Towards Human-Level Bimanual Dexterous Manipulation with Reinforcement Learning

Yuanpei Chen[1], **Yaodong Yang**[1,5,†], **Tianhao Wu**[2], **Shengjie Wang**[1], **Xidong Feng**[3], **Jiechuang Jiang**[1], **Stephen Marcus McAleer**[4], **Hao Dong**[2], **Zongqing Lu**[1], **Song-Chun Zhu**[1,5]
[1]Institute for AI, Peking University
[2]Center on Frontiers of Computing Studies, Peking University
[3]University College London
[4]Carnegie Mellon University
[5]Beijing Institute for General Artificial Intelligence

Abstract

Achieving human-level dexterity is an important open problem in robotics. However, tasks of dexterous hand manipulation, even at the baby level, are challenging to solve through reinforcement learning (RL). The difficulty lies in the high degrees of freedom and the required cooperation among heterogeneous agents (e.g., joints of fingers). In this study, we propose the **Bi**manual **Dex**terous **Hands** Benchmark (Bi-DexHands), a simulator that involves two dexterous hands with tens of bimanual manipulation tasks and thousands of target objects. Specifically, tasks in Bi-DexHands are designed to match different levels of human motor skills according to cognitive science literature. We built Bi-DexHands in the Issac Gym; this enables highly efficient RL training, reaching 30,000+ FPS by only one single NVIDIA RTX 3090. We provide a comprehensive benchmark for popular RL algorithms under different settings; this includes Single-agent/Multi-agent RL, Offline RL, Multi-task RL, and Meta RL. Our results show that the PPO type of on-policy algorithms can master simple manipulation tasks that are equivalent up to 48-month human babies (e.g., catching a flying object, opening a bottle), while multi-agent RL can further help to master manipulations that require skilled bimanual cooperation (e.g., lifting a pot, stacking blocks). Despite the success on each single task, when it comes to acquiring multiple manipulation skills, existing RL algorithms fail to work in most of the multi-task and the few-shot learning settings, which calls for more substantial development from the RL community. Our project is open sourced at https://github.com/PKU-MARL/DexterousHands[1].

1 Introduction

Humans have a skillful ability to manipulate objects of different shapes, sizes, and materials, which rely on the dexterity of our hands and fingers. Building a robot inspired by the human hands that can autonomously manipulate various objects has always been an important component of the robotics field [1]. The development of human dexterity begins in infancy and is influenced by what the physical environment provides, including the objects available to the child [2]. As infants and children develop physical and intelligence, they are more likely to attempt complex movements, and often learn dexterity through attempting movements and the consequences of their actions [3, 4, 5]. Similarly, robot dexterity can not be a constant program pre-set in the laboratory. To acquire the capability of object manipulations in the real world, robots must be able to learn dexterous manipulation skills as if

[1]Yuanpei Chen and Shengjie Wang worked as a research intern at Peking University. [†]Corresponding to <yaodong.yang@pku.edu.cn>

Preprint. Under review.

we were infants. As a result, we expect robots to learn to master the ability of dexterous manipulation at the human level from daily tasks.

Recently, reinforcement learning (RL) algorithms have outperformed human experts in many fields that require decision makings [6, 7]. In contrast to the traditional control methods, RL can complete some challenging tasks in learning dexterous in-hand manipulation [8, 9, 10]. However, manipulation that generates changes on the object is still difficult [11]. More difficult is generalization across tasks, although previous work can achieve simple level of tasks such as throwing [12], sliding [13], poking [14], pivoting [15], and pushing [16], but is still difficult to perform well in unstructured or contact-rich environments, which require the ability to combine and generalize complex manipulation skills. In a nutshell, reaching human-level sophistication of hand dexterity and bimanual coordination remains an open challenge for modern robotics researchers.

To help solve the problems mentioned above and let robots have the same dexterous manipulation ability as humans, we developed, in the Isaac Gym [17] simulator, a novel benchmark on bimanual dexterous manipulation for RL algorithms along with a diverse set og tasks and objects named **Bi-DexHands**. We follow the principle of Fine Motor Subtest (FMS) [18] to design tens of tasks, which provides the opportunities to observe and evaluate specific skills that demonstrate a child's ability to use their hands to play with toys, manipulate objects, and use tools. Next, we tested the baselines of various model-free RL algorithms to show the ability of the baseline algorithm in these tasks, not only the standard RL algorithms but also multi-agent RL (MARL), offline RL, multi-task RL, and Meta RL algorithms, each of them focuses on the bimanual collaboration, learning from demonstration, and task generalization, respectively. Our major goal is to facilitate researchers to master human-level bimanual dexterous manipulations with RL. Not limited to this, we also hope this study to provide a new platform for the community of RL, robotics, and cognitive science. Bi-DexHands are developed with the following key features:

- **Isaac Gym Efficiency**: Building on the Isaac Gym simulator, Bi-DexHands supports running thousands of environments simultaneously. On one NVIDIA RTX 3090 GPU, Bi-DexHands can reach 30,000+ mean FPS by running 2,048 environments in parallel.
- **Comprehensive RL Benchmark**: We provide the first bimanual manipulation task environment for common RL, MARL, offline RL, multi-task RL, and Meta RL practitioners, along with a comprehensive benchmark for SOTA continuous control model-free RL methods.
- **Heterogeneous-agent Cooperation**: Agents in Bi-DexHands (*i.e.*, joints, fingers, hands,...) are genuinely heterogeneous; this is different from common multi-agent environments such as SMAC [19] where agents can simply share parameters to solve the task.
- **Task Generalization**: We introduced a variety of dexterous manipulation tasks (e.g., hand over, lift up, throw, place, put...) as well as enormous target objects from the YCB [20] and SAPIEN [21] dataset, thus allowing meta-RL and multi-task RL algorithms to be tested on the task generalization front.
- **Cognition**: We provided the underlying relationship between our dexterous tasks and the motor skills of humans at different ages. This facilitates researchers on studying robot skill learning and development, in particular in comparison to humans.

2 Related Work

Today, robots are skilled in some repetitive and familiar environments like assembled in the factory. Grasping is a milestone in robotics manipulation. For decades, researchers have been working to establish a stable grasping theory [22]. However, most previous methods have relied on various assumptions, such as known object information or no uncertainty in the process. In recent years, data-driven approaches have been successful in this regard, being able to deal with uncertainty in perception and generate grasping methods for known, familiar, and even unknown objects in real-time [23]. Grasping is only a part of the manipulation. Today's robots can perform some simple behaviors like grasping, pushing, and throwing. But it is still difficult to manipulate in unstructured scenes and contact-rich situations. Moving objects while in-hand manipulation is also a complex challenge. One step to address this challenge is to use hands with intrinsic dexterity [24, 25], which often mimic human hands [26]. Another undeveloped area is bimanual manipulation, a method of using a second hand to provide additional dexterity [27, 28]. Learning for manipulation is important for robots to continuously learn and achieve intelligent control. It is especially suitable for modeling

manipulation on complex non-rigid objects and reducing control dimensions [29], but it still suffers from problems such as lack of accurate models, reality gaps, and difficulty in collecting expert data. Therefore, our work proposes a bimanual dexterous manipulation benchmark, hoping to facilitate researchers to address the challenges of robotic manipulation we mentioned above.

Dexterous five-finger hands provide an essential tool to perform a multitude of tasks in human-centric environments. However, such dexterous manipulation remains a challenging problem because of the high dimensional actuation space and contact-rich model. Before the emergence of RL-based controllers, a large variety of manipulation tasks highly relied on accurate dynamics models and trajectory optimization methods [30, 31, 32]. For example, Williams et al. [33] used the model predictive path integral control (MPPI) method to perform the task successfully, dexterous manipulation of a cube. Charlesworth et al. [34] improved the MPPI method to make the handing over task between two hands tractable. Since RL simplifies the design process of the controller, model-agnostic approaches have become more and more popular in the field of robotic control. In terms of dexterous manipulation, many works achieve a significant improvement compared with traditional controllers. OpenAI et al. [9] developed an RL-based controller to reorient a block or a Rubik's cube. Considering the poor generalization of current approaches, Chen et al. [10] presented an efficient system for learning how to reorient a large number of objects without access to shape information. While their studies demonstrate that RL enables efficient and scalable learning on single-hand manipulation, bimanual manipulation remains a hardship for model-free reinforcement learning [34]. In this paper, our benchmark provides a wide range of well-designed and challenging daily life scenarios for comprehensive RL algorithms, hoping to help the researcher toward master human-level bimanual dexterous manipulation.

3 Formulations & Algorithms

In order to create a platform toward master human-level dexterity, we use two Shadow hands to manipulate in our environment. Shadow hand [35] is a popular robotic hand usually used in some dexterous manipulation tasks. It is designed to resemble the typical human male hand in shape and size, and capable of performing a variety of flexible and delicate operations. Shadow hand's DoF is shown in Fig.1, designed to mimic the human skeleton as much as possible. Concretely, the 24-DoF hand is actuated by 20 pairs of agonist-antagonist tendons, while the other four joints remain underactuated.

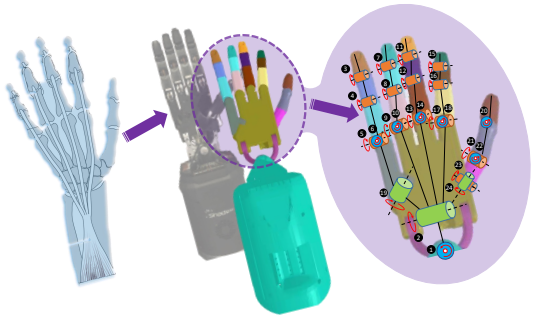


Figure 1: Degree-Of-Freedom (DOF) configuration of the Shadow Hand similar to the skeleton of a human hand.

Furthermore, our low-level controller runs at 1k Hz, as well as the RL-based policy outputs the relative positions of actuated joints at 30 Hz. It is worth noting that compared with previous studies, the base of the hand is not fixed in some tasks. Instead, the policy can control the position and orientation of the base within a restricted space, which takes advantage of the function of the wrist, thus making the Shadow hand more bio-mimetic. Meanwhile, we can efficiently perform the task in real-world applications by linking the base to a robotic arm. Refer to Appendix A.1 for more details about the physical parameters of the Shadow hand.

Our benchmark aims at providing solutions for bimanual dexterous manipulation in a comphensive field of RL. To achieve that, We consider five RL formulations including: Single-agent RL, Multi-agent RL (MARL), Offline RL, Multi-task RL, and Meta-RL in Bi-DexHands. In the following part, we will introduce the detailed formulation and the corresponding implemented algorithms in our benchmark of these five RL formulations.

RL/MARL. In order to evaluate the performance of RL/MARL, we formulate our scenarios as a decentralized partially observable MDP (Dec-POMDP). The Dec-POMDP consists of 10 components, $Z =< \mathcal{N}, \mathcal{M}, S, \boldsymbol{O}, \boldsymbol{A}, \boldsymbol{\Pi}, P, R, \rho, \gamma >$. Initially, the robotic hands are manually separated as $\mathcal{N}$ agents, the set of which represents $\mathcal{M}$. When starting the simulation, the state of the environment (i.e., the information of robots and objects) is set at $s_0 \in S$ according to the initial distribution of states $\rho(s_0)$. Then at the time step t, s_t represents the state, and the i-th agent receives an observation $o_t^i \in \boldsymbol{O}$ relying on s_t. Hereafter, the policy of the i-th agent, $\pi_i \in \boldsymbol{\Pi}$, takes the o_t^i as input, and

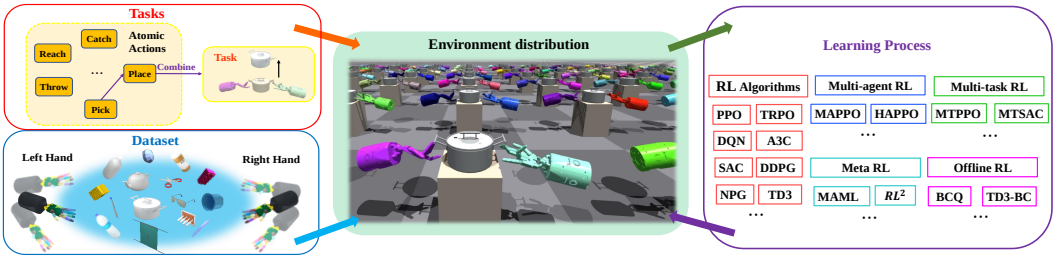


Figure 2: Framework of Bi-DexHands, a bechmark for learning bimanual dexterous manipulation.

outputs an action $a_t^i \in A_i$. Additionally, we denote the joint action of all agents by $\boldsymbol{a}_t \in \boldsymbol{A}$, and the equation $\boldsymbol{A} = [A_1, ..A_{\mathcal{N}}]$ is naturally satisfied. After that, i-th agent can obtain a reward r_t^i based on $R(s_t, \boldsymbol{a_t})$, as well as all agents transitions to the next state s_{t+1} with the possibility of the transition function $P(s_{t+1}|s_t, \boldsymbol{a}_t)$. The goal is to find the optimal policy $\boldsymbol{\Pi}$ to maximize the sum of rewards $\mathbb{E}_{\boldsymbol{\Pi}}[\sum_{t=0}^{T-1} \gamma^t \sum_{i=1}^{\mathcal{N}} r_t^i]$ in an episode with T time steps. It should be pointed out that when $\mathcal{N} = 1$, it is the problem formulation of single-agent RL.

In this setting, We implemented state-of-the-art continuous single-agent RL algorithms, such as PPO [36], SAC [37], TRPO [38], DDPG [39], and TD3 [40] algorithms. Taking our continuous control and fully cooperative environments into consideration, we introduced HAPPO [41], HATRPO [41], MAPPO [42], IPPO [43], and MADDPG [44] algorithms.

Offline RL. Offline RL follows the formulation of standard MDP, where the goal is to maximize the expected return $\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t]$. However, in offline RL, the agent has to learn policy only using the transitions in previously collected dataset $\mathcal{D} = \{(s_t, a_t, s_{t+1}, r_t)\}$, without interacting with the environment. The fundamental challenge of offline RL is value errors of out-of-distribution actions. We implemented BCQ [45], TD3+BC [46], and IQL [47] algorithms for offline RL.

Multi-task RL. Multi-task reinforcement learning aims to train a single policy $\pi(a|s, z)$, which can achieve good results on different tasks. z represents an encoding of the task ID. The goal of our policy is to maximize the reward given by $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathbb{E}_\pi[\sum_{t=0}^{T-1} \gamma^t r_t]$, where $p(\mathcal{T})$ is a task distribution in our benchmark. In practice, multi-task RL adds the context vector corresponding to the type of environment (e.g., one-hot task ID) into states to learn a general skill. We implemented multi-task PPO, multi-task TRPO, and multi-task SAC algorithms for multi-task RL.

Meta RL. Meta RL, also known as "learning to learn", aims to gain the ability to train on tasks to extract the common features of these tasks, so as to quickly adapt to new and unseen tasks. In Meta-RL, both training and test environments are assumed to follow the same task distribution $p(\mathcal{T})$. In Bi-DexHands, we design some common structures between different tasks for meta-training to ensure that it can adapt efficiently to new tasks. Compared with Multi-task RL, Meta RL is not allowed to get direct task-level information such as task ID. It needs to solve entirely new tasks by task inference and adaptation purely based on interactions. We implemented model-agnostic meta learning (MAML) [48] and proximal meta-policy search (ProMP) [49] algorithms for Meta RL.

4 Bimanual dexterous manipulation benchmark

In this section, we will discuss the construction of Bi-DexHands, a benchmark for bimanual dexterous manipulation over diverse scenarios.

4.1 System design

As we mentioned before, the core of Bi-DexHands is to build up a learning framework for two Shadow hands capable of diverse skills as humans, such as reaching, throwing, catching, picking and placing. To be specific, Bi-DexHands consists of three components: datasets, tasks, and learning algorithms, as shown in Fig.2. Varying worlds provide a large number of basic settings for robots, including the configuration of robotic hands and objects. Meanwhile, a variety of tasks corresponding to children's behaviors at different ages make it possible to learn dexterity like a human. Combining a dataset and task, we can generate a specific environment or scenario for the following learning. Eventually, our experiments demonstrate that reinforcement learning is able to facilitate the robots to

achieve some remarkable performance on such challenging tasks, and there is still some room for improvement and more difficult tasks for future work.

4.2 Construction of datasets

The construction of the datasets corresponds to the configuration of robots and objects. The core goal of datasets is to generate a large variety of scenes for robot learning. As we mentioned in the last part, the robots in our benchmark are two dexterous Shadow hands. Other than the robots, the objects also play an essential role in constructing the datasets. For extending the types of tasks, we introduced a variety of objects from the YCB [20] and SAPIEN [21] datasets. Two datasets contain many everyday objects. Notably, the SAPIEN dataset provides many articulated objects with motion annotations and rendering material, which means these objects are close to the real ones significantly. Therefore, it provides a natural way to build a connection between the worlds of our benchmark and scenes of daily life. Concretely, Fig.2 shows the construction of datasets, and we can see that the object includes pots, pens, eggs, scissors, eyeglasses, doors, and other common tools. After defining the configuration of robots and the type of objects, we build the specific world based on the Isaac Gym simulator. Meanwhile, each world defines variable initial poses of robots and objects, providing a diverse set of environments.

4.3 Design of tasks

An infant's behavior experiences a multi-stage development, such as social, communication, and physical parts [50]. Particularly in bimanual dexterous manipulation, there are some relationships between some common behaviors of babies and the ages. To gain insights into the underlying relationships, we conducted an in-depth analysis and built a mapping between the baby's age and tasks according to the Fine Motor Subtest (FMS) [18]. As the baby's age increases, the difficulty of completing the designed tasks also increases, because the baby can complete more and more difficult behaviors as the body develops. So it is also of great importance to evaluate the performance of trained agents. This is because we can precisely point out agents' intelligence level by analogy with a baby's movement for bimanual dexterous manipulation. An overview of the correspondence of our tasks to the FMS is shown in Table.1. For more details on the tasks, please refer to Appendix A.2.

4.4 Design of Multi-task/Meta RL

The design of our Multi-task/Meta RL categories is generally similar to meta-world [54], divided into ML1, MT1, ML4, MT4, ML20, and MT20. Each of our tasks has object variation, which as we can interact with different kinds of objects in daily life scenes, providing a foundation for us to learn dexterous manipulation like humans. In the following, we will introduce 6 tasks categories for Multi-task/Meta-RL. More details can refer to Appendix D.

MT1&ML1: Learning a multi-task policy&Few-shot adaptation within one task: Both ML1 and MT1 are categories for generalization ability within the same task, and their generalization ability is reflected in the ability to complete tasks under different goals. ML1 uses meta-reinforcement learning for few-shot adaptation, in which goal information will not be provided. MT1 uses the multi-task method for generalization, and the information on the goal will be provided in a fixed set.

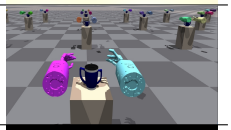
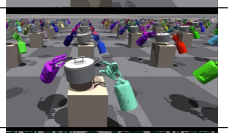
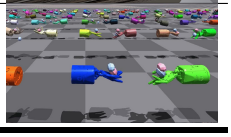
MT4&MT20: Learning a multi-task policy belonging to 4&20 training tasks: MT4 and MT20 conduct policy training in 4&20 tasks and hope to complete all tasks in only one policy. In MT4, we hope to learn policy with similar human skills, so we try to combine similar tasks as much as possible. MT20 uses all of our 20 tasks. In MT4 and MT20, we use a one-hot task ID to represent different tasks, and the information on the goal will be provided in a fixed set.

ML4&ML20: Learning a Few-shot adaptation for new 1&5 test tasks from 3&15 training tasks: ML4 and ML20 are categories for learning meta-policies in 3&15 tasks respectively and hoping to adapt to new 1&5 testing tasks. There is no doubt that this is a difficult challenge. We choose the tasks which using the catch behavior for design in ML4. The ML20 requires adaptation in all 15 tasks with large differences designed according to baby intelligence, which is the most difficult challenge in our benchmark. Similarly, we will variate the goal for each task, and will not provide task information, requiring Meta RL algorithms to identify the tasks.

5 Benchmarking reinforcement learning algorithms

In this section, we conduct a full benchmark of the RL algorithms in Bi-DexHands. We firstly quantify our environment speed to demonstrate the running efficiency of Bi-DexHands. Then We

Table 1: Task name and the description of the human skill in the corresponding age. References under the human age are the cognitive science literature referenced for the behavior designed, and the difficulty level of the tasks is under the task name. Easy level tasks are more basic skills, medium level tasks need more precise control and finger dexterity, and hard level tasks require handing dynamic interaction and tool use.

Task Name	Human's Skill Description	Age (months)	Demo
Push Block Easy	Child extends one or both arms forward and touches the block with any part of either hand	5-6 [18, Chapter 3]	
Open Scissor & Open Pen Cap Easy	They use one hand to hold a toy and the other hand manipulate it	7 [51, Chapter 4]	
Turn Button ON/OFF Easy	They can push and squish soft stuff or push hard things, like a button on a toy phone or popup toy	11 [52, 11 months]	
Swing Cup Easy	They can turn a ball on their toy mobile, a steering wheel on a toy car, or the faucet in the tub	11 [52, 11 months]	
Lift Pot & Lift Cup Easy	They can put a sippy cup to their mouth to drink	12 [52, 12 months]	
Door Open & Close Easy	Toddlers can open and close cupboards and oven doors	13 [52, 13 months]	
Re-Orientation Medium	Infant further refines this ability to differentiate individual finger movement and manipulate objects	18 [51, Chapter 4]	
Stack Block(2,6,8) Medium	Child stacks at least 2/6/8 blocks in any trial.	2:22-28 6,8:33-42 [18, Chapter 3]	
Pull a Ball into Bucket Medium	Child place s 10 pellets in the bottle in 60 seconds or less, one pellet at a time.	22-28 [18, Chapter 3]	
Open Bottle Cap Medium	Uses hands to twist things, like turning doorknobs or unscrewing lids	30 [53, Table 6]	
Catch Underarm Hard	Catches a large ball most of the time	48 [53, Table 6]	
Pour Water Hard	Serves himself food or pours water, with adult supervision	48 [53, Table 6]	
Two Catch Underarm Hard	Some adults can throw objects between two hands like magic	adult	

offer the benchmark results and corresponding discussion and analysis on those five RL formulations. All of our experiments are run with Intel i7-9700K CPU @ 3.60GHz and NVIDIA RTX 3090 GPU. For the hyperparameters of all algorithms, please refer to Appendix B.

5.1 Environmental speed

Thanks to Isaac Gym's high-performance GPU parallel simulating capabilities, we can greatly improve the sampling efficiency of our RL algorithm while using fewer computing resources. We believe that the high sampling efficiency improves the exploration ability of the RL algorithm, allowing us to successfully learn the bimanual dexterous manipulation policy. To demonstrate the Isaac Gym's efficiency of Bi-DexHands, We provided some results of environmental speed in Table.2 by running on-policy algorithms. Both PPO and HAPPO can achieve more than 20k FPS.

Table 2: Mean and standard deviation of FPS (frame per second) of the environments in Bi-DexHands.

Algorithms	CatchUnderarm	CatchOver2Underarm	CatchAbreast	TwoCatchUnderarm
PPO	35554 ± 613	35607 ± 344	35164 ± 450	32285 ± 898
HAPPO	23929 ± 98	23827 ± 135	23456 ± 255	23205 ± 168

5.2 RL/MARL results

Currently, we only evaluate the performance of PPO, SAC, MAPPO, and HAPPO algorithms on these 20 tasks, and we implemented the rest of the RL/MARL algorithms in our Github repository. The performance of each algorithm are shown in Figure 3. Note that the experiments of MARL algorithms run based on two agents, which means each hand represents an agent. It can be observed that the PPO algorithm performs well on most tasks. Although there are some tasks that require two-hand cooperation, PPO algorithm is still better than HAPPO, MAPPO algorithms in most cases. This may be because PPO algorithm is able to use all observations for training the policy, while MARL can only use partial observations. However, in most tasks, the more difficult and require the cooperation of both hands, the smaller performance gap between PPO and HAPPO, MAPPO, indicating that the multi-agent algorithm can improve the performance of bimanual cooperative manipulation. Another finding is that the SAC algorithm does not work on almost all tasks. It may be due to 1) the off-policy algorithm has a lower improvement in high sampling efficiency than on-policy. 2) The policy entropy of SAC brings instability to policy learning under the high-dimension input. We discuss this finding in detail in Appendix C.

5.3 Offline RL results

We build offline datasets with four datatypes, *i.e.*, random, replay, medium, and medium-expert. The data collection follows that in D4RL-MuJoCo [55], which is a standard offline benchmark, and the details are given in Appendix A.3. We evaluate behavior cloning (BC), BCQ [45], TD3+BC [46], and IQL [47] on two tasks, Hand Over and Door Open Outward, and report normalized scores in Table 3. BCQ and TD3+BC could obtain significant performance improvement compared with behavior policy (BC). However, the action space and state space in Bi-DexHands are much larger than that in MuJoCo, which means the problem of out-of-distribution action is more severe in Bi-DexHands datasets. That is the reason why IQL could only achieve performance improvement in several datasets. Due to the potential large distribution shift, we believe Bi-DexHands can be a more challenging and meaningful offline benchmark for offline RL research.

Table 3: Normalized score in offline tasks.

Tasks	Datasets	Online PPO	BC	BCQ	TD3+BC	IQL
Hand Over	random	100.0	0.7 ± 0.2	1.0 ± 0.1	0.9 ± 0.2	0.7 ± 0.4
	replay	100.0	17.5 ± 3.5	61.6 ± 4.9	$\mathbf{70.1} \pm 2.1$	43.1 ± 2.3
	medium	100.0	61.6 ± 1.0	$\mathbf{66.1} \pm 1.9$	$\mathbf{65.8} \pm 2.2$	57.4 ± 1.5
	medium-expert	100.0	63.3 ± 1.4	$\mathbf{81.7} \pm 4.9$	$\mathbf{84.9} \pm 5.3$	67.2 ± 3.6
Door Open Outward	random	100.0	2.1 ± 0.6	23.8 ± 2.9	$\mathbf{34.9} \pm 4.3$	3.8 ± 1.0
	replay	100.0	36.9 ± 4.3	48.8 ± 4.5	$\mathbf{60.5} \pm 2.6$	31.7 ± 2.0
	medium	100.0	63.9 ± 0.7	60.1 ± 2.3	$\mathbf{66.3} \pm 0.7$	56.6 ± 1.2
	medium-expert	100.0	69.0 ± 6.4	$\mathbf{73.7} \pm 4.5$	$\mathbf{71.9} \pm 3.5$	53.8 ± 1.8

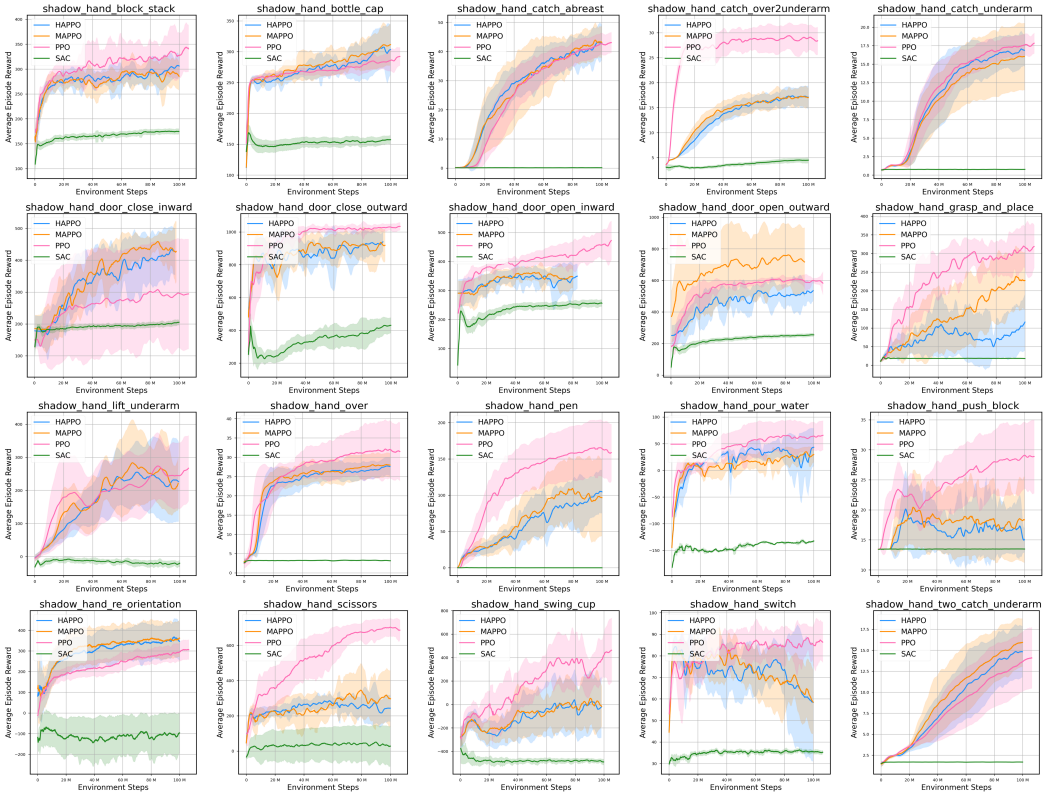


Figure 3: Learning curves for all 20 tasks. The shaded region represents the standard deviation of the score over 10 trials. Curves are smoothed uniformly for visual clarity. All algorithms interact with environments in 100M steps and the number of parallel simulations is 2048.

5.4 Generalization ability

The goals of our generalization evaluation is 1) to find out the ability of current multi-task and meta reinforcement learning algorithms to generalize on the tasks we designed. 2) to find out whether the tasks that are harder for babies are also harder for RL. The previous RL/MARL results have proved that our individual task is solvable. For goal 1), we evaluate the multi-task PPO [36] and ProMP [49] algorithms on MT1, ML1, MT4, ML4, MT20, and ML20. We also provided the results of random policy and using the PPO algorithm in individual task as the ground truth for comparison. The average reward for each training is shown in Table 4. We can observe that the multi-task PPO does not perform well, and the ProMP have tiny performance improvement compared with random policy. It may because it's hard to learn policy from individually each task itself in Bi-DexHands. Therefore, we still have a lot of room to improve the generalization ability of bimanual dexterous hands under cross-task setting, which is a meaningful open challenge for the community.

Table 4: The average reward of all tasks for MT1, ML1, MT4, ML4, MT20, and ML20 on 10 seeds.

Method	MT1	MT4	MT20	Method	ML1		ML4		ML20	
					train	test	train	test	train	test
Ground Truth	15.2	24,3	32.5	Ground Truth	15.0	15.8	28.0	13.1	33.7	26.1
Multi-task PPO	9.4	5.4	8.9	ProMP	0.95	1.2	2.5	0.5	0.02	0.36
Random	0.61	1.1	-2.5	Random	0.59	0.68	1.5	0.24	-2.9	0.27

For goal 2), we use random and ground truth reward to normalize the results of all tasks in MT20 and arrange them in the order of increasing age. The results is shown in Fig.4. It can be seen that in general, as the age of the person corresponding to the task increases, the difficulty for RL also increases, which proves that our task design is designed with rationality and relevance to people.

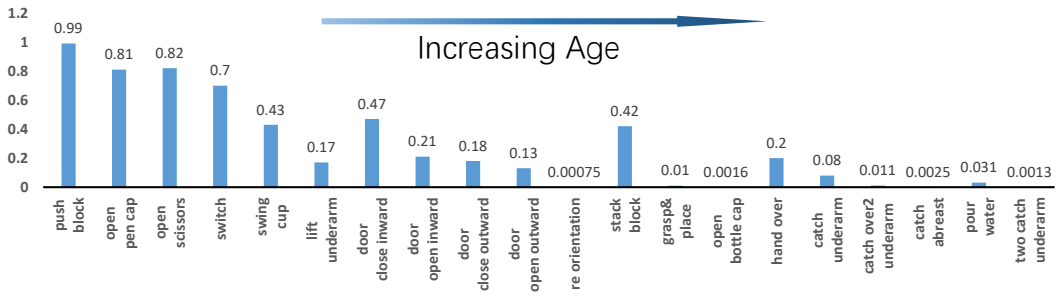


Figure 4: The normalized reward run by the MTPPO algorithm under the MT20 setting. The tasks from left to right according to the increase of corresponding age. The normalized score is computed by score $= \frac{\text{reward-random reward}}{\text{ground truth reward-random reward}}$.

6 Conclusion and Future Work

We introduced a benchmark, Bi-DexHands, which consists of well-designed tasks and a large variety of objects for learning bimanual dexterous manipulation. We investigated the motor development process of infants' dexterity from cognitive science, and carefully designed more than twenty tasks for RL based on the results, hoping that robots can learn dexterity like humans. With the help of the Isaac Gym simulator, it can run thousands of environments in parallel, improving the sample efficiency for RL algorithms. Moreover, the implemented RL/MARL/offline RL algorithms achieve superior performance on tasks with simple manipulation skills required. Meanwhile, complex manipulations still remain challenging. In particular, when the agent is trained to master multiple manipulation skills, the results of multi-task/Meta RL are not satisfactory. Interestingly, we found that under the multi-task setting, RL exhibited results associated with the development of human intelligence, that is, the trend of RL performance matches with the development of human ages. So far, in bimanual dexterous robot hand manipulation, the current reinforcement learning can reach the level of 48-months infants.

We identify four main future directions toward mastering human-level bimanual dexterous manipulation. **1)** Learning from demonstration: our platform needs some human teaching data to study learning from demonstration. **2)** Soft body and deformable objects simulation: we need a better physics engine to support our research on software and task design, to be more specific daily life scenes. **3)** Current meta/multi-task RL algorithms are unable to perform all tasks in our benchmark successfully, which calls for substantial further development on the algorithmic design end. **4)** We would like address the sim-to-real gap by transferring the simulation result on real dexterous hands. In particular, we hope our benchmark results can serve as a start point to help researchers transfer RL-learned skills to reality and help real-world robots to learn dexterous manipulation.

References

[1] Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.

[2] Leah Nellis and Betty E Gridley. Review of the bayley scales of infant development—second edition. *Journal of School Psychology*, 32(2):201–209, 1994.

[3] Gilbert Gottlieb. *Synthesizing nature-nurture: Prenatal roots of instinctive behavior*. Psychology press, 2014.

[4] Jeffrey J Lockman. A perception–action perspective on tool use development. *Child development*, 71(1):137–144, 2000.

[5] Sarah E Berger and Karen E Adolph. Learning and development in infant locomotion. *Progress in brain research*, 164:237–255, 2007.

[6] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[7] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[8] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[9] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[10] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[11] Mevlana C Gemici and Ashutosh Saxena. Learning haptic representation for manipulating deformable food objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 638–645. IEEE, 2014.

[12] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

[13] Jian Shi, J Zachary Woodruff, Paul B Umbanhowar, and Kevin M Lynch. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017.

[14] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[15] Rika Antonova, Silvia Cruciani, Christian Smith, and Danica Kragic. Reinforcement learning for pivoting task. *CoRR*, abs/1703.00472, 2017.

[16] J Zachary Woodruff and Kevin M Lynch. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4066–4073. IEEE, 2017.

[17] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In *NeurIPS Datasets and Benchmarks*, 2021.

[18] Nancy Bayley, Scales Infant, et al. Bayley scales of infant and toddler development–third edition: Technical manual. 2006.

[19] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[20] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[21] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[22] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[23] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

[24] Walter G Bircher, Aaron M Dollar, and Nicolas Rojas. A two-fingered robot gripper with large object reorientation range. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3453–3460. IEEE, 2017.

[25] Nahian Rahman, Luca Carbonari, Mariapaola D'Imperio, Carlo Canali, Darwin G Caldwell, and Ferdinando Cannella. A dexterous gripper for in-hand manipulation. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 377–382. IEEE, 2016.

[26] Ryuta Ozawa and Kenji Tahara. Grasp and dexterous manipulation of multi-fingered robotic hands: a review from a control view point. *Advanced Robotics*, 31(19-20):1030–1050, 2017.

[27] Nikolaus Vahrenkamp, Markus Przybylski, Tamim Asfour, and Rüdiger Dillmann. Bimanual grasp planning. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 493–499. IEEE, 2011.

[28] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V Dimarogonas, and Danica Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.

[29] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[30] Uikyum Kim, Dawoon Jung, Heeyoen Jeong, Jongwoo Park, Hyun-Mok Jung, Joono Cheong, Hyouk Ryeol Choi, Hyunmin Do, and Chanhun Park. Integrated linkage-driven dexterous anthropomorphic robotic hand. *Nature communications*, 12(1):1–13, 2021.

[31] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

[32] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[33] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[34] Henry J Charlesworth and Giovanni Montana. Solving challenging dexterous manipulation tasks with trajectory optimisation and reinforcement learning. In *International Conference on Machine Learning*, pages 1496–1506. PMLR, 2021.

[35] ShadowRobot. Shadowrobot dexterous hand. https://www.shadowrobot.com/dexterous-hand-series/, 2005.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[37] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018.

[38] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[39] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[40] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[41] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *CoRR*, abs/2109.11251, 2021.

[42] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[43] Christian Schröder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *CoRR*, abs/2011.09533, 2020.

[44] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[45] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning (ICML)*, 2019.

[46] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[47] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[48] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[49] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. In *ICLR (Poster)*. OpenReview.net, 2019.

[50] Marco Del Giudice and Jay Belsky. The development of life history strategies: Toward a multi-stage theory. *The evolution of personality and individual differences*, pages 154–176, 2011.

[51] Lawrence G Weiss, Thomas Oakland, and Glen P Aylward. *Bayley-III clinical use and interpretation*. Academic Press, 2010.

[52] FIRST WORDS Project. 16 actions with objects by 16 months, Feb. 16, 2018 [Online].

[53] Jennifer M Zubler, Lisa D Wiggins, Michelle M Macias, Toni M Whitaker, Judith S Shaw, Jane K Squires, Julie A Pajek, Rebecca B Wolf, Karnesha S Slaughter, Amber S Broughton, et al. Evidence-informed milestones for developmental surveillance tools. *Pediatrics*, 149(3), 2022.

[54] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[55] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

A Task Specifications

A.1 Physical parameters of Shadow hand

The limits of each joint in Shadow hand are as Table 5. The thumb has 5 degrees of freedom with 5 joints, the other fingers are all 3 degrees of freedom and 4 joints, and the joints at the ends of each finger are uncontrollable. The distal joints of the fingers are coupled like that of human fingers, making the angle of the middle joint always bigger or equal to the angle of the distal joint. This allows the middle phalange is curved, while the distal phalange is straight. There is an extra joint (LF5) at the end of the little finger to allow the little finger to rotate in the direction of the thumb. There are two joints at the wrist, which guarantees that the entire hand can rotate 360 degrees.

Table 5: Finger range of motion.

Joints	Corresponds to the number of 1	Min	Max
Finger Distal (FF1,MF1,RF1,LF1)	15, 11, 7, 3	0°	90°
Finger Middle (FF2,MF2,RF2,LF2)	16, 12, 8, 4	0°	90°
Finger Base Abduction (FF3,MF3,RF3,LF3)	17, 13, 9, 5	-15°	90°
Finger Base Lateral (FF4,MF4,RF4,LF4)	18, 14, 10, 6	-20°	20°
Little Finger Rotation(LF5)	19	0°	45°
Thumb Distal (TH1)	20	-15°	90°
Thumb Middle (TH2)	21	-30°	30°
Thumb Base Abduction (TH3)	22	-12°	12°
Thumb Base Lateral (TH4)	23	0°	70°
Thumb Base Rotation (TH5)	24	-60°	60°
Hand Wrist Abduction (WR1)	1	-40°	28°
Hand Wrist Lateral (WR2)	2	-28°	8°

Stiffness, damping, friction, and armature are also important physical parameters in robotics. For each Shadow hand's joint, we show our DoF properties in Table 6. This part can be adjusted in the Isaac Gym simulator.

Table 6: DoF properties of Shadow hand.

Joints	Stifness	Damping	Friction	Armature
WR1	100	4.78	0	0
WR2	100	2.17	0	0
FF2	100	3.4e+38	0	0
FF3	100	0.9	0	0
FF4	100	0.725	0	0
MF2	100	3.4e+38	0	0
MF3	100	0.9	0	0
MF4	100	0.725	0	0
RF2	100	3.4e+38	0	0
RF3	100	0.9	0	0
RF4	100	0.725	0	0
LF2	100	3.4e+38	0	0
LF3	100	0.9	0	0
LF4	100	0.725	0	0
TH2	100	3.4e+38	0	0
TH3	100	0.99	0	0
TH4	100	0.99	0	0
TH5	100	0.81	0	0

A.2 Detailed components of tasks

In this section, we detailed the components of tasks in Bi-DexHands. We refer to some designs of existing dexterous hand environments, integrate their advantages, and expand some new environments

and unique features for single/multi-agent reinforcement learning. Our environments focus on the application of RL algorithms to dexterous hand control, which is challenging in traditional control algorithms. The difficulty of our environment is not only reflected in the challenging task content but also reflected in the high-dimensional continuous space control. The state space dimension of each environment is up to 400 dimensions in total, and the action space dimension is up to 40 dimensions. A multi-agent feature of our environment is that we use five fingers and palms of each hand as a minimum agent unit. It is mean that you can use each finger and palm as an agent, or combine any number of them as an agent by yourself. All environments are goal-based, and each epoch will randomly reset the object's starting pose and target pose to improve generalization. All objects type can be selected in the config, the basis is egg, block, and pen. We also provide objects type in the YCB dataset as an extension, you can customize the object type they want to use.

An overview of our tasks is shown in Fig.5. Next, we will introduce the basic description, action space, observation space, and reward function of each task. We only use the shadow hand and object state values as observation at present, but we also provide an interface for using point cloud as observation in our Github repository for researchers to study in the future. The observation of all tasks is composed of three parts: the state values of the left and right hands, and the information of objects and target. The state values of the left and right hands were the same for each task, including hand joint and finger positions, velocity, and force information. The state values of the object and goal are different for each task, which we will describe in the following. Table.7 gives the specific information of the left-hand and right-hand state values. Note that the observation is slightly different in the HandOver task due to the fixed base.

Table 7: Observation space of dual shadow hands.

Index	Description
0 - 23	right shadow hand dof position
24 - 47	right shadow hand dof velocity
48 - 71	right shadow hand dof force
72 - 136	right shadow hand fingertip pose, linear velocity, angle velocity (5 x 13)
137 - 166	right shadow hand fingertip force, torque (5 x 6)
167 - 169	right shadow hand base position
170 - 172	right shadow hand base rotation
173 - 198	right shadow hand actions
199 - 222	left shadow hand dof position
223 - 246	left shadow hand dof velocity
247 - 270	left shadow hand dof force
271 - 335	left shadow hand fingertip pose, linear velocity, angle velocity (5 x 13)
336 - 365	left shadow hand fingertip force, torque (5 x 6)
366 - 368	left shadow hand base position
369 - 371	left shadow hand base rotation
372 - 397	left shadow hand actions

Under the multi-agent setting, the partial observation of each agent depends on the observation of the hand it belongs to. For example, if the left distal finger, left thumb, and right distal finger are one agent respectively, the observation of the left distal finger and left thumb are the observation of the entire left hand in the Table 7 plus the object and target information. The obs of the right distal finger is the observation of the entire right hand in the Table 7 plus object and target information. The action of each agent depends on the multi-agent setting (*i.e.*, fingers, hands,...), and the output by each agent is the joint degree of itself. Bi-DexHands is a fully-cooperative game where all agents have the same reward. Therefore, the setting of multi-agent can be completely inferred from the setting of single-agent.

A.2.1 Hand Over

This environment consists of two shadow hands with palms facing up, opposite each other, and an object that needs to be passed. In the beginning, the object will fall randomly in the area of the shadow hand on the right side. Then the hand holds the object and passes the object to the other hand. Note that the base of the hand is fixed. More importantly, the hand which holds the object

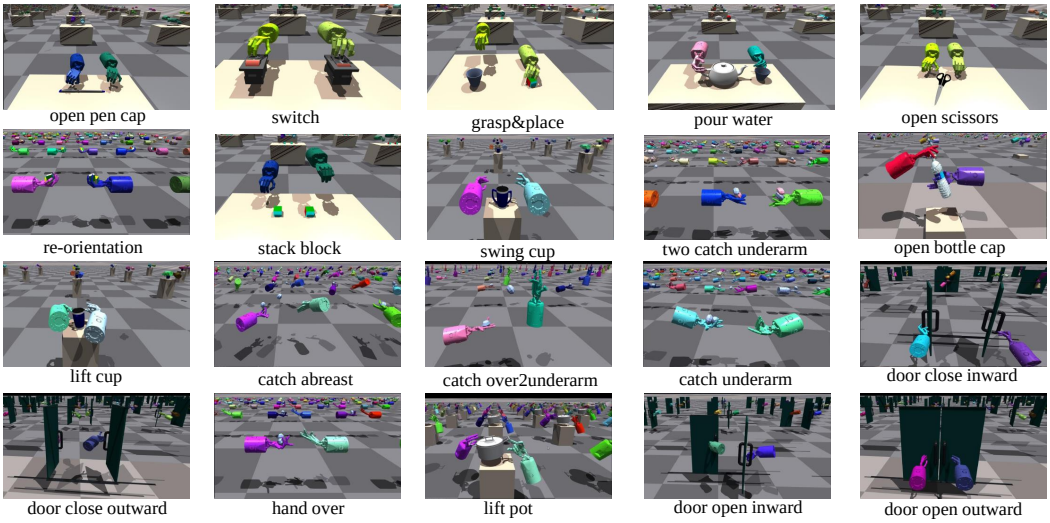


Figure 5: An overview of all tasks.

initially can not directly touch the target, nor can it directly roll the object to the other hand, so the object must be thrown up and stays in the air in the process. There are 398-dimensional observations and 40-dimensional actions in the task. Additionally, the reward function is related to the pose error between the object and the target. When the pose error gets smaller, the reward increases dramatically. Specifically, the observation space of each agent is detailed in the following Table 8, and the action space is shown in Table 9.

Observations The 398-dimensional observational space for Hand Over task is shown in Table 8. It should be noted that since the base of the dual hands in this task is fixed, the observation of the dual hands is compared to the Table 7 of reduced 24 dimensions.

Table 8: Observation space of Hand Over.

Index	Description
0 - 373	dual hands observation shown in Table 7
374 - 380	object pose
381 - 383	object linear velocity
384 - 386	object angle velocity
387 - 393	goal pose
394 - 397	goal rot - object rot

Actions The 40-dimensional action space for one hand in Hand Over task is shown in Table 9.

Table 9: Action space of Hand Over.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 39	left shadow hand actuated joint

Rewards Denote the object and goal position as x_o and x_g respectively. Then, the translational position difference between the object and the goal d_t is given by $d_t = \|x_o - x_g\|_2$. Denote the angular position difference between the object and the goal as d_a, then the rotational difference d_r is given by $d_r = 2 \arcsin clamp(\|d_a\|_2, max = 1.0)$. Finally, the rewards are given by the following specific formula:

$$r = exp[-0.2(\alpha d_t + d_r)] \tag{1}$$

where α is a constant balancing translational and rotational rewards.

A.2.2 Catch Underarm

In this problem, two shadow hands with palms facing upwards are controlled to pass an object from one palm to the other. What makes it more difficult than the Handover problem is that the hands' translation and rotation degrees of freedom are no longer frozen but are added into the action space.

Observations The 422-dimensional observational space as shown in Table 10.

Table 10: Observation space of Catch Underarm.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot

Actions The 52-dimensional action space as shown in Table 11.

Table 11: Action space of Catch Underarm.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards Denote the object and goal position as x_o and x_g respectively. Then, the translational position difference between the object and the goal d_t is given by $d_t = \|x_o - x_g\|_2$. Denote the angular position difference between the object and the goal as d_a, then the rotational difference d_r is given by $d_r = 2 \arcsin clamp(\|d_a\|_2, max = 1.0)$. Finally, the rewards are given by the following specific formula:

$$r = exp[-0.2(\alpha d_t + d_r)] \tag{2}$$

where α is a constant balancing translational and rotational rewards.

A.2.3 Catch Over2Underarm

This environment is like made up of half Hand Over and Catch Underarm, the object needs to be thrown from the vertical hand to the palm-up hand.

Observations The 422-dimensional observational space as shown in Table 12.

Table 12: Observation space of Catch Over2Underarm.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot

Actions The 52-dimensional action space as shown in Table 13.

Table 13: Action space of Catch Over2Underarm.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards Denote the object and goal position as x_o and x_g respectively. Then, the translational position difference between the object and the goal d_t is given by $d_t = \|x_o - x_g\|_2$. Denote the angular position difference between the object and the goal as d_a, then the rotational difference d_r is given by $d_r = 2 \arcsin clamp(\|d_a\|_2, max = 1.0)$. Finally, the rewards are given by the following specific formula:

$$r = exp[-0.2(\alpha d_t + d_r)] \tag{3}$$

where α is a constant balancing translational and rotational rewards.

A.2.4 Two Catch Underarm

This environment is similar to Catch Underarm, but with an object in each hand and the corresponding goal on the other hand. Therefore, the environment requires two objects to be thrown into the other hand at the same time, which requires a higher manipulation technique than the environment of a single object.

Observations The 446-dimensional observational space as shown in Table 14.

Table 14: Observation space of Two Catch Underarm.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object1 pose
405 - 407	object1 linear velocity
408 - 410	object1 angle velocity
411 - 417	goal1 pose
418 - 421	goal1 rot - object rot
422 - 428	object2 pose
429 - 431	object2 linear velocity
432 - 434	object2 angle velocity
435 - 441	goal2 pose
442 - 445	goal2 rot - object2 rot

Actions The 52-dimensional action space as shown in Table 15.

Table 15: Action space of Two Catch Underarm.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards For the reward part, we use subscripts 1,2 to distinguish the 2 objects.

Denote the object and goal position as x_{o_1},x_{o_2} and x_{g_1},x_{g_2} respectively. Then, the translational position difference between the object and the goal d_{t_1},d_{t_2} is given by $d_{t_i} = \|x_{o_i} - x_{g_i}\|_2$, where $i = 1, 2$. Denote the angular position difference between the object and the goal as d_{a_1},d_{a_2}, then the rotational difference d_{r_1},d_{r_2} is given by $d_{r_i} = 2 \arcsin clamp(\|d_{a_i}\|_2, max = 1.0)$. Finally, the rewards are given by the following specific formula:

$$r = exp[-0.2(\alpha d_{t_1} + d_{r_1})] + exp[-0.2(\alpha d_{t_2} + d_{r_2})] \tag{4}$$

where α is a constant balancing translational and rotational rewards.

A.2.5 Catch Abreast

This environment consists of two shadow hands placed side by side in the same direction and an object that needs to be passed. Compared with the previous environment which is more like passing objects between the hands of two people, this environment is designed to simulate the two hands of the same person passing objects, so different catch techniques are also required and require more hand translation and rotation techniques.

Observations The 422-dimensional observation space as shown in Table 16.

Table 16: Observation space of Catch Abreast.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot

Actions The 52-dimensional action space as shown in Table 17.

Table 17: Action space of Catch Abreast.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards Denote the object and goal position as x_o and x_g respectively. Then, the translational position difference between the object and the goal d_t is given by $d_t = \|x_o - x_g\|_2$. Denote the angular position difference between the object and the goal as d_a, then the rotational difference d_r is given by $d_r = 2 \arcsin clamp(\|d_a\|_2, max = 1.0)$. Finally, the rewards are given by the following specific formula:

$$r = exp[-0.2(\alpha d_t + d_r)] \tag{5}$$

where α is a constant balancing translational and rotational rewards.

A.2.6 Lift Underarm

This environment requires grasping the pot handle with two hands and lifting the pot to the designated position. This environment is designed to simulate the scene of lift in daily life and is a practical skill.

Observations The 428-dimensional observation space as shown in Table 18.

Table 18: Observation space of Lift Underarm.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	object right handle position
425 - 427	object left handle position

Table 19: Action space of Lift Underarm.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Actions The 40-dimensional action space as shown in Table 19.

Rewards The reward consists of three parts: the distance from the left hand to the left handle, the distance from the right hand to the right handle, and the distance from the object to the target point. The position difference between the object to the target point d_{target} is given by $d_{target} = \|x_{obj} - x_{goal}\|_2$. The position difference between the left hand to the left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$.The position difference between the right hand to the right handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. The reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 3 * d_{target} \tag{6}$$

A.2.7 Door Open Outward/Door Close Inward

These two environments require a closed/opened door to be opened/closed and the door can only be pushed outward or initially open inward. Both these two environments only need to do the push behavior, so it is relatively simple.

Observations The 428-dimensional observation space as shown in Table 20.

Table 20: observation space of Door Open Outward/Door Close Inward.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	door right handle position
425 - 427	door left handle position

Actions The 52-dimensional action space as shown in Table 21.

Rewards The reward consists of three parts: the distance from the left hand to the left handle, the distance from the right hand to the right handle, and the distance between the two handles. The

Table 21: Action space of Door Open Outward/Door Close Inward.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

distance between the two handles d_{target} is given by $d_{target} = \|x_{lhandle} - x_{rhandle}\|_2$. The position difference between the left hand to the left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$.The position difference between the right hand to the right handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. For DoorOpenOutward, the reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 2 * d_{target} \tag{7}$$

For DoorCloseInward, the reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 2 * (1 - d_{target}) \tag{8}$$

A.2.8 Door Open Inward/Door Close Outward

These two environments also require a closed/opened door to be opened/closed and the door can only be pushed inward or initially open outward, but because they can't complete the task by simply pushing, which need to catch the handle by hand and then open or close it, so it is relatively difficult.

Observations The 428-dimensional observation space as shown in Table 22.

Table 22: Observation space of Door Open Inward/Door Close Outward.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object pose
405 - 407	object linear velocity
408 - 410	object angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	door right handle position
425 - 427	door left handle position

Actions The 52-dimensional action space as shown in Table 23.

Table 23: Action space of Door Open Inward/Door Close Outward.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards The reward consists of three parts: the distance from the left hand to the left handle, the distance from the right hand to the right handle, and the distance between the two handles. The distance between the two handles d_{target} is given by $d_{target} = \|x_{lhandle} - x_{rhandle}\|_2$. The position difference between the left hand to the left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$.The

position difference between the right hand to the right handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. For DoorOpenInward, the reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 2 * d_{target} \tag{9}$$

For DoorCloseOutward, the reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 2 * (1 - d_{target}) \tag{10}$$

A.2.9 Bottle Cap

This environment involves two hands and a bottle, we need to hold the bottle with one hand and open the bottle cap with the other hand. This skill requires the cooperation of two hands to ensure that the cap does not fall.

Observations The 414-dimensional observation space as shown in Table 24.

Table 24: Observation space of Bottle Cap.

0 - 397	dual hands observation shown in Table 7
398 - 404	bottle pose
405 - 407	bottle linear velocity
408 - 410	bottle angle velocity
411 - 413	bottle cap position

Actions The 52-dimensional action space as shown in Table 25.

Table 25: Action space of Bottle Cap.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards The reward also consists of three parts: the distance from the left hand to the bottle cap, the distance from the right hand to the bottle, and the distance between the bottle and bottle cap. The distance between the bottle and bottle cap d_{target} is given by $d_{target} = \|x_{bottle} - x_{bottlecap}\|_2$. the distance from the left hand to the bottle cap d_{left} is given by $d_{left} = \|x_{lhand} - x_{bottlecap}\|_2$. the distance from the right hand to the bottle d_{right} is given by $d_{right} = \|x_{rhand} - x_{bottle}\|_2$. The reward is given by this specific formula:

$$r = 0.2 - d_{left} - d_{right} + 30 * d_{target} \tag{11}$$

A.2.10 Push Block

This environment involves two hands and two blocks, we need to use both hands to reach and push the block to the desired goal separately. This is a relatively simple task.

Observations The 417-dimensional observation space as shown in Table 26.

Actions The 52-dimensional action space as shown in Table 27.

Rewards The reward consists of three parts: the distance from the left hand to block1, the distance from the right hand to block2, and the distance between the block and desired goal. The distance between the block and desired goal d_{target} is given by $d_{target} = \|x_{block1} - x_{block1_goal}\|_2 + \|x_{block2} - x_{block2_goal}\|_2$. the distance from the left hand to the block1 d_{left} is given by $d_{left} = \|x_{lhand} - x_{block1}\|_2$. the distance from the right hand to the block2 d_{right} is given by $d_{right} = \|x_{rhand} - x_{block2}\|_2$. The reward is given by this specific formula:

$$r = 2 - d_{left} - d_{right} + 5 * (0.8 - d_{target}) \tag{12}$$

Table 26: Observation space of Push Block.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	block1 pose
405 - 407	block1 linear velocity
408 - 410	block1 angle velocity
411 - 413	block1 position
414 - 416	block2 position

Table 27: Action space of Push Block.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

A.2.11 Swing Cup

This environment involves two hands and a dual handle cup, we need to use two hands to hold and swing the cup together.

Observations The 428-dimensional observation space as shown in Table 28.

Table 28: Observation space of Swing Cup.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	cup pose
405 - 407	cup linear velocity
408 - 410	cup angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	cup right handle position
425 - 427	cup left handle position

Actions The 52-dimensional action space as shown in Table 29.

Rewards The reward consists of three parts: the distance from the left hand to the cup's left handle, the distance from the right hand to the cup's right handle, and the rotating distance between the cup and desired goal. The rotate distance between the cup and desired goal d_{target} is given by $d_{target} = 2 * \arcsin q_{cup} * q_{target}$. the distance from the left hand to the cup left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$. the distance from the right hand to the cup right handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. The reward is given by this specific formula:

$$r = -d_{left} - d_{right} + 1/(abs(d_{target}) + 0.1) * 5 - 1 \tag{13}$$

A.2.12 Open Scissors

This environment involves two hands and scissors, we need to use two hands to open the scissors.

Observations The 428-dimensional observation space as shown in Table 30.

Actions The 52-dimensional action space as shown in Table 31.

Table 29: Action space of Swing Cup.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Table 30: Observation space of Open Scissors.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	scissors pose
405 - 407	scissors linear velocity
408 - 410	scissors angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	scissors right handle position
425 - 427	scissors left handle position

Rewards The reward consists of three parts: the distance from the left hand to the scissors' left handle, the distance from the right hand to the scissors' right handle, and the target angle at which the scissors need to be opened. The distance between the scissors dof angle and target dof angle d_{target} is given by $d_{target} = \|x_{scissorsdof} - x_{targetdof}\|$. the distance from the left hand to the scissors left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$. the distance from the right hand to the scissors left handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. The reward is given by this specific formula:

$$r = 2 - d_{left} - d_{right} + (0.59 - d_{target}) * 5 \tag{14}$$

A.2.13 Re Orientation

This environment involves two hands and two objects. Each hand holds an object and we need to reorient the object to the target orientation.

Observations The 446-dimensional observation space as shown in Table 32.

Actions The 52-dimensional action space as shown in Table 33.

Rewards The reward consists of three parts: the distance from the left object to the left object goal, the distance from the right object to the right object goal, and the distance between the object and desired goal. The distance between the object and desired goal d_{target} is given by $d_{target} = 2 * \arcsin q_{object1} * q_{target} + 2 * \arcsin q_{object2} * q_{target}$. the distance from the left hand to the scissors left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{lhandle}\|_2$. the distance from the right hand to the scissors left handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{rhandle}\|_2$. The reward is

Table 31: Action space of Open Scissors.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Table 32: Observation space of Re Orientation.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	object1 pose
405 - 407	object1 linear velocity
408 - 410	object1 angle velocity
411 - 417	goal1 pose
418 - 421	goal1 rot - object rot
422 - 428	object2 pose
429 - 431	object2 linear velocity
432 - 434	object2 angle velocity
435 - 441	goal2 pose
442 - 445	goal2 rot - object2 rot

Table 33: Action space of Re Orientation.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

given by this specific formula:

$$r = d_{left} * -10 + d_{right} * -10 + d_{target} * 1.5 \tag{15}$$

A.2.14 Open Pen Cap

This environment involves two hands and a pen, we need to use two hand to open the pen cap.

Observations The 428-dimensional observation space as shown in Table 34.

Table 34: Observation space of Open Pen Cap.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	pen pose
405 - 407	pen linear velocity
408 - 410	pen angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	pen body position
425 - 427	pen cap position

Actions The 52-dimensional action space as shown in Table 35.

Rewards The reward consists of three parts: the distance from the left hand to the pen body, the distance from the right hand to the pen cap, and the distance between the pen body and pen cap. The distance between the pen body and pen cap d_{target} is given by $d_{target} = \|x_{penbody} - x_{pencap}\|$. the distance from the left hand to the scissors left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{penbody}\|_2$. the distance from the right hand to the scissors left handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{pencap}\|_2$. The reward is given by this specific formula:

$$r = exp(-10 * d_{left}) + exp(-10 * d_{right}) + d_{target} * 5 - 0.8 \tag{16}$$

Table 35: Action space of Open Pen Cap.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

A.2.15 Switch

This environment involves dual hands and a bottle, we need to use dual hand fingers to press the desired button.

Observations The 428-dimensional observation space as shown in Table 36.

Table 36: Observation space of Switch.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	switch1 pose
405 - 407	switch1 linear velocity
408 - 410	switch1 angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	switch1 position
425 - 427	switch2 position

Actions The 52-dimensional action space as shown in Table 37.

Table 37: Action space of Switch.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Rewards The reward consists of three parts: the distance from the left hand to the left switch, the distance from the right hand to the right switch, and the distance between the button and button's desired goal. The distance between the button and the button's desired goal d_{target} is given by $d_{target} = \|x_{button1} - x_{target1}\|_2 + \|x_{button2} - x_{target2}\|_2$. the distance from the left hand to the scissors left handle d_{left} is given by $d_{left} = \|x_{lhand} - x_{switch1}\|_2$. the distance from the right hand to the scissors left handle d_{right} is given by $d_{right} = \|x_{rhand} - x_{switch2}\|_2$. The reward is given by this specific formula:

$$r = 2 - d_{left} - d_{right} + (1.4 - d_{target}) * 50 \tag{17}$$

A.2.16 Stack Block

This environment involves dual hands and two blocks, and we need to stack the block as a tower.

Observations The 428-dimensional observation space as shown in Table 38.

Table 38: Observation space of Stack Block.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	block1 pose
405 - 407	block1 linear velocity
408 - 410	block1 angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	block1 position
425 - 427	block2 position

Table 39: Action space of Stack Block.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

Actions The 52-dimensional action space as shown in Table 39.

Rewards The reward consists of three parts: the distance from the left hand to block1, the distance from the right hand to block2, and the distance between the block and desired goal. The distance between the block and desired goal d_{target} is given by $d_{target} = \|x_{block1} - x_{target1}\|_2 + \|x_{block2} - x_{target2}\|_2$. the distance from the left hand to the block1 d_{left} is given by $d_{left} = \|x_{lhand} - x_{block1}\|_2$. the distance from the right hand to the block2 d_{right} is given by $d_{right} = \|x_{rhand} - x_{block2}\|_2$. The reward is given by this specific formula:

$$r = 1.5 - d_{left} - d_{right} + (0.24 - d_{target}) * 2 \tag{18}$$

A.2.17 Pour Water

This environment involves two hands and a bottle, we need to Hold the kettle with one hand and the bucket with the other hand, and pour the water from the kettle into the bucket. In the practice task in Isaac Gym, we use many small balls to simulate the water.

Observations The 428-dimensional observation space as shown in Table 40.

Table 40: Observation space of Pour Water.

Index	Description
0 - 397	dual hands observation shown in Table 7
398 - 404	kettle pose
405 - 407	kettle linear velocity
408 - 410	kettle angle velocity
411 - 417	goal pose
418 - 421	goal rot - object rot
422 - 424	kettle handle position
425 - 427	bucket position

Actions The 52-dimensional action space as shown in Table 41.

Rewards The reward consists of three parts: the distance from the left hand to the bucket, the distance from the right hand to the kettle, and the distance between the kettle spout and desired goal.

Table 41: Action space of Pour Water.

Index	Description
0 - 19	right shadow hand actuated joint
20 - 22	right shadow hand base translation
23 - 25	right shadow hand base rotation
26 - 45	left shadow hand actuated joint
46 - 48	left shadow hand base translation
49 - 51	left shadow hand base rotation

The distance between the kettle spout and desired goal d_{target} is given by $d_{target} = \|x_{spout} - x_{goal}\|_2$. the distance from the left hand to the bucket d_{left} is given by $d_{left} = \|x_{lhand} - x_{bucket}\|_2$. the distance from the right hand to the kettle d_{right} is given by $d_{right} = \|x_{rhand} - x_{kettle}\|_2$. The reward is given by this specific formula:

$$r = 1 - d_{left} - d_{right} + (0.5 - d_{target}) * 2 \tag{19}$$

A.3 Offline Data Collection

We follow the data collection of D4RL[55] mujoco tasks. The medium dataset is generated by first training a policy online using PPO, early-stopping the training, and collecting 10^6 samples (s_t, a_t, s_{t+1}, r_t) using this medium policy. The random dataset is collected by a randomly initialized policy and contains 10^6 samples. The replay dataset consists of 10^6 experienced samples during training of the medium policy. The medium-expert dataset contains 2×10^6 samples by mixing equal amounts of samples collected by expert policy and medium policy. To facilitate comparison across tasks, following the setting of D4RL[55], we normalize scores for each task to the range between 0 and 100 , by computing normalized score $= 100 * \frac{\text{return-random return}}{\text{expert return-random return}}$. A normalized score of 0 corresponds to the average return of an agent taking actions uniformly at random across the action space. A score of 100 corresponds to the average return of an expert policy.

B Training details

Isaac Gym is different from other simulators in that it can simulate completely on the GPU, so there is no need to exchange data between the GPU and the CPU during the training process. Therefore we reproduced the existing RL algorithm in our Github repository to accommodate this feature. We implemented many different algorithms in the comprehensive RL domain, but only evaluated some of them. We will give a brief introduction to these algorithms below and give the hyperparameters of the algorithms we used in our evaluation.

B.1 Single-agent algorithms

B.1.1 Trust Region Policy Optimization

TRPO is a basic policy optimization algorithm, with theoretically justified monotonic improvement. Based on the theorem1 in the original paper by John Schulman et. al. $\eta(\pi_{new}) \geq L_{old}(\pi_{new}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$,where$\epsilon = \max_{s,a} |A_\pi(s,a)|$,$\eta$ is the objective function and L_{old} is a surrogate objective: $L_\pi(\hat{\pi}) = \eta(\pi) + E_{s\sim\rho_\pi, a\sim\pi}(A_\pi(s,a))$, providing feasible approximation of η according to the theorem. To empirically allow for larger update steps, the optimization problem is adjusted to $\pi_{\theta_{new}} = \max_\theta L_{\theta_{old}}(\theta)$ subject to $D_{KL}^{max}(\theta_{old}, \theta) \leq \delta$. To yield a practical algorithm, TRPO makes a bit of approximation like optimizing with conjugate gradient method followed by a line search.

B.1.2 Proximal Policy Optimization

PPO is a policy optimization algorithm enjoying simpler implementation, more general application and better sample complexity over TRPO. Based on the surrogate objective in TRPO: $L^{CPI}(\theta) = \hat{E}_t[r_t(\theta)\hat{A}_t]$, PPO proposed a new approximate surrogate function $L^{CLIP}(\theta) =$

$\hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$, which restricts policy optimization step by removing the incentive for r_t to move outside of the interval $[1-\epsilon, 1+\epsilon]$. Another alternative surrogate objective is given by incorporating a penalty on KL divergence, and adapting the penalty coefficient. During traning, PPO uses a combined objective, consisting of surrogate objective for the policy, value function loss for the critic and a bonus entropy term: $L^{CLIP+VF+S}(\theta) = \hat{E}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$.

Table 42: Hyperparameters of PPO.

Hyperparameters	Other Tasks	Lift Underarm	Stack Block
Num mini-batches	4	4	8
Num opt-epochs	5	10	2
Num episode-length	8	20	8
Hidden size	[1024, 1024, 512]	[1024, 1024, 512]	[1024, 1024, 512]
Clip range	0.2	0.2	0.2
Max grad norm	1	1	1
Learning rate	3.e-4	3.e-4	3.e-4
Discount (γ)	0.96	0.96	0.9
GAE lambda (λ)	0.95	0.95	0.95
Init noise std	0.8	0.8	0.8
Desired kl	0.016	0.016	0.016
Ent-coef	0	0	0

B.1.3 Deep Deterministic Policy Gradient

DDPG, based on the DPG algorithm, is a model-free, off-policy actor-critic algorithm using deep function approximators that can learn policies in high-dimensional, continuous action spaces. It uses a copy of the actor and critic networks $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ to calculate the target values, and use "soft" target updates to update the target networks more stably by having them slowly track the learned networks: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$. It follows an exploration policy μ' by adding noise sampled from a noise process $\mathbf{N}$: $\mu'(S_t) = \mu(s_t|\theta_t^\mu) + \mathbf{N}_t$. The critic is updated by minimizing the loss: $L(\phi) = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$ where $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$, and the actor is updated using sampled policy gradient: $\nabla_\theta J \approx \frac{1}{N}\sum_i \nabla_\alpha Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta\mu}\mu(s|\theta^\mu)|_{s_i}$.

B.1.4 Twin Delayed Deep Deterministic policy gradient

TD3 is an actor-critic algorithm which applies its modifications to the state of the art actor-critic method for continuous control, DDPG. It focused on two outcomes that occur as the result of estimation error, overestimation bias and a high variance build-up. It uses Clipped Double Q-learning method to reduce overestimation bias: $y \leftarrow r + \gamma min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$, where $\tilde{a} \leftarrow \pi_{\phi'}(s) + \epsilon, \epsilon \sim clip(\mathbf{N}(0, \tilde{\sigma}), -c, c)$, which uses target policy smoothing regularization to avoid overfitting and enforce the value similarity between similar actions, It uses delayed policy and target network updates to ensure small value error.

B.1.5 Soft Actor-Critic

SAC is an off-policy maximum entropy actor-critic algorithm. It considers a more general maximum entropy objective: $J(\pi) = \sum_{t=0}^{T} \mathbf{E}_{(s_t,a_t)\sim\mathbf{D}_\pi}[r(s_t, a_t) + \alpha\mathbf{H}(\pi(\cdot|s_t))]$, in which the temperature parameter α determines the relative importance of the entropy term. The soft value function $V_\psi(s_t)$ is trained to minimize the squared residual error: $L_v(\psi) = \mathbf{E}_{s_t\sim\rho}[\frac{1}{2}(V_\psi(s_t) - \mathbf{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t, a_t) - \log\pi_\phi(a_t|s_t)])^2]$. The soft Q-function parameters can be trained to minimize the soft Bellman residual: $L_Q(\theta) = \mathbf{E}_{(s_t,a_t)\sim\mathbf{D}}[\frac{1}{2}(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t))^2]$, in which $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbf{E}_{s_{t+1}\sim p}[V_{\bar{\phi}}(s_{t+1})]$ and $V_{\bar{\phi}}$ is the target value network. The policy parameters can be learned by directly minimizing the expected KL-divergence: $KL_\pi(\phi) = \mathbf{E}_{s_t\sim\rho}[D_{KL}(\pi_\phi(\cdot|s_t)||\frac{exp(Q_\theta(s_t,\cdot))}{Z_\theta(s_t)})]$, in which $Z_\theta(s_t)$ normalizes the distribution.

Table 43: Hyperparameters of SAC.

Hyperparameters	Other Tasks	Lift Underarm	Stack Block
Num opt-epochs	1	1	1
Num. mini-batches	4	4	4
Hidden size	[1024, 1024, 1024]	[1024, 1024, 1024]	[1024, 1024, 1024]
Learning rate	3.e-4	3.e-4	3.e-4
ReplayBuffer size	5000	5000	5000
Discount (γ)	0.96	0.96	0.96
Polyak ($1-\tau$)	0.99	0.99	0.99
Entropy coef	0.2	0.2	0.2
Reward scale	1	1	1
Max grad norm	1	1	1
Batch size	32	32	32

B.2 Multi-agent algorithms

B.2.1 Independent Proximal Policy Optimization

IPPO (Independent PPO) is a multi-agent variant of proximal policy optimization(PPO). It uses PPO to learn decentralized policies π^i for agents with indicidual policy clippng based on the objective: $\mathbf{L}^i(\theta) = \mathbf{E}_{s_t^i, a_t^i}[min(\frac{\pi_\theta(a_t^i|s_t^i)}{\pi_{\theta_{old}}(a_t^i|s_t^i)}A_t^i, clip(\frac{\pi_\theta(a_t^i|s_t^i)}{\pi_{\theta_{old}}(a_t^i|s_t^i)}, 1-\epsilon, 1+\epsilon)A_t^i)]$, and the advantage function is based on independent learning, where each agent a learns a local observation based critic $V_\phi(z_t^i)$ parameterised by ϕ using GAE. Additionally, it uses value clipping to restrict the update of critic function for each agent i: $\mathbf{L}^i(\phi) = \mathbf{E}_{z_t^i}[min\{(V_\phi(z_t^i) - \hat{V_t^i})^2, (V_{\phi_{old}}(z_t^i) + clip(V_\phi(z_t^i) - V_{\phi_{old}}(z_t^i), -\epsilon, +\epsilon) - \hat{V_t^i})^2\}]$. The overall learning loss additionally adds an entropy regularization term of policy π^i.

B.2.2 Heterogenous-Agent Trust Region Policy Optimization

HATRPO is a multi-agent algorithm developed from TRPO. With the advantage decomposition lemma, the algorithm is proposed to implement a multi-agent policy iteration procedure with monotonic improvement guarantee. It requires no homogeneity of agents, nor any restrictive assumptions on the decomposibility of joint Q-functions. At each iteration k+1, given a random permutation of agents $i_{1:n}$, agent i_m sequentially optimizes its own policy parameter $\theta_{k+1}^{i_m}$ by maximizing the objective: $\theta_{k+1}^{i_m} = argmax_{\theta^{i_m}} \mathbf{E}_{s\sim\rho_{\theta_k}, a^{i_{1:m-1}}\sim\pi_{\theta_{k+1}^{i_{1:m-1}}}^{i_{1:m-1}}, a^{i_m}\sim\pi_{\theta^{i_m}}^{i_m}}[A_{\pi_{\theta_k}}^{i_m}(s, a^{i_{1:m-1}}, a^{i_m})]$, subject to $\mathbf{E}_{s\sim\rho_{\theta_k}}[D_{KL}(\pi_{\theta_k^{i_m}}^{i_m}(\cdot|s), \pi_{\theta^{i_m}}^{i_m}(\cdot|s))] \leq \delta$. Apply a linear approximation to the objective function and a quadratic approximation to the KL constraint: $\theta_{k+1}^{i_m} = \theta_k^{i_m} + \alpha^j \sqrt{\frac{2\delta}{g_k^{i_m}(H_k^{i_m})^{-1}g_k^{i_m}}}(H_k^{i_m})^{-1}g_k^{i_m}$, in which $H_k^{i_m}$ is the Hessian of the expected KL-divergence, $g_k^{i_m}$ is the gradient of the objective function, and $\alpha^j < 1$ is a positive coefficient. Estimate the advantage function $\mathbf{E}[A_{\pi_{\theta_k}}^{i_m}(s, a^{i_{1:m-1}}, a_{i_m})]$ with $(\frac{\pi_\theta^{i_m}(a_{i_m}|s)}{\pi_{\theta_k}^{i_m}(a_{i_m}|s)} - 1)M^{i_{1:m}}(s, a)$, where $M^{i_{1:m}} = \frac{\bar{\pi}^{i_{1:m-1}}}{\pi_{i_{1:m-1}}}\hat{A}(s, a)$ and $\bar{\pi}^{i_{1:m-1}} = \prod_{j=1}^{m-1}\bar{\pi}^{i_j}$ is the policies of agents $i_{1:m-1}$ just updated in the same iteration k+1.

B.2.3 Heterogeneous-Agent Proximal Policy Optimisation

HAPPO is a multi-agent policy optimization algorithm that follows the centralized training decentralized execution (CTDE) paradigm. HAPPO doesn't assume homogeneous agents and doesn't require decomposibility of the joint value function. The theoretical core of extending PPO to multi-agent settings is the advantage decomposition lemma(Lemma 1 in the original paper). As a result of it, similar to single agent PPO, we have a theoretical monotonic improvement guarantee for the multi-agent setting: $J(\bar{\pi}) \geq J(\pi) + \sigma_{m=1}^n[L_\pi^{i_{1:m}}(\bar{\pi}^{i_{1:m-1}}, \bar{\pi}^{i_m}) - CD_{KL}^{max}(\pi^{i_m}, \bar{\pi}^{i_m})]$(Lemma 2 in the original paper). This lemma yields a similar policy optimization iteration: $\pi_{k+1}^{i_m} =$

$arg\max_{\pi^{i_m}}[L_{\pi}^{i_{1:m}}(\pi^{i_{1:m-1}}, \pi^{i_m}) - CD_{KL}^{max}(\pi^{i_m}, \pi^{i_m})]$. To avoid maintaining value functions for each single agent, the following proposition is used: $E[A_{\pi}^{i_m}(s, a^{i_{1:m-1}}, a^{i_m})] = E[(\frac{\hat{\pi}^{i_m}(a^{i_m}|s)}{\pi^{i_m}(a^{i_m}|s)} - 1)\frac{\overline{\pi}^{i_{1:m-1}}(a^{i_{1:m-1}}|s)}{\pi^{i_{1:m-1}}(a^{i_{1:m-1}}|s)}A_{\pi}(s, a)])$, so that it only need to keep one value function $A_{\pi}(s, a)$ for all agents.Finally, it uses the clipping trick similar to single agent PPO, obtaining the final practical algorithm, for details, please refer to (11) in the original paper.

Table 44: Hyperparameters of HAPPO.

Hyperparameters	Other Tasks	Lift Underarm	Stack Block
Num mini-batches	1	1	1
Num opt-epochs	5	10	5
Num episode-length	8	20	8
Hidden size	[1024, 1024, 512]	[1024, 1024, 512]	[1024, 1024, 512]
Use popart	True	True	True
Use value norm	True	True	True
Use proper time limits	False	False	False
Use huber loss	True	True	True
Huber delta	10	10	10
Replay Size	10000	10000	10000
Polyak	0.995	0.995	0.995
Reward scale	1	1	1
Clip range	0.2	0.2	0.2
Max grad norm	1	1	1
Learning rate	1.e-4	1.e-4	1.e-4
Discount (γ)	0.96	0.96	0.96
GAE lambda (λ)	0.95	0.95	0.95
Init noise std	1	1	1
Ent-coef	0	0	0

B.2.4 Multi-Agent Proximal Policy Optimization

MAPPO (Multi-Agent PPO) is an application of the actor-critic single-agent PPO algorithm to multi-agent tasks. It follows the CTDE structure. Each agent i follows a shared policy $\pi_{\theta}(a_i|o_i)$ based on local observation $o_i = O(s; i)$ at global state s, takes its action a_i and optimizes its reward $J(\theta) = E_{a^t,s^t}[\sum_t \gamma^t R(s^t, a^t)]$. The actor network maximizes: $L(\theta) = [\frac{1}{Bn}\sum_{i=1}^{B}\sum_{k=1}^{n} min(r_{\theta,i}^{(k)}A_i^{(k)}, clip(r_{\theta,i}^{(k)}, 1 - \epsilon, 1 + \epsilon)A_i^{(k)})] + \sigma\frac{1}{Bn}\sum_{i=1}^{B}\sum_{k=1}^{n} S[\pi_{\theta}(o_i^{(k)})]$, where n refers to the agent number, $A_I^{(k)}$ is computed using GAE method, S is policy entropy and σ is entropy coefficient hyper-parameter. The critic network minimizes: $L(\phi) = \frac{1}{Bn}\sum_{i=1}^{B}\sum_{k=1}^{n}(max[(V_{\phi}(s_i^{(k)}) - \hat{R}_i)^2, (clip(V_{\phi}(s_i^{(k)}), V_{\phi_{old}}(s_i^{(k)}) - \epsilon, V_{\phi_{old}}(s_i^{(k)}) + \epsilon) - \hat{R}_i)^2])$, where $\hat{R}_i$ is reward-to-go.

B.2.5 Multi-Agent Deep Deterministic Policy Gradient

MADDPG (Multi-Agent DDPG) is an actor-critic deep policy gradient algorithm solving multi-agent tasks. Based on DDPG, it uses CTDE structure, in which the critic uses global information to optimize Q-function while training and the actor uses local observation to take actions while testing. For each agent i, update the critic by minimizing the loss function: $L^i(\phi) = \frac{1}{S}\sum_j (y^j - Q_i^{\mu}(\mathbf{x}^j, a_1^j, \ldots, a_N^j))^2$, where $y^j = r_i^j + \gamma Q_i^{\mu'}(\mathbf{x}'^j, a_1', \ldots, a_N')|_{a_k'=\mu_k'(\sigma_k^j)}$, and update actor using the sampled policy gradient: $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_i(\sigma_i^j)\nabla_{a_i}Q_i^{\mu}(\mathbf{x}^j, a_i^j, \ldots, a_i, \ldots, a_N^j)|_{a_i=\mu_i(\sigma_i^j)}$, where S is the size of the mini-batch.

Table 45: Hyperparameters of MAPPO.

Hyperparameters	HandCatch	HandLift	Stack Block
Num mini-batches	1	1	1
Num opt-epochs	5	10	5
Num episode-length	8	20	8
Hidden size	[1024, 1024, 512]	[1024, 1024, 512]	[1024, 1024, 512]
Use popart	True	True	True
Use value norm	True	True	True
Use proper time limits	False	False	False
Use huber loss	True	True	True
Huber delta	10	10	10
Clip range	0.2	0.2	0.2
Max grad norm	10	10	10
Learning rate	5.e-4	5.e-4	5.e-4
Opt-eps	5.e-4	5.e-4	5.e-4
Discount (γ)	0.96	0.96	0.96
GAE lambda (λ)	0.95	0.95	0.95
Std x coef	1	1	1
Std y coef	0.5	0.5	0.5
Ent-coef	0	0	0

B.3 Offline algorithms

B.3.1 BCQ

BCQ constrains the selected actions to be in the action distribution of the dataset. It trains a Q-network Q, a perturbation network ξ, and a conditional VAE $G = \{E(\mu, \sigma|s, a), D(a|s, z \sim (\mu, \sigma))\}$. The agent generates n actions by G, adds small perturbations $\in [-\Phi, \Phi]$ on the actions using ξ, and then selects the action with the highest value in Q. The policy can be written as

$$\pi(s) = \underset{a^j + \xi(s,a^j)}{\operatorname{argmax}} Q\left(s, a^j + \xi\left(s, a^j\right)\right), \quad \text{where } \left\{a^j \sim G(s)\right\}_{j=1}^{n}.$$

Q is updated by minimizing $\mathbb{E}_{(s,a,s')\sim\mathcal{D}}|Q(s,a) - y|^2$, where $y = r + \gamma\hat{Q}(s', \hat{\pi}(s'))$. y is calculated by the target networks $\hat{Q}$ and $\hat{\xi}$, where $\hat{\pi}$ is correspondingly the policy induced by $\hat{Q}$ and $\hat{\xi}$. ξ_i is updated by maximizing $\mathbb{E}_{(s,a)\sim\mathcal{D}} Q(s, a + \xi(s, a))$.

B.3.2 TD3+BC

TD3+BC simply adds the behavior clone term into the objective of policy optimization in TD3 to constrain the learned policy to be close to the behavior policy. Specifically,

$$\pi = \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\lambda Q(s, \pi(s)) - (\pi(s) - a)^2\right].$$

B.3.3 IQL

IQL avoids to query the values of any out-of-distribution actions without explicit constraints. It approximates an upper expectile of the value distribution by simply modifying the loss function in a SARSA-style TD backup, without ever using out-of-distribution actions in the target value. The V values are updated by minimizing

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q(s,a) - V(s)\right)\right],$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$. And Q values are updated by minimizing

$$\mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V\left(s'\right) - Q(s,a)\right)^2\right].$$

After the Q values have converged, the policy are updated by advantage-weighted behavioral cloning:

$$\mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta\left(Q(s,a) - V(s)\right)\right)\log\pi(a \mid s)\right].$$

Table 46: Hyperparameters of offline algorithms.

Hyperparameters	BCQ	TD3+BC	IQL
Hidden size	[400,300]	[256,256]	[256,256]
Learning rate	1.e-3	3.e-4	3.e-4
Discount (γ)	0.99	0.99	0.99
Polyak ($1-\tau$)	0.995	0.995	0.995
Batch size	100	256	256
Φ	0.05	-	-
generated actions	10	-	-
α	-	0.2	-
β	-	-	3.0
τ (IQL)	-	-	0.7

Most of parameters of offline algorithms follow the official settings. We find that a small α for TD3+BC would achieve better performance and we choose 0.2 rather than 2.5 (official setting). BC is TD3+BC with $\alpha = 0$.

B.4 Multi-task RL algorithms

B.4.1 Multi-task PPO/SAC/TRPO

Multi-task PPO, Multi-task SAC, and Multi-task TRPO are basically the same as the original PPO, SAC, and TRPO, except for a small change called "disentangled alphas" in the Multi-task SAC algorithm. Alpha is the entropy coefficient used to control policy exploration. Disentangled alpha means that the learning of each task has a separate alpha coefficient for better exploration between different tasks.

Table 47: Hyperparameters of Multi-task PPO.

Hyperparameters	MT1, MT4, and MT20
Num mini-batches	4
Num opt-epochs	5
Num episode-length	8
Hidden size	[2048, 1024, 512]
Clip range	0.2
Max grad norm	1
Learning rate	3.e-4
Discount (γ)	0.96
GAE lambda (λ)	0.95
Init noise std	0.8
Desired kl	0.016
Ent-coef	0

B.5 Meta RL algorithms

B.5.1 MAML

MAML is a model-agnostic algorithm for meta learning, it can be used for both supervised learning and reinforcement learning. In reinforcement learning, the goal of meta-learning is to allow the agent to quickly acquire policy for new tasks through only a small amount of experience samples in the testing phase. A task is an MDP, and any aspect of the MDP may change across tasks in the task distribution $p(\mathcal{T})$. At this time, the f_θ represents the agent's policy (a mapping from state $\mathbf{x}_t$ to action $\mathbf{a}_t$), and the loss function of each task $\mathcal{T}_i$ is:

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = -\mathbb{E}_{\mathbf{x}_t, \mathbf{a}_t \sim f_\phi, p(\mathcal{T}_i)}[\sum_{t=1}^{H} R_i(\mathbf{x}_t, \mathbf{a}_t)]$$

where H is the horizon of MDP. In a K shot reinforcement learning, K rollouts $(\mathbf{x}_1, \mathbf{a}_1, ..., \mathbf{x}_H)$ generated from f_θ, task $\mathcal{T}_i$, and their corresponding rewards $R(\mathbf{x}_t, \mathbf{a}_t)$ will be used to adapt to the new task $\mathcal{T}_i$.

B.5.2 ProMP

ProMP (Proximal Meta-Policy search) proposes a novel meta-learning algorithm based on the MAML. It combines the PPO algorithm with the idea of MAML and improves the efficiency and stability of the meta-learning training process by controlling the statistical distance of both pre-adaptation and adapted policies. In general, ProMP optimizes

$$\mathcal{L}_{\mathcal{T}}^{ProMP}(\theta) = \mathcal{L}_{\mathcal{T}}^{CLIP}(\theta') - \eta \mathcal{D}_{\mathcal{KL}}(\pi_{\theta_i}, \pi_\theta) \; s.t. \; \theta' = \theta + \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}}^{LR}(\theta), \; \mathcal{T} \sim p(\mathcal{T})$$

where $\mathcal{L}_{\mathcal{T}}^{CLIP}(\theta')$ is the same as PPO which allows it to safely use a single trajectory for multiple gradient update steps, and $\mathcal{L}_{\mathcal{T}}^{LR}(\theta)$ results in the following objective:

$$\mathcal{L}_{\mathcal{T}}^{LR}(\theta) = \mathbb{E}_{\tau \sim P_{\mathcal{T}}(\tau, \theta_o)}[\sum_{t=1}^{H-1} \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_o}(\mathbf{a}_t|\mathbf{s}_t)} A_{\pi_{\theta_o}}(\mathbf{a}_t, \mathbf{s}_t)]$$

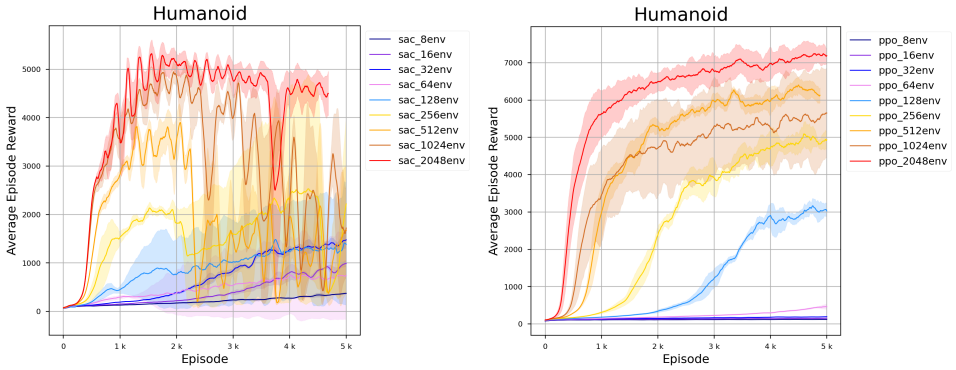
Table 48: Hyperparameters of ProMP.

Hyperparameters	ML1, ML4, and ML20
Num. mini-batches	1
Inner loop opt-epochs	1
Outer loop opt-epochs	3
Num. episode-length	8
Hidden size	[2048, 1024, 512]
Clip range	0.2
Max grad norm	1
Outer loop learning rate	3.e-4
Inner loop learning rate	3.e-4
Discount (γ)	0.9
GAE lambda (λ)	0.95
Init noise std	1
Desired kl	0.016
Ent-coef	0

C Performance discussion of PPO and SAC

In our RL/MARL experiments, we found that SAC does not work on almost all tasks, which is an anomalous phenomenon. Firstly, bimanual dexterous manipulation is a challenging task, and previous studies have shown that simple model-free RL is basically unable to complete the task. So why do we get such good performance with PPO, and SAC almost all fail? We speculate that it is because the success of PPO relies on the huge improvement in sampling efficiency brought by 2048 parallel environments. Empirically, the gain of on-policy RL due to the improvement of sampling efficiency is larger than that of off-policy RL, so SAC can not be improved to the extent that it can complete the task of bimanual dexterous manipulation. In other words, it is normal that SAC can not complete our task, and PPO can complete it because of the high sampling efficiency brought by Isaac Gym. To verify our conjecture, we tested the SAC and PPO algorithm in different environments number (8, 16, 32, 64, 128, 256, 512, 1024, 2048) in the humanoid environment officially implemented by NVIDIA [17]. The results are shown in the Figure.6. It can be seen that the performance of the SAC algorithm is better than that of the PPO below 128 environments, indicating that the implementation of our SAC algorithm is good and meets our expectations. After more than 128 environments, the performance improvement of PPO by the increase of the number of environments is apparent, while the training of the SAC algorithm is unstable, and the performance is obviously inferior to the PPO. This proves our previous conjecture and explains why SAC performs so poorly on Bi-DexHands. In addition, because the action dimension of the Bi-DexHands has 50+ dimensions, the policy entropy

method used by the SAC algorithm is easy causes instability during training. This instability appears to be exacerbated in the case of high sampling efficiency, and may also be a reason for the poor performance of SAC. In general, RL algorithms with high sampling efficiency will show some different characteristics. We also hope that Bi-DexHands can help researchers to study how to design RL algorithms with high sampling efficiency.



(a) SAC (b) PPO

Figure 6: Performance of SAC and PPO algorithms on humanoid with different numbers of environments

Table 49: Hyperparameters of SAC.

Hyperparameters	Humanoid
Num opt-epochs	2
Num. mini-batches	1
Num episode-length	32
Hidden size	[1024, 1024, 1024]
ReplayBuffer size	40000
Learning rate	3.e-4
Discount (γ)	0.99
Polyak ($1-\tau$)	0.995
Ent-coef	0.2
Reward scale	1
Max grad norm	1
Batch size	64

Table 50: Hyperparameters of PPO.

Hyperparameters	Humanoid
Num mini-batches	4
Num opt-epochs	5
Num episode-length	32
Hidden size	[1024, 1024, 1024]
Clip range	0.1
Max grad norm	1
Learning rate	3.e-4
Discount (γ)	0.99
GAE lambda (λ)	0.95
Init noise std	1.0
Desired kl	0.01
Ent-coef	0

D Details of multi-task/Meta RL training

In order to better take advantage of Isaac Gym's large-scale parallel simulation, the design of our multi-task/Meta RL pipeline is different from all existing benchmarks. The largest difference is that we do not need to only sample part of all tasks for training, all tasks are trained at the same time. I will introduce our pipeline and the detail of the multi-task/Meta RL categories respectively below.

D.1 High performance multi-task/meta RL pipeline using Isaac Gym

Isaac Gym is a recent promising simulator for reinforcement learning. Different from previous simulators that can only use CPU to simulate, it can put all simulation calculations in GPU. Benefiting from the powerful parallel computing capability of GPU and avoiding switching data between CPU and GPU, Isaac Gym is able to create a large number of simulation environments in parallel without consuming many resources. This improvement in sampling efficiency is helpful for reinforcement learning, especially in on-policy RL and multi-task/meta RL. It also has a problem that Isaac Gym only allows one single environment instance to be created on a single GPU, so we can not create multiple gym-like environments at the same time as other simulators. So we designed a pipeline that runs through the entire training pipeline of one single environment instance, to make the multi-

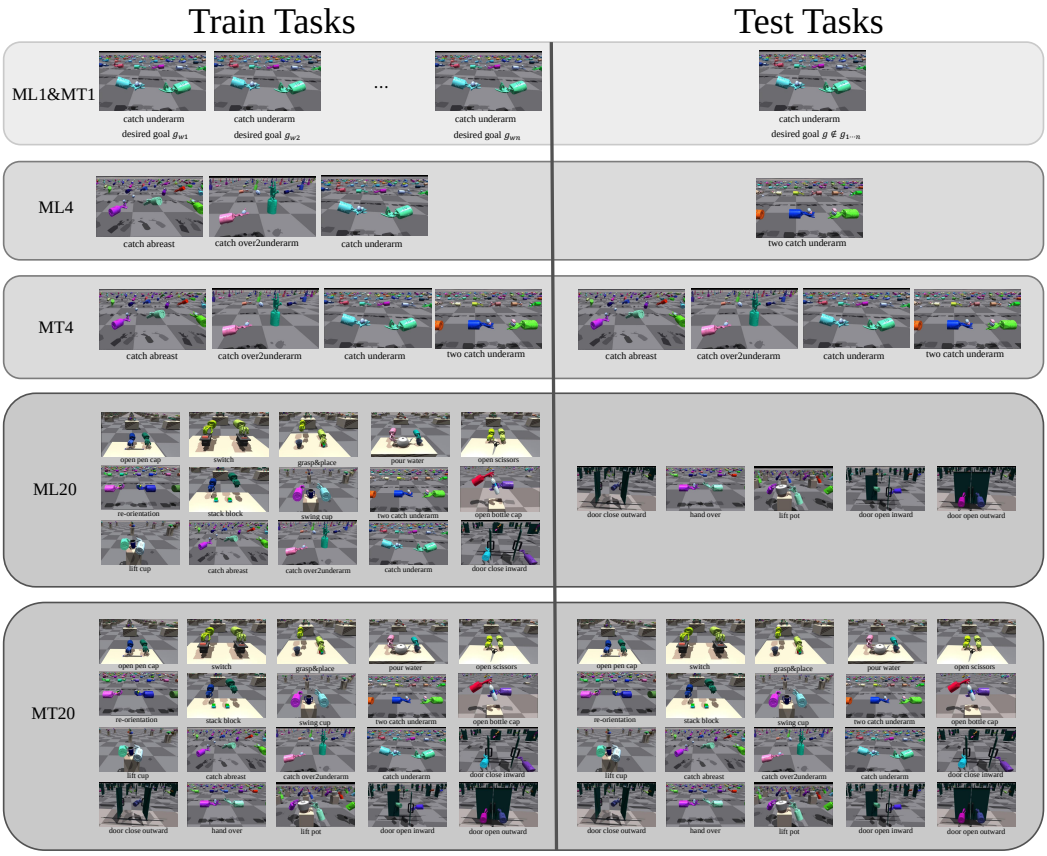


Figure 7: Detail implementations of multi-task/meta settings.

task/meta RL algorithm better leverage Isaac Gym's advantages. We directly load all tasks into an environment instance when initializing the environment, and use all tasks for data sampling and policy update at the same time, which is equivalent to that we sample all the environments every time in other simulators. In this way, each task can be trained synchronously, and the FPS is not significantly lower than one single task in parallel environments. To the best of our knowledge, our benchmark is the first to use Isaac Gym as a simulator for multi-task/meta RL. The sampling efficiency is greatly improved compared to previous simulators that rely on python parallel programs, which is helpful for multi-task/meta RL training. We hope that this will facilitate the research of multi-task/meta RL.

D.2 Detail implementation of MT1, ML1, MT4, ML4, MT20, and ML20

Our multi-task/meta RL categories are formed by our carefully designed combinations of individual tasks detailed above. According to what we said above, the ML category is that all tasks are trained and tested at the same time. Therefore, MT1 and ML1, MT4 and ML4, MT20 and ML20 are all the same in terms of category settings. The difference is 1) ML categories only use a part of tasks as meta-train sets, and the other part is used for meta-test sets, while the MT categories are all trained together. 2) From the perspective of observation, multi-task adds a one-hot vector to represent task ID, while meta masks the observation related to the goal, which requires the Meta RL algorithm to learn by itself. Figure.2 visualizes the detailed design of our multi-task and meta categories. Let's introduce their settings in detail separately:

MT1&ML1: These two categories are only trained and tested in one type of task, only the pose of the goal is different between different tasks. We use Catch Underarm as the basic category, and translate the goal pose to the left, right, and back by 0.03cm, plus the goal of the original pose to form the task of MT1&ML1. ML1 train on left, right translation, and in-position tasks, and have to quickly adapt to backward translation tasks.

MT4&ML4: These two categories consist of 4 tasks, namely Catch Underarm, Hand Over, Catch Abreast, and Two Catch Underarm. The main reason for choosing these four tasks is that they are all object throwing and catching tasks, and the skills required are relatively similar, which is conducive to multi-task and Meta RL. It should be noted that to maintain the consistency of the environment, we no longer fix the base of the handover task. ML4 train on Catch Underarm, handover, catch abreast tasks and have to adapt to two Catch Underarm tasks.

MT20&ML20: These two categories are composed of all of the 20 designed tasks. Due to the large span between different tasks, they are undoubtedly the most challenging tasks in Bi-DexHands. But it is also the most meaningful task because it covers the development of human dexterity and provides a good environment for us to master human-level dexterity. Note that there are some orders of magnitude differences in rewards between tasks. To make their rewards as close as possible, we scale the rewards in Grasp&Place, Door Open Outward, Door Open Inward, Bottle Cap, Block Stack, Door Close Inward, Door close Outward, Lift Underarm, Re Orientation, Scissors, and Swing Cup tasks by 0.1 factor to ensure the order of magnitude consistency between their rewards. ML20 needs to adapt quickly in Door Close Outward, Hand Over, Lift Pot, Open Scissors, and Two Catch Underarm tasks. All the remaining environments are given for training.