Mixline: A Hybrid Reinforcement Learning Framework for Long-horizon Bimanual Coffee Stirring Task

Zheng Sun[1], Zhiqi Wang[1], Junjia Liu[1], Miao Li[2] (✉), and Fei Chen[1] (✉)

[1] The Chinese University of Hong Kong, Hong Kong, China feichen@cuhk.edu.hk
[2] Wuhan University, Wuhan, China miao.li@whu.edu.cn

Abstract. Bimanual activities like coffee stirring, which require coordination of dual arms, are common in daily life and intractable to learn by robots. Adopting reinforcement learning to learn these tasks is a promising topic since it enables the robot to explore how dual arms coordinate together to accomplish the same task. However, this field has two main challenges: coordination mechanism and long-horizon task decomposition. Therefore, we propose the *Mixline* method to first learn sub-tasks separately via the online algorithm and then compose them together based on the generated data through the offline algorithm. We constructed a learning environment based on the GPU-accelerated Isaac Gym. In our work, the bimanual robot successfully learned to grasp, hold and lift the spoon and cup, insert them together and stir the coffee. The proposed method has the potential to be extended to other long-horizon bimanual tasks.

Keywords: Reinforcement learning · Bimanual coordination · Isaac Gym

1 Introduction

The rapid development of Reinforcement Learning (RL) has provided new ideas for robot control [1][2], and the training of bimanual robots with coordination has become a hot topic in reinforcement learning. In this paper, we are interested in how to make bimanual robots learn human's daily activities. These activities are usually long-horizon and need the coordination of dual human arms. These two challenges limit the use of RL in complex tasks and make it unavailable to real-world scenarios. Here we design a long-horizon and bimanual coffee stirring task as an example to study the potential solution of these kinds of tasks. The illustration of coffee stirring movements is shown in Fig. 1. It contains movements like grabbing, lifting, inserting, and stirring. We can obtain the proper ways to decompose and recompose long-horizon tasks and coordinate between the dual arms by solving this task, which will be essential while extending to other long-horizon and bimanual tasks.

First Author and Second Author contribute equally to this work.

Fig. 1: Bimanual coffee stirring

In recent research, Zhang et al. proposed a novel disentangled attention technique, which provides intrinsic regularization for two robots to focus on separate sub-tasks and objects [3]. Chitnis et al. decompose the learning process into a state-independent task schema to yield significant improvements in sample efficiency [4]. Liu et al. applied dual-arm Deep Deterministic Policy Gradient (DDPG) [5] based on Deep Reinforcement Learning (DRL) with Hindsight Experience Replay (HER) [6] to achieve cooperative tasks based on "rewarding cooperation and punishing competition" [7]. Chiu et al. achieved bimanual regrasping by using demonstrations from a sampling-based motion planning algorithm and generalizing for non-specific trajectory [8]. Rajeswaran et al. augmented the policy search process with a small number of human demonstrations [9]. Cabi et al. introduced reward sketching to make a massive annotated dataset for batch reinforcement learning [10]. Mandlekar et al. presented Generalization Through Imitation (GTI), using a two-stage intersecting structure to train policies [11]. Zhang et al. gave a deep analysis of Multi-Agent RL theory and assessment of research directions [12]. Liu et al. combined Transformer with Graph neural network as *Structured-Transformer* to learn the spatio-temporal relationship of dual arms from human demonstration and achieve a traditional Chinese cooking art with stir-fry [13].

This paper first adopts the vanilla Proximity Policy Optimization (PPO) algorithm to learn the whole task directly and found that coordinated and long-horizon task properties make it hard to learn the desired stirring. Thus, we propose to use a task division process first to decompose the long-horizon task and learn each sub-task separately. Moreover, we propose wait training method to solve the poor coordination. Then by adopting a conservative Q-learning (CQL), we can combine the offline data generated via the separate learning process to achieve the learning of the whole task. We regard this hybrid reinforcement learning method which contains both online and offline RL algorithms, as *Mixline*. Besides, to speed up the learning process of the proposed algorithm, we choose to use Isaac Gym as our simulation platform, which can provide GPU-accelerated parallel computing and allow us to train in a colony [14].

2 Task Definition and Environment

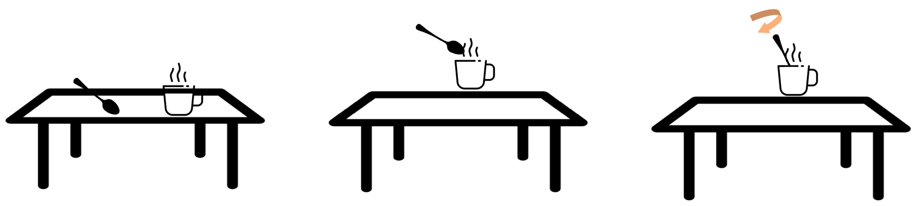
2.1 Task Definition

When we drink coffee every day, we pick up the spoon and the coffee cup with our left and right hands and use the spoon to stir the coffee in the cup. Therefore our training goal is to train two Franka robotic arms to do the same action. One arm grabs a spoon, the other grabs a cup, then lifts and inserts the spoon into the cup for a stirring action.

Moreover, we found that the previous work still has some flaws through the research of related work. For example, there is no straightforward dominant technique of robot reinforcement learning, especially in bimanual coordination. In robot learning, high-dimensional action, observation space, and sparse reward environments affect the training result seriously. Also, the exploration of long-horizon tasks is a challenge. According to the characteristics of the Isaac gym simulator, during the training process, we can obtain the observation and reward of any environment and agent at any time, which makes it possible to split the task training. Therefore, in our task, we split the whole task into three stages to solve the sparse reward and long-horzion problems. The three stages are:

1. Grasp and pick up the cup and spoon;
2. Insert the spoon into the cup;
3. Stir in the cup without hitting the cup overly.

The diagrams about three stages are shown in Fig. 2.



(a) Hold and lift (b) Insert (c) Stir overly

Fig. 2: Separate the coffee stirring task into three stages. From (a) to (c), there are diagrams about stage 1, stage 2, and stage 3.

By setting the reward function for the three stages separately, we solve the existing sparse reward problem by setting the reward function for the three stages separately. Splitting into three stages helps us solve the long-horizon and poor coordination problems.

2.2 Isaac Gym Environment

Isaac Gym offers a high-performance learning platform to train policies for a wide variety of robotic tasks directly on GPU [14]. It performs significantly better than other standard physics simulators using CPU clusters. Isaac Gym has many advantages in reinforcement learning:

- Providing a high-fidelity GPU-accelerated robot simulator platform;
- With the help of Tensor API, Isaac Gym goes through CPU bottlenecks by wrapping physics buffers into PyTorch [15] tensors;
- Tens of thousands of simultaneous environments on a single GPU [16].

Compared with other simulators, the advantage of Isaac Gym is that all the reward calculation processes and the physical simulation process are completed in the GPU with the help of the Tensor API, thus skipping the CPU bottleneck to speed up training significantly.

3 Methodology

3.1 Background of Reinforcement Learning

The dynamical system of reinforcement learning is fully defined by a Markov decision process (MDP) [17]. MDP can be defined as $M = (S, A, T, d_0, r, \gamma)$, where S is states, A means actions, T defines a conditional probability distribution that describes the dynamic system, d_0 is the initial state distribution, r is reward function, γ means discount factor.

The goal of reinforcement learning is to learn a policy $\pi(a, s)$. And the RL objective $J(\pi)$ can be derived as an expectation under the trajectory distribution, as Equ. 1. τ is the whole trajectory given by sequence of states s and actions a, and $p_\pi(\tau)$ is the trajectory distribution of the given policy π.

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)}[\sum_{t=0}^{H} \gamma^t r(s_t, a_t)] \tag{1}$$

3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [18] is an on-policy algorithm, which is the most widely used algorithm in reinforcement learning. It first makes improvements based on Policy Gradient (PG) Methods. The gradient estimator is written in the form [19].

$$\hat{g} = \hat{E}_t[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2}$$

where π is the state distribution. Therefore, for getting the gradient expression, the objective function can be constructed in the form (5).

$$L^{PG}(\theta) = \hat{E}_t[log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{3}$$

And the gradient or the objective function is usually modified by the method of important sampling [20]. Suppose the target policy is θ and the behavior policy is θ'. The modified objective function is

$$\hat{g}' = \hat{E}_{(s_t,a_t)}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \nabla_\theta log \pi_\theta(a_t|s_t)\hat{A^{\theta'}}_t] \tag{4}$$

$$\hat{L'}^{PG}(\theta) = \hat{E}_{(s_t,a_t)}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\hat{A^{\theta'}}_t] \tag{5}$$

As mentioned above, we do not want the gap between θ and θ' to be too large; hence we find a way to constrain it. KL divergence, also called relative entropy, can be used to measure the difference between two distributions. Therefore the most direct way is to add a penalty on KL divergence. The modified objective function is

$$\hat{J'}_{PPO}(\theta) = \hat{E}_\pi[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}A^{\theta'}(s,a)] - \beta KL(\theta,\theta^{'}) \tag{6}$$

3.3 Wait training mechanism for bimanual learning

We introduce a new training method to help us solve the problem of bimanual coordination. We call this training method wait training. The specific principle is as follows: we focus on the training situation of one robotic arm each time we train. Taking the first stage as an example, after the training starts, we lock the movement of the right arm so that the training process focuses on the grasping of the spoon by the left arm. When the left arm grabs the spoon, lock the movement of the left arm and unlock the movement of the right arm so that the training process focuses on grasping the cup by the right arm. When we successfully train dual-arm to grasp, unlock the arms movement and focus the training process on the lift of the arms. Through wait training, we have solved the problem of poor coordination between arms under the PPO algorithm.

3.4 Conservative Q-learning for sub-task composition

First, we trained the policy using PPO algorithm to complete each sub-task. Then we can derive the bimanual trajectory and combine them to form the whole long-horizon trajectory. We suppose the combined trajectory as our expert dataset, or offline dataset, and we can use them to do offline training.

To utilize offline data, we introduced conservative Q-learning (CQL) [21]; when doing offline training, it can be implemented on top of soft actor-critic (SAC) [22]. The primary purpose of CQL is to augment the standard Bellman error objective with a simple Q-value regularizer. In experience, the conservative-Q way is proved to be effective in mitigating distributional shift compared with other offline algorithms.

The conservative policy evaluation is first to minimize the Q value of all actions, then add a term to maximize the Q value of actions from the expert dataset to encourage actions that conform to the offline dataset and restrain actions beyond the dataset, where Equ. 7. In practice, we only need to add a simple Q-value regularizer like the KL-divergence regularizer. With the term added, we ensure the expected value of the policy under the existing Q-function is bounded under its true value, which means "conservative." Also, CQL uses an explicit Q-function penalty instead of the policy constraint method, which makes the method simple.

$$\begin{aligned}\hat{Q}^{k+1} \leftarrow \arg\min_{Q}\ \alpha \cdot (\mathbb{E}_{s\sim D, a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim D, a\sim\hat{\pi}_{\beta}(a|s)}[Q(s,a)]) \\ + \frac{1}{2}\mathbb{E}_{s,a,s'\sim D}[(Q(s,a) - \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(s,a))^{2}]\end{aligned} \tag{7}$$

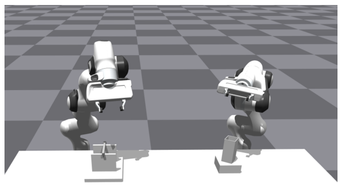
In our work, this *Mixline* method can help collect enormous datasets, and we only have to train sub-task using the online algorithm, which reduces the learning difficulty for long-horizon tasks.

4 Experiments

4.1 Simulation Environment

In our work, we set seven objects in each working space in the Isaac Gym environment. There are two Franka robots, one spoon on a shelf, one cup on a box shelf, and a table, as shown in the Fig. 3a.

Simulation: Every Franka robot has seven revolute joints and two prismatic joints. The action space is continuously between -1 to 1 and uses relative control. Our basic RL settings is shown in Fig. 3b.


(a) Franka robots and assets in Isaac Gym environment

item	Description
Action buffer size	18
Range of action values	[-1 to 1]
Observation buffer size	74
Number of environments	1024

(b) RL settings

Fig. 3: Simulation environment.

To facilitate the state of the grasp in the training process, we divided the observation space in each workspace into 74. The Table 1 records the allocation of each part of the buffer. We labeled one Franka arm as Franka-spoon and the other as Franka-cup.

RL flow: In our experiment,we use PyTorch [15][23] as base RL library. In the RL iteration, the agents sample actions from the policy and give the actions to the environment for physics simulation; then, the environment gives back all the buffer needed to compute for further steps such as reward calculation.

Table 1: Obversation buffer table

index	Description
0-8	the scale of Franka-spoon's joint positions
9-17	the scale of Franka-cup's joint positions
18-26	the velocity of Franka-spoon's joints
27-29	the relative position between the top of gripper and spoon
30-38	the velocity of Franka-cup's joints
39-41	the relative position between the top of gripper and cup
42-48	spoon's position and rotation
49-55	cup's position and rotation
56-64	Franka-spoon's joint positions
64-73	Franka-cup's joint positions

4.2 Design of Reward Function

In general, all the rewards and penalties are given a scale to get the total reward.

Reward

Distance reward: Distance reward is relative to the grasping point. The reward lifts when the robot hand moves towards the object's grasp point.

$$r_{dist} = \begin{cases} \frac{2.0}{1.0+d^2} & d \leq threshold \\ \frac{1.0}{1.0+d^2} & \text{otherwise} \end{cases} \tag{8}$$

Rotation reward: We pre-define the axis to ensure alignment. The rotation reward gets its maximum value when the specified robot plane coincides with the pre-defined plane.

$$\begin{aligned} dot_i &= v_i \cdot v_i' \\ r_{rot} &= \frac{1}{2}(\text{sign}(dot_1) * {dot_1}^2 + \text{sign}(dot_2) * {dot_2}^2) \end{aligned} \tag{9}$$

Around reward: When the robot grasps something, we have to make two fingers at the different sides of the object, not only orientation and distance. Only in this situation can the object be taken up. This reward is the prerequisite for taking up the object.

$$r_{around} = \begin{cases} 0.5 & pos_{lfinger} < pos < pos_{rfinger} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Finger distance reward: If the around reward is satisfied, it can grasp the object when the gripper is close. Meantime, the distance should be small to ensure both the finger and robot hand is close enough, and the gripper holds tighter, the reward is bigger. Here LF means left finger, RF means right finger.

$$\begin{aligned} d_f &= d_{LF} + d_{RF} \\ r_{fingdist} &= \begin{cases} 0.08 - d_f & r_{around} > 0 \\ 100 \times (0.08 - d_f) & r_{around} > 0 \text{ and } d \leq threshold \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

Lift reward: The target of stage 1 is to take up the object. Hence the reward boosts when the robot hand grasps the object successfully and lifts the object. In the meantime, we expect the gripper holds the object tightly so that the object will not drop during fast movement.

$$r_{lift} = \begin{cases} h \times (1 + r_{around}) & h < 0 \\ h \times (1 + r_{around} + 10 r_{fingdist}) & h \geq 0 \end{cases} \tag{12}$$

Movement reward: For the long-horizon task, in stage 3, the robot arm needs to stir the coffee constantly. We set the linear velocity as a reward to keep the robot moving to achieve the goal.

$$r_{move} = \begin{cases} v_{spoon} & \text{fulfill the stage 3 pre-conditions} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Penalty

In general, the penalty term prevents the agent do unexpected actions.

Action penalty: Making a regularization of the actions means more actions and more penalties. We expect the robot can finish the task as soon as possible.

$$p_{action} = \sum action^2 \tag{14}$$

Collision penalty: The robot may hit the table if the link contact force exceeds the limit. An unexpected collision gives a penalty to the total reward.

$$p_{collision} = \begin{cases} -1 & \text{force} > threshold \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Loosen gripper penalty: After holding the object, the gripper should keep its relative position; if the gripper opens again, giving a penalty.

$$p_{gripper} = \begin{cases} -1 & d_{finger} > threshold \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

Fall penalty: The robot receives a fall penalty if the object is knocked down by the robot arm when it tries to grasp or in other similar circumstances.

Wrong pose penalty: The goal is to let the robots achieve bimanual coordination so that the trajectory should always be reasonable, not only the final target position. For example, the cup should not be inverted during the movement. Fall penalty can be added into this penalty.

$$p_{wrong} = \begin{cases} -1 & \text{unexpected cases} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

4.3 Reset Condition

The robot should always stay in its safe operational space. We don't expect the robot to be close to its singularity position. We reset the environment if the pose is unreasonable. For example, in our task, we hope the robot should not let the object drop; if the object drops on the table or ground, we directly reset it and give a penalty. In common, the environment should be reset if the task step exceeds the maximum task length. In our experiment, we set this value to 1k, more significant than the step needed in policy to succeed.

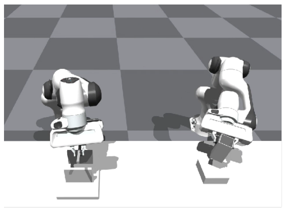
4.4 Results

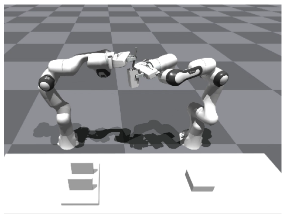
In our work, we generated expert datasets with online algorithm, then proposed combining the sub-task offline dataset to train the policy. In stage 1, we trained to grasp cup and spoon in the PPO algorithm, used the wait training method to solve coordination problems, and the CQL+PPO method in several different environment settings. In stage 2 & 3, we set the end of stage 1 as the initial state and trained the robot arm to complete the task. With the help of the above methods, We finish our training goals and results as shown in Fig.4. After combining all stages, we proposed to train the policy using the offline algorithm to finish the whole long-horizon task. The training results are consistent with our expectations, proving that the method proposed in bimanual reinforcement learning is effective.

4.5 Ablation Study

We tested the performance in Ant, Humanoid, and our task environments in general PPO method and variant CQL implement based on PPO, as shown in Fig. [5a, 5b, 5c].

The result in Fig. 5 shows that CQL(ρ) performs significantly better than the general PPO algorithm in our task. In typical environments like Ant and Humanoid, PPO and CQL act nearly the same. This result is expected because, in the on-policy algorithm, the behavior policy is the same as the learned policy, therefore the effect of optimizing terms to promote learning efficiency is limited.


(a) Stage 1: Hold and lift


(b) Stage 2: Insert

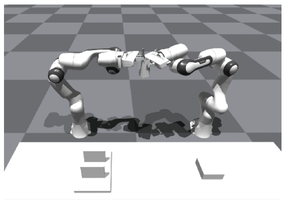
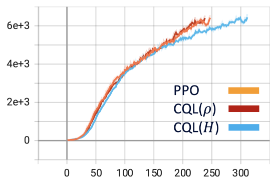
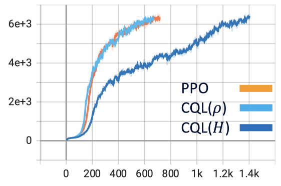

(c) Stage 3: Stir overly

Fig. 4: Result of each stage

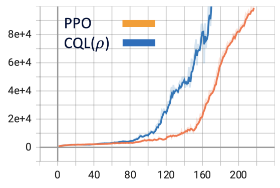
Our *Mixline* method to separate the whole long-horizon task into sub-tasks effectively generates the trajectory and can be easily extended to use in the offline algorithm.



(a) Ant



(b) Humanoid



(c) Our task

Fig. 5: Performance of PPO and CQL+PPO in different task environments

5 Conclusion

In this paper, we propose a novel hybrid reinforcement learning method for learning long-horizon and bimanual robot skills. To overcome the long-horizon issue, we combine the online and offline reinforcement learning algorithms as the *Mixline* method to learn sub-task separately and then compose them together. Besides, we design a wait training mechanism to achieve bimanual coordination. The experiments are conducted in parallel based on the Isaac Gym simulator. The results show that by using the *Mixline* method, we can solve the long-horizon and bimanual coffee stirring task, which is intractable by just using online algorithms.

The proposed method has the potential to be extended to other long-horizon and bimanual tasks. Moreover, combining online and offline RL algorithms might allow us to add human demonstration as the initial offline data to boost policy learning. Another further direction is to model the coordination mechanism by neural network rather than setting waiting manually.

References

1. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017). https://doi.org/10.1109/MSP.2017.2743240
2. Ding, Z., Huang, Y., Yuan, H., Dong, H.: Introduction to reinforcement learning. In: Deep reinforcement learning, pp. 47–123. Springer (2020)
3. Zhang, M., Jian, P., Wu, Y., Xu, H., Wang, X.: Disentangled attention as intrinsic regularization for bimanual multi-object manipulation. arXiv e-prints pp. arXiv–2106 (2021)
4. Chitnis, R., Tulsiani, S., Gupta, S., Gupta, A.: Efficient bimanual manipulation using learned task schemas. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 1149–1155. IEEE (2020)
5. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
6. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W.: Hindsight experience replay. Advances in neural information processing systems **30** (2017)
7. Liu, L., Liu, Q., Song, Y., Pang, B., Yuan, X., Xu, Q.: A collaborative control method of dual-arm robots based on deep reinforcement learning. Applied Sciences **11**(4), 1816 (2021)
8. Chiu, Z.Y., Richter, F., Funk, E.K., Orosco, R.K., Yip, M.C.: Bimanual regrasping for suture needles using reinforcement learning for rapid motion planning. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 7737–7743. IEEE (2021)
9. Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., Levine, S.: Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. arXiv preprint arXiv:1709.10087 (2017)
10. Cabi, S., Colmenarejo, S.G., Novikov, A., Konyushkova, K., Reed, S., Jeong, R., Zolna, K., Aytar, Y., Budden, D., Vecerik, M., et al.: Scaling data-driven robotics with reward sketching and batch reinforcement learning. arXiv preprint arXiv:1909.12200 (2019)
11. Mandlekar, A., Xu, D., Martín-Martín, R., Savarese, S., Fei-Fei, L.: Learning to generalize across long-horizon tasks from human demonstrations. arXiv preprint arXiv:2003.06085 (2020)
12. Zhang, K., Yang, Z., Başar, T.: Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of Reinforcement Learning and Control pp. 321–384 (2021)
13. Liu, J., Chen, Y., Dong, Z., Wang, S., Calinon, S., Li, M., Chen, F.: Robot cooking with stir-fry: Bimanual non-prehensile manipulation of semi-fluid objects. IEEE Robotics and Automation Letters **7**(2), 5159–5166 (2022)
14. Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G.: Isaac gym: High performance gpu-based physics simulation for robot learning (2021)
15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett,

R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

16. Wan, W., Harada, K.: Developing and comparing single-arm and dual-arm regrasp. IEEE Robotics and Automation Letters **1**(1), 243–250 (2016)
17. Sutton, R.S., Barto, A.G., et al.: Introduction to reinforcement learning (1998)
18. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
19. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems **12** (1999)
20. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning. pp. 1889–1897. PMLR (2015)
21. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems **33**, 1179–1191 (2020)
22. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. International Conference on Machine Learning (ICML) (2018)
23. Yadan, O.: Hydra - a framework for elegantly configuring complex applications. Github (2019), https://github.com/facebookresearch/hydra