Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Control of Space Flexible Manipulator Using Soft Actor-Critic and Random Network Distillation *

Chen Yang[1] and Jun Yang[2]

1. *Center for Artificial Intelligence and Robotics*
Tsinghua Shenzhen International Graduate School
518055 Shenzhen, Guangdong Province, China

cyang18@mails.tsinghua.edu.cn
yangjun603@tsinghua.edu.cn

Xueqian Wang[1, *] and Bin Liang[2, 3, *]

2. *Department of Automation, Tsinghua University*
100084 Beijing, China
3. *Research Institute of Tsinghua University in Shenzhen*
518057 Shenzhen, Guangdong Province, China

wang.xq@sz.tsinghua.edu.cn
liangbin@tsinghua.edu.cn

Abstract – **Due to the excessive degree of freedom of the space flexible manipulator, we can hardly obtain its accurate dynamic model for its motion planning. In this work, we formulate the precise motion control of the free-floating space piecewise constant curvature (FSPCC) continuum manipulator (*i.e.* space flexible manipulator) as a sparse reward problem in reinforcement learning, and use the Soft Actor-Critic (SAC) algorithm along with the Random Network Distillation (RND) method to train the optimal policies. Firstly, we use the RND method to jointly train a predictive network and a fixed network. The discrepancy between the output values of the two networks is served as an internal reward for the environment. Secondly, the SAC algorithm aims to maximize the expected return and the entropy of the policy. Policies with high entropy will successfully complete the task while acting as randomly as possible. Finally, the internal rewards tend to incentivize the agent to explore more widely for faster convergence of the algorithm. We applied this method to the FSPCC continuum manipulator simulation model and the results demonstrate that the SAC algorithm together with RND method can control the FSPCC continuum manipulator to catch the target quickly, even in the presence of sparse reward.**

Index Terms - Sparse reward, space free-floating piecewise constant curvature, soft actor-critic, random network distillation.

I. Introduction

The diversity of space on-orbit tasks has led to more and more research on space robots. However, the space rigid-link manipulators cannot complete the complex space tasks smoothly due to the complexity of the space tasks and the limitation of the working space. So the space continuum manipulators are required to accomplish these tasks because of their flexibility. Research on space continuum manipulators is still in its infancy, but there are many methods for controlling and modeling space robot and continuum manipulators. For space robots, Li et al. [1] used hand-eye camera and Kalman filter to get target feature, pose and velocity and then proposed an autonomous path planning for space robot to capture target. Shui et al. [2] proposed the null space approach to reparameterize the path and found the optimal joint velocity with the Newton iteration method. Yan et al. [3] used a new method named soft q-learning (SQL) [4] for free-floating space robots to capture targets. For continuum manipulators, Mahl et al. [5] proposed a kinematic modeling approach to establish a kinematic model of continuum manipulators which divides a single part into a number of virtual units with piecewise constant curvature. Chawla et al. [6] introduced a new approximate kinematic model based on phase and actuator length differences and compared a lot of kinematic modes with the new model for continuum manipulators. Huang et al. [7] built the piecewise constant curvature (PCC) model in V-REP environment for the continuum robot and used a model-based policy search algorithm named guided policy search (GPS) [8] to control a PCC continuum manipulator to capture the target. Therefore, the methods proposed in [1, 2, 5, 6] are based on accurate dynamic models for space robots and continuum manipulators. The biggest difficulty in controlling the space continuum manipulator is to obtain its dynamic model due to the non-holonomic constraints and dynamic singularities of the space continuum manipulator system and the elastic components and strong coupling inside the space continuum manipulator. In order to avoid this difficulty, we proposed a new method to control the space continuum manipulator to capture the target without knowing what the environment is.

Reinforcement learning (RL) provides us with a new framework to solve complex sequential decision-making problems such as robotic control problems. And RL algorithms can be divided into two methods: model-based and model-free methods. These methods give us some new ideas for solving complex and difficult problems by trail-and-error ways because they do not need to know what the environment is. Model-based methods have better sampling efficiency, these methods require some additional cost of human supervision and model estimation and they are difficult to implement. On the contrary, Model-free methods have better asymptotic performance and they are easier to implement. However, this kind of methods typically suffer from two major challenges: high sample complexity and brittle convergence properties.

RL algorithms can also be divided into on-policy learning algorithms and off-policy learning algorithms. On-policy learning algorithms always need to collect new samples for each gradient step such as trust region policy optimization

* This work is supported by the National Natural Science Foundation of China (No.U1813216 and No.61803221), the Science and Technology Planning Project of Guangdong Province (No. 2017B010116001), and the Basic Research Program of Shenzhen (JCYJ20160428182227081 and JCYJ20170817152701660).

(TRPO) [9], proximal policy optimization (PPO) [10] or asynchronous advantage actor-critic (A3C) [11]. Hence, the number of samples per step increases as the complexity of the task increases, which often requires a large price. On the contrary, off-policy algorithms often reuse past samples. However, this kind of algorithms suffer from a major challenge for stability and convergence in continuous state and action space. To explore an efficient and stable off-policy RL algorithm for continuous state and action space, Haarnoja et al. [12] proposed an actor-critic RL algorithm named soft actor-critic (SAC) under the maximum entropy RL framework recently. Compared with other deep RL algorithms, the SAC algorithm has the advantage of strong exploration ability, high sample efficiency and good robustness because it also trains maximum entropy stochastic policies with fewer samples, so it is more suitable for application in simulation and real robots.

Generally speaking, the problem of the robotic manipulator grabbing the target is a dense reward problem. However, in some cases where the simulation model and the external environment are very complicated, simply setting the dense reward problem may not complete the task well. But in the sparse reward problem we can use some methods to introduce internal rewards and increase exploration, so setting it to a sparse reward issue may make it easier to complete the task. Kulkarni et al. [13] proposed a hierarchical-DQN (h-DQN) framework to accomplish tasks by setting and completing many sub-goals in the sparse reward problem. Burda et al. [14] introduced the random network distillation (RND) to combine intrinsic and extrinsic rewards and to provide an improvement in exploration.

In this work, we use soft actor-critic (SAC) algorithm and the random network distillation (RND) to control the FSPCC continuum manipulator to capture the target. There are two main reasons for using SAC algorithm and RND rather than other RL algorithms: 1) SAC is one of the model-free algorithms with the highest sampling efficiency and robustness, so that it can be used well in real manipulators instead of simulation. 2) RND can promote the exploration ability of SAC and let it explore areas that have never been explored and make it converge faster.

The structure of this paper is as follows: Section 2 introduces the principles of SAC algorithms and RND. Section 3 briefly introduces the space continuum manipulator and explains the details of the SAC algorithm and RND controlling the continuous manipulator to capture the target. The simulation results are shown in section 4 and the section 5 gives the conclusions and future work.

II. Problem Formulization

We first explain notation and maximum entropy reinforcement learning, then briefly introduce soft actor-critic algorithm and random network distillation.

A. Notation

Markov decision process (MDP) provides theoretical support for the development of deep reinforcement learning and we assume that the action space A and state space S are continuous. The state transition probability $P(s_{t+1}|s_t,a_t)$ as a dynamic model of reinforcement learning can be expressed as the probability density of transitioning to the next state s_{t+1} under a given state s_t and given action a_t. We don't need to know this state transition probability. Furthermore, the environment will give an immediate reward for each transition. It is worth mentioning that the reward discount factor $\gamma \in [0,1)$ is to ensure that the accumulated reward of the infinite-horizon Markov decision process is bounded. $\rho_\pi(s_t)$ and $\rho_\pi(s_t,a_t)$ are the state and state-action marginal distributions of the trajectory caused by the policy $\pi(a_t \mid s_t)$.

B. Maximum Entropy Reinforcement Learning

The goal of maximum entropy reinforcement learning is to maximize the accumulated rewards and entropy of the policy:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))], \tag{1}$$

Where $\mathcal{H}(\pi(\cdot \mid s_t)$ represents information entropy. Maximizing information entropy is to increase the exploration ability of the algorithm, so that the learned policies are not special but more stochastic policies. The temperature parameter α indicates the degree of randomness of the learned policies, that is, the degree of the generalization and the maximum entropy reinforcement learning degenerates into traditional reinforcement learning when α=0 . In general, the entropy of the resulting policies is high, which allows the agent to explore more widely.

C. Soft Policy Iteration

We first introduce soft policy iteration because it is the theoretical basis of soft actor-critic algorithm, and soft policy iteration is a method of learning policies by alternating between policy evaluation and policy improvement under the framework of maximum entropy reinforcement learning.

In the policy evaluation, the soft-Q value of the fixed policy can be calculated iteratively, the formula is expressed as ,

$$\mathcal{T}^\pi Q(s_t,a_t) \overset{\Delta}{=} r(s_t,a_t) + \gamma \mathbb{E}_{S_{t+1}\sim\rho}[V(s_{t+1})], \tag{2}$$

Where $\mathcal{T}^\pi$ is a modified Bellman backup operator and $V(s_t)$ is the soft state value function, that is,

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t,a_t) - \alpha \log \pi(a_t \mid s_t)]. \tag{3}$$

Consider the soft Bellman backup operator $\mathcal{T}^\pi$ and we can define the soft Q-function as,

$$Q^{k+1} = \mathcal{T}^\pi Q^k. \tag{4}$$

We can get the soft Q-function of policy π by repeatedly applying the formula (2) and the formula (4) defined above.

In the policy improvement, we update the policy of each state in Kullback-Leibler divergence, that is,

$$\pi_{new} = \arg \min_{\pi' \in \Pi} D_{KL}\left(\pi'(\cdot \mid s_t) \middle\| \frac{\exp(\frac{1}{\alpha} Q^{\pi_{old}}(s_t, \cdot))}{Z^{\pi_{old}}(s_t)} \right), \quad (5)$$

Where symbol Π represents the set of policies and the formula (5) shows that the soft Q-function value of the new policy is higher than the value of the old policy.

D. *Soft Actor-Critic*

The soft policy iteration method mentioned above may allow us to find the optimal maximum entropy policy, but we can only execute it in a precise form in the case of a table. We need to use function approximators to approximate both the policy and the soft Q-function in the continuous domains. So we consider to choose two neural networks $Q_\theta(s_t, a_t)$ and $\pi_\phi(a_t \mid s_t)$ parameterized by θ and ϕ to approximate both the soft Q-function and the policy. We can train the parameters of the soft Q-function by minimizing the soft Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}\sim\rho}\left[V_{\bar{\theta}}(s_{t+1})\right]\right)\right)^2\right], \quad (6)$$

Where parameter $\bar{\theta}$ is the parameter of target soft Q network. And we can derive that the policy parameters can be optimized by minimizing the expected KL-divergence according to equation (5), that is,

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_\phi}\left[\alpha \log(\pi_\phi(a_t \mid s_t)) - Q_\theta(s_t, a_t)\right]\right]. \quad (7)$$

Then we use the neural network with parameter ϕ to reparameterize the policy, i.e.,

$$a_t = f_\phi(\varepsilon_t; s_t), \quad (8)$$

Where ε_t is the input noise vector sampled from the fixed distribution. Finally, we can rewrite equation (7) as,

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D, \varepsilon_t\sim\mathcal{N}}\left[\alpha \log \pi_\phi(f_\phi(\varepsilon_t; s_t) \mid s_t) - Q_\theta(s_t, f_\phi(\varepsilon_t; s_t))\right]. \quad (9)$$

E. *Random Network Distillation*

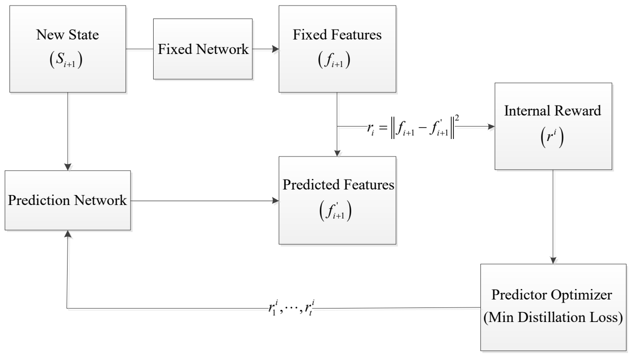
New State (S_{t+1})
Fixed Network
Fixed Features (f_{t+1})
$r_i = \|f_{t+1} - f'_{t+1}\|^2$
Internal Reward (r^i)
Prediction Network
Predicted Features (f'_{t+1})
Predictor Optimizer (Min Distillation Loss)
$r_1^i, \cdots, r_t^i$

Fig. 1 The principle of RND.

The idea of RND is simplified based on the intrinsic curiosity module (ICM) architecture [15]. It mainly uses two neural networks. After each network is randomly initialized, one of the networks acts as the target network and remains fixed, the other network reduces the prediction difference with the target network through continuous learning. In general, the predicted difference in the visited state will be smaller, and the predicted difference in the unvisited state will be larger. Using the prediction difference as an intrinsic reward can increase the exploration ability of the algorithm. The principle can be represented by Fig.1.

III. Training Policies Via Soft Actor-Critic And Random Network Distillation

In this section, we briefly introduce the FSPCC continuum manipulator model, and illustrate the details of the continuum manipulator controlled by the SAC algorithm and RND.

We built a two-segment space continuum manipulator model in the V-REP simulation environment as shown in Fig.2.

The model described in this section contains two segments, dark green and pink, which are connected in turn and each segment contains 6 subsegments. Each subsegment consists of a cross shaft and a rigid body. Each cross shaft has two degrees of freedom (DOF) because it contains two mutually perpendicular revolute joints as shown in Fig.3. So this space continuum manipulator has a total of 24 DOF.

Furthermore, it is worth mentioning that the usual continuum manipulator is cable-driven with the characteristic of piecewise constant curvature. So it is difficult to build its model in the simulation environment. In order to make the model as close as possible to the real situation, we have made the following improvements to the simulation model: 1) We directly control the joint torque to achieve the same effect as controlling the tension of the driving cables to avoid simulating the driving cables. 2) We define the joints of the first cross shaft of each segment as the controlled joint and the rest are the uncontrolled joints so that we need to control four joints in total. At the same time, we use PD control to keep the angles of controlled joint and the uncontrolled joint remain the same, which ensures that the simulation model has the characteristics of PCC.

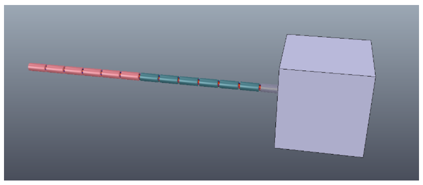

Fig. 2 Two-segment space continuum manipulator.

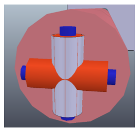

Fig. 3 The cross shaft.

The difference between our model and the model proposed by Huang et al. [7] lies in the base and control method, the latter is controlled by the joint speed and its base is fixed, while the former is controlled by the joint torque and its base is free floating. In addition, the space continuum manipulator is easy to break when selecting the torque control in the simulation environment. However, when we choose the speed control method, the algorithm can't converge very well, and when the maximum torque is set to be small, the space continuum manipulator will not break, so we choose the torque control method. In this paper, the maximum torque is set to $0.2N \cdot m$.

We summarize the process of training space continuum manipulator with SAC and RND as Algorithm.

```
Algorithm Training process for space continuum manipulator
  Initialization
    Initialize the parameters $\theta_1$, $\theta_2$, $\phi_1$ and $\phi_2$;
    Initialize the parameters of target Q-network $\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$;
    Build an empty experience replay buffer D;
  for each time step do
      Sample an action using $a_t = f_\phi(\varepsilon_t, s_t)$, where $\varepsilon_t \sim \mathcal{N}(0,1)$;
      Sample the next $s_{t+1}$ from the environment;
      Calculate intrinsic reward $r_t = \left\| f_{t+1} - f'_{t+1} \right\|^2$;
      Add the new samples $\{(s_t, a_t, r(s_t, a_t), s_{t+1}, r_t)\}$ to D;
    Randomly extract N samples from D as a mini-batch;
    Calculate forward loss as in [14];
    Update the prediction network parameter $\phi_2$ w.r.t. forward loss;
    Update the Q-function parameters $\theta_1$ and $\theta_2$ as in [12];
    Update the policy network parameters $\phi_1$ as in [12];
    Update the target Q-function parameters $\bar{\theta}_1$ and $\bar{\theta}_2$ as in [12];
  end for
```

IV. Experiments

In this section, we demonstrate that the proposed algorithm is feasible in controlling FSPCC continuum manipulator. In addition, we prove that RND is only suitable for the sparse reward problem and can greatly improve the exploration ability of the algorithm.

A. *Experiment setting*

The previous section already said that we only need to control the four joints in model, which means that the state and action space of the system only contain the values of the four joints. We assume that the agent can fully observe the state of the system, that is, the observations $o_t = s_t$ at any time. In this section, we divide the state of the system into four parts: the manipulator, the base, the end-effector and the target satellite, expressed as $s = \left[s_{manipulator}, s_{base}, s_{end}, s_{target}\right]$. The manipulator part can be defined as $s_{manipulator} = \left[\theta_1, \theta_2, \theta_3, \theta_4, \omega_1, \omega_2, \omega_3, \omega_4\right]$, where $\theta_1, \theta_2, \theta_3, \theta_4$ and $\omega_1, \omega_2, \omega_3, \omega_4$ represent the angle and angular velocities of the four controlled joint, respectively. The free-floating base part can be defined as a vector $s_{base} = \left[\alpha, \beta, \gamma, p_x, p_y, p_z, v_x, v_y, v_z, \dot{\alpha}, \dot{\beta}, \dot{\gamma}\right]$, where $\alpha, \beta,$ and γ represent the three Euler angles of the free-floating base, p_x, p_y, p_z and v_x, v_y, v_z represent the position coordinates and velocity of the base centroid on the x, y and z axes, respectively, and Euler angular velocities are express as $\dot{\alpha}, \dot{\beta}, \dot{\gamma}$. The end-effector part consists of a 3-dimensional end-effector position vector and a 1-dimensional end-effector velocity vector. The reason for the end part is that there is an error between the end part value calculated from the simulation model and the actual end part value. In addition, we also need to know the 3-dimensional position vector of the target satellite. So the state of the entire system is a 27-dimensional vector. The action taken by the space robot is the torque of the four controlled joints.

The experiment setting is as follows: The base is free-floating with the initial position of [0.0475, 0, 3]m and the length of each subsegment is 0.09m. We define each episode to be $T = 100$ time steps with a time interval of $dt = 50ms$ per step. The simulation environment will be reset when each episode ends. The initial angles of the controlled joints are set to [0, 0 ,0, 0] rad. The initial positions of the end-effector and target satellite are [1.5, 0, 3]m and [1.26, -0.29, 2.99]m, respectively. At the initial time, the velocity and angular velocity are both set to zero. Unlike [5], we don't need to set constraint in the model, we only need to set the maximum joint torque to $0.2N \cdot m$ in the code, because the experimental results show that the excessive joint torque will affect the stability of the model. When the distance between the end-effector and the target satellite is less than 0.05m, we believe that the target satellite is successfully captured and the episode ends.

Since the general reinforcement learning problem of the robot manipulator captures the target star is a dense reward problem, that is, we generally set the reward function as $-\left(\omega_1 d_t + \omega_{\log} \log(d_t + \delta) + \omega_2 \left\|a_t\right\|_2^2\right)$, where these hyper-parameters are set to ω_1=5, $\omega_{\log} = 5$, and ω_2=0.05. d_t is the distance from the end-effector to the target satellite, and a_t is the action of the robot manipulator at the current time. However, unlike the rigid-body robot, when we set the reward function to be the same as above, we find that the FSPCC continuum manipulator circles around the target satellite as long as possible to get more rewards than reaching the target satellite straightforward because of more degrees of freedom the continuum manipulator has. Then we set reward to $-\left(\omega_1 d_t + \omega_2 \left\|a_t\right\|_2^2\right)$, the simulation process shows that the FSPCC continuum manipulator prefers to circle around the target but does not reach the target. so it is not good to learn the policies to reach the target satellite in the dense reward setting. Instead, we set the problem of catching the target with the continuum manipulator as a sparse reward problem. When the distance $d_t < 0.05m$, give an immediate reward of 500, otherwise the reward is 0. Numerically, the reward function can be expressed as.

$$r_t = \begin{cases} 500 & d_t \leq 0.05m \\ 0 & \text{else} \end{cases} \tag{10}$$

We first simulated 5500 epochs for the SAC algorithm and RND methods to control the FSPCC continuum manipulator capture target satellite in the case of sparse reward. The training process is shown in Fig.4. It is interesting that the sum of the rewards of each episode in the early stage of training varies greatly because the maximum entropy policies are very good at exploring while the RND method promotes the exploration ability of the algorithm. We can find that the training curve has an overall upward trend despite the existence of some jitter. When the algorithm trains more than 300 episodes, the FSPCC continuum manipulator can capture the target satellite. When the algorithm trains more than 5000 episodes, the FSPCC continuum manipulator can stably capture the target satellite and at the same time the algorithm converges. It shows that the SAC and RND methods can well accomplish the task of capturing the target satellite by the FSPCC continuum manipulator in the case of sparse reward. In addition, we found that nearly 7000 episodes were run in this experiment because an epoch can have multiple episodes when the continuum manipulator reach the target quickly.

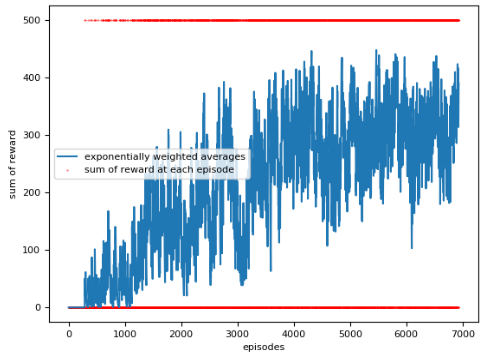

Fig. 4 Training process of experiment 1. (The blue line represents the exponentially weighted averages with a smoothing factor 0.9)

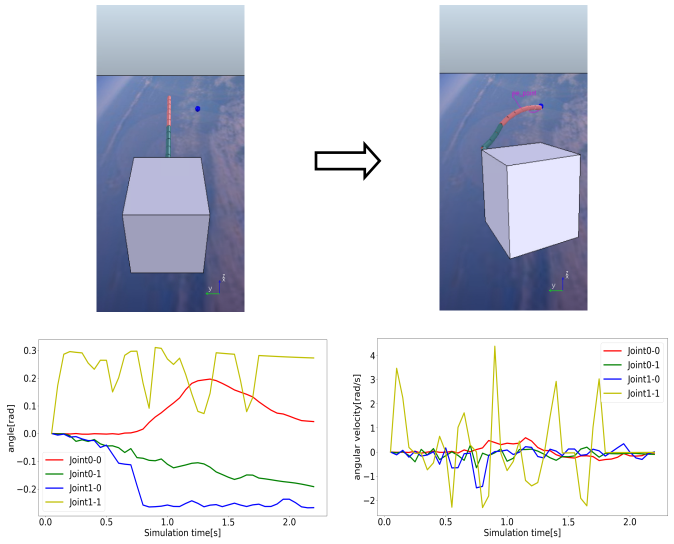

Fig. 5 Performance of trained policies in experiment 1. (On the top: the process of the space continuum manipulator capturing the target. At the bottom: the variation of the four controlled joint angles and joint velocities during the successful capture of the target.)

B. *Ablation study on RND*

In order to prove that the RND method can increase the exploration ability of SAC in the case of sparse reward, we have done an ablation study on RND. We removed the RND method based on experiment 1, that is, in the case of sparse reward, we only use the SAC algorithm to control the space continuum manipulator to capture the target (the experimental model, hyperparameters and experimental settings are the same as experiment 1). The training process is shown in Fig.6. The result of experiment 2 (Fig.6) shows that the algorithm does not converge at the end, and the space continuum manipulator doesn't complete the task. Compared with Fig.4, we can conclude that the RND method can promote the exploration ability of SAC in the case of sparse reward.

C. *Dense reward*

In order to find out whether the RND method is only suitable for sparse reward and whether the effect of using the RND method is the best in the presence of sparse reward. We did two experiments (experiment 3 and experiment 4) with and without the RND method in the case of continuum reward (both of which use the SAC algorithm) and the experimental results are shown in Fig.7. The dense reward function is set to the form described in the above experiment details and all experimental parameters and settings are the same as the previous experiment 1 and 2. Comparing the training curves of experiments 3 and 4, we concluded that the effect of the RND method used in dense reward is not so significant. It is worth mentioning that when we compare the effect of experiment 1 with experiments 3 and 4, we found that the curve of experiment 1 showed a consistent ascending trend and the curves of experiments 3 and 4 descended in the latter stage of training. We set the same number of epochs in these experiments and found that the agent in experiment 1 has reached nearly 7000 episodes and both agents in experiments 3 and 4 have reached less than 6000 episodes, which indicates that the number of times the target has been captured in experiment 1 is much more than that of experiments 3 and 4. Thus the policies trained in experiment 1 are better than those trained in experiments 3 and 4. So it is better to use SAC algorithm combined with RND method to control the FSPCC continuum manipulator to capture the target in sparse reward settings.

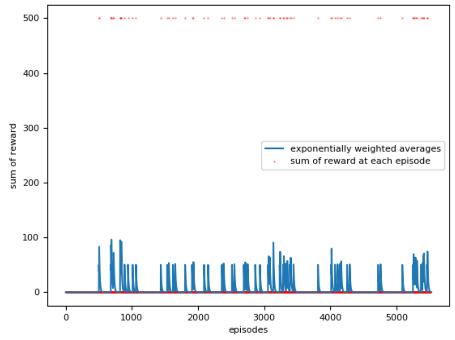

Fig. 6 Training process of experiment 2.

Conclusion And Future Works

Due to the excessive degree of freedom of the FSPCC continuum manipulator, it is rather intractable for the conventional model-free RL algorithm to control it to complete the space tasks under the dense reward setting. In this paper, we discuss how the SAC algorithm and the RND method control the FSPCC continuum manipulator to capture the target in the case of sparse reward and whether the RND is only suitable for sparse reward. The simulation results show that our method provides an effective solution for FSPCC continuum manipulator to capture the target in the presence of sparse reward, and RND is only suitable for sparse reward. In addition, we set the problem of catching the target with the continuum manipulator as a sparse reward problem, which can effectively solve the problem of the FSPCC continuum manipulator circling around the target as long as possible without reaching the target quickly in the dense reward setting.

A challenging direction for our future work is to avoid the obstacles by controlling the FSPCC continuum manipulator of the method described in this paper. Moreover, we may try to use this method for the dynamic target capture of space robots due to the strong exploration ability of the method.

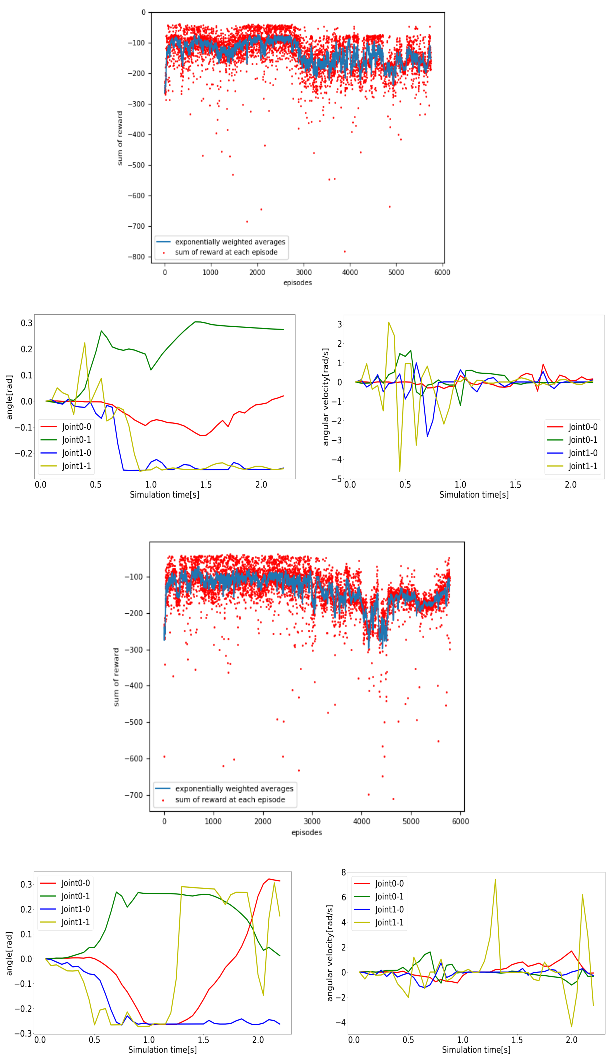

Fig. 7. The effects of experiment 3 and 4. (In the first line: the training curve of experiment 3 (with RND method). In the second line: the variation of the four controlled joint angles and joint velocities during the successful capture of the target in experiment 3. In the third line: the training curve of experiment 4 (without RND method). In the fourth line: the variation of the four controlled joint angles and joint velocities during the successful capture of the target in experiment 4.)

References

[1] C. Li, B. Liang, and W. Xu, "Autonomous Trajectory Planning of Free-floating Robot for Capturing Space Target," *in IEEE international Conference on Intelligent Robots and Systems*, Beijing, China, pp.1008-1013, October 2006.

[2] H. Shui, J. Wang, and H. Ma, "Optimal Motion Planning for Free-Floating Space Robots Based on Null Space Approach," *in IEEE International Conference on Measuring Technology and Mechatronics Automation*, Hunan, China, pp.845-848, April 2009.

[3] 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 International Conference on Robotics and Biomimetics*, Kuala Lumpur, Malaysia, December 2018.

[4] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement Learning with Deep Energy-Based Policies," *arXiv preprint arXiv:1702.08165* (2017).

[5] T. Mahl, A. Mayer, A. Hildebrandt, and O. Sawodny, "A variable curvature modeling approach for kinematic control of continuum manipulators," *in American Control Conference*, Washington, U.S.A, pp. 4945–4950, June 2013.

[6] A. Chawla, C. Frazelle, and I. Walker, "A Comparison of Constant Curvature Forward Kinematics for Multisection Continuum Manipulators," *in Second IEEE International Conference on Robotic Computing*, 2018.

[7] S. Huang, Q. Zhang, Z. Liu, X. Wang, and B. Liang, "Control of a piecewise constant curvature continuum manipulator via policy search method," *in International Conference on Robotics and Biomimetics*, Kuala Lumpur, Malaysia, December 2018.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334-1373, 2015

[9] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *in International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347* (2017).

[11] V. Mnih, A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *in International Conference on Machine Learning (ICML),* NY, USA, June 19-24, 2016.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *arXiv preprint arXiv:1801.01290v1* (2018).

[13] T. Kulkarni, K. Narasimhan, A. Saeedi and J. Tenenbaum, "Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation," *arXiv preprint arXiv:1604.06057 (2016)*.

[14] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation," *arXiv preprint arXiv:1810.12894v1* (2018).

[15] D. Pathak, P. Agrawal, A. Efros, and T. Darrell, "Curiosity-driven exploration by selfsupervised prediction," *in International Conference on Machine Learning (ICML)*, Sydney, Australia, August 2017.