effectors of which need to reach the targets in the work space individually.  $p_{e_i}$  and  $v_{e_i}$  denote the position and velocity of the end-effector of Arm- $i$ .  $\theta_i^j$  is the angle of  $j$ -th joint of Arm- $i$ , ranging from  $[-\pi, \pi]$ ,  $\Theta_i = [\theta_i^1, \theta_i^2, \dots, \theta_i^6]$ . In this case,  $\dot{\theta}_i^j$  naturally represents the angular velocity, and  $\tau_i^j$  means the torque of the joint. Moreover, the pose vector of the end-effector and base are expressed as  $r_{e_i}$  and  $r_b$ .

Based on kinematic equation, the velocity of the end-effector of Arm- $i$  can be formulated as:

$$\dot{r}_{e_i} = J_b \dot{r}_b + J_r^i \dot{\theta}_i \quad (1)$$

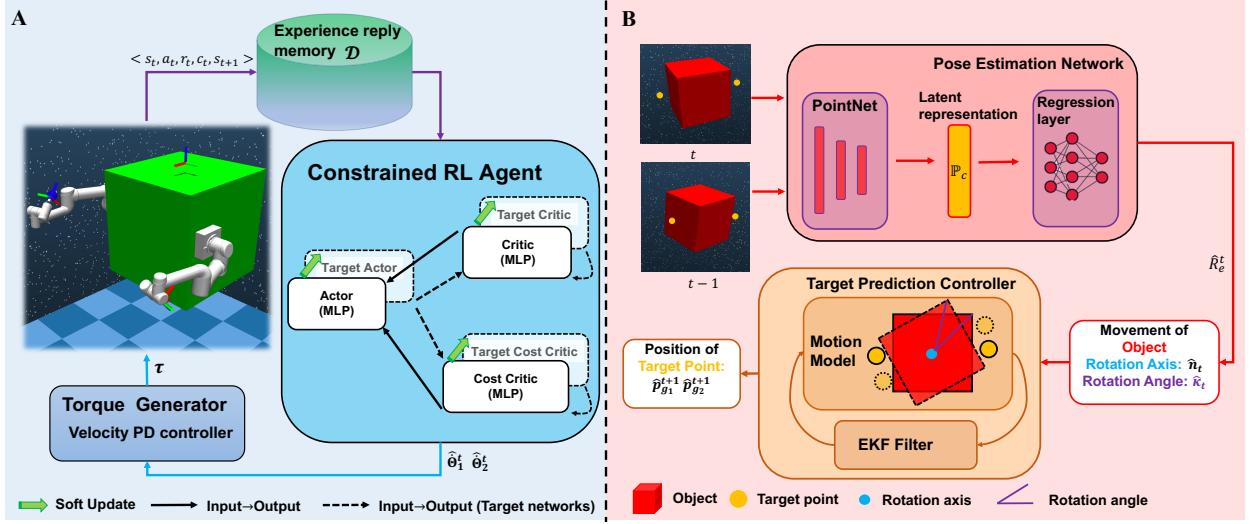


Figure 2: **A:** Framework of Module I: constrained multi-target trajectory planning via reinforcement learning. **B:** Framework of Module II consists of pose estimation and target prediction.

Where  $J_b$  is the Jacobian matrix with regard to the base, and  $J_r^i$  is the Jacobian matrix of Arm- $i$ .

Assuming the initial linear and angular momentum of system is zero, we can derive the momentum conservation equation:

$$H_b \dot{r}_b + H_r^1 \dot{\Theta}_1 + H_r^2 \dot{\Theta}_2 = 0 \quad (2)$$

where  $H_b$ ,  $H_r^1$  and  $H_r^2$  are the coupling inertia matrices of base, Arm-1 and Arm-2, respectively.

Therefore, the velocity of the end-effector of Arm-1 can be rewritten as:

$$\dot{r}_{e_i} = -(J_b H_b^{-1} H_r^1 - J_q^1) \dot{\Theta}_1 - J_b H_b^{-1} H_r^2 \dot{\Theta}_2 \quad (3)$$

As we can see, the sophisticated coupling relationships between base and two manipulators makes kinematic equation contain many dynamic parameters hard to be identified. Meanwhile, the current state of Arm-1 depends on not only the its last joint trajectories, but also the joint trajectories in Arm-2 and vice versa. Thus, different from the trajectory planning of robots on the ground, trajectory planning of a free-float space manipulator is a complex time-varying problem.

In addition, the dynamic model of a free-float dual-arm space manipulator can be derived as:

$$H_s \times \begin{bmatrix} \ddot{\Theta}_1 \\ \ddot{\Theta}_2 \end{bmatrix} + C_s \times \begin{bmatrix} \dot{\Theta}_1 \\ \dot{\Theta}_2 \end{bmatrix} + C_b \times \dot{r}_b = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (4)$$

where the detailed illustration of  $H_s$ ,  $C_s$  and  $C_b$  can be found in [6].

### 3.2 Constrained Markov Decision Process

Before constructing the constrained reinforcement learning algorithm, we need to introduce the formulation of Constrained Markov Decision Process (CMDP), which contains  $\langle S, A, P, \rho, R, C, \gamma_r, \gamma_c \rangle$ . Among them,  $S$  is the state space, and  $A$  is the action space. Additionally,  $P : S \times A \times S \rightarrow \mathbb{R}$  represents the probabilistic transition function, and  $\rho$  is the initial state distribution.  $R : S \times A \rightarrow \mathbb{R}$  denotes the reward function with respect to the state and action, and  $C : S \times A \rightarrow \mathbb{R}$  represents the cost function. Finally,  $\gamma_r$  and  $\gamma_c$  are the discount factor corresponding to reward and cost respectively, ranging from  $[0, 1]$ . When training at time step  $t$ , the agent stays in a state  $s_t$ , then executes an action  $a_t$ , receives a reward  $r_t$  and a cost  $c_t$ , and enters into a new state  $s_{t+1}$ . Notably, the action  $a_t$  is generated by a policy  $\pi(a_t | s_t)$ , made by a deterministic policy  $\mu(s_t)$  with a Gaussian noise in this paper. Besides, the total time steps in an episode  $m$  is  $T$ , and the sum of episodes are  $M$ .

In order to maximize the total reward and satisfy the constraints, the objective of our problem is expressed as:

$$\begin{aligned} \max \quad & \mathcal{J}_r = \mathbb{E}_{s_0 \sim \rho, a_{0:T-1} \sim \pi, s_{1:T} \sim P} \left[ \sum_{t=0}^{T-1} \gamma_r^t r_t \right] \\ \text{s.t. } & \mathcal{J}_c = \mathbb{E}_{s_0 \sim \rho, a_{0:T-1} \sim \pi, s_{1:T} \sim P} \left[ \sum_{t=0}^{T-1} \gamma_c^t c_t \right] \leq C_s \end{aligned} \quad (5)$$

where  $C_s$  represents the maximum threshold of the total cost. To estimate the relationships between reward, action and state, the state-action value and state-value functions have been constructed by:

$$\begin{aligned} Q_\pi^r(s, a) &= \mathbb{E}_{s_t \sim P, a_t \sim \pi} \left[ \sum_{t=0}^{T-1} \gamma_r^t r_t | s_0 = s, a_0 = a \right] \\ V_\pi^r(s) &= \mathbb{E}_{a \sim \pi} [Q_\pi^r(s, a)] \end{aligned} \quad (6)$$

Similarly, we can define the state-action cost value and state cost value functions as  $Q_\pi^c(s, a)$  and  $V_\pi^c(s)$  respectively. The detailed derivation is omitted because of the same principle.

## 4 Two-Module Motion Planning System

The specific architecture of our two-module system is shown in Fig. 2. Aimed at tracking the moving target points on the object, the whole system consists of Module I and Module II. Firstly, Module I acts as an important tool for the trajectory planning of FFDASM for the targets within a large workspace. Importantly, the potential positions of targets are included in our workspace, thus making it possible to further track the targets. With the contribution of reinforcement learning, the constrained optimization problem we proposed can be solved successfully. Secondly, the core task of Module II is to estimate the pose of object, and then predict the position of targets on the object. According to the prediction, it can assist Module I to track the target points. As shown in Fig. 2, Point Cloud Net takes the point clouds of object between two frames as input, to estimate the rotation axis  $\hat{n}_t$  and angle  $\hat{\kappa}_t$ . Hereafter, we built the kinematic model of target points on the surface of object, and then applied Extended Kalman Filter to predict the positions of target points next step. Finally, Module I takes as input the prediction and other feedback to control the space manipulator. Different from the end-to-end model, our two-module system offers some advantages, like more scalable and interpretable learning process. It is also worthy noting that our study initially opens up new possibilities for implementing the combination of representation learning and reinforcement learning in the trajectory planning of FFDASM. In the following, we described the details of main parts of our two-module learning system.

## 5 Module I: Multi-Target Trajectory Planning Method

### 5.1 Formulation of Optimization Problem

As discussed in the previous part, we transformed the trajectory planning problem into a reinforcement learning problem. To illustrate our optimization problem, we first introduced some concrete settings of components in MDP.

**State  $S$ :** the state vector  $s_t$  at the step  $t$  consists of the angular positions and velocities of joints, the positions of end-effectors and the position of target points of two manipulators, which means  $s_t = [\Theta_1^t, \dot{\Theta}_1^t, \Theta_2^t, \dot{\Theta}_2^t, p_{e1}^t, p_{e2}^t, p_{g1}^m, p_{g2}^m]$ .  $p_{gi}^m$  represents the target point of Arm- $i$ , which is sampled in the work space  $\mathcal{W}_i$  at the episode  $m$ .

**Action  $A$ :** the action vector at the step  $t$  is the desired velocities of joints,  $a_t = [\hat{\Theta}_1^t, \hat{\Theta}_2^t]$ . Then a velocity tracking PD controller takes  $a_t$  as input, to generate the torques of joints. Given that position controller performs less smoothly, it is better to choose the velocity controller [25].

**Probabilistic transition function  $P$ :** the core task of probabilistic transition function is to describe the stochastic transition  $s_{t+1} \rightarrow s_t$ . As we expected,  $P$  is corresponding to the dynamic model of our system, thereby resulting in the formulation:

$$s_{t+1} \sim P(s_{t+1} | s_t, a_t) = \mathcal{F}(s_t, a_t, \delta_t) \Leftrightarrow \begin{cases} (3) \\ (4) \\ \tau_t = K_{pd}(\hat{\Theta}_1^t, \hat{\Theta}_2^t) \\ \|*\| \leq U_* * \in [\Theta, \dot{\Theta}, \tau] \\ S_{SM} \cap \Omega = \emptyset \end{cases} \quad (7)$$

where  $K_{pd}$  represents PD controller,  $U_*$  is the maximum range of variables without the sake of safety, and  $S_{SM}$  and  $\Omega$  denote the space of space manipulators and base respectively. Additionally,  $\delta_t$  is the uncertainty of our model.

**Reward function  $R$ :** after executing an action, the agent will receive a reward based on the reward function  $R(s_t, p_{g_{1:2}}^m)$ , where  $p_{g_{1:2}}^m$  represents two target points at the episode  $m$ . In this paper, we utilized the sparse reward function, as Eq. (8) shows. As we can see, when the errors between end-effectors and target points are less than the thresholds  $U_{e_1}$  and  $U_{e_2}$ , the reward is 0, else -1 at the step  $t$ .

$$r_t = \begin{cases} 0, & \text{if } \|p_{e_1}^t - p_{g_1}^m\| \leq U_{e_1}, \|p_{e_2}^t - p_{g_2}^m\| \leq U_{e_2} \\ -1, & \text{else} \end{cases} \quad (8)$$

**Cost function  $C$ :** the prior studies [18, 19] didn't go further to explore the results under constraints. However, the violent fluctuation of base has a significant impact on communication of space manipulator, thereby leading to a failed task. In this case, we built a cost function with respect to the movement of base, as shown in Eq. (9).

$$c_t = \kappa(r_b^t - r_b^0)t \quad (9)$$

Noticeably, if the final pose of base is closer to the initial pose, the sum of cost becomes fewer.

Hereafter, following the above design, the trajectory planning of FFDASM can be defined as a reinforcement learning problem shown in Eq. (5), maximizing the objective function  $\mathcal{J}_r$  with the constraint  $\mathcal{J}_c$ .

$$\begin{aligned} \max_{\pi} \mathcal{J}_r &= \mathbb{E}_{s_0, \delta} \left[ \sum_{t=0}^{T-1} \gamma_r^t r_t \right] \\ \text{s.t. } \mathcal{J}_c &= \mathbb{E}_{s_0, \delta} \left[ \sum_{t=0}^{T-1} \gamma_c^t c_t \right] \leq C_s \\ s_{t+1} &= \mathcal{F}(s_t, a_t, \delta_t) \end{aligned} \quad (10)$$

## 5.2 Constrained Reinforcement Learning Algorithm

Considering the constrained optimization problem and the requirement of multiple target points, we proposed a multi-target constrained reinforcement learning algorithm, Constrained Hindsight Experience Replay **CHER**. Firstly, the method constructs three networks including a state-action value network  $Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi)$ , a deterministic policy network  $\pi(\cdot | s_t, p_{g_{1:2}}^m; \psi)$  and a cost-action value network  $Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)$ .  $\phi, \psi$  and  $\eta$  are the parameters of networks.

As we all know, the objective of  $Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi)$  and  $Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)$  is to estimate the sum of reward and cost with regard to the policy. Therefore, we can update  $\phi$  by minimizing MSE loss of the TD error [28]:

$$\begin{aligned} L_Q(\phi) &= E_{\mathcal{D}} \left[ \frac{1}{2} (y_t - \nabla_{\phi_1} Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi))^2 \right] \\ y_t &= r_t + \gamma_r \bar{Q}_{\pi}^r(s_{t+1}, \bar{\pi}(\cdot | s_{t+1}, p_{g_{1:2}}^m; \psi'), p_{g_{1:2}}^m; \phi') \end{aligned} \quad (11)$$

where  $\mathcal{D}$  is the relay buffer including interaction information  $< s_t, a_t, p_{g_{1:2}}^m, r_t, s_{t+1} >$  at the step  $t$  in the episode  $m$ ,  $\bar{Q}_{\pi}^r$  is the target Q network parameterized by  $\phi'$ , and  $\bar{\pi}$  is the policy network parameterized by  $\psi'$ . During the training,  $\phi$  and  $\psi$  are softly copied to  $\phi'$  and  $\psi'$  after some episodes repeatedly, which is shown as follows:

$$\phi' = \chi \phi' + (1 - \chi) \phi, \quad \psi' = \chi \psi' + (1 - \chi) \psi \quad (12)$$

where  $\chi \ll 1$ , meaning that the update is slow and stable. The update of  $Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)$  is similar to Eq. (11), so we omitted that for simplicity.

**Penalty method:** To address the problem, the first solution is the penalty method [29]. Literally, the penalty method adds a penalty term with respect to the constraints into the previous objective, leading to a new objective function:

$$\max_{\pi} \mathcal{J} = \mathcal{J}_r - \lambda_p (\mathcal{J}_c - C_s) \quad (13)$$

wherein  $\lambda_p \geq 0$  is a constant depending on different tasks. Thanks to introducing  $Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi)$  and  $Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)$ , we can replace the sum of reward with the estimated Q function. That means the minimization objective of the policy network can be formulated as:

$$L_{\pi}(\psi) = E_{\mathcal{D}} [-Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi) + \lambda_p (Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta) - C_s)] \quad (14)$$

When  $\lambda_p$  is larger, the policy will avoid the risk of increasing the cost more carefully.

**Algorithm 1** Constrained Hindsight Experience Replay (CHER)

---

```

1: Orthogonal initialize  $\phi, \phi', \psi, \psi', \eta, \eta', \lambda_p, \lambda_l$ 
2: Initialize replay buffer  $\mathcal{D}$ 
3: for episode  $m = 1, M$  do
4:   Sample targets  $p_{g_1}^m, p_{g_2}^m$  and initial state  $s_0$ 
5:   for step  $t = 0, T - 1$  do
6:     Sample a action from  $\pi_b(\cdot | s_t, p_{g_{1:2}}^m; \psi)$ 
7:     Execute the action  $a_t$  and observe a new state  $s_{t+1}$ 
8:   end for
9:   for step  $t = 0, T - 1$  do
10:     $r_t := R(s_t, p_{g_{1:2}}^m)$ 
11:    Store  $< s_t, a_t, p_{g_{1:2}}^m, r_t, s_{t+1} >$  into  $\mathcal{D}$ 
12:     $[\hat{p}_{g_1}^m, \hat{p}_{g_2}^m] := [p_{e_1}^T, p_{e_2}^T]$ 
13:     $r'_t := R(s_t, \hat{p}_{g_{1:2}}^m)$ 
14:    Store  $< s_t, a_t, \hat{p}_{g_{1:2}}^m, r'_t, s_{t+1} >$  into  $\mathcal{D}$ 
15:  end for
16:  for iteration  $n = 1, N$  do
17:    Sample a minibatch  $\mathcal{B}$  from the replay buffer  $\mathcal{D}$ 
18:    if Penalty method then
19:      Update the  $\psi, \psi', \phi, \phi'$  via Eq.(11),(12),(14) using minibatch  $\mathcal{B}$ 
20:    end if
21:    if Lagrangian method then
22:      Update the  $\psi, \psi', \phi, \phi', \lambda_l$  via Eq.(11),(12),(16),(17) using minibatch  $\mathcal{B}$ 
23:    end if
24:  end for
25: end for

```

---

**Lagrangian method:** other than the penalty method, the optimization problem can also be transformed into its Lagrangian dual form to optimize [30]. The optimal parameters of policy network is defined as:

$$\psi^* = \arg \max_{\psi} \min_{\lambda_l \geq 0} E_{\mathcal{D}}[Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi) - \lambda_l(Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta) - C_s)] \quad (15)$$

where  $\lambda_l$  is the Lagrangian multiplier that strike the balance between the constraint and objective. Similar to the penalty method, the update of policy network is approximated by according to stochastic gradient descent:

$$\nabla_{\psi} L_{\pi}(\psi) \propto E_{\mathcal{D}}[(-\nabla_{a_t} Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi) + \lambda_l \nabla_{a_t} Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)) \nabla_{\psi} \pi(\cdot | s_t; \psi)] \quad (16)$$

Furthermore, during training, the Lagrangian multiplier  $\lambda_l$  is updated by the following objective:

$$\begin{aligned} L(\lambda_l) &= \lambda_l(Q_{\pi}^c(s_t, a_t; \eta) - C_s) \\ \lambda_l &\leftarrow \max(0, \lambda_l + \zeta \nabla_{\lambda_l} L(\lambda_l)) \end{aligned} \quad (17)$$

where  $\lambda_l$  is the learning rate of Lagrangian multiplier.

**Hindsight Experience Replay (HER):** Moreover, to mitigate the sparsity of reward in multi-goal problem, we utilized the Hindsight Experience Replay (HER) method to improve the sample efficiency [31]. To be specific, the method assumes that the positions of two target points  $p_{g_1}^m$  and  $p_{g_2}^m$  at the episode  $m$  are the previous goals, and then the final positions of end-effectors  $p_{e_1}^T$  and  $p_{e_2}^T$  replace them as new goals. That means  $[p_{e_1}^T, p_{e_2}^T] \Rightarrow [\hat{p}_{g_1}^m, \hat{p}_{g_2}^m]$ . Meanwhile, the reward will be recalculated based on new goals, thus converting a failed experience into a successful experience.

To recap, after  $M$  episodes, the optimal parameters of three networks  $Q_{\pi}^r(s_t, a_t, p_{g_{1:2}}^m; \phi)$ ,  $Q_{\pi}^c(s_t, a_t, p_{g_{1:2}}^m; \eta)$  and  $\pi(\cdot | s_t, p_{g_{1:2}}^m; \psi)$  can be provided by the above minimization objectives. Algorithm 1 depicts the pseudo code for training CHER. Among them,  $\pi_b(\cdot | s_t, p_{g_{1:2}}^m; \psi)$  denotes a noisy version of current policy for exploration during the training.

## 6 Module II: Pose Estimation and Prediction Method

### 6.1 Pose Estimation based on Point Clouds

The essential function of Module II is to construct an integrated system that can estimate the pose of the object and then predict the position of target points for trajectory planning. Notably, under the intense influence of sunlight in space, it is difficult for space manipulators to estimate the pose via RGBD images. Therefore, we just consider the point clouds of objects obtained by LiDAR sensors, which are more accurate in real-world practices.

To be specific, an intuitive approach is to take the point cloud at this frame as input, and the estimated pose of the object as output. However, the results extremely depend on the global coordinate frame. Thus, it is beneficial to utilize the point clouds between two frames to evaluate the relative rotation matrix. Referring to [32], we also built a 3-layered PointNet [33] parameterized by  $\omega$ , to encode two point clouds. Then, we extracted the latent representations  $\mathbb{P}_c$  by the MLP neural network. Additionally, the batch normalization layer is added behind each fully connected layer to accelerate the training. Finally, the network outputs the predicted pose vector  $\vec{r}_e$ , and rewrite it as  $\hat{R}_e$ . Following [34], the training objective can be expressed as:

$$\min_{\omega} L_R = 2 \arcsin\left(\frac{1}{2\sqrt{2}} \|R_p - \hat{R}_e(\mathbb{P}_c; \omega)\|_F\right) \quad (18)$$

where  $R_p$  is the ground truth of the rotation matrix. In the training process, we sampled random points on the surface of objects, and then constructed two point clouds by randomly rotating the original point cloud. Thus, the label is the true relative rotation between them, as well as the input of networks is two generated point clouds.

To sum up, the first part of Module II achieves the relative pose estimation between two frames. Although the predicted rotation matrix still can not directly provide the prior information for the latter target prediction, we found that the predicted rotation can facilitate the recovery of the relative rotation axis  $\hat{n}_t$  and angle  $\hat{\kappa}_t$  of object at the step  $t$  based on Eq. (19).

$$\hat{\kappa}_t = \arccos\left(\frac{\text{tr}(\hat{R}_e^t) - 1}{2}\right), \quad \hat{R}_e^t \hat{n}_t = \hat{n}_t \quad (19)$$

Furthermore, it is worth noting that though the traditional ICP [35] method can recover the rotation axis and angle by an iterative optimization, the critical advantage of a learning system is higher scalability and generalization. The latent representation  $\mathbb{P}_c$  provided by PointNet can benefit the classification and detection of objects after, which is discussed in the experiments part.

### 6.2 Target Prediction via Extended Kalman Filter

According to the estimated relative rotation axis  $\hat{n}_t$  and angle  $\hat{\kappa}_t$  of the object at the step  $t$ , we can construct the kinematic model of the target point on the surface of the object. Firstly, without the effect of external forces, we assume the object moves in a regular circle around the rotation axis  $n_r$ , which means  $\hat{n}_t$  is close to a constant. In practice, this assumption is easily satisfied in a low-gravity environment [19]. Secondly, the target point is attached to the surface of object, thereby with the same rotating velocity.

In the context of the above assumptions, we built the motion and observation models for target point as shown in Eq. (20), where the state vector  $X_t^i$  represents  $[x_t^i, y_t^i, \zeta_t^i, v_t^i]$  observation vector  $Z_t^i$  represents  $[\tilde{x}_t^i, \tilde{y}_t^i]$ , and  $\delta_t$  is the control cycle. Among them,  $x_t^i$  and  $y_t^i$  are the position of the target point in the rotating plane  $\mathcal{S}_{\perp n_r}$ ,  $\zeta_t^i$  is the rotation angle, and  $v_t^i$  is the velocity of the target point. Notably,  $x_t^i$  and  $y_t^i$  can be derived by the projection transformation  $\mathbb{T}(p_{g_i}^t; \mathcal{S}_{\perp n_r})$ , where  $p_{g_i}^t$  is the position of the target point of Arm- $i$ . To indicate the gap between the model and reality, some uncertain terms are included in two models. Specifically,  $\varpi_t^i$  and  $\varrho_t^i$  denote the process and measurement noises.

$$\begin{cases} x_{t+1}^i = x_t^i + v_t^i \cos(\zeta_t^i) \delta_t \\ y_{t+1}^i = y_t^i + v_t^i \sin(\zeta_t^i) \delta_t \\ \zeta_{t+1}^i = \zeta_t^i + \hat{\kappa}_t^i \delta_t \\ v_{t+1}^i = v_t^i \end{cases} \Rightarrow X_{t+1}^i = f(X_t^i) + \varpi_t^i, \quad \varpi_t^i \sim N(\mathbf{0}, \mathcal{Q}^i) \quad (20)$$

$$Z_{t+1}^i = [\mathbf{I}_{2 \times 2} \ \mathbf{0}] X_{t+1}^i + \varrho_t^i = h(X_{t+1}^i) + \varrho_t^i, \quad \varrho_t^i \sim N(\mathbf{0}, \mathcal{R}^i) \quad (21)$$

Since the models are non-linear, we can use the Extended Kalman Filter (EKF) [36] to obtain the predicted state vector  $\hat{X}_{t+1}^i$ , as Eq. (22) shows.

$$\hat{X}_{t+1}^i = f(X_t^i) + \mathbf{K}_t^i (Z_t^i - h(f(X_t^i))) \quad (22)$$

$\mathbf{K}_t^i$  represents the Kalman gain derived by EKF method at the step  $t$ , and mathematical details can be found in [36]. Following the above equation, we have the access to predict the position of the target point in real time. Based on the previous projection transformation, the position of the target point for Arm- $i$  can be written as:

$$\hat{p}_{g_i}^{t+1} = \mathbb{T}(\hat{x}_{t+1}^i, \hat{y}_{t+1}^i; \mathcal{S}_{\perp n_r}) \quad (23)$$

Remarkably, when  $\hat{p}_{g_i}^{t+1}$  is taken as an input in Module I at the step  $t$ , the strategy can reach the dynamic target point within a bounded time and error. We proposed an upper bound of convergent time.

**Theorem 1:** The velocity of target point is less than the velocity of end-effector, which means  $v_t \leq \dot{p}_{e_i}^t, t \in T$ , and the error between  $\hat{p}_{g_i}^{t+1}$  and  $p_{g_i}^{t+1}$  is bounded by  $\epsilon_{p_{g_i}}$ . Thus, we can guarantee the formulation satisfied, when the initial error is  $d_{e_i}$ :

$$\|p_{e_i}^{T_B} - p_{g_i}^{T_B}\| \leq U_{e_i} + \epsilon_{p_{g_i}}, d_{e_i} \leq \int_t^{T_B} (\dot{p}_{e_i}^t - v_t) dt \quad (24)$$

The minimum of  $T_B$  is the upper bound of convergent time, thereby meaning the end-effector will track the target point after  $T_B$ .

## 7 EXPERIMENTS

### 7.1 Comparison with Other Baselines

The goal of Module I is to provide a trajectory planning method for multiple targets in a large work space. We chose 3 state-of-the-art RL-based methods for FFDASM as other baselines.

- **Wu’s method [18]:** Wu’s method realized the trajectory planning for 8-Dof free-float dual-arm space manipulators for the single target.
- **Wang’s method [25]:** Wang’s method solved the task of multiple targets for a single arm based on an improved version of PPO algorithm.
- **SAC-D:** Considering Soft Actor-Critic (SAC) algorithm achieves superior performance in many robotic control task [37], we designed SAC-D algorithm for our task, in which we applied the similar reward function in [25].

Compared with other baselines for FFDASM, the experimental results of our method illustrate a significant improvement in planning accuracy. Concretely, as shown in Fig. 3-A, our method **CHER-P** (Penalty method  $\lambda_p = 0.5$ ) facilitates the end-effector to achieve the task within the least number of episodes. Admittedly, other algorithms have performed well in some tasks. However, multi-target trajectory planning for 12-DoF dual-arm space manipulators remains a critical challenge for most algorithms, as shown in Fig. 3-A. This is because the increasing dimensions of state and action have a huge impact on the efficiency of exploration, especially for multiple targets. Our method (CHER-P( $\lambda_p = 0.5$ )) introduced HER method to improve sample efficiency, facilitating to speed up the convergence. Note that because other algorithms is unable to satisfy the basic need of planning accuracy, we overlook the requirements of constraints in their training.

Furthermore, to evaluate the performance of accuracy and constraints together, we plotted  $e_1$ ,  $e_2$  and cost value in testing episodes for 3 algorithms, which are as follows:

- **CHER-P** (Penalty method) :  $\lambda_p$  is a constant coefficient of constraints .
- **CHER-L** (Lagrangian method):  $\lambda_l$  is a varying coefficient of constraints, which can be updated by Eq. (17) iteratively.
- **HER-R+C**: the method based Hindsight Experience Replay (HER) models the trajectory planning as a Markov Decision Process (MDP) instead of CMDP. Thus, A new reward function can be derived as  $R_{\text{new}} = R + C$ , where  $R$  and  $C$  are above-described.

As Fig. 3-B shows, it can be observed that the algorithms based HER except for **HER-R+C** achieve comparable performances on planning accuracy. That means if the reward function contains multiple types of objectives (rewards or costs), it is hard to strike a balance between these demands during training. Additionally, from the perspective of the cost value, although **CHER-L** can adjust the Lagrangian multiplier  $\lambda_l$  according to the violation of constraints dynamically, the final stage in the training will exist significant constraint violations. Alternatively, in this paper, **CHER-P** ( $\lambda_p = 0.5$ ) strikes the most suitable balance between reward and cost. To recap, the penalty coefficient of constraints plays an important role in practical usage. To sum up, Table 1 clearly illustrates learned policy performance

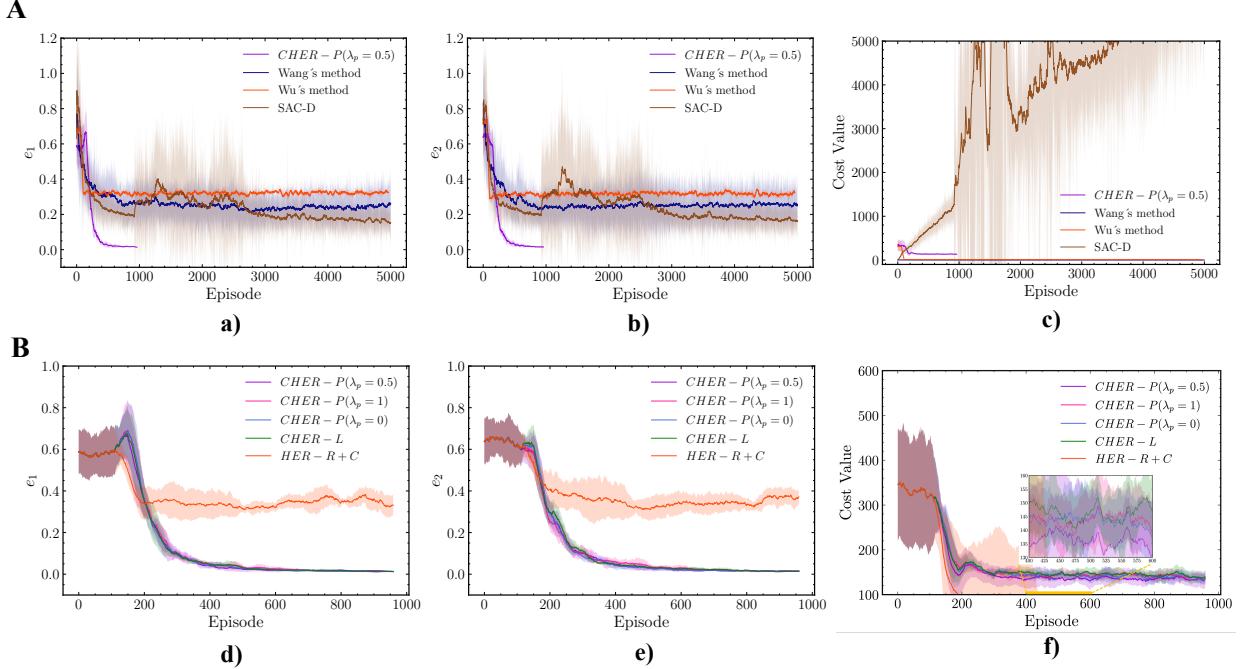


Figure 3: **A:** a-c) Position errors and cost value using our method and baseline methods. **B:** d-f) Position errors and cost value based on different constrained setting. (Curves smoothed report the mean of 5 random seeds, and shadow areas show the variance.)

for different reinforcement learning algorithms, and we can see that the versions of our method have superior results in contrast to others.

Finally, It is worthy noting that all algorithms used the same model to built networks, which consists of four fully connected layers with ReLU activation. The number of hidden units per layer is 256, and the activation of output layer in policy network is Tanh function. Additionally, the input and output are also the same variables, and other parameters in each algorithm are provided by some papers [18, 25] or OpenAI baselines [38].

Table 1: The best performance of different learning algorithms. The best results are marked as bold. Note that we considered its cost value is valid only if an algorithm meets the demand,  $e_1$  or  $e_2 \leq 0.05$ .

Algorithm	$e_1$ (m)	$e_2$ (m)	Cost Value
<b>Wu's method [18]</b>	$0.324 \pm 0.012$	$0.323 \pm 0.010$	$9.765 \pm 0.839$
<b>Wang's method [25]</b>	$0.255 \pm 0.110$	$0.171 \pm 0.085$	$8.196 \pm 0.384$
<b>SAC-D</b>	$0.096 \pm 0.077$	$0.100 \pm 0.031$	$5844 \pm 1846$
<b>HER-R+C</b>	$0.331 \pm 0.052$	$0.371 \pm 0.035$	$40.462 \pm 8.620$
<b>CHER-L</b>	$0.013 \pm 0.002$	<b><math>0.014 \pm 0.002</math></b>	$139.207 \pm 12.170$
<b>CHER-P(<math>\lambda_p = 0</math>)</b>	<b><math>0.012 \pm 0.002</math></b>	<b><math>0.014 \pm 0.002</math></b>	$135.440 \pm 11.918$
<b>CHER-P(<math>\lambda_p = 1</math>)</b>	<b><math>0.012 \pm 0.002</math></b>	$0.014 \pm 0.003$	$134.865 \pm 11.404$
<b>CHER-P(<math>\lambda_p = 0.5</math>)</b>	$0.014 \pm 0.002$	$0.015 \pm 0.004$	<b><math>131.838 \pm 10.558</math></b>

## 7.2 Robustness of Trajectory Planning

Utilizing the model trained by **CHER-P** algorithm, we evaluated the performance of trajectory planning under random five pairs of starts and ends in working space. Most importantly, the testing environment is different from that of the training, because the initial position of end-effector is fixed in training. In this case, 100 repeated results shown in Fig. 3-C demonstrate our method has a strong robustness. What we should also highlight is the smoothness of trajectory our method generated. We can observe that two end-effectors reach the targets quickly and converge to a narrow region near targets. Although the final positions of the two end-effectors meet the demand of the threshold we set, the final error has a slight difference from each other. Interestingly, we found the same situation in many previous studies [18, 19].

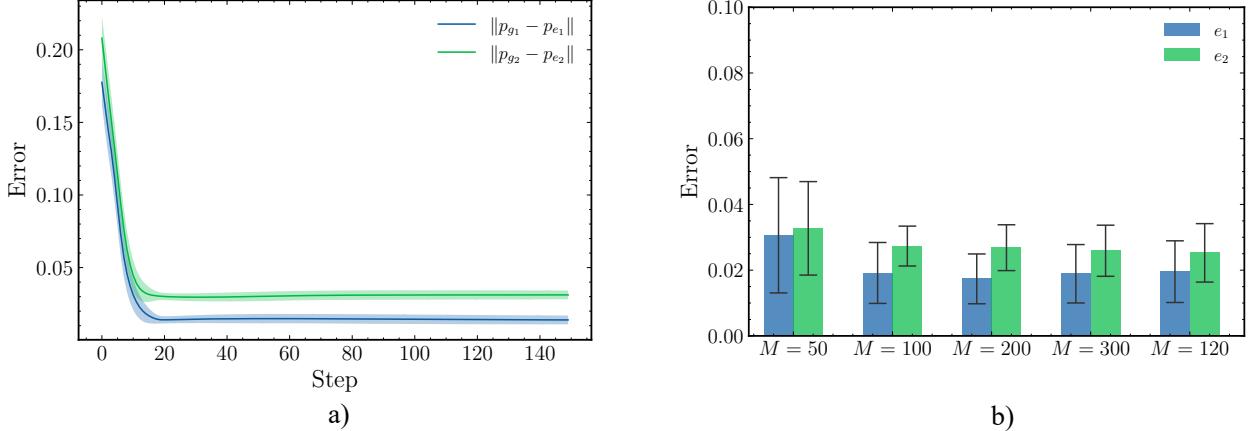


Figure 4: a): Errors  $e_1$  and  $e_2$  during an episode under different initial position of end-effectors. b): Errors  $e_1$  and  $e_2$  under different masses of base.

Intuitively, the reason behind that is the coupling effect between two arms make it hard to balance two individual reaching tasks for them. Also, we guessed multi-agent learning will amend the bias, which will be discussed in future work.

In addition, considering the gap between the real world and simulation, we tested our trained policy under different scenarios where the mass of base is changing. Fig. 4 shows the robust property of our method clearly. Even though the mass of the base decreases by nearly 50%, the planning error doesn't saw a significant increase finally. This is because such a sufficient exploration in multi-target space reduces the sensitivity of disturbance. The above results demonstrate our method have obtained considerable robustness and generalization abilities, so as to have the potential to transfer the model to the real world.

### 7.3 Generalization of Pose Estimation

In the first part of Module II, the pose estimation network based on point clouds of the object recovers the rotation axis and angle of object in space. In order to enhance the generalization, we didn't make full use of point clouds, but sampled 128 points from the surface of the object, adding some Gaussian noises to the sampled points. Meanwhile, we introduced their normal vector to facilitate the convergence rate. Furthermore, although the shape of object mainly is a cube, we extended the types of object, adding two open-source object datasets, ContactDB [39] and YCB [40]. Actually, a better way is to use specific objects in space, but there is no related sources, which will be our future direction. Because the scale of objects are not consistent, we normalized the point clouds of objects before training. Fig. 4-a illustrates the pose errors in the testing environment, where the cubic object (a non-cooperative object) rotates in 5 different speeds. We can see that the relative errors remain at the lowest point under the highest spinning speed. This is because the relative difference of two point clouds is slight in two frames, leading to the ambiguity of latent representation  $\mathbb{P}_c$ . As a matter of fact, in real-world practices, we run the process of pose estimation in a low frequency to solve the problem, because the object's movement is sustained for a long term in space. Furthermore, to better exploit  $\mathbb{P}_c$ , we took  $\mathbb{P}_c$  as input for the classification task. Based on the MLP network and cross-entropy loss, the testing accuracy is  $93.34 \pm 5.3\%$ , illustrating our framework is scalable and effective. Finally, we appreciate the released code in [32], because we designed our experiments referring to it.

After estimating the relative pose of the object between two frames, we can obtain the rotation axis and angle of the object during the period according to Eq. (19). Then, according to our motion model of target points, we can get the predicted position of target points based on EKF method next step.

## 8 Validation of Target Tracking

After estimating the relative pose of the object between two frames, we can obtain the rotation axis and angle of object during the period according to Eq. (19). Then, according to our motion model of target points, we can get the predicted position of target points based on EKF method next step. With the access to predicted target positions, our method can

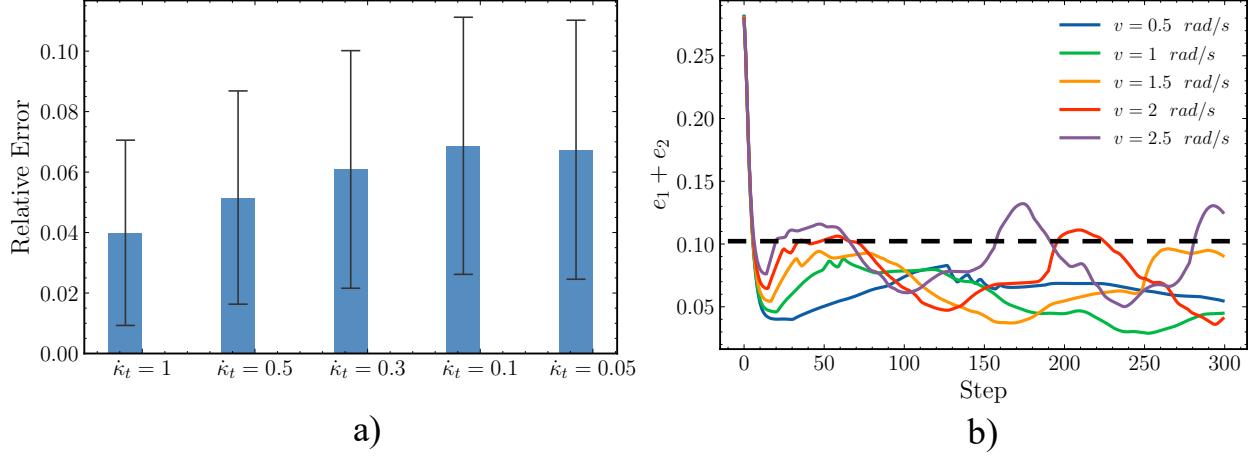


Figure 5: a): Relationship between relative error and spinning speed of object. b): Curves of the sum of errors  $e_1 + e_2$  under different spinning speeds.

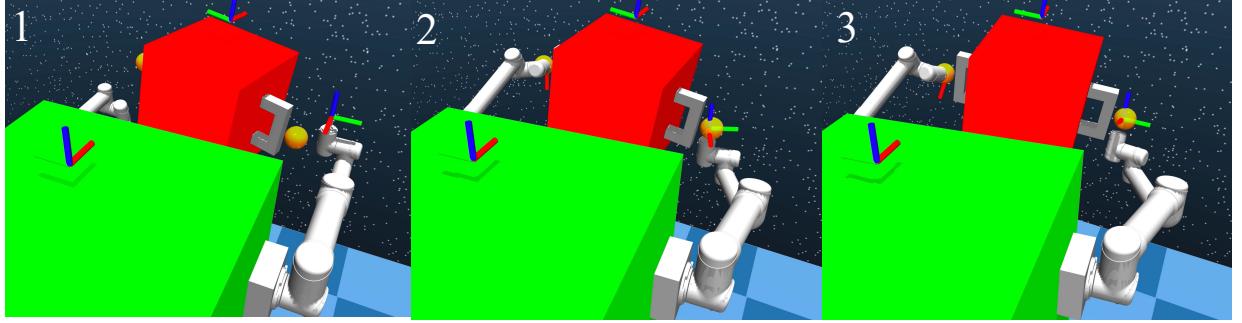


Figure 6: Video snapshots of the process that space manipulator tracks the target points.

realize tracking dynamic targets in space. Firstly, we set a testing environment, where the spinning speed of targets is ranging from  $0.5 \text{ rad/s}$  to  $2.5 \text{ rad/s}$ , and the radius of rotation is  $0.15 \text{ m}$ . As shown in Fig. 5, the end-effector will track the target point successfully. However, we can notice that when the spinning speed is  $2.5 \text{ rad/s}$ , the sum of errors  $e_1 + e_2$  shows an upward trend that means a sign of un-convergence. The underlying reason can be analyzed by Theorem 1. Specifically, when the velocities of end-effectors are less than those of targets, convergent time  $T_B$  is  $\infty$  and the sum of errors can not be bounded by  $U_{e_i} + \epsilon_{p_{g_i}}$ .

Furthermore, we simulated a practical scenario to better exploit the advantage of our system. After an occurrence of a failure, a target satellite uncontrollably rotates around an axis. The commander wants a free-float dual-arm space manipulator to catch the target satellite. After several orbital transfers, our manipulator stays a certain distance with the target satellite. Then, LiDAR sensors on space manipulator receives the point clouds of target satellite, and the pose estimation network take them as input to estimate the rotation axis and angular speed. Importantly, based on kinematic characteristics of target satellite, target prediction controller can predict the future positions of target points for end-effectors. Finally, because the trajectory of target points is involved in our training work space, Module I allows the end-effectors to track the moving target points. Fig. 6 shows the video snapshots of the process. While in this work we didn't consider the behavior capturing the target satellite, it can be further extended, which we leave as an avenue for future work.

## 9 CONCLUSIONS

We proposed a learning-based motion planning system for a free-float dual-arm space manipulator to track dynamic targets, which contains Module I and Module II. For Module I, the comparison experiments demonstrate our multi-target trajectory planning algorithm is able to reach the targets within a large workspace for a 12-DoF dual-arm space manipulator. Meanwhile, taking advantage of Penalty method and Lagrangian method, our algorithm strikes a balance between objective and constraints efficiently. In this case, our study facilitates the RL-based method to be applied in space with multiple sophisticated constraints. Moreover, the combination of representation learning and EKF method allows Module II to estimate the pose of object and predict the position of target points. Importantly, the extended experiments illustrate our method has high scalability and generalization so as to achieve a promising result on a practical scenario of tracking targets on a non-cooperative object.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant U21B6002.

## References

- [1] Mitsuhige Oda, Kouichi Kibe, and Fumio Yamagata. Ets-vii, space robot in-orbit experiment satellite. In *Proceedings of IEEE international conference on robotics and automation*, volume 1, pages 739–744. IEEE, 1996.
- [2] Kazuya Yoshida, Kenichi Hashizume, and Satoko Abiko. Zero reaction maneuver: Flight validation with ets-vii space robot and extension to kinematically redundant arm. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 441–446. IEEE, 2001.
- [3] Hanlei Wang and Yongchun Xie. Passivity based adaptive jacobian tracking for free-floating space manipulators without using spacecraft acceleration. *Automatica*, 45(6):1510–1517, 2009.
- [4] Yicheng Liu, Kedi Xie, Tao Zhang, and Ning Cai. Trajectory planning with pose feedback for a dual-arm space robot. *Journal of Control Science and Engineering*, 2016, 2016.
- [5] Kazuya Yoshida, Dimitar Dimitrov, and Hiroki Nakanishi. On the capture of tumbling satellite by a space robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4127–4132. IEEE, 2006.
- [6] Wen Yan, Yicheng Liu, Qijie Lan, Tao Zhang, and Haiyan Tu. Trajectory planning and low-chattering fixed-time nonsingular terminal sliding mode control for a dual-arm free-floating space robot. *Robotica*, pages 1–21, 2021.
- [7] Panfeng Huang, Jianping Yuan, and Bin Liang. Adaptive sliding-mode control of space robot during manipulating unknown objects. In *2007 IEEE International Conference on Control and Automation*, pages 2907–2912. IEEE, 2007.
- [8] Mingming Wang, Jianjun Luo, Jing Fang, and Jianping Yuan. Optimal trajectory planning of free-floating space manipulator using differential evolution algorithm. *Advances in Space Research*, 61(6):1525–1536, 2018.
- [9] Mingming Wang, Jianjun Luo, and Ulrich Walter. Trajectory planning of free-floating space robot using particle swarm optimization (pso). *Acta Astronautica*, 112:77–88, 2015.
- [10] Zhengcang Chen and Weijia Zhou. Path planning for a space-based manipulator system based on quantum genetic algorithm. *Journal of Robotics*, 2017, 2017.
- [11] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- [12] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.
- [13] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.
- [14] Lianglun Cheng Jie Zhong, Tao Wang. Collision-free path planning for welding manipulator via hybrid algorithm of deep reinforcement learning and inverse kinematics. *Complex Intelligent Systems*, 2, 2021.
- [15] Salhotra Gautam Pertsch Karl Yamada Jun, Lee Youngwoon. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. 10 2020.
- [16] Piastra Marco Ferrara Antonella Sangiovanni Bianca, Incremona Gian Paolo. Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning. *IEEE Control Systems Letters*, PP:1–1, 06 2020.

- [17] Changzhi Yan, Qiyuan Zhang, Zhaoyang Liu, Xueqian Wang, and Bin Liang. Control of free-floating space robots to capture targets using soft q-learning. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 654–660. IEEE, 2018.
- [18] Y. H. Wu, Z. C. Yu, C. Y. Li, M. J. He, and Z. M. Chen. Reinforcement learning in dual-arm trajectory planning for a free-floating space robot. *Aerospace Science and Technology*, 98(1):105657, 2020.
- [19] Yinkang Li, Xiaolong Hao, Yuchen She, Shuang Li, and Meng Yu. Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning. *Aerospace Science and Technology*, 109:106446, 2021.
- [20] Vassilios Tsounis, Mitja Alge, Joonho Lee, Farbod Farshidian, and Marco Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.
- [21] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.
- [22] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, 2018.
- [23] Evan Prianto, MyeongSeop Kim, Jae-Han Park, Ji-Hun Bae, and Jung-Su Kim. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor-critic with hindsight experience replay. *Sensors*, 20(20):5911, 2020.
- [24] Huy Ha, Jingxi Xu, and Shuran Song. Learning a decentralized multi-arm motion planner. *arXiv preprint arXiv:2011.02608*, 2020.
- [25] Shengjie Wang, Xiang Zheng, Yuxue Cao, and Tao 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)*, pages 3724–3730, 2021.
- [26] Shengjie Wang, Yuxue Cao, Xiang Zheng, and Tao Zhang. Collision-free trajectory planning for a 6-dof free-floating space robot via hierarchical decoupling optimization. *IEEE Robotics and Automation Letters*, 7(2):4953–4960, 2022.
- [27] Wenshan Zhu Jun Sun Xiaolong Zhang Shuang Li Yinkang Li, Danyi Li. Constrained motion planning of 7-dof space manipulator via deep reinforcement learning combined with artificial potential field. *Aerospace*, 9:163–181, 2022.
- [28] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [29] Peter Geibel and Fritz Wysotski. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.
- [30] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.
- [31] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.
- [32] Wenlong Huang, Igor Mordatch, Pieter Abbeel, and Deepak Pathak. Generalization in dexterous manipulation via geometry-aware multi-task learning. *arXiv preprint arXiv:2111.03062*, 2021.
- [33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [34] Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *Advances in neural information processing systems*, 31, 2018.
- [35] François Fleuret, François Fleuret, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.
- [36] Keisuke Fujii. Extended kalman filter. *Reference Manual*, pages 14–22, 2013.
- [37] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

- [38] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. <https://github.com/openai/baselines>, 2017.
- [39] Samarth Brahmbhatt, Cusuh Ham, Charles C Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8709–8719, 2019.
- [40] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.