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.

The objects in the YCB dataset are used for our object-catching tasks. Because our object-catching environment is only related to the pose of the object, we can arbitrarily replace objects of suitable size in the YCB dataset. Other tasks use items from the Sapien dataset, and can also use other objects from the same category in Sapien dataset. However, because it is related to the shape of the object, some additional operations are required. We have added examples to Github to show how to use objects from YCB and Sapien datasets, see here.

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.

Designing a reward function is very important for an RL task. I would like to introduce the method of our reward design in detail. In general, our reward design is goal-based and follows the same set of logic. For object-catching tasks, our reward is simply related to the difference between the pose of the object and the target. The closer the object is to the target, the greater the reward. For other tasks that require the hand to hold the object, our reward generally consists of three parts: the distance from the left hand to the target point on the object that the left-hand needs to operate, the distance from the right hand to the target point on the object that the right-hand needs to operate, and the distance from the object to the target. The principle of our design is to conform to human intuition based on completing the task and to make the reward function structure as unified as possible. This unified reward function structure is also one of the requirements of Meta RL and Multi-task RL environment design. However, because the scenarios of each task are different, the hyperparameters of the reward function will inevitably be different. We have tried our best to avoid manual reward shaping for each task provided that RL can be successfully trained.

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 initially can not directly touch the target, nor can it directly roll the object to the other hand, so the

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

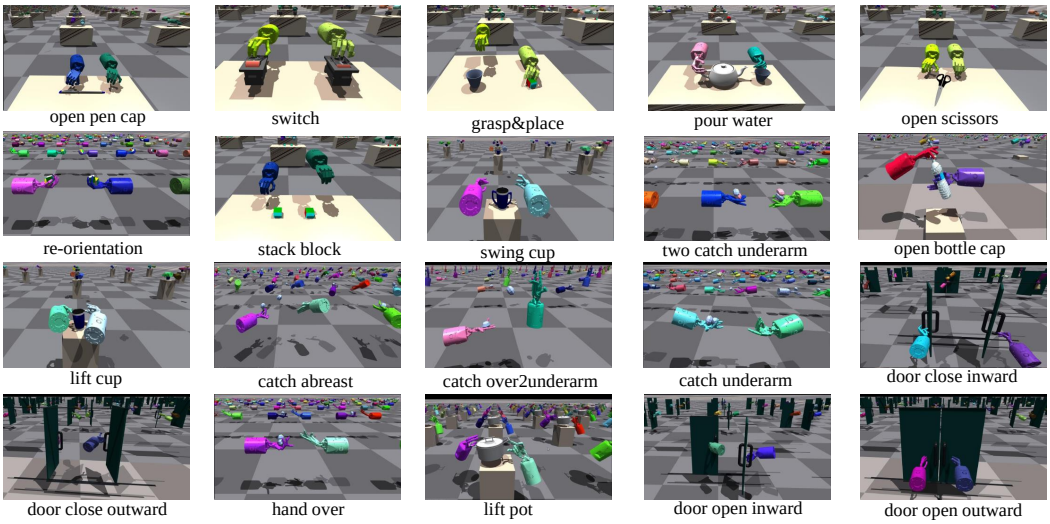


Figure 5: An overview of all tasks.

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.

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

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

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

Table 9: Action space of Hand Over.

Index	Description
0 - 19	right Shadow Hand actuated joint
20 - 39	left Shadow Hand actuated joint

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.

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.

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

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

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

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.

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

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

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

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

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 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.

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

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

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

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

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

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:

$$r = 0.2 - d_{left} - d_{right} + 3 * (0.985 - d_{target}) \tag{6}$$

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.

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

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 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.

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

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

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

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

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.

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:

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

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

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.

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

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

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

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:

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

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.

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

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

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

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.

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

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

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

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

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.

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

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

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

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 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.

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

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:

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

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

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

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

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.

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

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. 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[68] 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[68], 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.

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

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

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

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

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.

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:

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

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

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

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:

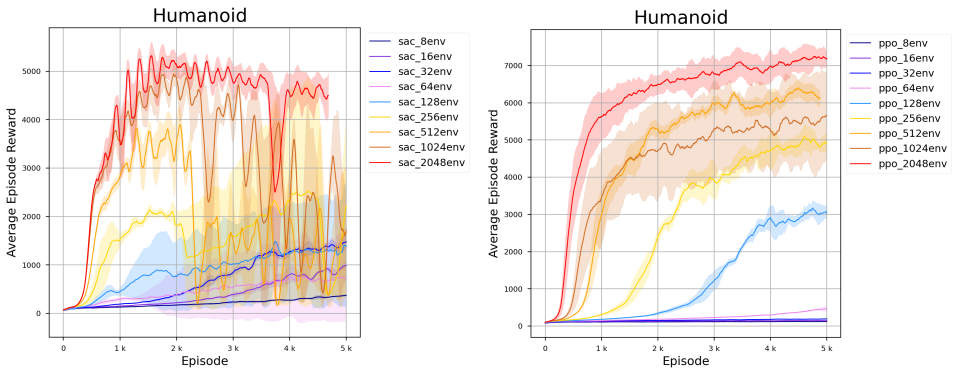
$$\mathcal{L}_{\mathcal{T}}^{LR}(\theta) = \mathbb{E}_{\tau \sim P_{\mathcal{T}}(\tau, \theta_o)}[\sum_{t=1}^{H-1} \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\theta_o}(\mathbf{a}_t|\mathbf{s}_t)} A_{\pi_{\theta_o}}(\mathbf{a}_t, \mathbf{s}_t)]$$

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

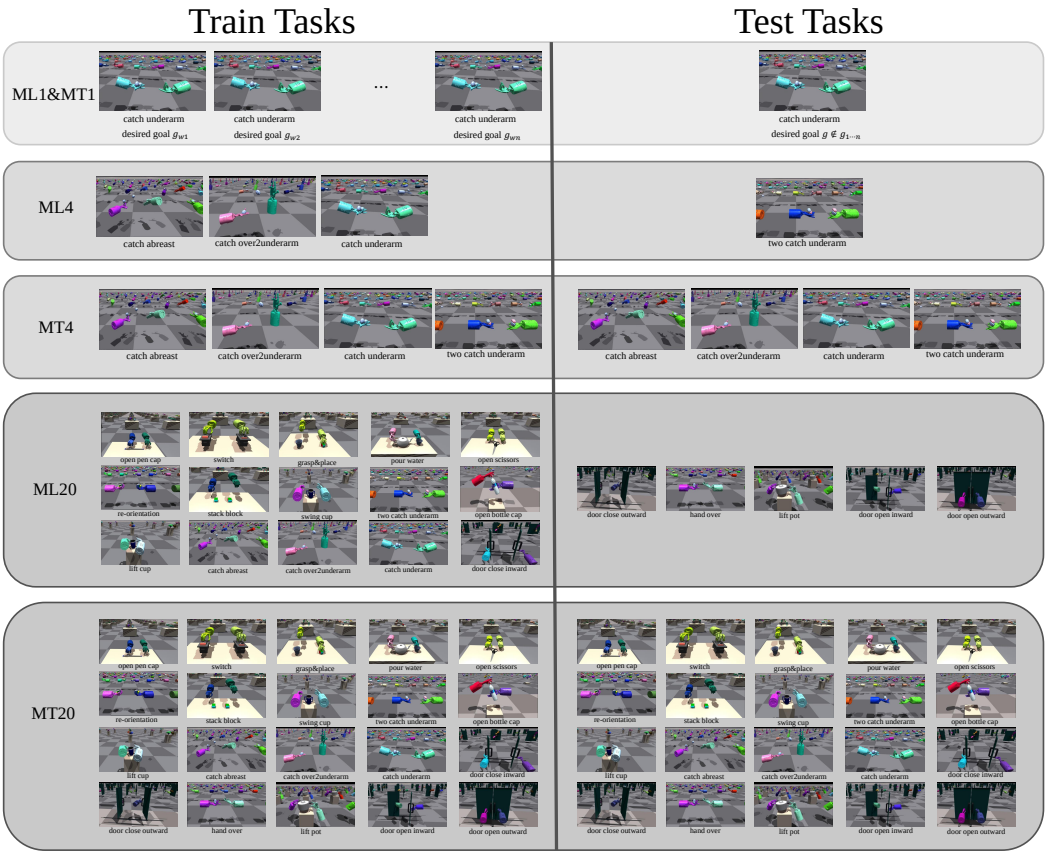


Figure 7: Detail implementations of multi-task/meta settings.

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.

Figure 8: Simple point cloud RL experiment. **Left image** is an RGB image taken with the RGBD camera that comes with isaacgym and is used to convert it into a point cloud. **Middle image** is a point cloud image converted from an RGB image. **Right image** is the experimental result of using PPO to run 6000 episodes in 256 parallel ShadowHandOver environments. The purple line is the result of RL training using point cloud input, and the blue line is the result of RL training using state input. Other parameters and settings are the same as baseline.

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.

E Visual observation about the Bi-DexHands

Currently, the observations of Bi-DexHands are state-based. This is good for a beginning research, but not a realistic setting. In the real world, the agent always needs to estimate the states of other objects by visual observations, so using visual input RL is very important for sim2real transfer. Isaac Gym can use RGBD cameras to provide us with visual information, which can be directly used as image input or processed into a point cloud. We have tried point cloud RL in Bi-DexHands, but it still has some problems. Below I will detail why we are not using visual input in Bi-DexHands.

The problem is that the parallelism of Isaac Gym's cameras is not very good. It can only obtain images one by one env serially, which will greatly slow down the running speed. At the same time, the training of the dexterous hand is very difficult and greatly depends on the high sampling efficiency. we do a simple experiment and result in Figure.8. We replace the object information with point clouds in the case of a small number of environments, and use PointNet to extract point cloud features. It can be seen that under the same episode and same number of environments, the performance of point cloud input is not as good as full state input, but it can also achieve some performance. But also using an RTX 3090 GPU, the point cloud RL has only 200+ fps, and the full state can reach 30000+. In fact, we can only open up to 256 environments when using point clouds. This was a problem with Isaac Gym's poor parallel support for cameras, so we didn't use point clouds or other visual inputs as our baselines, but they certainly could.