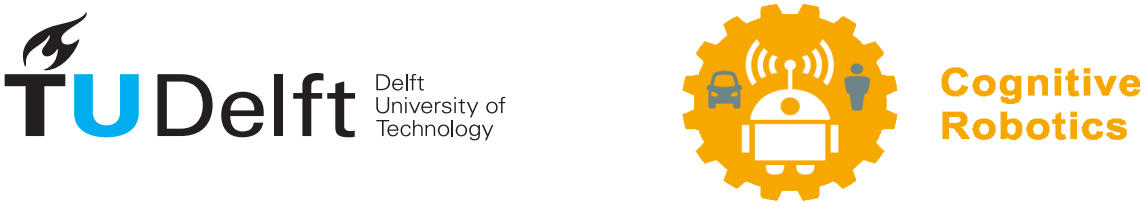
Teaching Bimanual dexterous manipulations with Interactive Demonstrations and Reinforcement Learning

Zheyu Du

Delft University of Technology

TU Delft



Delft University of Technology

Department of Mechanical, Maritime and Materials Engineering

For Master Thesis in Robotics

Teaching Bimanual dexterous manipulations with Interactive Demonstrations and Reinforcement Learning

Supervisor
Dr.-Ing. Jens Kober
Delft University of Technology

Daily supervisor
Giovanni Franzese
Delft University of Technology

Master Candidate
Zheyu Du

Student ID
5270510

Academic Year
2020-2023

Acknowledgments

I would like to express my gratitude to all my teachers from the Cognitive Robotics (CoR) Department at TU Delft for bringing me into the world of Robotics and Machine Learning. In the last two years, I have gained new knowledge and much research experience from the courses, especially in the learning and control area, which prepared me well for my future career.

My sincerest thanks to my daily supervisor Giovanni Franzese and my supervisor Jens Kober for their advice and feedback during my thesis research. I have learned many new things beyond the courses and gained much experience in academic research through this process. And I could not smoothly complete the project without their guidance. I would further thank Giovanni Franzese for his accompany and mentorship throughout the research assignment, the thesis project, and the master defense. Through uncountable discussions and meetings with him in the past two years, I have gained more insights into robot learning and control and further developed and completed my ideas for the project.

At last, I also want to thank my parents, friends, and classmates. Their accompany and support in the past three years encouraged me to get through all the difficulties and always keep energetic and optimistic to face my study and my life.

Teaching Bimanual dexterous manipulations with Interactive Demonstrations and Reinforcement Learning

Zheyu Du, Giovanni Franzese, and Jens Kober

***Abstract*— Robot dexterous manipulation research has drawn more attention in recent years since the development of various learning methods makes it possible for robots to achieve dexterity at the human level. Many attempts have been made to integrate human knowledge into Reinforcement Learning (RL) processes for faster learning speed and better performance. Despite their successes in many aspects, there are two open problems that still need to be carefully considered: 1. The effect of demonstrations gradually vanishes during RL. 2. In most cases, only imperfect demonstrations are available to robots. In this work, we proposed a new learning framework - Interactive Behavioural Cloning for faster Reinforcement Learning (IBC-RL), which could alleviate problems in complex manipulation tasks with long horizons. Different demonstrations are shown to robots at different learning stages. Robots learn complex tasks step by step with interactive demonstrations from human teachers. The framework is evaluated with four dexterous manipulation tasks simulated with the Isaac Gym engine. Human teachers perform demonstrations by controlling the simulated robot hands through a hand-tracking system (See Fig. 1). The results of the experiments could demonstrate the efficiency of IBC-RL in guiding and accelerating the learning processes with imperfect demonstrations.**

I. INTRODUCTION

Robot dexterous manipulation researches aim to enable robots to flexibly manipulate whatever objects they may have in hand, such as reorienting a cube [1], flipping a box [2], and using various tools [3]. Human beings have an extraordinary ability to handle these kinds of tasks and handle objects with various shapes, textures, and sizes. However, these are complicated tasks for robots, which challenge both the design and the driven algorithms of robot hands. As a matter of fact, the most common robots currently used in industrial manufacturing still have no ability to react to changes in the working environments. They can only execute motion sequences that human engineers predefined. But thanks to the development of new sensing technologies, more powerful computation ability, and more innovative driven algorithms, robots are becoming increasingly intelligent and dexterous in recent years [4].

Nevertheless, they are still underperforming compared to humans, especially in learning complex manipulation tasks where dexterous hands with many degrees of freedom are necessary. The dexterous hands indeed improve the ability of robots for complex operations and more human-like behaviors. On the other hand, they also largely increase the difficulty of learning an optimal strategy. In particular, learning bimanual manipulation tasks is challenging without the availability of expert demonstrations and many hours of self-exploration [5], [6].

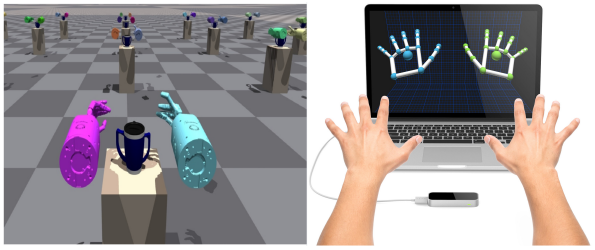


Fig. 1: Left figure: An example of the bimanual manipulation task - Swing the cup, which is simulated in Isaac Gym [9]. The robot uses both hands to grab the cup and then rotate it into the target orientations. Right figure: a human-robot interface - Leap Sensor. It captures the real-time movements of human hands and outputs the states of the simulated hands.

Many new learning methods and various control schemes have been proposed in recent years, which aim to accelerate the learning process and improve the performance of the trained controller. However, Reinforcement Learning (RL) methods still inevitably require massive data from interactions with the environments, although the amount could be reduced with model-based RL methods [7]. Imitation Learning (IL) methods require a considerable number of demonstrations from human teachers, and the reliance on expert knowledge cannot be entirely avoided [6], [8]. It is not a new idea to integrate human knowledge with the RL methods and benefit from the strengths of both two kinds of techniques. For example, the most common way is to pretrain the controller with data collected from human teachers and give the RL algorithm a warm start. Nevertheless, there are two main problems that have not been sufficiently studied so far. First, the effect of human demonstrations gradually vanishes during the RL process: robots will progressively forget the knowledge they gained from human teachers. Second, due to the limits of the human-robot interfaces and other possible interruptions, the recorded human demonstrations may be imperfect, containing suboptimal strategies, unnecessary movements, and action noises.

In this paper, to alleviate the problems mentioned above, we introduce Interactive Behavioural Cloning for faster Reinforcement Learning (IBC-RL), where human demonstrations are not used to only pretrain the policy but show robots different sub-skills according to their current learning stage and guide the explorations during RL. The proposed technique is motivated by the idea of propaedeutic skill learning where, for example, we first teach robots how to grasp objects, and only when they can successfully perform the grasping we show them how to stack the objects together. This is also the

most natural way when a human child is taught to perform a task by a teacher. The teaching is always step-by-step, and demonstrations targeting the current learning stage are shown to the child.

In the rest of this paper, some related works are introduced in Sec. II, which includes a brief review of previous research on dexterous manipulations, human-robot interfaces, and RL methods with expert or non-expert demonstrations. Sec. III introduces the background knowledge of the proposed learning framework, which briefly reviews the basic concepts of RL and Proximal policy optimization (PPO), presents the reward functions, and formalizes the imperfect demonstrations. Sec. IV introduces the IBC-RL framework in detail and formalizes the methods used. Sec. V presents the results of the evaluation experiments, some ablation studies, and the limitations of the framework.

II. Related work

Robot manipulation tasks are mainly learned with methods based on dynamic analysis or reinforcement learning (RL). The conventional dynamic analysis methods need to model the dynamics of the entire system and then treat the control of actuators as optimization problems with chosen constraints. These methods are normally limited to simple tasks with simple end effectors, like grippers [10]. Recently, RL methods have achieved success in more complex and contact-rich tasks with dexterous robot hands. Model-based RL methods are well-known for higher adaptability and data efficiency [7], which makes them seem to be suitable for practical problems. However, despite having many appealing features, model-based RL methods still suffer from the inaccuracy of the learned transition model in practice, which makes it hard to scale to various high-dimensional tasks [11], [12]. On the contrary, model-free RL methods directly learn policies through interactions with the environments. They are gaining popularity in recent years because of their scalable learning ability. However, it is undeniable that outstanding performance comes with huge costs. These methods usually require a considerable amount of data for training and typically take a long time, which is prohibitive in real applications. To alleviate this problem, an effective way is to utilize human knowledge to help with learning [10]. Expert demonstrations have been widely studied and successfully integrated with RL in different ways to accelerate the learning processes and improve the performance of trained controllers. But the study on imperfect demonstrations is still insufficient.

In this section, relevant research about robot dexterous manipulations is briefly reviewed and introduced in Sec. II-A. Sec. II-B introduces and compares different ways to integrate human demonstrations with RL. Sec. II-C introduces different human-robot interfaces used in previous research to collect demonstrations. At last, the learning frameworks that were used to deal with different kinds of imperfect demonstrations are reviewed in Sec. II-D.

A. Robot dexterous manipulation

Dexterous manipulation tasks were formulated by Salisbury [13]. They require the coordination of robot fingers and even arms to control the objects continuously and always maintain contact in the process, which puts more challenges on the design of robot hands and driven algorithms compared with discrete picking and placing. Some essential works were done to include the rolling [14], [15], sliding [16], pivoting [17], and finger gaiting [18] into the control loop, to optimize the initial grasping position and orientation [19] and to optimize the hand design for easier control or better performance [20], [21], [22]. Based on them, many control schemes have been proposed, especially in recent years.

Conventional methods usually require a high-level planner and a low-level controller. High-level planner explores the state and action spaces with various planning techniques for an optimized trajectory. The low-level controller maps the friction forces at contact points to the accelerations of robot fingers and then calculates appropriate actions for the actuators to follow the planned trajectory. Contact and grasping states are either modeled mathematically [23], [24], [25] or directly obtained from various sensors [26]. However, dynamic modeling and trajectory planning for dexterous hands severely increase the difficulty of the tasks and calculation burden. A simpler way is utilizing extrinsic dexterity to transfer the requirement of dexterity from robot hands to the external world [27]. But this means robots cannot perform the manipulation tasks alone without an additional surface like a wall or a table.

In recent years, machine learning techniques are gaining prosperity. RL methods, especially those with deep neural networks, are playing active roles in the robot control area. They were first used for dexterous manipulation tasks in 2015 [28]. After that, more research with various model-free RL methods has been done to optimize the learning processes and improve the performance of trained controllers. For example, the learning could be more efficient by parallelizing neural networks and shaping the reward function with instructive information [29]. The robustness of the learned controller could be improved by training with domain randomization [30] and taking advantage of past experience with a recurrent neural network [1]. Until now, a large number of generalized manipulation tasks could be learned with this kind of method [1], [3], [31]. Model-based RL methods have also been used in dexterous manipulation tasks, but less than the model-free ones. Anusha Nagabandi, et al proposed a model-based RL controller (PDDM), which uses a deep neural network to model the transition dynamics of the system [7]. It has shown higher data efficiency in some dexterous manipulation tasks.

Our work builds upon the model-free deep RL to control anthropomorphic double hands. By utilizing a small number of human demonstrations, even the imperfect ones, the learning processes could be significantly accelerated and more stable with fewer performance drops. Previous work has demonstrated learning of multiple bimanual manipulation

tasks with various model-free RL controllers [5]. Based on them, we further improve the learning algorithms to accelerate learning and reduce interactions with the environments.

B. Reinforcement learning with human demonstrations

An effective way to improve the data efficiency and the performance of learning methods is by integrating them with the knowledge of human beings. One approach is to pretrain the controller to mimic the expert policies extracted from human demonstrations. Then the controller is further optimized through reinforcement learning, guided by a shaped reward function [32], [33]. However, some RL algorithms, such as maximum entropy RL [34], are designed to encourage random explorations, which is conflicted with the idea of initializing the RL with imitation learning. Another problem is that shaping the reward function with human demonstrations is indispensable for these methods. Otherwise, the robot will gradually forget the knowledge learned from human teachers during the RL. An alternate approach is to train an off-policy RL with a replay buffer of human demonstrations [35] or do the imitation learning and policy searching at the same time [3], [36]. The robot could then benefit from both expert demonstrations and interactions with the environment in the policy update. However, these methods heavily rely on the high quality of the demonstrations, and some of them require annotating the demonstrated state-action pairs with the expert rewards, which are inaccessible in some cases.

Our method is inspired by the work of M. Vecerik, et al [35]. We use imitation learning for controller initialization and experience replay, and we have released the burden of shaping the reward function and annotating all the data.

C. Human-robot interfaces

Besides, due to the morphological differences between robot hands and human hands, the demonstrations usually could not be directly used for learning. It is still an open question of how to perform the demonstrations for robots effectively. Henry Zhu, et al gave demonstrations through kinesthetic teaching [2]. It is straightforward since the human teacher could directly manipulate the robot. However, it is unsuitable for high-DoF robot hands since the demonstrations will not be continuous and accurate enough. Aravind Rajeswaran, et al used a CyberGlove to record the movements of fingers and an HTC Vive tracker to track the trajectories of human hands [3]. Abhishek Gupta, et al. proposed a novel method [37] to perform demonstrations. Human experts perform multiple object-centric demonstrations, then each of them will be weighted for each individual controller. In this way, controllers only learn from the demonstrations that share similar initial conditions with themselves and at the same time are feasible for learning.

Our work uses a Leap Sensor to collect demonstrations, which is convenient and cheap. It can collect real-time states of human hands and fingers. However, the sensing area is limited, and sensing with infrared rays is less accurate.

D. Learning with imperfect human demonstrations

Most previous research studied how to learn a policy from high-quality demonstrations from human experts effectively. Imitation learning methods have been proven to work well in benchmark tasks [38]. However, they could not show satisfactory performance when dealing with practical problems since the quality of demonstrations could not always be guaranteed. They might be unavailable at all or too expensive to have a large number of them for accurate learning [39]. Therefore, many researchers have moved their attention to learning with non-expert demonstrations.

Some researchers have studied how to learn from suboptimal demonstrations with imitation learning and reinforcement learning techniques. However, most of these methods require different information or other assumptions to update the policy. The Distance minimization inverse RL (DM-IRL) assumes that the true reward function is linear in the predefined feature function and requires human teachers to estimate the cumulative reward [40]. Semi-supervised IRL (SSIRL) is able to recognize optimal demonstrations from a mixture of optimal and suboptimal ones [8]. The reliance on expert demonstrations could still not be avoided.

Another way to deal with this problem is by utilizing the nature of RL. Noises learned could be gradually filtered out through more explorations. Instead of trying to find optimal demonstrations, robots could learn from the imperfect ones and make further optimizations and filter out the noises through the explorations during RL. Normalized Actor-Critic (NAC) trains the controller for auto-driving with imperfect human demonstrations and then refines the learned policy with RL [41]. It is indeed able to utilize imperfect demonstrations to help with the learning, but they are only used for the initialization of neural networks. Relative Entropy Q-learning (REQ) collects new data with a mixture of action from RL and human policy [42]. It has been tested and evaluated with several bimanual manipulation tasks and has shown excellent performance and potential. However, it only deals with the suboptimal strategies in the demonstrations and also requires the guidance of optimal ones.

Previous research aimed to deal with the demonstrations with suboptimal strategies, some meant to filter out the noises, and there are also works tried to learn from demonstrations with different strategies. Only a few of them tried to solve all the problems at the same time. Our work is inspired by NAC and makes further improvements to utilize imperfect demonstrations during RL. We use the data directly collected from the sensor and aim to deal with all the problems mentioned above, which is a more realistic situation in practical tasks.

III. Background on Robot Learning

This section briefly reviews the main concepts of reinforcement learning in III-A. Then III-A.1 introduces the Proximal Policy Optimization method [43], which is used in our framework (IBC-RL). III-A.2 compares three ways to define the reward function. At last, III-B formalizes the imperfect

demonstration studied in this paper and distinguishes it from other kinds of non-expert demonstrations studied before.

A. Reinforcement Learning

Reinforcement Learning (RL) methods are able to deal with the control problems of complex dynamic systems, which are usually defined by Markov decision processes (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, r\}$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, and T defines the transition dynamics of the system. The robot explores the environment and learns a policy π which could maximize the cumulative reward

$$R^{\pi}(\boldsymbol{s}_0) = \sum_{t=0}^{\infty} \gamma^t r_{t+1} = \sum_{t=0}^{\infty} \gamma^t \rho(\boldsymbol{s}_t, \pi(\boldsymbol{s}_t)), \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor that encodes the increasing uncertainty about the future and helps with the convergence of learning. The state-value (V) function and action-value (Q) function are defined as

$$V^{\pi}(\boldsymbol{s}_t) = \mathbb{E}[R^{\pi}(\boldsymbol{s}_t)|\boldsymbol{s}_t]; \tag{2}$$

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \mathbb{E}[R^{\pi}(\boldsymbol{s}_t)|\boldsymbol{s}_t, \boldsymbol{a}_t], \tag{3}$$

respectively. The advantage function

$$A^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) - V^{\pi}(\boldsymbol{s}_t), \tag{4}$$

indicates how much is the action $\boldsymbol{a}_t$ taken at state $\boldsymbol{s}_t$ better or worse than the expected return when following policy π, which could be used to update the policy.

*1) **Proximal Policy Optimization**:* Proximal policy optimization (PPO) is one of the most widely used model-free RL algorithms, which optimizes the approximated value function and the control policy periodically [1], [43]. For k-step returns, the estimator of the value function $V^{\pi}(s_t, a_t)$ could be defined as

$$\hat{V}_t^{(k)} = \sum_{i=t}^{t+k-1} \gamma^{i-t} r_i + \gamma^k V(\boldsymbol{s}_{t+k}). \tag{5}$$

Then, to compare the action $\boldsymbol{a}_t$ with the average action taken at state $\boldsymbol{s}_t$, the advantage function $A^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is used as an indication, which could be estimated with the generalized advantage estimator (GAE) in the following way:

$$\hat{V}_t^{GAE} = (1 - \lambda) \sum_{k>0} \lambda^{k-1} \hat{V}_t^{(k)}, \tag{6}$$

$$\hat{A}_t^{GAE} = \hat{V}_t^{GAE} - V(\boldsymbol{s}_t). \tag{7}$$

During each trial, the transition data of each step are collected into the update buffers with a fixed length N_u. For every N_u steps, the policy and value function estimators are optimized with the mini-batch stochastic gradient descent method (See Algorithm 1, line 15-18). The clipped objective of this periodic optimization could be defined as

$$\begin{aligned} L_{PPO} &= \mathbb{E} \min(R_p \hat{A}_t^{GAE}, clip(R_p, 1 - \epsilon, 1 + \epsilon) \hat{A}_t^{GAE}), \\ R_p &= \frac{\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{old}(\boldsymbol{a}_t|\boldsymbol{s}_t)}, \end{aligned} \tag{8}$$

in which R_p is the ratio of the probability of taking action $\boldsymbol{a}_t$ with the updated policy π to the probability of taking action $\boldsymbol{a}_t$ with the old policy π_{old}. ϵ is a hyper-parameter that indicates the range of deviation of the new policy from the old one.

PPO explores the environment by sampling new actions randomly around the actions given by the old policy. This randomness will progressively decrease and the update rule encourages the algorithm to exploit the previous experience more, which may increase the chance of getting the algorithm stuck in local minima, especially when dealing with complex tasks. Besides, although limited by ϵ, it is possible that the updated policy deviates significantly from the old one and results in the deterioration of performance. Many different tricks have been used in the variants of PPO to deal with these problems. In this paper, we will show that they could also be alleviated with the help of human demonstrations, even with imperfect ones.

*2) **Reward Function**:* The reward function is one of the most important pieces in RL that guides the explorations and determines the goals that robots will try to reach. In the real world, most of the tasks are considered to be episodic [44]. The most natural and intuitive way of awarding a learner is to give positive feedback if a certain goal has been reached within a set of constraints, like

$$r = \begin{cases} R & if\ |\boldsymbol{s}_t - \boldsymbol{s}_g| < \xi \\ 0 & else, \end{cases} \tag{9}$$

in which $\boldsymbol{s}_g$ is the goal state and ξ defines the region of success. This kind of sparse reward function does not need any prior information about the environment, but it cannot provide enough guidance for learning since robots could only receive feedback if reaching certain regions in the state space [29], [44].

Since the complexity of the reward function is proportional to the complexity of the policies that robots could learn, another choice is to carefully shape a reward function that encodes all the intended motions. These methods include Inverse Reinforcement Learning (IRL), apprenticeship learning, and inverse optimal control [45]. However, shaping the reward with human knowledge is usually avoided in robotics research. The designed functions are typically task-specific, which cannot be generalized to other tasks. The computational costs increase significantly with the complexity of the tasks. And the model that transforms human knowledge to a reward function is usually not available [45].

Considering the computational complexity and generalization to various tasks, reward functions that are structured and continuous in the state space [2], [7], or at least smoothly varying across different regions [1], [29], are mostly used in dexterous manipulation tasks. For example, the reward could be measured with the distance between the current state and the goal state as

$$\begin{aligned} r &= -w_d * |d\boldsymbol{s}| + R_g * \mathbb{1}_{\{|d\boldsymbol{s}| < \xi\}} \\ d\boldsymbol{s} &= \boldsymbol{s}_t - \boldsymbol{s}_g. \end{aligned} \tag{10}$$

The robot will receive higher rewards when the current state $\boldsymbol{s}_t$

is closer to the goal state s_g. If the task is completed $|ds| < \epsilon$, the robot will receive an additional reward R_g. Only a little prior knowledge has been used in the reward function, and it could provide more guidance for explorations.

Most of the dexterous manipulation tasks could be divided into a sequence of subtasks [46]. Robots could achieve the final goal step by step by achieving all the goals of the subtasks. For example, the task in which a robot needs to hit a nail with a hammer could be divided into three subtasks ('get to hammer', 'hammer to nail', and 'hit the nail') and thus three subgoals. The reward function of such complex manipulation tasks is usually defined as

$$\begin{aligned} r &= r^1_{sub} + r^2_{sub} + ... + r^n_{sub} \\ r^n_{sub} &= f(|\boldsymbol{s}_t - \boldsymbol{s}^n_g|), \end{aligned} \tag{11}$$

in which r^n_{sub} and $\boldsymbol{s}^n_g$ are the reward and goal state of n^{th} subtask, respectively. $f(\cdot)$ is the function that calculates the reward value given the current distance to the goal state [3], [5]. This structure is also used in our work.

B. Formalization of Imperfect Demonstration

Previous researches mainly focus on expert demonstrations $\tau_{sa} : \boldsymbol{a} = \pi^*(\boldsymbol{s})$, which adopt the optimal or a near-optimal strategy with respect to the definition of the reward function. Some researchers studied suboptimal demonstrations $\tau'_{sa} : \boldsymbol{a}' = \pi_{sub}(\boldsymbol{s})$ which are probably collected from non-expert users, and demonstrations corrupted with different degrees of Gaussian noises $\tau'_{sa} : \boldsymbol{a}' = \pi^*(\boldsymbol{s}) + \boldsymbol{a}_n, \boldsymbol{a}_n \sim \mathcal{N}(\mu, \sigma^2)$. But none of these could represent the most common case in practice.

Most robot users are not experts in the given tasks in practical problems. Their demonstrations are based on their own preferences and experience, which are suboptimal and probably of different lengths. Depending on different human-robot interfaces used for demonstration collection, sensing noises and errors are usually inevitable. The robot learner could tolerate small noises and errors, but large ones may cause trouble to learning. Certain limitations of the interfaces may also result in some unnecessary movements. For example, due to the limited sensing area, human teachers sometimes move their hands out of the sensing area and need to move them back. This behavior is not a part of the strategy to perform the task, but this kind of unnecessary movement is inevitable with this kind of human motion-tracking device. Also, since most human-robot interfaces have no feedback on human hands, human teachers have no idea of the contact situation, which may also lead to unnecessary movements like shaking, regrasping, and other adjustments of hand poses. These movements mainly depend on the sensing device used and the personal preferences of human teachers. In most cases, they could not be distinguished from sensing noises and measured statistically.

The imperfect demonstrations τ_{su} considered in this paper, on the other hand, contain all the deficiencies mentioned above, which could be defined as

$$\tau_{su} : \boldsymbol{u} = \pi_{sub}(\boldsymbol{s}, k) + \boldsymbol{u}_{noise}, \boldsymbol{s} \in \mathcal{S}, \boldsymbol{u} \in \mathcal{A}, \tag{12}$$

Algorithm 1 Interactive Behavior Cloning for Reinforcement Learning (IBC-RL)

Human: Divide task into subtasks: $M = \{m_1, m_2, ..., m_n\}$
Step 1: Collect human demonstrations
while *Recording demonstrations* **do**
  $\tau^1_i \leftarrow (\boldsymbol{s}^1_t, \boldsymbol{a}^1_t),\ \tau^2_i \leftarrow (\boldsymbol{s}^2_t, \boldsymbol{a}^2_t), ...,\ \tau^n_i \leftarrow (\boldsymbol{s}^n_t, \boldsymbol{a}^n_t)$
  $\mathcal{D}^n \leftarrow filter(\tau^n_i)$
end while
Step 2: Learning subtasks with diluted demo replay
Reset environment and controller: $(\boldsymbol{s}_0, \boldsymbol{\theta}, \mathcal{R}, \mathcal{S}, \mathcal{A}, \mathcal{U})$
for $i \le n$ **do** ▷ Learning ith subtask
  Train controller with $\mathcal{D}^i$: $BC(\pi_\theta, \mathcal{D}^i, n_e)$
  while $t < G_i$ **do**
    $\boldsymbol{s}_{t+1}, r_t, \boldsymbol{a}^{old}_t$ = Step $(\boldsymbol{s}_t, \boldsymbol{a}_t = \pi_\theta(\boldsymbol{s}_t))$
    $\mathcal{S} \leftarrow \boldsymbol{s}_t,\ \mathcal{A} \leftarrow \boldsymbol{a}_t,\ \mathcal{R} \leftarrow r_t,\ \mathcal{U} \leftarrow \boldsymbol{a}^{old}_t$
    Update $\bar{r}_{max}$ with eq.(18) and eq.(19)
    if controller needs to be updated **then**
      Optimize $\boldsymbol{\theta}$ with eq.(8) ▷ Update periodically
      Empty $\mathcal{R}, \mathcal{S}, \mathcal{A}, \mathcal{U}$
    end if
    if learning gets stuck **then** ▷ Condition in eq.(19)
      Modify $\boldsymbol{\theta}$ with $\mathcal{D}^i_m$: $BC(\pi_\theta, \mathcal{D}^i_m, n'_e)$
    end if
  end while
end for

in which $\pi_{sub}(\boldsymbol{s}, k)$ is the suboptimal policy shown by human teachers in k^{th} demonstration and $\boldsymbol{u}_{noise}$ is the noise signal due to the sensing errors, unnecessary movements, and dynamics difference. At state $\boldsymbol{s}_t$, the human teacher takes a suboptimal action $\boldsymbol{u}_t \sim p_n(\boldsymbol{u}_t|\boldsymbol{s}_t, \boldsymbol{a}_t, k), \boldsymbol{u}_t \in \mathcal{A}$. Therefore, imperfect demonstrations could be considered to be drawn from the following probability density:

$$\begin{aligned} p_d(\tau_{su}) = &\prod_{t=1}^{T} p_s(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{u}_t) \\ &\cdot p_n(\boldsymbol{u}_t|\boldsymbol{s}_t, \boldsymbol{a}_t, k), \boldsymbol{u}_t \in \mathcal{A}, \end{aligned} \tag{13}$$

in which T is the control horizon, p_s is the transition probability, and p_n is the noise density indicating the level of suboptimal strategies and the quality of collected data [47]. An ideal control framework should enable robots to extract useful information from such imperfect demonstrations and use it to help with learning.

IV. Interactive Teaching Framework

The goal of the proposed learning framework is to enable robots to benefit from (imperfect) human demonstrations

during the RL in long-horizon tasks, see Algorithm. 1. With IBC-RL, robots learn the entire task by gradually learning all the subtasks sequentially with the segmented demonstrations, which is a similar learning logic as a human teacher teaches a child. The general learning pipeline of IBC-RL is introduced in Sec. IV-A. Sec. IV-B explains how robots learn useful information from imperfect human demonstrations. Sec. IV-C explains how different demonstration buffers are constructed based on human knowledge and used to help with learning. At last, Sec. IV-D explains how human demonstrations are used to guide the exploration during RL.

A. Learning pipeline

The entire learning pipeline of IBC-RL is shown in Fig. 2. Before the learning starts, human teachers first decompose the task and then perform demonstrations for each subtask, as explained in Sec. IV-C. Optionally, they could also choose to perform the related demonstrations later during the learning when robots ask for their help.

After the learning starts, robots start from the first subtask and finally learn all the subtasks sequentially. The number of learning iterations for i^{th} subtask G_i $(i < n)$ is determined by the experience and observations of human teachers in the following way:

$$\begin{aligned} r^i_{thre} &= w^i_p \cdot \bar{r}^i_h, \quad w^i_p \in [0,1] \\ G_i &= \{N_e \mid r^{i-1}_{thre} \le \bar{r} < r^i_{thre}\}, \end{aligned} \tag{14}$$

in which r^i_{thre} is the threshold reward of i^{th} subtask. w^i_p is the weight factor of i^{th} subtask and is determined by the human teacher. $\bar{r}^i_h$ is the average reward that human teachers could get when they complete i^{th} subtask. $\bar{r}$ is the average reward of robot learners, which is introduced and formalized in Sec. IV-D. If the performance of the controller reaches the threshold ($\bar{r} \ge r^i_{thre}$), it means robots have roughly learned i^{th} subtask, and they will start to learn $(i+1)^{th}$ subtask. Therefore, G_i is the number of iterations N_e when $\bar{r}$ is in range $[r^{i-1}_{thre}, r^i_{thre}]$. w^i_p needs to be chosen carefully based on the observations of training processes. Being either too small or too large will slow down learning. G_n is a relatively large number from the start of the last subtask to the maximum learning iterations, and the learning could be stopped if it converges early.

The learning procedure of each subtask is shown in the blue dotted box in Fig. 2. $\mathcal{D}^i$ and $\mathcal{D}^i_m$ are the datasets of collected human demonstrations of i^{th} subtask, which will be used when and only when learning this subtask. The controller is first trained on the demonstrations in $\mathcal{D}^i$ through Behavior Cloning (BC), which is explained in Sec.IV-B. Robots will ask for $\mathcal{D}^i$ if the demonstrations for this subtask are unavailable. Then the RL starts again, and robots update the controllers through further explorations with the optimization logic explained in Sec. III-A.1. During this process, if the learning gets stuck, the controller will be modified with the demonstrations in $\mathcal{D}^i_m$ also through BC, but with much fewer training epochs ($n'_e < n_e$). This kind of active demonstration replay is detailed in Sec. IV-D. After training for G_i steps, robots will move on to the next subtask and repeat this learning procedure.

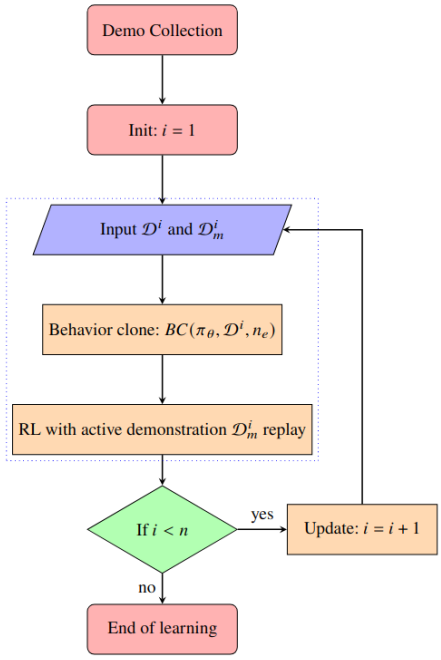


Fig. 2: The flow chart of sequentially learning n subtasks with IBC-RL. The blue dotted box illustrates the procedure by which robots learn each subtask. $\mathcal{D}^i$ is used for BC. $\mathcal{D}^i_m$ is used for the active demonstration replay, which could be the same set with $\mathcal{D}^i$ or a different data set collected from human teachers trying to help the robot get out of the local minima.

In this learning pipeline, human teachers show the correct ways to robots at the beginning of each subtask and guide their self-learning processes through demonstrations. The noisy knowledge is demonstrated to robots in a diluted form to overcome the forgetting problem, and the noises could be filtered out quickly through RL.

B. Behavior cloning with imperfect demonstrations

The method used in IBC-RL to extract useful information from imperfect demonstrations is Behavior Cloning (BC) with restricted learning epochs. The data collected needs to be filtered at first to compensate for the noises to a certain degree:

$$\tau^{inter}_i = \{(\boldsymbol{s}, \boldsymbol{a}) \mid (\boldsymbol{s}, \boldsymbol{a}) \in \tau_i, |a| > a_{threshold}\} \tag{15}$$

$$\tau^*_i = \{\tau^{inter}_i \mid \bar{a}_{knn} - \epsilon < a < \bar{a}_{knn} + \epsilon\} \tag{16}$$

τ_i is the trajectory collected in i^{th} demonstration, $\bar{a}_{knn}$ is the average action values of k nearest data points and ϵ is the tolerance to the noise. In the first filter, small actions whose values are smaller than the threshold $a_{threshold}$ will be set to zero. In the second filter, actions whose values are not close to the average action values of k nearest data points will be assigned with the average values. The datasets after filtering are indeed much cleaner. However, it is inevitable to lose some detailed information, and the noises cannot be fully filtered out. The suboptimal strategies and the remaining noises could still cause trouble in learning.

With the filtered demonstration dataset $\mathcal{D} : \{\tau^*_i\}_N$, which is formed with N different trajectories, the parameters $\boldsymbol{\theta}$ of the

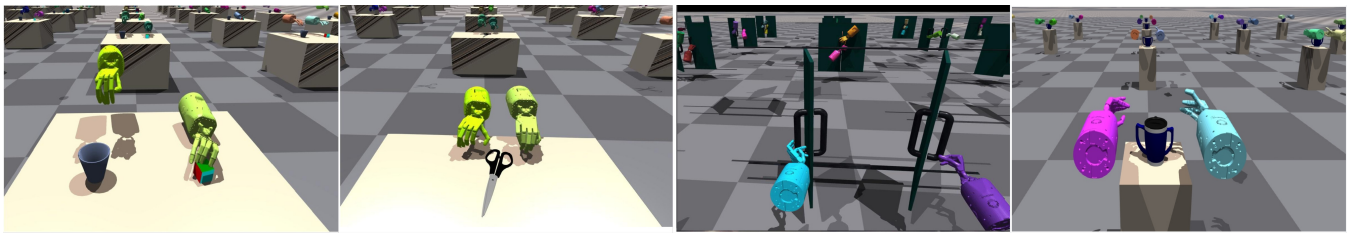


Fig. 3: The bimanual manipulation tasks used to evaluate the control framework: 'Grasp and Place', 'Open the Scissor', 'Close the Door', and 'Swing the Cup' (From left to right). The definition of the tasks and the simulation environments are provided by Yuanpei Chen, et al [5] with modifications made by ourselves.

policy neural network are modified to reduce the mean squared error (MSE)

$$\mathcal{L}_{MSE}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{s},\boldsymbol{a})\in\mathcal{D}} ||\pi_\theta(\boldsymbol{s}) - \boldsymbol{a}|| \tag{17}$$

between the outputs of the policy neural network and the demonstrated actions. $(\boldsymbol{s}, \boldsymbol{a})$ are state-action pairs in the dataset $\mathcal{D}$. Unlike the original BC, which requires the learned policy to mimic the demonstrated ones as closely as possible, IBC-RL requires the trained controller to avoid overfitting any imperfect human policy. Therefore, the dataset $\mathcal{D}$ should be formed with multiple demonstrations: the average value of the demonstrations sampled from human teachers is able to provide general guidance to the robot. Besides, a hyper-parameter n_e is used to control how many epochs the controller is trained on the training set and stop the learning at a proper time. In this way, overfitting could be avoided, and the robot only learns the general direction for further explorations.

C. Demonstrations for subtasks

Like teaching children to perform a task, human teachers could teach a robot step by step to ensure that necessary knowledge is shown to it at the proper time. As explained in Sec. III-A.2, most of the manipulation tasks could be decomposed into a sequence of subtasks. When the robot needs to hit a nail, it needs to 'grab the hammer', 'move to the nail', and finally 'hit the nail'. The subtasks must be completed sequentially. There is no way that the robot could move the hammer to the nail if it has not grabbed the hammer. It could not hit the nail if the hammer is not near the nail. The related reward functions of these tasks are defined with the structure shown in eq. 11, which guides the robot to perform the entire task by achieving all the subgoals sequentially.

Therefore, the task M could be predefined into a sequence of subtasks $M = \{m_1, m_2, ..., m_n\}$ based on human experience and the reward function (See Algorithm 1, line 1). Demonstrations for each subtask are collected separately from human teachers before the RL start or during the learning when robots need them. For the n^{th} subtask, the demonstrated trajectories $\tau_i^n : (\boldsymbol{s}, \boldsymbol{u})$ are collected. After filtering, these trajectories are put into the related datasets $\mathcal{D}^n$ or $\mathcal{D}_m^n$ (See Algorithm 1, line 3-6). $\mathcal{D}_m^n$ could be the same dataset as $\mathcal{D}^n$ or a different one collected later during the learning. When robots need to learn the n^{th} subtask, only $\mathcal{D}^n$ or $\mathcal{D}_m^n$ will be shown to them instead of the demonstration $\mathcal{D}$ for the entire task. The noises and the suboptimal strategies in other data sets will not corrupt the learning repeatedly.

By decomposing the entire task into multiple subtasks, the two main problems mentioned above could be alleviated simultaneously. It could help with the forgetting problem since the learning periods of the subtasks are much shorter, and the related demonstrations are shown to robots only when they need them. More importantly, it also avoids unnecessary confusion and undesired corruption to the learning processes when working with imperfect human demonstrations.

D. Active demonstration replay

Although being only a part of the entire task, some complex subtasks may still be quite challenging to learn and require many learning iterations. While learning these subtasks, the robot could still gradually forget human knowledge. On the other hand, imperfect demonstrations may fail to provide enough guidance for learning these subtasks. Therefore, the learning may still get stuck, which could be corrected by adjusting the controller with the human demonstrations again.

Many methods could be used to indicate if the learning gets stuck in local minima or if the robot learns the wrong strategies that could not lead to the success of the following subtasks. The most straightforward one is measuring the average rewards $\bar{r}$ of a fixed number of trials completed recently. During learning, robots will keep and continuously update a reward buffer $\mathcal{R}_e$ with a fixed length N. Once a robot worker has completed a trial, the cumulative reward of this episode is added to $\mathcal{R}_e$, and the earliest reward value stored in $\mathcal{R}_e$ is deleted. In other words, $\mathcal{R}_e$ only stores N recent reward values. The average reward

$$\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i,\ r_1, r_2, ..., r_N \in \mathcal{R}_e \tag{18}$$

can reflect how well the robot could perform the task at the current learning stage. During learning, $\bar{r}$ of the past M steps are recorded in a buffer $\mathcal{R}_u$ of fixed length M and the maximum value is recorded in $\bar{r}_{max}$ as

$$\bar{r}_{max} = \max\{\bar{r}_1, \bar{r}_2, ..., \bar{r}_M\}, \bar{r}_1, \bar{r}_2, ..., \bar{r}_M \in \mathcal{R}_u. \tag{19}$$

If $\bar{r}$ fails to increase in M steps: $\bar{r}_{max} = \bar{r}_1$, which means the learning gets stuck, robots will use human demonstrations $\mathcal{D}_m^i$ to adjust the controller to get the learning back into the right track (See Algorithm 1, line 14, 19-21). In cases where human teachers continuously supervise the learning, they could also

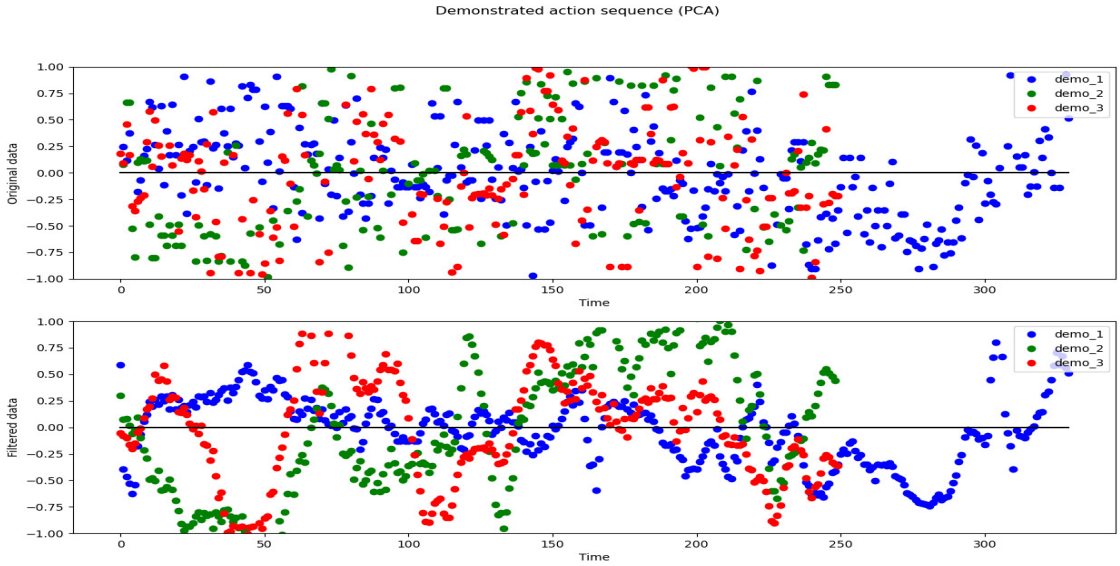


Fig. 4: The upper figure shows the original action sequences collected from the sensor, and the lower figure shows the action sequences after filtering with Equation.15 and 16. The X-axis is time and the Y-axis is the main component of collect actions transformed with PCA. Different strategies are shown in different demonstrations. *demo*_2 and *demo*_3 have similar strategies, which are very different from that of *demo*_1.

manually stop the learning and replay the demonstrations if unintended behaviors or strategies are observed.

Unlike the previous research, in which human demonstrations are replayed to robots regularly, we only replay them when the learning gets stuck. And we only use the demonstrations to modify the controller to help it get rid of the local minima instead of forcing the robot to mimic them. Although imperfect demonstrations indeed could provide useful guidance for learning, they will also corrupt the learning every time shown to the robot.

V. Bimanual Manipulation Experiments

TABLE I: Parameters of the simulated environments

Parameters	Description	Value
D_s	Dimension of the state space	428
D_a	Dimension of the action space	52
n_a	Number of working agents	512
f_c	Control frequency	60 Hz
f_n	Frequency of noise generation	600 steps
N_o	The range of noise in observations	[0, 0.002]
N_a	The range of noise in actions	[0, 0.05]

Various experiments have been done to validate the problems when integrating imperfect human demonstrations with reinforcement learning and evaluate the effectiveness and robustness of our framework. The first part of the experiments demonstrates the problems that human knowledge will be gradually forgotten during reinforcement learning and how the noises will cause trouble to the learning processes. The second part of the experiments explains why multiple demonstrations are used instead of a single one and why demonstrations of the subtasks are used instead of those of the entire task. The last part of the experiments evaluates the effectiveness of the framework with four bimanual dexterous manipulation tasks (see Fig.3) and explains the limitations of the framework.

All the test environments are built and simulated with the Isaac Gym engine [9]. The important parameters are listed in table I. Each human-like robot hand has 26 DoF, which is dexterous enough to perform various tasks like pushing, grasping, catching, etc. For the collection of human demonstrations, human teachers control the simulated robot hands with the human motion tracking device, which captures the movements of human hands with infrared rays at 100 Hz. The sensor collects the real-time states of human hands and fingers. After some transformations which could compensate for the morphology difference, the robot will receive the difference between its current state and the next state $s_{t+1} - s_t$, which is sent to the PD controllers for the trajectory following. Then the robot mimics human movements by sequentially visiting all the collected states of human hands in a demonstration. The state-action pairs of the simulated robot hands are recorded at 60 Hz. The demonstrations used in the experiments were collected from two demonstrators. They performed the tasks using the hand-tracking device without much practice and tried to show different but reasonable and human-like strategies.

Demonstrations collected with the human motion tracking device contain various suboptimal strategies, unnecessary movements, and noises, which could be depicted with eq. 12 and 13. For each collected demonstration, when replaying the collected actions in sequence, the robot can repeat human behaviors, but suboptimal strategies, unnecessary movements, and noises could be obviously noticed. It becomes much

cleaner after filtering, as shown in Fig.4. Assigning exotic actions with average values could indeed filter out much noise. However, it will also lose many details of human movements.

Our work is built upon a baseline of bimanual manipulation tasks of [5]. We designed new functions and APIs to collect and visualize human demonstrations and made necessary modifications to the PPO controller and task definitions. The main parameters of the PPO controller are shown in table II. These values are shared in all the following experiments. Because of the randomness when sampling new actions at each step during the exploration, the learning processes of the same task could be very different. Therefore, multiple tests are done for each task with the same settings. The mean and variance of the learning curves are shown in the figures for comparison.

TABLE II: Parameters of the PPO controller

Parameters	Description	Value
γ	Discount rate	0.96
lr	Learning rate	3×10^{-4}
n_s	Update intervals	8
n_m	Number of mini-batches	4
ϵ	Clip range of the policy update	0.2
H_p	Size of the policy network	[1024, 1024, 512]
H_v	Size of the value network	[1024, 1024, 512]
F_a	Activation function	Elu
N_{it}	Max number of iteration for each trial	4000

The learning framework IBC-RL is evaluated with four kinds of bimanual dexterous manipulation tasks in Sec. V-A. The results of some ablation studies are shown in Sec. V-B, which validate the main problems and support the design of the framework. At last, the limitations of the framework exposed in the experiments are summarized in Sec. V-C.

A. Evaluation of the framework

The new learning framework IBC-RL is evaluated with four tasks: 'Open the Scissor', 'Grasp and Place', 'Close the Door', and 'Swing the cup' (see Fig. 3). All the tasks have been divided into two subtasks: 'Reach' and 'Manipulation'. In the 'Reach' subtask, robots learn how to move their hands from where they are initialized to the target objects. In the 'Manipulation' subtask, robots learn how to manipulate the objects to transform them from the initial state to the target state. The results of the experiments could demonstrate the validity of IBC-RL and also illustrate its limitations.

In all four tasks, the following distances need to be measured

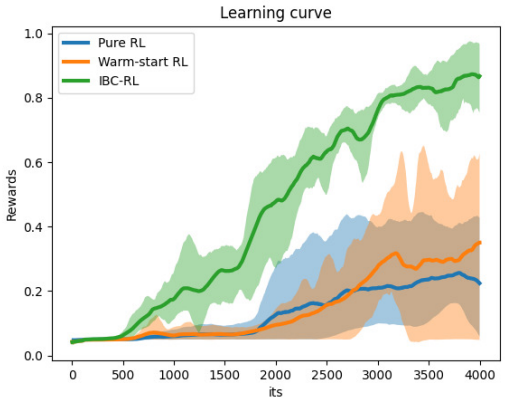


Fig. 5: The results of the 'Grasp and Place' task. When learning with IBC-RL, the learning of the second subtask 'manipulation' starts from 1200 its. The rewards have been regularized into the range [0, 1] for comparison. For each method, the mean and variance of the results of 5 repeated tests have been plotted in the figure.

and are used in the reward functions:

$$\begin{aligned} r_d &= \sum_{i=1}^{5} D(\boldsymbol{p}_{rf}[i] - \boldsymbol{p}_{rh}) \\ l_d &= \sum_{i=1}^{5} D(\boldsymbol{p}_{lf}[i] - \boldsymbol{p}_{lh}) \\ D(\boldsymbol{x}) &= \sqrt{\Sigma_{i=1}^{n} \left(\boldsymbol{x}[i]\right)^2}, \end{aligned} \tag{20}$$

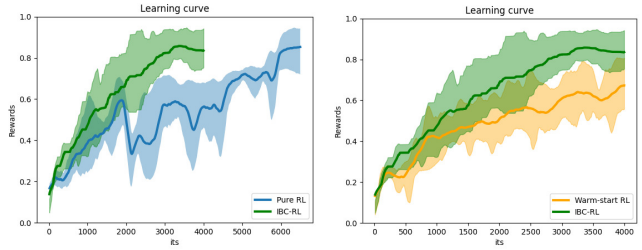
in which r_d is the distance between the right hand and the right handle of the object. $\boldsymbol{p}_{rf}$ is the array that stores the positions of five fingers of the right hand. $\boldsymbol{p}_{rh}$ is the position vector of the right handle of the object. l_d, $\boldsymbol{p}_{lf}$, and $\boldsymbol{p}_{lh}$ have similar meanings, but they are about the left hand and the left handle of the object.

1) ***Grasp and Place****:* In this task, robots need to put the cube into the cup. The reward function

$$\begin{aligned} r = &\exp(-10 \cdot r_d) + \exp(-10 \cdot l_d) \\ &+10 \cdot \exp(-10 \cdot D(\boldsymbol{p}_{rh} - \boldsymbol{p}_{lh})) \end{aligned} \tag{21}$$

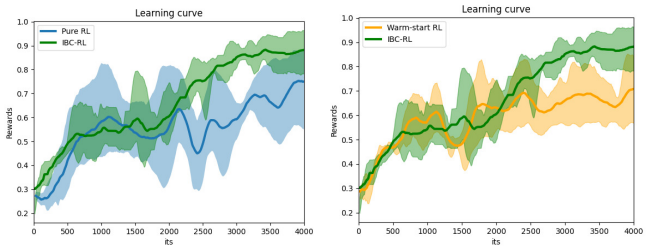
punishes the distances between the hands and the objects, and the distance between two objects. The results of the experiments are shown in Fig. 5.

Pure PPO controller is not intelligent enough to learn a valid policy by itself in difficult tasks like 'Grasp and Place' [5]. Due to the imperfection of human demonstrations, initializing the learning with entire demonstrations could only offer limited help. But with IBC-RL, robots are able to perfectly perform the task within 4000 iterations. They could roughly learn how to reach the objects after 1200 iterations. Then demonstrations of the 'manipulation' subtask are fed to robots. In the following iterations, they gradually learn how to put the cube into the cup and also optimize the detailed operations (See the recorded video, which visualizes the learning process [48]). After training, robots can learn a much better strategy than those demonstrated by human teachers. The movements of



(a) IBC-RL vs Pure RL **(b)** IBC-RL vs Warm-start RL

Fig. 6: The results of the 'Open the Scissor' task. When learning with IBC-RL, the learning of the second subtask 'manipulation' starts from 500 its. The rewards have been regularized into the range $[0, 1]$ for comparison. For each method, the mean and variance of the results of 5 repeated tests have been plotted in the figure.



(a) IBC-RL vs Pure RL **(b)** IBC-RL vs Warm-start RL

Fig. 7: The results of the 'Close the door' task. When learning with IBC-RL, the learning of the second subtask 'manipulation' starts from 1500 its. The rewards have been regularized into the range $[0, 1]$ for comparison. For each method, the mean and variance of the results of 5 repeated tests have been plotted in the figure.

robots are also more efficient and precise. The controllers trained with different methods are compared with each other (See the recorded video [49]).

It is worth noting that robots are still unable to learn specific behaviors with IBC-RL, like picking up the cube, even if human teachers have shown similar behaviors in demonstrations. Because such specific behavior is not defined in the reward function, robots will not receive any rewards for doing these human-like behaviors. They will learn a more direct and concise way to reach a higher reward.

2) ***Open the Scissor****:* In this task, the robot needs to open a scissor that is initialized to be closed. The reward function

$$r = 2 - r_d - l_d + (0.59 + \theta_o) \cdot 5 \tag{22}$$

punishes the distances between the hands and the scissor handles and awards the distance between the two handles. θ_o is the joint angle of the scissor, which will increase when opening the scissor. The results of the experiments are shown in Fig. 6.

Fig. 6a compares the learning processes with pure RL and IBC-RL. Pure PPO controller is able to learn a valid policy but requires about 6000 iterations. With IBC-RL, it could achieve the same performance with only about 3000 iterations. Also, by replaying the demonstrations of subtasks, the learning becomes more stable without many performance drops. On the other hand, the performance of a pure PPO controller drops every time the exploration gets stuck or goes wrong. Fig. 6b compares the learning processes with warm-start PPO and IBC-RL. Apparently, our framework outperforms the other one which cannot converge within 4000 iterations and suffers from performance drops.

In this task, the contact of robot hands and objects is not steady during the operations, which increases the uncertainty of explorations and may also mislead the learning. Replaying demonstrations to correct the direction of exploration could offer more help in this case.

3) ***Close the Door****:* In this task, robots need to close the doors which are initialized to be open. The reward function

$$r = 3 - r_d - l_d - |h_{rh} - h_{lh}| \cdot 2 \tag{23}$$

punishes the distances between the hands and the door handles, and the distance between the two handles. Robots need to continuously apply forces on the doors to keep them closed. h_{rh} and h_{lh} are the horizontal positions of the right handle and left handle of the door, respectively. The results of the experiments are shown in Fig. 7.

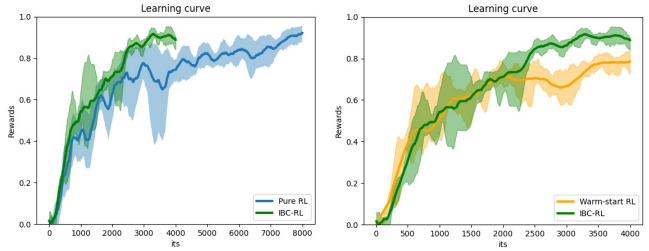
Fig. 7a compares the learning processes with pure RL and IBC-RL in the 'Close the door' task. This task is simpler than 'Open the Scissor,' and the pure PPO controller can roughly learn the task after about 5000 iterations, but it cannot stably perform the task. And it is still much slower than IBC-RL, which could perfectly perform the task after only 3000 iterations. Also, the learning processes of IBC-RL are more stable with fewer performance drops. Fig. 7b compares the learning processes with warm-start PPO and IBC-RL. Initializing the controller with entire demonstrations nearly offers no help to the learning here.

In this task, demonstrations given by human teachers do not respect the definition of the reward function. In demonstrations, we use hands to directly push the door rather than pushing the handles, which may be a more natural way for most people. This could explain why the average performance of the controller with IBC-RL could not outperform the others when learning the first subtask. The learning is guided by the reward function, meaning that robots cannot utilize human knowledge in the first subtask and must learn by themselves. For the second subtask, robots could learn from human demonstrations again, and thus the learning has been accelerated.

4) ***Swing the Cup****:* In this task, robots need to hold the cup handles and rotate the cup 90 degrees. The reward function

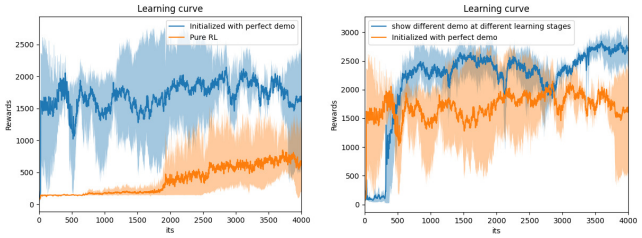
$$\begin{aligned} rot &= 2 \cdot \arcsin(D(\boldsymbol{q}_d)) \\ r &= -r_d - l_d + 1/(|rot| + 0.1) \cdot 5 - 1 \end{aligned} \tag{24}$$

punishes the distances between the hands and the cup handles and the difference between the current pose and the target pose of the cup. $\boldsymbol{q}_d$ is the difference between the quaternions of the object rotation and the target rotation. Robot hands are initialized near the cup handles, and robots do not need to switch their hands for a rotation of 90 degrees, which makes this task easier. The results of the experiments are shown in



(a) IBC-RL vs Pure RL **(b)** IBC-RL vs Warm-start RL

Fig. 8: The results of the 'Swing the Cup' task. When learning with IBC-RL, the learning of the second subtask 'manipulation' starts from 800 its. The rewards have been regularized into the range $[0, 1]$ for comparison. For each method, the mean and variance of the results of 5 repeated tests have been plotted in the figure.



(a) Entire demo vs Pure RL **(b)** Segmented vs Entire

Fig. 9: The left figure compares the learning processes of pure PPO (orange line) and warm-starting PPO with demonstrations of the entire task (blue line). The right figure compares the learning processes of warm-starting PPO with demonstrations of the entire task (orange line) and PPO with demonstrations for subtasks (blue line)

Fig. 8.

Fig. 8a compares the learning processes with pure RL and IBC-RL. In simple tasks like 'Swing the Cup', a pure PPO controller is already able to learn the task quite fast. Therefore, IBC-RL could not accelerate the learning much at an early stage, but it could still help with the convergence. Human demonstrations could help the controller learn how to get close to the target position more effectively and keep the cup in the target pose, which saves about two or three thousand iterations of learning. Fig. 8b compares the learning processes with warm-start PPO and IBC-RL. It again proves that IBC-RL is able to solve the forgetting problem and always enables robots to utilize human knowledge during learning.

B. Ablation studies

Some ablation studies are done to validate the main problems in designing the learning framework and support the main choices we made.

1) ***Human demos gradually vanish during the RL process:*** This is a common problem when integrating human knowledge with the RL [3], [35]. If only imitating human demonstrations before the RL starts, the robot could not precisely repeat the movements shown in the demonstrations again after RL and thus could not reach a higher reward. In other words, the knowledge learned from demonstrations will be gradually forgotten in the RL. In the beginning, entire demonstrations are shown to the robot, but it could only utilize the initial part of them.

We proved that this problem does exist and could be alleviated by learning from demonstrations of subtasks instead of learning the entire task in a single time. We use the expert demonstrations collected with a well-trained controller, which can demonstrate a near-optimal policy. In the 'Grasp and Place' task, the segmented demonstrations have been proven to be able to offer more help and result in a controller with better performance (See Fig. 9).

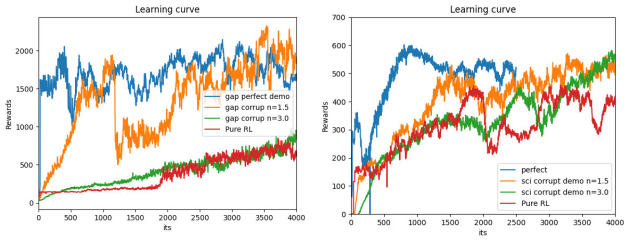
Fig. 9a compares the learning processes of pure PPO and that initialized with the demonstrations of the entire task. Pure PPO is not able to learn the task. The learning initialized with complete expert demonstrations is able to converge within several hundred iterations. However, the robot fails to learn how to perform the second half of the task well, which is moving the cube exactly into the cup. And with more explorations in the following RL iterations, the performance becomes less stable. On average, the trained robot could not learn to perform the task well as the demonstrations did. The second half of the demonstrations is gradually forgotten by robots during RL. Similar findings are shown in [3]. In the experiment where the robot needs to grab a hammer and hit on a nail, it could hold the hammer but cannot precisely hit on the nail after RL.

Fig. 9b compares the learning processes initialized with the demonstrations of the entire task and learns two subtasks ('reach' and 'manipulation') sequentially. In the 'reach' subtask, we show the robot how to put its hands on the cup and the cube, respectively. After it learns for 300 iterations with RL, it will move to the 'manipulation' subtask, and we will then show it how to put the cube into the cup. In this way, the learning process becomes more stable and could reach a higher reward. The performance of the controller after training is comparable to the expert demonstrations.

2) ***Effect of action noises:*** We studied the influence of action noises on the learning processes by corrupting expert demonstrations. At each step, a new action is sampled within a multivariate normal distribution centered at the optimal action of the current state. The variance of the distribution is measured by a parameter n. Actions carried out by robots during the explorations will be noisier with a larger n. The degree of noise on expert demonstrations is gradually increased in the 'Grasp and Place (gap)' task (See Fig. 10a) and the 'Open the Scissor (sci)' task (See Fig. 10b). In both tasks, demonstrations with more noise contribute less to learning. With $n = 3.0$, they almost provide no help at all.

The imperfect demonstrations considered in our work contain not only random sensing noises but also suboptimal strategies and other unnecessary movements, which makes it even more difficult for robots to utilize human knowledge.

3) ***Multiple human demos or single human demo:*** Learning from a single human demonstration will make the controller overfit a suboptimal strategy and the noises, which increases the workload of later optimizations through RL. On the other hand, robot learning from multiple human demon-



(a) Grasp and Place (gap) **(b)** Open the Scissor (sci)

Fig. 10: Comparison of the learning processes with different degrees of noise (n from 0 to 3) in the tasks 'Grasp and Place (gap)' (left) and 'Open the Scissor (sci)' (right). Demonstrations are helpless when $n = 3$ (The green line)

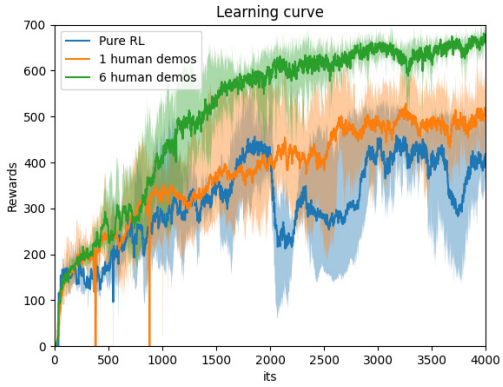


Fig. 11: Comparison of the learning processes of the robot that learns with multiple human demonstrations and with a single demonstration. In the green curve, robots learn with all six demonstrations, and the green shade indicates the variance of the results of repeated tests. In the orange curve, robots learn with only one of the six demonstrations, and the orange shade indicates the variance of the learning results with different demonstrations.

strations may suffer from conflicted labels [50]. Therefore, several tests have been done with the 'Open Scissor' task to validate our choice.

Fig. 11 compares the learning processes with six human demonstrations and with one of them each time. Training with multiple human demonstrations outperforms that with a single demonstration, even compared with the best one with the highest reward. The learning processes with multiple demonstrations in Fig. 11 are also more stable since the learning is heavily influenced by the quality of the demonstration when only a single one is used. In the 'Grasp and Place' task, the controller cannot even learn anything with only a single human demonstration since the strategy demonstrated for this task is far from the optimal one with a pretty low reward. The results of the experiments in Fig. 11 also prove the fact that with multiple human demonstrations, robots will be able to learn the general direction for further explorations, which could reduce the influence of conflicted labels and avoid focusing too much on the details of each imperfect demonstration.

4) ***Entire demo or demo for the subtasks****:* In previous research, which works with expert demonstrations [3], [35], demonstrations of the entire tasks are fed to the robot and

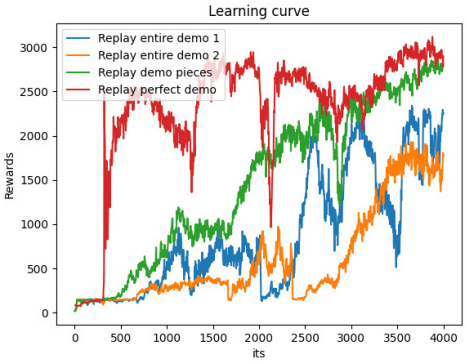


Fig. 12: Comparison of replaying demonstrations of the entire task and those of the subtasks. In the red curve, perfect demonstrations are replayed around 1300 and 2200 iterations. In the green curve, demonstrations of the first subtask are fed to robots at the beginning. Demonstrations of the second subtask are fed to robots at 400 its and then replayed at 2800 iterations. In blue and orange curves, human demonstrations of the entire task are fed to robots at the beginning and then replayed around 2000 and 2400 iterations, respectively.

used for policy updates during the learning. Several tests have been done to determine if it is still appropriate to replay entire demonstrations or if it is better to replay the demonstrations of the subtasks (see Fig. 12).

When robots are retrained with perfect demonstrations (red curve in Fig. 12), they could recover soon from the interruptions and continue to learn. However, replaying entire demonstrations severely corrupts learning when robots are fed with imperfect human demonstrations. It needs much more iterations to recover (blue and orange curves in Fig. 12). Every time the demonstrations are fed to robots, the learning suffers from the noise signals, which significantly slows down the learning process. After each interruption, the learning requires more than 500 RL iterations to recover. On the other hand, replaying demonstrations for the subtasks avoids feeding unnecessary noisy data repeatedly to robots and thus does not corrupt the learning too much (green curve in Fig. 12). The learning could recover soon from corruption and avoid unnecessary interactions with the environment. It can recover within 100 or 200 iterations in our tests. Therefore, when learning with imperfect human demonstrations, replaying demonstrations of the subtasks could be a better choice, which can help the controller get rid of the local minima while also having less interruption on the learning.

C. Limitations of the framework

1) ***Imperfect demonstrations make fewer improvements on simple tasks****:* Unlike the expert ones, imperfect human demonstrations could only provide general guidance to the controller during the explorations, which means the learning still heavily relies on the explorations of RL. Therefore, as shown in Fig. 8, IBC-RL could not accelerate the learning much at an early stage for simple tasks like 'Swing the Cup.' But the imperfect demonstrations could still be useful to accelerate the convergence of learning to reach the target pose and hold the cup there.

*2) **Human demonstrations may not follow the definition of the reward function**:* The strategies shown in human demonstrations result in human-like behaviors, but they may be given low scores if assessed with the reward function. For example, in the 'Close the Door' task, human teachers may prefer to close the door with the hands put directly on the body of the doors, while the reward function requires the hands to be fixed on the handles. Since our method still relies on the exploration of RL, which is guided by the reward function, the first part of the human demonstrations (put the hands on the door) is actually an error to the controller which needs to be fixed in the later learning iterations. In this task, as shown in Fig.7, the learning could not be accelerated in the first 2000 iterations since the controller needs to fix the error and learn how to push the door with the hands on the handles by itself.

*3) **More expressive reward functions are necessary**:* With the LfD methods, robots are able to learn almost the same behaviors from human teachers. Again, since IBC-RL could not learn a good enough policy directly from human demonstrations and still relies on the exploration of RL, the reward function fully decides what the controller is able to learn, which may cause trouble in some cases. For example, in the 'Grasp and Place' task, it could not learn how to pick up the cube since the reward function only punishes the distance between the hands and the objects, and the robot could already get the highest reward without holding the cube in hand. Similarly, in the 'Open the door' task, the robot could not learn to hold the handles even if we show this behavior in demonstrations.

VI. CONCLUSIONS

We have proposed a new learning framework (IBC-RL) that enables robots to utilize imperfect human demonstrations in RL. Under this framework, robots gradually learn all the subtasks in sequence. Related demonstrations are fed to robots just as guidance and only when necessary to avoid repeated corruption to the learning. With four dexterous bimanual manipulation tasks, we have demonstrated that IBC-RL could enable robots to learn much faster and steadier. It could even empower robots to perform more complicated tasks that cannot be learned with pure RL or other RL-based methods when expert demonstrations are unavailable. Our work could release the reliance on human experts and accurate human motion-tracking devices to a certain degree.

According to the results of the experiments, there are also a few limitations of IBC-RL that one needs to be aware of before applying it. The learning with this framework still relies on the explorations of RL, which means the reward function fully determines what the robot can learn and what it will try to learn. Therefore, one could not expect the robot to master the skills that are not defined in the reward function, even if human teachers have demonstrated them. For the same reason, the demonstrations fed to the robot should also respect the reward function. Otherwise, they will be useless or even harmful to learning. The robot could get only general guidance for later explorations from imperfect demonstrations. Therefore, they may not be very helpful in tasks that are easy to learn.

In our work, we have primarily demonstrated the efficiency of the IBC-RL framework when working with imperfect demonstrations. It still has great potential that could be further developed. In the future, it could be applied to more complex manipulation tasks, which could be decomposed into more subtasks since the step-by-step learning method of IBC-RL is practically more suitable for long-horizon tasks. Robots could also break the requirement for expert teachers and accurate teaching interfaces and learn various complex tasks with all kinds of human demonstrators. The framework could be further developed by replacing BC with a more effective method that could enable robots to benefit from very different suboptimal strategies demonstrated by human teachers. By taking the average, BC cannot efficiently deal with the different suboptimal strategies since many details have been abandoned. In the meantime, more tests need to be done to further evaluate and optimize this learning framework. For example, it should be tested with other human-robot interfaces and more demonstrators and also compared with other learning frameworks.

REFERENCES

[1] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[2] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[3] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[4] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.

[5] Y. Chen, Y. Yang, T. Wu, S. Wang, X. Feng, J. Jiang, Z. Lu, S. M. McAleer, H. Dong, and S.-C. Zhu, "Towards human-level bimanual dexterous manipulation with reinforcement learning," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. [Online]. Available: https://openreview.net/forum?id=D29JbExncTP

[6] F. Xie, A. Chowdhury, M. De Paolis Kaluza, L. Zhao, L. Wong, and R. Yu, "Deep imitation learning for bimanual robotic manipulation," *Advances in neural information processing systems*, vol. 33, pp. 2327–2337, 2020.

[7] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*. PMLR, 2020, pp. 1101–1112.

[8] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6818–6827.

[9] "Isaac gym - preview release," Jul 2022. [Online]. Available: https://developer.nvidia.com/isaac-gym

[10] C. Yu and P. Wang, "Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: a review," *Frontiers in Neurorobotics*, vol. 16, 2022.

[11] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[12] A. Plaat, W. Kosters, and M. Preuss, "High-accuracy model-based reinforcement learning, a survey," *Artificial Intelligence Review*, pp. 1–33, 2023.

[13] M. T. Mason and J. K. Salisbury Jr, "Robot hands and the mechanics of manipulation," 1985.

[14] A. Bicchi and R. Sorrentino, "Dexterous manipulation through rolling," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1995, pp. 452–457.

[15] L. Han, Y.-S. Guan, Z. Li, Q. Shi, and J. C. Trinkle, "Dextrous manipulation with rolling contacts," in *Proceedings of International Conference on Robotics and Automation*, vol. 2. IEEE, 1997, pp. 992–997.

[16] M. Cherif and K. K. Gupta, "Planning quasi-static fingertip manipulations for reconfiguring objects," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 837–848, 1999.

[17] Y. Aiyama, M. Inaba, and H. Inoue, "Pivoting: A new method of graspless manipulation of object by robot fingers," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, vol. 1. IEEE, 1993, pp. 136–143.

[18] L. Han and J. C. Trinkle, "Dextrous manipulation by rolling and finger gaiting," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 1. IEEE, 1998, pp. 730–735.

[19] K. Hertkorn, M. A. Roa, and C. Borst, "Planning in-hand object manipulation with multifingered hands considering task constraints," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 617–624.

[20] Q. Lu, N. Baron, A. B. Clark, and N. Rojas, "Systematic object-invariant in-hand manipulation via reconfigurable underactuation: Introducing the ruth gripper," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1402–1418, 2021.

[21] C. Konnaris, C. Gavriel, A. A. Thomik, and A. A. Faisal, "Ethohand: A dexterous robotic hand with ball-joint thumb enables complex in-hand object manipulation," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*. IEEE, 2016, pp. 1154–1159.

[22] D. Andronas, S. Xythalis, P. Karagiannis, G. Michalos, and S. Makris, "Robot gripper with high speed, in-hand object manipulation capabilities," *Procedia CIRP*, vol. 97, pp. 482–486, 2021.

[23] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "Planar in-hand manipulation via motion cones," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 163–182, 2020.

[24] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," in *Robotics Research*. Springer, 2020, pp. 523–539.

[25] M. Pfanne, M. Chalon, F. Stulp, H. Ritter, and A. Albu-Schäffer, "Object-level impedance control for dexterous in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2987–2994, 2020.

[26] Q. Li, C. Elbrechter, R. Haschke, and H. Ritter, "Integrating vision, haptics and proprioception into a feedback controller for in-hand manipulation of unknown objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2466–2471.

[27] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1578–1585.

[28] H. Van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 121–127.

[29] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[31] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*. PMLR, 2022, pp. 297–307.

[32] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

[33] P. Rivera, J. Oh, E. Valarezo, G. Ryu, H. Jung, J. H. Lee, J. G. Jeong, and T.-S. Kim, "Reward shaping to learn natural object manipulation with an anthropomorphic robotic hand and hand pose priors via on-policy reinforcement learning," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 167–171.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[35] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[36] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 921–928.

[37] A. Gupta, C. Eppner, S. Levine, and P. Abbeel, "Learning dexterous manipulation for a soft robotic hand from human demonstrations," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3786–3793.

[38] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[39] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[40] B. Burchfiel, C. Tomasi, and R. Parr, "Distance minimization for reward learning from scored trajectories," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[41] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[42] R. Jeong, J. T. Springenberg, J. Kay, D. Zheng, Y. Zhou, A. Galashov, N. Heess, and F. Nori, "Learning dexterous manipulation from suboptimal experts," *arXiv preprint arXiv:2010.08587*, 2020.

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

[44] S. Amarjyoti, "Deep reinforcement learning for robotic manipulation-the state of the art," *arXiv preprint arXiv:1701.08878*, 2017.

[45] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 1395–1476, 2021.

[46] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 1503–1510.

[47] V. Tangkaratt, B. Han, M. E. Khan, and M. Sugiyama, "Variational imitation learning with diverse-quality demonstrations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 9407–9417.

[48] Z. Du, "Training process of ibc-rl with 'grasp and place' task," Feb 2023. [Online]. Available: https://youtu.be/LdILg6l2bEQ

[49] ——, "The comparison of the controllers trained with three different methods," Feb 2023. [Online]. Available: https://youtu.be/p4yV-gE0H5w

[50] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 617–624.

A

Appendix - Demonstration Collection

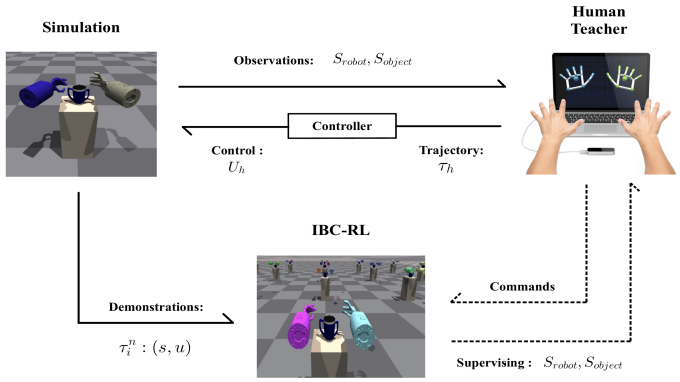


Figure A.1: Overview of the teaching scheme

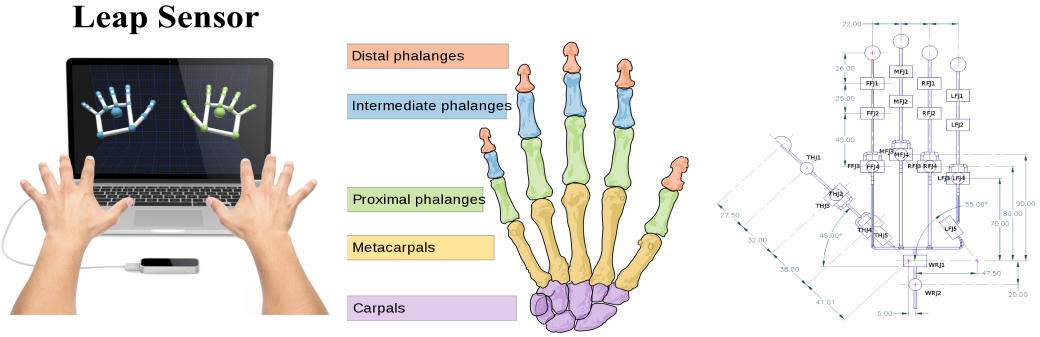
The general teaching scheme is shown in Fig. A.1. When performing demonstrations for a task, one unit of robot hands and the object is simulated with Isaac Gym. By observing the state of the robot's hands S_{robot} and the object S_{object}, the human teacher moves his hands to complete the task. The trajectories of the human hands τ_h are recorded and followed by the robot by taking actions U_h. Some transformations on the collected trajectory are needed here to compensate for the morphology difference. In the meantime, the states and actions of the robot τ_i^n are recorded as demonstrations for task learning with IBC-RL. Human teachers could supervise the learning process and interrupt the learning by sending commands, such as replaying demonstrations, starting the next subtask, etc. If necessary, new demonstrations could be recorded for robots.

When teleoperating the simulated robot hands, a trajectory of human hands $\tau_h : \{X_1, X_2, ..., X_n\}$ is recorded by the motion-tracking device. In the meantime, the state changes of human hands $\triangle X = X_n - X_{n-1}$ are sent to the controller for state tracking. In this way, the simulated robot hands could repeat the movements of human hands simultaneously. Sec. A.1 introduces in detail how the trajectories of human hands are recorded and the transformations made for the morphology difference. Sec. A.2 explains how the robot follows the trajectories and how the demonstrations for later learning are recorded.

A.1 Trajectory Recording

A.1.1 Human-Robot Interface

A Leap Sensor is used as the human motion tracking device in our experiments [3]. As shown in Fig. A.2a, it captures the real-time movements of human hands and repeats them with a pair of modeled hands without noticeable delay. The hand model in Leap Sensor has a similar structure compared with the human hand shown in Fig. A.2b. Without much morphology difference, the repeated movements are



(a) Leap Sensor **(b)** Human hand **(c)** Robot hand

Figure A.2: Leap sensor and its hand model are shown in Fig. A.2a. The structure of a human hand is shown in Fig. A.2b [1]. And the structure of the robot hand simulated with Isaac Gym is shown in Fig. A.2c [2]

close to the original ones shown by human demonstrators. However, due to the limited sensing area and the inaccuracy of sensing with infrared rays, it is hard for human teachers to perform optimal strategies, and the collected data are noisy.

We built our work based on the official ROS driver for the Leap Motion Controller [4] and further extended their work for motion tracking of both hands simultaneously. The states X_h of the hand models are represented with the Cartesian coordinates of the 52 joints and the orientations of two palms. The message containing these states is published at 100 Hz, and our trajectory generator, which subscribes to the publisher, receives this message at 60 Hz. A recorded trajectory is formed with a sequence of states $\tau' = \{X'_1, X'_2, ..., X'_n\}$, which are expressed in the coordinate frame of the Leap Sensor.

A.1.2 State Transformation

The transformations of the collected states are necessary mainly for two reasons. First, the structure of the hand models in Leap Sensor is different from that of the simulated shadow robot hand [2] as shown in Fig. A.2c. Such morphology difference needs to be compensated for. Otherwise, the collected demonstrations will be meaningless and not instructive enough. Second, most of the actuators on robot hands only generate torques and control the rotations of the joints. However, only positions of the joints are available from the Leap sensor, and certain analytical computations are necessary to acquire their angles.

The cosine of the angle of joint b could be derived from its coordinates and the coordinates of the two joints a, c next to it doing the inner product between the two vectors $\vec{bc}$ and $\vec{ba}$. Thus,

$$\begin{aligned} cos(\theta_b) &= cal\_angle(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \\ &= \frac{(x_c - x_b) \cdot (x_a - x_b) + (y_c - y_b) \cdot (y_a - y_b) + (z_c - z_b) \cdot (z_a - z_b)}{\sqrt{(x_c - x_b)^2 + (y_c - y_b)^2 + (z_c - z_b)^2} \cdot \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2}}, \end{aligned} \tag{A.1}$$

in which (x_a, y_a, z_a) are the 3-D coordinates of joint a, (x_b, y_b, z_b) of b, and (x_c, y_c, z_c) of c.

All the joint angles on fingers are calculated with eq. A.1 and then minus π to match the expressions in the robot hand system. The joints THJ4, FFJ3, MFJ3, RFJ3, and LFJ3 rotate fingers horizontally and are approximated using eq. A.1 but with the intermediate joint of the same finger and the proximal joints of the finger next to it. The results are further tuned manually to compensate for the morphology difference.

The angles of joints THJ2 and THJ3 could not be calculated accurately enough since their movements will affect each other. They are assigned with fixed values $\theta_{THJ2} = 0, \theta_{THJ3} = 1.1$. The coordinates and orientations of the palms remain unchanged. At last, all the angles are trimmed into the ranges of the joint movement. The calculations are shown in the table. A.1. After transformations, each state X_n contains 40 angles of the joints and also the coordinates and orientations of two hands. The collected states $\{X_1, X_2, ..., X_n\}$ are then passed to the controller sequentially in time for the trajectory following.

Table A.1: Transformations of states

Joints	Calculation	Range
THJ0	$cal\_angle(thumb\_tip, thumb\_distal, thumb\_intermediate) - \pi$	[-1.571, 0]
THJ1	$cal\_angle(thumb\_distal, thumb\_intermediate, thumb\_proximal) - \pi$	[-0.524, 0.524]
THJ2	0.0	[-0.209, 0.209]
THJ3	1.1	[0, 1.222]
THJ4	$cal\_angle(thumb\_intermediate, thumb\_proximal, palm\_normal) + 1.1$	[-1.047, 1.047]
FFJ1	$cal\_angle(index\_distal, index\_intermediate, index\_proximal) - \pi$	[0, 1.571]
FFJ2	$cal\_angle(index\_intermediate, index\_proximal, index\_metacarpal) - \pi$	[0, 1.571]
FFJ3	$cal\_angle(index\_intermediate, index\_proximal, middle\_proximal) - 1.71$	[-0.349, 0.349]
MFJ1	$cal\_angle(middle\_distal, middle\_intermediate, middle\_proximal) - \pi$	[0, 1.571]
MFJ2	$cal\_angle(middle\_intermediate, middle\_proximal, middle\_metacarpal) - \pi$	[0, 1.571]
MFJ3	$cal\_angle(middle\_intermediate, middle\_proximal, index\_proximal) + 1.57$	[-0.349, 0.349]
RFJ1	$cal\_angle(ring\_distal, ring\_intermediate, ring\_proximal) - \pi$	[0, 1.571]
RFJ2	$cal\_angle(ring\_intermediate, ring\_proximal, ring\_metacarpal) - \pi$	[0, 1.571]
RFJ3	$cal\_angle(ring\_intermediate, ring\_proximal, middle\_proximal) + 1.3$	[-0.349, 0.349]
LFJ1	$cal\_angle(pinky\_distal, pinky\_intermediate, pinky\_proximal) - math.pi$	[0, 1.571]
LFJ2	$cal\_angle(pinky\_intermediate, pinky\_proximal, pinky\_metacarpal - \pi$	[0, 1.571]
LFJ3	$cal\_angle(pinky\_intermediate, pinky\_proximal, middle\_proximal) + 1.3$	[-0.349, 0.349]
LFJ4	$cal\_angle(pinky\_proximal, palm\_pos, palm\_normal) + 1.0$	[0, 0.785]
WRJ0	$-palm\_rot[1] * 1.5$	[-0.698, 0.489]
WRJ1	$-palm\_rot[0] * 0.5 - 0.5$	[-0.489, 0.14]

A.2 Replicate movements in simulation

The robot hands follow the trajectories with PD controllers, as shown in Fig. A.3, where the control signals are calculated with the state errors and their derivatives as

$$\begin{aligned} u_t &= K_p \cdot e(s_t) + K_d \cdot \frac{d}{dt} e(s_t) \\ e(s_t) &= e_h(s_t) + e_s(s_t), \quad e_r(s_t) = \triangle X = X_{t+1} - X_t \end{aligned} \tag{A.2}$$

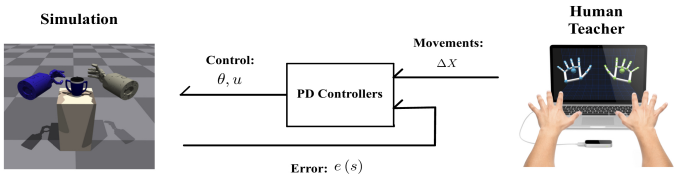


Figure A.3: Trajectory following with PD controllers

in which K_p and K_d are weighting constants of the proportional and derivative sections, respectively. $e_h(s_t)$ defines the target movements that should be made at time step t. And $e_r(s_t)$ is the residual error in previous movements at time step t. PD controllers are easy to implement without much manual tuning of the parameters. They could enable robots to follow the given state sequences and do not need any additional knowledge or a model of the system. However, the control is not optimal, and delays are inevitable. This is one of the reasons for the suboptimal strategies in demonstrations. But it could be ignored compared with other factors considered in our experiments.

Our work uses the Issac Gym engine, and relevant packages for simulation [5]. They already provide a function ($gym.set\_dof\_position\_target\_tensor(\theta)$) to control the rotations of all the joints with inner-defined PD controllers. Therefore, we use this function to follow the states of rotating joints by providing the target angles. For the control of the movements and rotations of entire robot hands, we built PD controllers as shown in table. A.2.

Table A.2: Parameters of PD controllers

	K_p	K_d
Position	1000	200
Rotation	3000	500

The controllers for the movements in the x, y, and z axes change the positions of robot hands, in which $K_p = 1000, K_d = 200$. The controllers for the 'roll,' 'yaw,' and 'pitch' rotations change the orientations of robot hands, in which $K_p = 3000, K_d = 500$. After calculation, all the control signals are regularized into [-1,1]. At each time step, $e(s_t) = e_h(s_t) + e_r(s_t)$ is taken as the input of PD controllers, and the control signals θ, u are passed to the robot. The robot will repeat his movements when the human demonstrator performs the task. In this process, the state and action pairs (s, u) of the robot are recorded at 60 Hz and then used for learning. See the recorded video [6], which is an example of recorded demonstrations for the 'Grasp and Place' task.

The collected demonstrations are imperfect in the experiments mainly for the following reasons. First, they are collected from non-expert users of Leap Sensor without much practice. The limited sensing area and the transformation errors have increased the difficulty in controlling the simulated robot hands. Therefore, unnecessary movements and suboptimal strategies are inevitable. Second, the human demonstrator has intentionally shown different strategies. Unfortunately, we did not invite many demonstrators for the experiments since the focus of our work is not to check if humans are good at demonstrating but to deal with various demonstrations they may give. The demonstrations were collected from two demonstrators, and they tried to show different but reasonable and human-like strategies. At last, the performance of the demonstrators is also restricted by the limited control horizon.

References

[1] "Dactyly," Jan 2023. [Online]. Available: https://en.wikipedia.org/wiki/Dactyly

[2] "Shadow dexterous hand series - research and development tool," Nov 2022. [Online]. Available: https://www.shadowrobot.com/dexterous-hand-series/

[3] Ultraleap, "Tracking: Leap motion controller." [Online]. Available: https://www.ultraleap.com/product/leap-motion-controller/

[4] warp1337, "Warp1337/rosleapmotion: Leap motion ros integration." [Online]. Available: https://github.com/warp1337/rosleapmotion

[5] "Isaac gym - preview release," Jul 2022. [Online]. Available: https://developer.nvidia.com/isaac-gym

[6] Z. Du, "An example of demonstrations shown to the robot in 'grasp and place' task," Feb 2023. [Online]. Available: https://youtu.be/G2ZaoOWVPpA