A Postprint Study on Goal-based Domain Randomization Methods for Robot Manipulation

Authors: Zhang Xiayu, Chen Xiaoping

Date: 2022-05-18T16:08:25Z

Abstract

Employing reinforcement learning to address robot manipulation problems offers numerous advantages; however, traditional reinforcement learning algorithms encounter challenges associated with sparse rewards, and the policies obtained are difficult to directly deploy in real-world environments. To enhance the success rate of policy transfer from simulation to reality, we propose a goal-oriented domain randomization methodology: utilizing goal-based reinforcement learning algorithms for model training, which can effectively mitigate the sparse reward problem in robot manipulation tasks, yielding policies that perform well in simulated environments. Concurrently, the algorithm incorporates a goal-driven domain randomization approach, which proves effective in improving policy generalization and bridging the gap between simulation and real-world environments, thereby facilitating the transfer of policies from simulation to reality for successful execution. Experimental results demonstrate that reinforcement learning algorithms employing the goal-oriented domain randomization method contribute to improving the success rate of policy transfer from simulation to reality.

Full Text

Research on Goal-Based Domain Randomization Method in Robot Manipulation

Zhang Xiayu†, Chen Xiaoping
(University of Science & Technology of China, Hefei 230026, China)

Abstract: Reinforcement learning offers numerous advantages for solving robot manipulation problems. However, traditional reinforcement learning algorithms face the challenge of sparse rewards, and the learned policies are difficult to apply directly to real-world environments. To improve the success rate of policy transfer from simulation to reality, this paper proposes a goal-based do-

main randomization method. The approach employs goal-based reinforcement learning algorithms for model training, which effectively addresses the sparse reward problem in robot manipulation tasks. The resulting policies perform well in simulation environments. Simultaneously, the algorithm incorporates goal-conditioned domain randomization, which demonstrates excellent effectiveness in improving policy generalization and bridging the gap between simulation and reality. Policies trained in simulation can be easily transferred to real-world environments and executed successfully. Results show that the reinforcement learning algorithm using the goal-based domain randomization method helps improve the success rate of policy migration from simulation to reality.

Key words: reinforcement learning; domain randomization; robot manipulation; sim-to-real transfer

0 Introduction

With the development of artificial intelligence technology and the proliferation of automation equipment, robot manipulation plays an increasingly important role in real-world applications. Unlike traditional task planning methods, reinforcement learning enables agents to discover optimal policies autonomously through interaction with the environment and feedback from reward functions, without requiring designers to concern themselves with specific implementation details. This provides a natural advantage for solving robot manipulation problems. For instance, the OpenAI team has achieved complex manipulation tasks with dexterous robotic hands, while domestic research teams have implemented object grasping on Kinova robotic arms using reinforcement learning.

Robot manipulation tasks involve enormous state spaces and sparse rewards, while complex manipulation tasks are difficult to equip with manually defined reward functions. Consequently, reinforcement learning-based algorithms still face significant challenges in practical applications. Moreover, applying reinforcement learning in real-world manipulation tasks presents numerous difficulties. Direct sampling and training in real environments is often impractical due to low sampling efficiency and safety risks to personnel and equipment during training and testing. While transferring policies learned in simulation to real-world scenarios appears feasible, modeling errors between simulators and physical environments, along with discrepancies between simulated and actual data acquisition, render such policies unusable. Therefore, addressing the mismatch between simulation and real-world environments and overcoming the "Reality Gap" represents a key direction for solving robot manipulation tasks through reinforcement learning.

Various methods exist to address this reality gap. The problem of transferring control policies from simulation to reality can be viewed as an instance of domain adaptation, where a model trained in a source domain is transferred to a new target domain. These methods rely on a key assumption: different do-

mains share common features, enabling representations and behaviors learned in one domain to be leveraged in another. Domain adaptation methods learn a mapping from simulation and real-world states to a shared latent space. Training occurs in simulation using the mapped state space, and when transferring to reality, states are similarly mapped to the latent space before applying the simulation-trained model. Domain randomization, conversely, randomizes information or parameters in the simulation environment. From a theoretical perspective, Chen Xiaoyu et al. provide a theoretical explanatory model for the classic sim-to-real transfer problem, particularly explaining why and when domain randomization algorithms are effective. Through POMDP model analysis, they demonstrate that domain randomization effectively solves sim-to-real transfer problems in robot manipulation and offers excellent performance guarantees, with well-designed methods theoretically capable of training without any real-world data. Peng et al. randomize physical parameters, optimizing the expected cumulative return across numerous virtual environments with different physical parameters to produce more robust policies. Chebotar et al. build upon this by using existing policies to generate trajectories in both virtual and real environments from the same initial state, then adjusting randomized physical parameters by comparing trajectory differences. Tobin et al. apply domain randomization to visual representations of environments. Niu et al. use domain randomization to improve the robustness of autonomous driving training in simulation.

These methods primarily focus on objective environmental differences (typically physical parameter variations) while paying less attention to differences in the driven entities within the environment. In fact, robot manipulation itself is an underactuated system, where actual control outcomes often deviate from model expectations. This discrepancy primarily stems from actuator driving methods and sensor feedback biases rather than environmental parameter errors. While higher-precision actuators can reduce this gap, they cannot eliminate it entirely, and randomization of environmental parameters often proves ineffective in this regard.

Beyond domain-related methods, Andrei et al. extend a special type of progressive neural network to reinforcement learning for model training, while Christiano et al. employ inverse dynamics models. However, these methods are relatively model-dependent and exhibit limited generalization across different manipulation tasks.

In summary, existing methods each have strengths in overcoming simulation-reality discrepancies, yet none can guarantee transfer success rates while simultaneously maintaining training speed and algorithm generalization across tasks. Therefore, this paper proposes a goal-based domain randomization method that addresses sparse rewards in reinforcement learning through experience replay while improving policy adaptability to differences between real and simulated environments via domain randomization. This approach not only outperforms other domain randomization algorithms in training efficiency but also ensures

high success rates when executing tasks in real-world environments.

1.1 Reinforcement Learning Formulation for Robot Manipulation

Based on the above, this paper first formulates robot manipulation as a reinforcement learning problem. In robot manipulation tasks, a standard reinforcement learning model can be described as a process where an agent maximizes returns through interaction with the environment. For convenience in subsequent descriptions, this paper assumes fully observable environments. A deterministic policy maps from state space S to action space A, with each policy query sampling actions from a specific distribution. The reward function $r : S \times A \times R \to S$ represents the value of executing a specific action in a given state. The state transition probability $p(\cdot|s_t, a_t)$ denotes the probability distribution of transitioning to state s_{t+1} after executing action a_t in state s_t. At each time step t, the agent generates an action from the policy based on the current state:

$$a_t \sim \pi(\cdot|s_t)$$

and receives a new state from the state distribution $s_{t+1} \sim p(\cdot|s_t, a_t)$. The agent' s objective is to maximize its expected return $G_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$, where $\gamma \in [0, 1]$ is the discount factor. The action-value function is defined as $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_t, a_t}[R_t|s_t, a_t]$. A policy π^* is considered optimal if for any π, we have $Q^{\pi^*}(s, a) \geq Q^{\pi}(s, a)$ for all s, a. All optimal policies share the same Q function, called the optimal Q function and denoted as Q^*. The optimal function satisfies the following Bellman equation:

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a')|s, a]$$

1.2 Hindsight Experience Replay

Due to the large task space and sparse rewards in robot manipulation, conventional reinforcement learning algorithms struggle to converge. Learning successful policies from sparse binary rewards poses a significant challenge for most reinforcement learning algorithms. Consider the simple bit-flipping task as an example: the state is a binary sequence $\{0, 1\}^n$, and actions flip any one of the n positions. If the reward function is set to 0 when the sequence is correct and -1 otherwise, traditional reinforcement learning algorithms fail to train when the sequence length exceeds 20 using binary rewards.

Another example is the common pushing task in robot manipulation. In typical simulation environments, the state space S is defined as all reachable positions of the agent, while the action space is a 2D tuple (x, y) representing the distance

traveled in the x and y directions, usually with a step size of 0.01 seconds and a maximum of 200 steps. A simple and easily constructed reward function is the binary reward $r(s, g)$, which only indicates whether the given state satisfies the goal. For each episode, both initial and goal states are sampled uniformly. The policy receives a -1 reward until reaching the goal state:

$$r(s,g) = \begin{cases} 0, & \text{if goal} \\ -1, & \text{otherwise} \end{cases}$$

In the pushing task, using this reward function means rewards are only obtained when the box is pushed to the target position, resulting in excessively sparse rewards that prevent most algorithms from converging when the task space is large.

For tasks that can be fully modeled, reinforcement learning algorithms can guide agents toward the overall objective through carefully designed reward functions. For instance, in the aforementioned bit-flipping task, if the reward function is designed as $r(s_t, a_t, s_{t+1}) = -\|s_{t+1} - g\|_2^2$, general reinforcement learning algorithms can demonstrate excellent performance. However, for complex problems, designing reward functions is often extremely difficult and may lead to suboptimal behaviors. Real-world robot manipulation environments are complex, making it impractical to construct specialized reward functions for most tasks. Therefore, this paper utilizes Hindsight Experience Replay (HER) to address this issue, enabling policy training with sparse and non-specialized binary reward functions.

The HER algorithm is based on a simple idea: training in sparse reward spaces generates numerous failure trajectories. If these failed trajectories can be leveraged to improve learning efficiency, it becomes possible to train viable policies using simple, non-specialized reward functions. In a failed trajectory, the true goal G remains unachieved throughout. Since the trajectory fails to reach the goal, the agent cannot update its policy from such reward information, as each time step only provides a -1 reward that is difficult for general reinforcement learning algorithms to utilize.

During replay, for trajectories that fail to reach the goal, the algorithm extracts achieved states from these trajectories as virtual goals. Using these virtual goals, it computes rewards and trains with reinforcement learning methods. Although the trajectory is unsuccessful for the original goal, it becomes successful under the new virtual goal. Consequently, rewards computed based on virtual goals are not limited to -1. By replaying past experiences, the agent can train with more successful examples than originally recorded.

HER is a goal-based reinforcement learning algorithm that augments the input state with a goal g, representing the state as (s, g). The strategy for selecting new goals involves randomly choosing k states from the same trajectory after

state s_t as the goal g'. After setting the goal, the algorithm generates new rewards using the corresponding trajectory.

HER employs the Deep Deterministic Policy Gradient (DDPG) algorithm as the off-policy reinforcement learning method. DDPG is designed for continuous control problems and maintains two neural networks: a Critic network for evaluating Q-values and an Actor network for generating the target policy.

Algorithm 1: Hindsight Experience Replay (HER)
Input: Off-policy reinforcement learning algorithm $\mathcal{A}$; goal selection strategy π_g; reward function r
Output: Trained policy network π

1. Initialize $\mathcal{A}$, replay buffer $\mathcal{R}$
2. for episode in range do
3. Obtain initial state s_0 and goal g
4. for t in range $(0, T-1)$ do
5. `Compute next action $a_t$ based on current state $s_t$ and goal $g$`
6. `Execute action $a_t$ to obtain new state $s_{t+1}$`
7. end for
8. for t in range $(0, T-1)$ do
9. `Store $(s_t, a_t, r_t, s_{t+1}, g)$ in buffer $\mathcal{R}$ for experience replay`
10. end for
11. Optimize policy using $\mathcal{A}$ and $\mathcal{R}$
12. end for

---

2 Goal-Based Domain Randomization Algorithm

However, policies obtained through experience replay alone exhibit poor generalization and are difficult to apply in real-world environments. Training fails to converge for complex tasks with varying objectives. Furthermore, current technology cannot create simulators that perfectly model real environments. Differences between simulation and reality lead to low success rates when transferring policies to physical execution. Using the pushing task as an example, simulation environments can obtain accurate coordinates of objects and robot end-effectors, but in real-world experiments, such coordinates can only be acquired through positioning systems like MCS, whose data is susceptible to environmental noise and sampling errors. Moreover, physical properties of real environments are difficult to replicate completely in simulation. Executing identical actions from the

same state may yield different outcomes than simulation predictions, with variations occurring across repeated executions. Specifically, sampling frequency, latency, jitter during policy execution, and subtle unmodeled environmental differences all introduce interference.

This paper addresses these issues using domain randomization: a complementary adaptation technique that models differences between source and target domains as variability within the source domain. Considering that discrepancies between real and simulated environments result from multiple factors that ultimately affect task states during execution, this work randomizes goals during training to simulate errors between actual states and targets caused by the reality gap, thereby producing more generalizable and robust policies.

To enhance policy generalization and improve success rates in real-world execution, this paper proposes a goal-based domain randomization algorithm. The objective is to train a policy that performs effectively in both real and simulated environments. Given the difficulty of sampling in real environments, training occurs in simulation where success rates and training speed are tested before transferring the learned policy to real-world environments for further evaluation. The policy is represented by trained neural networks, with real-world execution involving feeding sampled environmental states into the network to obtain action strategies at each step.

2.1 Domain Randomization Method

In sim-to-real transfer problems, the physical characteristics of target real environments are difficult to simulate completely with current technology. This has led to physics-based randomization methods that randomize dynamics-related parameters including robot link masses, joint damping, object masses, friction coefficients between objects and tables, table height, position sensor data, and time steps between actions. Randomizing these parameters aims to reduce dynamics-related differences between simulation and reality, thereby improving task success rates.

The proposed method recognizes that simulation environments can never be perfectly identical to real ones. Goal-conditioned algorithms involve goals similar to those in human operation. Randomizing these goals within the operational space according to specific distributions enables trained policies to better accommodate discrepancies between achieved results and intended targets during task execution.

Vision-based randomization methods randomize visual information including clutter shapes and quantities, object textures and positions, background textures of tables and floors, simulated camera positions and orientations, lighting sources, and surface reflectance properties. Such randomization acknowledges that agents have limited observation capabilities and cannot access material properties that significantly impact training effectiveness. These methods share a similar fundamental approach with dynamics randomization—expand-

ing datasets to narrow the simulation-reality gap, which currently cannot be eliminated entirely.

This paper randomizes goals in goal-conditioned reinforcement learning, which represent unique states distinct from previously randomized environmental parameters. Goal randomization fully leverages the characteristics of goal-driven algorithms with practical significance: robot manipulation is inherently an underactuated system where actual control outcomes often deviate from model expectations. This discrepancy arises not only from physical parameter differences but also from actuator motion strategies. While higher-precision actuators can reduce this gap, they cannot eliminate it completely. Randomizing goals enables learned policies to adapt to underactuation-induced errors, thereby improving manipulation success rates. Additionally, since simulation errors in physical parameters typically compound with system underactuation to produce final goal errors, goal randomization addresses this combined effect.

2.2 Goal Randomization Method

HER is naturally a goal-conditioned algorithm, making it possible to improve policy generalization by processing training goals. During training, the goal g in each episode is randomized with parameters drawn from a specific distribution to simulate real-world conditions. The operation involves generating a new goal for each sampled state transition: given that this paper focuses on robot manipulation tasks where goals are object coordinates (e.g., the endpoint coordinate where the object should be manipulated, or the midpoint coordinate of a door handle axis for door-opening tasks), the new goal selection employs future state prediction (using a state several steps ahead in the trajectory as the new goal for that transition).

The purpose of introducing goal randomization parameters is to process the generated new goals in a specific manner (since manipulation task goals are object state coordinates, randomization involves multiplying coordinate values in the goal by a random coefficient). The randomized goals should follow a specific distribution relative to the original goals within the task space.

Finally, the randomized goals are combined with state transitions and stored in the buffer for training. The experience replay component uses DDPG, an off-policy algorithm for continuous control. The following describes the sampling and training process in simulation with goal domain randomization, as shown in Algorithm 2.

Algorithm 2: Goal-Based Domain Randomization Algorithm
Input: Off-policy reinforcement learning algorithm $\mathcal{A}$; goal randomization parameter strategy π_g; reward function r
Output: Trained policy network π

1. Initialize $\mathcal{A}$, replay buffer $\mathcal{R}$
2. for episode in range do

3. Obtain initial state s_0 and final goal g
4. for t in range $(0, T-1)$ do
5. `Compute next action $a_t$ based on current state $s_t$ and goal $g$`
6. `Execute action $a_t$ to obtain new state $s_{t+1}$`
7. `$r_t = r(s_t, a_t, s_{t+1}, g)$`
8. end for
9. for t in range $(0, T-1)$ do
10. Obtain goal g' via HER
11. Generate randomization coefficient
12. for k in range do
13. `$g'' = g' \times \text{randomization coefficient}$`
14. end for
15. Store $(s_t, a_t, r_t, s_{t+1}, g'')$ in buffer $\mathcal{R}$
16. end for
17. Optimize policy using $\mathcal{A}$ and $\mathcal{R}$
18. end for

---

3 Experiments and Performance Evaluation

This section evaluates the proposed algorithm through comparative experiments in both simulation and real-world environments, assessing success rates in simulation tasks, training convergence speed, algorithm generalization under varying task parameters, and success rates after real-world transfer.

3.1 Environment Configuration

The simulation environment is built on Ubuntu using PyTorch for network construction and Mujoco as the physics engine. To ensure that trained policies can be tested in real-world environments, the simulation models the physical setup including the UR5e robotic arm, electric gripper, experimental platform, and objects. The simulation scenarios are visualized in Figure 1.

For goal randomization during sampling, this paper employs normally distributed random numbers. Specifically, goals sampled during off-policy training are randomized following an $N(0, 0.1)$ distribution.

3.2 Simulation Results Analysis

To evaluate the proposed algorithm' s performance, comparisons are made with three reinforcement learning algorithms: standard reinforcement learning (RL), Hindsight Experience Replay (HER), and Dynamics Randomization. Standard RL algorithms perform poorly in large search spaces, with near-zero success rates for complex tasks without carefully designed initial states. While HER addresses the search space issue, it cannot adapt well when initial states vary randomly. Both dynamics randomization and the proposed goal-based randomization improve upon HER, with the goal-based method demonstrating stronger adaptability to complex tasks through its focus on goal analysis.

To further compare generalization performance across tasks, the initial state randomness coefficient is increased to 2.5. Table 2 shows the resulting success rates for different algorithms.

The algorithm is evaluated through three distinct task scenarios:

1. **Pushing:** The scene contains a table, an object on the table surface, and a robotic arm. The goal is to push the object to a designated position on the table. Since grasping is not involved, the gripper remains closed throughout the task.
2. **Pick-and-Place:** Similar to pushing, but the goal is to grasp the object and move it to a specified position in the air. This task involves gripper control for grasping. To ensure training success, the initial state is set with the gripper already holding the object.
3. **Door-Opening:** This more complex task involves a fixed door frame and a robotic arm. The objective is to grasp the door handle, pull it down to unlatch the door, and then pull the door open to a certain angle. Given the task complexity, the initial state is set with the gripper already holding the door handle.

Sampled states include the robotic arm end-effector position in the Mujoco environment and coordinates of all objects. Algorithm 2' s key operation is the goal randomization during iteration. In each iteration, the goal g' is first obtained via HER, then randomized into set g'' according to the task space (distribution range of actuators and objects) and randomization coefficients (e.g., Gaussian distribution functions). The resulting set g'' follows the randomization coefficient' s probability distribution relative to the initial goal g within the task space, while being constrained by the task space to prevent extreme values from affecting training. This randomization of g' partially reflects underactuated system errors, improving success rates after real-world transfer. Finally, the randomized elements from set g'' are combined with sampled trajectories (s_t, a_t) and stored in buffer $\mathcal{R}$ for policy optimization.

3.3 Real-World Environment Validation

To validate the reliability of policies trained in simulation, real-world experiments use identical hardware configuration (UR5e, Universal Robots). To accurately obtain object coordinates, a multi-camera system (Opti-track) is employed, with markers placed at key positions on object surfaces and edges. For the door-opening task, multiple markers are placed on the door frame, panel, and handle to overcome occlusion issues. To unify the MCS and robot coordinate systems, the robotic arm is fixed at the experiment' s start, and markers are placed on critical arm positions and the end-effector to calibrate the arm' s coordinates in the MCS system.

Real-world testing focuses on the door-opening task with randomized initial positions. The gripper' s initial state matches simulation, already holding the door handle. To improve experimental success rates, a flexible grasping mechanism is used, with a 3D-printed structure at the gripper-flange connection that can be replaced in case of serious accidents, preventing damage to critical equipment. Table 3 shows task success rates under different initial state randomness coefficients.

Results demonstrate that when executing simulation-trained policies in real environments, success rates remain high with small initial state variations. However, when initial state variations are large, real-world success rates drop significantly compared to simulation. This occurs because door handles and doors are constrained rigid structures; manipulation errors cause sharp increases in force feedback, triggering the robot' s protection mechanisms and causing task failure. Without such protection, the door and gripper would be easily damaged during operation.

---

4 Conclusion

This paper proposes a goal-based domain randomization algorithm that combines experience replay with goal-based domain randomization. The approach demonstrates good convergence speed, better adaptation to environmental changes, maintains strong performance with large initial state variations, and achieves notable success rates after real-world transfer.

The method fully leverages the characteristics of goal-based algorithms, which not only improve training efficiency in sparse reward environments but also enhance adaptability and robustness across different tasks when combined with domain randomization. This environmental difference adaptability effectively improves policy transfer success rates from simulation to reality.

However, real-world manipulation tasks often involve more complex procedural steps, and task goals may extend beyond specific coordinate points, making them difficult to define. In such cases, goal-based algorithms may struggle. Defining task goals appropriately and selecting suitable randomization methods

for complex scenarios remains a primary challenge. Extending the proposed method to more diverse robot manipulation tasks represents the main direction for future work.

References

[1] Jens, Kober, J, et al. Reinforcement learning in robotics: A survey [J]. *The International Journal of Robotics Research*, 2013.

[2] Kroemer O, Niekum S, Konidaris G D. A review of robot learning for manipulation: Challenges, representations, and algorithms [J]. *Journal of machine learning research*, 2021, 22 (30).

[3] Andrychowicz M, Baker B, Chociej M, et al. Learning dexterous in-hand manipulation [J]. *The International Journal of Robotics Research*, 2020, 39 (1): 3-20.

[4] 张智广. 基于深度强化学习的机械臂抓取方法研究 [D]. 哈尔滨工业大学, 2021. (Zhang Zhiguang. Research on manipulator grasp based on deep reinforcement learning [D]. Harbin Institute of Technology, 2021.)

[5] Gupta A, Devin C, Liu Y X, et al. Learning invariant feature spaces to transfer skills with reinforcement learning [J]. arXiv, 2017.

[6] Chen X, Hu J, Jin C, et al. Understanding Domain Randomization for Sim-to-real Transfer [J]. arXiv, 2021.

[7] Peng X B, Andrychowicz M, Zaremba W, et al. Sim-to-real transfer of robotic control with dynamics randomization [C]// 2018 IEEE international conference on robotics and automation (ICRA). IEEE, 2018: 3803-3810.

[8] Chebotar Y, Handa A, Makoviychuk V, et al. Closing the sim-to-real loop: Adapting simulation randomization with real world experience [C]// 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019: 8973-8979.

[9] Tobin J, Fong R, Ray A, et al. Domain randomization for transferring deep neural networks from simulation to the real world [C]// 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2017: 23-30.

[10] Niu H, Hu J, Cui Z, et al. DR2L: Surfacing corner cases to robustify autonomous driving via domain randomization reinforcement learning [C]// The 5th International Conference on Computer Science and Application Engineering. 2021: 1-8.

[11] Rusu A A, Večerík M, Rothörl T, et al. Sim-to-real robot learning from pixels with progressive nets [C]// Conference on Robot Learning. PMLR, 2017: 262-270.

[12] Christiano P, Shah Z, Mordatch I, et al. Transfer from simulation to real world through learning deep inverse dynamics model [J]. arXiv preprint arXiv: 1610. 03518, 2016.

[13] Sutton R S, Barto A G. Reinforcement learning: An introduction [M]. MIT press, 2018.

[14] Andrychowicz M, Wolski F, Ray A, et al. Hindsight experience replay [J]. *Advances in neural information processing systems*, 2017, 30.

[15] Ding Y, Florensa C, Abbeel P, et al. Goal-conditioned imitation learning [J]. *Advances in neural information processing systems*, 2019, 32.

[16] Dhiman V, Banerjee S, Siskind J M, et al. Learning goal-conditioned value functions with one-step path rewards rather than goal-rewards [J]. arXiv preprint arXiv: 1906. 02170, 2019.

[17] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning [J]. arXiv preprint arXiv: 1509. 02971, 2015.

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in IROS, 2012, pp. 5026–5033.

[19] 吴璞, 夏长林, 景鸿翔. UR5 机器人运动学分析与轨迹规划研究 [J]. 煤矿机械, 2021, 42 (04): 55-58. DOI: 10. 13436/j. mkjx. 202104018. (Wu Pu, Xia Changlin, Jing Hongxiang Research on kinematics analysis and trajectory planning of UR5 robot [J] *Coal Mine Machinery*, 2021, 42 (04): 55-58.)

[20] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms [J]. arXiv preprint arXiv: 1707. 06347, 2017.

Note: Figure translations are in progress. See original paper for figures.