Learning to Engage: An Application of Deep Reinforcement Learning in Living Architecture Systems

by

Lingheng Meng

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Matthew Gombolay Assistant Professor School of Interactive Computing Georgia Institute of Technology
Supervisors:	Dana Kulić Professor Dept. of Electrical and Computer Engineering University of Waterloo — Monash University
	Rob Gorbet Associate Professor Dept. of Knowledge Integration Dept. of Electrical and Computer Engineering University of Waterloo
Internal Members:	Stephen L. Smith Professor Dept. of Electrical and Computer Engineering University of Waterloo
	Mark Crowley Associate Professor Dept. of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Edith Law Associate Professor David R. Cheriton School of Computer Science University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Lingheng was the sole author of **Chapter 1**, **2**, **3**, **9** and **10** which were written under the supervision of Dr. Dana Kulić and Dr. Rob Gorbet and were not written for publication. Exceptions to sole authorship of material are as follows:

Chapter 4
The two physical testbeds, namely *Aegis Canopy* and *Meander* are developed by Philip Beesley Studio Inc.[1]. For the LAS Simulation Toolkit, the LAS-Behavior-Engine is developed by the engineer team of Philip Beesley Studio Inc. and their collaborators. The author of this thesis solely developed the rest of the simulation toolkit.

Chapter 5:
This research was conducted at the Royal Ontario Museum by Lingheng Meng and Daiwei Lin under the supervision of Dr. Dana Kulić, and was published in the ACM Transactions on Human-Robot Interaction (THRI). Lingheng Meng implemented Single Agent Raw Action Space (SARA) and Agent Community Raw Action Space (ACRA) and contributed to the data analysis, while Daiwei Lin implemented Parameterized Learning Agent (PLA).

Lingheng Meng, Daiwei Lin, Adam Francey, Rob Gorbet, Philip Beesley, and Dana Kulić (2020). *Learning to Engage with Interactive Systems: A Field Study on Deep Reinforcement Learning in a Public Museum.* J. Hum.-Robot Interact. 10, 1, Article 5 (March 2021), 29 pages.
DOI: 10.1145/3408876

Chapter 6
A version of this chapter was published in the proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR).

Lingheng Meng, Rob Gorbet and Dana Kulić (2021). *The Effect of Multi-step Methods on Overestimation in Deep Reinforcement Learning.* 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 2021, pp. 347-353.
DOI: 10.1109/ICPR48806.2021.9413027.

Chapter 7
A version of this chapter was published in the proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[1] https://www.philipbeesleystudioinc.com

Lingheng Meng, Rob Gorbet and Dana Kulić (2021). *Memory-based Deep Reinforcement Learning for POMDPs.* 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 2021, pp. 5619-5626,
DOI: 10.1109/IROS51168.2021.9636140.

Abstract

Physical agents that can autonomously generate engaging, life-like behavior will lead to more responsive and interesting robots and other autonomous systems. Although many advances have been made for one-to-one interactions in well controlled settings, future physical agents should be capable of interacting with humans in natural settings, including group interaction. In order to generate engaging behaviors, the autonomous system must first be able to estimate its human partners' engagement level, then take actions to maximize the estimated engagement. In this thesis, we take Living Architecture Systems (LAS), architecture scale interactive systems capable of group interaction through distributed embedded sensors and actuators, as a testbed and apply Deep Reinforcement Learning (DRL) by treating the estimate of engagement as a reward signal in order to automatically generate engaging behavior. However, applying DRL to LAS is difficult, because of DRL's low data efficiency, overestimation problem, and issues with state observability, especially considering the large observation and action space of LAS.

We first propose an approach for estimating engagement during group interaction by simultaneously taking into account active and passive interaction, and use the measure as the reward signal within a reinforcement learning framework to learn engaging interactive behaviors. The proposed approach is implemented in a LAS in a museum setting. We compare the performance of the learning system to that of a baseline design using prescripted interactive behavior. Analysis based on sensory data and survey data shows that adaptable behaviors within an expert-designed action space can achieve higher engagement and likeability. However, this initial approach relies on a manually defined reward and assumes a known, concise definition of the state and action space to address issues of slow learning, sample efficiency and state/action specification of DRL.

To relax these restrictive assumptions, we first analyze the effect of multi-step methods on alleviating the overestimation problem in DRL, and building on top of Deep Deterministic Policy Gradient (DDPG), propose Multi-step DDPG (MDDPG) and Mixed Multi-step DDPG (MMDDPG). Empirically, we show that both MDDPG and MMDDPG are significantly less affected by the overestimation problem than vanilla DDPG, which consequently results in better final performance and learning speed. Then, to handle Partially Observable Markov Decision Processes (POMDPs), we propose Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3) by introducing a memory component to TD3, and compare its performance with other DRL algorithms in both MDPs and POMDPs. Our results demonstrate the significant advantages of the memory component in addressing POMDPs, including the ability to handle missing and noisy observation data. After that, we investigate partial observability as a potential failure source

of applying DRL to robot control tasks, which can occur when researchers are not confident whether the observation space fully represents the underlying state. We compare the performance of TD3, Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) under various partial observability conditions, and find that TD3 and SAC become easily stuck in local optima and underperform PPO. We propose multi-step versions of the vanilla TD3 and SAC to improve their robustness to partial observability.

Based on our study with manually designed reward functions, which is the estimate of engagement, and the fundamental research on DRL, we further reduce the reliance on designers' field knowledge, and propose to learn a reward function from human preferences on engaging behavior by taking advantage of preference learning algorithms. Our simulation results show that the reward function induced from human preference is able to lead to a policy that generates engaging behavior.

Acknowledgements

First of all, I would like to thank my supervisors, Prof. Dana Kulić and Prof. Rob Gorbet, for the great opportunity and precious experience to be guided by and work with them over the years. They have been always supportive, patient and inspiring to me in both academics and life. I sincerely appreciate your encouragement and patience when the experiment did not go well, and admire your optimism and passion to research. You not only teach me how to do research and think critically, but also set a role model for me on how to be a good supervisor, how to collaborate with others, how to make contributions to the society, and how to balance work and life, etc. A special thank goes to Prof. Dana Kulić for the wonderful visiting opportunity to Monash University in Australia.

I would also want to thank the members of my committee, Prof. Matthew Gombolay, Prof. Edith Law, Prof. Stephen L. Smith and Prof. Mark Crowley, for their professional and illuminating questions as well as their inspiring and motivating feedback throughout my PhD process, all of which significantly improved the final thesis. I want to thank you all for taking the time to review my thesis and for being there for me at each milestone.

I am appreciative to our collaborators from Philip Beesley Studio Inc., Prof. Philip Beesley, Adam Francey, Michael Lancaster, Matt Gorbet, Kevan Cress, etc., for providing the experiment testbeds and planning the fantastic workshops and symposia. The Royal Ontario Museum, Tapestry Hall, and our research participants, who made this work possible, are especially appreciated for enabling us to conduct our field study and for their kind assistance.

I would like to express my gratitude to all great members in Dana's research group at both University of Waterloo in Canada and Monash University in Australia. It is always eye-opening to learn from others' research, and inspiring to discuss with others. Thanks to Jonathan Feng-Shun Lin for his help in both academics and life. Thanks Nils Wilde for his help with sharing his PhD thesis proposal which helped me to organize my proposal. Thanks Pamela Carreno-Medrano, Tina Wu, and Rachel Love for organizing so many great social activities. Thanks Calvin Vong for making my visit to Monash University more smooth. Most notably, thanks Daiwei Lin for working with me on our field study and accompanying on the trips to our collaborators. I would like to express my gratitude to every single person who has contributed to our lab, but in order to preserve space, I will instead store their names in my heart.

Without the resources and assistance offered by the University of Waterloo and Monash University, this thesis would not be possible. The research in this thesis was also funded by the Social Sciences and Humanities Research Council, and it was made possible in part by assistance from the Digital Research Alliance of Canada.

I would especially like to thank my friend Shuyu for the many precious days we spent together, from working out to sharing the significant occasions of our two families.

Finally, I want to thank my family for their unending moral and financial support, as well as their encouragement and patience along the arduous road. My wife deserves the deepest gratitude for her steadfast, unwavering support and for bringing my baby to me. I also want to express my gratitude to my daughter for coming into my life, for being so cooperative, and for making our family laugh so much.

Dedication

This thesis is dedicated to my wife, Feijie and my daughter, Minyan.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

As robots enter human environments, engaging with human occupants in a suitable manner becomes increasingly important [247, 227, 38, 54, 248, 144, 11, 133]. To facilitate long-term interaction, interactive systems should be able to continuously adapt in order to generate engaging and appropriate behavior. Although for simple interactive systems it may be possible to manually design engaging behaviors, this approach is time-consuming and sometimes unfeasible for complex non-anthropomorphic interactive systems with many degrees of freedom (DOF), e.g. hundreds of DOF. In addition, manually designed behaviors constrain the system to a limited set of reactions, whereas autonomous generation of actions provides the possibility of adaption and continuous behavioral evolution in a nonstationary environment. Therefore, understanding how to autonomously generate engaging behavior is necessary for long-term Human Robot Interaction (HRI), and learning algorithms can be advantageous.

When introducing learning into an interactive system, human input could be explicit or implicit. An example of explicit input is a human taking a teacher role, giving feedback to guide the robot to achieve some goals. With implicit input, a learning signal, either hand-crafted or learned, is generated based on observations of human behaviors during interaction, without requiring active feedback. For successful long-term interaction in social and public settings, both implicit and explicit input are important, where the implicit input is used for learning from the natural interaction between the users and the interactive system with less cognitive demand, while the explicit input is used for learning from experts/designers with the goal of transferring human knowledge to the interactive system

explicitly. Finally, for successful interaction in social and public settings, the system should successfully engage with and learn from both individual and group interactions.

(a) *Radiant Soil*, installed in Daejeon Museum of Art in Daejeon, South Korea, 2018.

(b) *Epiphyte Spring*, installed in Design Institute of Landscape and Architecture in Hangzhou, China, 2015.

(c) *Hylozoic Series: Sibyl*, installed in the Industrial Precinct in Sydney, Australia, 2012.

(d) *Aegis*, installed in the Royal Ontario Museum in Toronto, Canada, 2018.

(e) *Meander*, installed in Tapestry Hall in Cambridge, Canada, 2020.

Figure 1.1: Living Architecture Systems. (Photos courtesy of Philip Beesley Studio Inc.)

In this thesis, we study the challenge of long term autonomy with Living Architecture Systems (LAS). LAS are interactive systems at architectural scale that emulate living environments aiming to engage occupants in long-term interaction (see Fig. 1.1 for sample installations by the Living Architecture Systems Group (LASG) and Philip Beesley Studio Inc. (PBSI)[1]). An immersive LAS can have dozens of sensors and hundreds of actuators with various modalities to enable interaction with visitors, which poses a challenge when

[1]More LAS environments by LASG/PBAI can be found at https://www.philipbeesleystudioinc.com/sculptures

designing effective interaction control strategies. Historically LASs have used pre-scripted, human-designed behaviors. However, we are interested in the potential for adaptive machine learning behavioral algorithms to generate interactive behaviors. LAS aims for long-term continuous interaction, and this necessitates the capability of LAS to evolve with and to learn in a non-stationary environment, rather than using a single pre-scripted solution or assuming the existence of a single optimal solution.

Although LAS aims for long-term engagement with occupants, the architectural scale and non-anthropomorphic, immersive nature of LAS makes it distinctive compared to robots more typically studied in HRI [244, 107, 139, 278, 100, 303]. Since LAS is intended to create life-like behaviors at architectural scale, it facilitates accommodating multiple occupants and group interaction, rather than the one-to-one interaction most commonly studied in HRI. In addition, unlike HRI studies with humanoid robots that can be directly inspired by human-human interactions, LAS is a non-anthropomorphic robot, making the design of interactive behavior less intuitive. While it is possible to manually design life-like interactive behaviors to some extent, it is very time-consuming and requires significant expertise, and the resulting behaviors are non-adaptive. Therefore, implementing interactive systems that can adapt and learn in dynamic crowd settings as well as from expert feedback and can autonomously generate engaging behavior is critically important, not only because it is useful in many applications, including public spaces, schools, workplaces, and family residences etc, but also because HRI in such interactive systems could extend our understanding of socially acceptable and engaging interactive behavior.

Manually designed behavior patterns can generate engaging life-like behaviors based on the designer's understanding [30, 29], but this can be time-consuming and result in behaviors that become predictable during long term interaction. To overcome this limitation, machine learning may be used to automatically generate behavior and study whether such behavior is attractive or engaging for participants. For example, in [57, 58], a Curiosity-Based Learning Algorithm (CBLA) was implemented in a LAS to automatically generate behavior based on a computational notion of curiosity [211] of the LAS. However, the action produced by the CBLA is purely intrinsically motivated by the curiosity mechanism and does not consider any extrinsic motivation, e.g., maximizing measures of human response, which could play a more important role when an interactive system aims to engage participants. Another issue with [57, 58] and many works on social robots [139, 42] is that the proposed approach is only designed and tested for one-to-one interaction in a controlled setting, rather than group interactions in natural settings, i.e., multiple people interacting with an interactive system simultaneously without instruction or guidance by the researchers.

Learning interactive behavior for an extrinsically motivated LAS in natural settings

with group interactions is a difficult challenge that is not typically considered in Reinforcement Learning (RL) [265] tasks in controlled settings with only one-to-one interaction. Firstly, LAS has very large state and action spaces, typically having dozens of sensors and hundreds of actuators. The learning challenge is exacerbated by a complicated and non-stationary environment, where the LAS might be manipulated by occupants with different cultural backgrounds, interests and personalities. In addition, interactions happening in the real world happen less frequently than interactions in the simulators or video games typically studied in RL. This results in fewer interactions, which poses a huge challenge for data-driven learning algorithms. Last but not the least, there is no standard measurement for estimating engagement, i.e., the extrinsic motivation of LAS. If a measurement of engagement is sparse and time-delayed, learning interactive behavior becomes more difficult.

Considering these challenges, Deep Reinforcement Learning (DRL) [192, 193] offers advantages because of its powerful representation capacity to deal with large observation and action space. DRL has been intensively studied in simulated environments, such as games [192] and simulated robots [164], as well as in real-world studies, such as robotics control [104, 299, 267] and human-robot interaction [222, 60, 187]. However, applying DRL to real world robots [117, 80], including LAS, is not straightforward because of the aforementioned challenges and the limitations of DRL. For example, DRL suffers from low data efficiency, instability, i.e., drastic fluctuations in accumulated reward increase rather than a smooth increase [117, 264, 130], and incapacity of dealing with Partially Observable Markov Decision Process (POMDP). These disadvantages hinder DRL from broad use in applications where interactive data collection, which is ubiquitous in HRI, is time-consuming, smooth adaption of behavior is crucial to maintain engagement, e.g. interactive robots [186], and POMDP is inevitable due to lack of knowledge of the structure of the dynamics, sensor limitations, missing data, etc. Therefore, more fundamental research is also needed to advance our understanding of DRL when facing these challenges and facilitate the successful application of DRL to LAS in order to automatically generate engaging behavior.

Within the RL framework, an agent could learn engaging behavior from implicit input, i.e., interactive data, given a well-defined reward function. However, manually designed reward needs domain expertise and is hard to specify when the environment dynamics involving human participants is very complicated and even non-stationary and/or the dimensionality of the observation space is very high. Therefore, explicitly incorporating human knowledge either into a reward function or directly into the policy, along with learning from implicit input, is appealing. Popular approaches to incorporating human knowledge are Preference Learning [294, 157], Inverse Reinforcement Learning [203], Learning from Demonstration [225], Interactive Reinforcement Learning where either action correction

[55, 56, 217] or reward [263, 272] is provided by a user. These approaches come with different mental and physical demands on teachers, and some of them are very hard to apply to LAS. For LAS, an agent either acts on a low-dimensional but highly abstracted parameterized action space, or acts on a high-dimensional raw action space with hundreds of actuators, which deters the recruitment of methods needing action guidance from human. Therefore, inducing a reward function from human preference is more attractive to our application. Different from manually designed reward function that relies on expertise and is time-consuming to specify, preference based reward function learning may be less demanding of human teachers.

This thesis aims to advance our understanding on *how to learn engaging behavior for crowd and long-term interaction for architectural scale interactive systems*. Such architectural-scale interactive systems normally have dozens of sensors and hundreds of actuators, which enable crowd interaction, i.e., interacting with multiple people simultaneously, but also make control more challenging. In addition, these systems usually target long-term interaction: their behavior should continue to engage their occupants when deployed in long-term installations. For these reasons, in this thesis Machine Learning (ML) techniques, such as Deep Reinforcement Learning (DRL) and Preference Learning (PL), are proposed to enable autonomous generation of engaging behavior.

When applying learning techniques to such systems, the following key challenges arise:

- Estimation of engagement: In order to maximise engagement, a measure of engagement, to be used as a reward signal for learning, must first be formulated. One approach is to manually formulate a proxy metric, considering the capacity of the sensors embedded in an interactive system. A more general approach is to automatically learn the engagement metric in order to further increase the flexibility and adaptability of the control algorithm and reduce the reliance on a manually designed reward;

- Ecological evaluation: To validate the proposed algorithms, the integrated system should be evaluated in target environments and crowd settings with diverse visitors.

- Data efficiency: learning algorithms may display low data efficiency, especially considering the large number of actuators and sensors embedded in these interactive systems;

- Partially observable environment: due to the architectural scale of the interactive system, the limited sensor coverage, and the latency of the internal state of the visitors or occupants, etc., the observation of the environment for a learning agent may be partially observable, which challenges the learning of engaging behavior;

- Simulation infrastructure: Experiments with physical robots are time-consuming, so simulation infrastructure is desired and beneficial for algorithm development and experiment reproducibility.

1.2 Contributions

In this thesis, we take LAS as our testbeds to investigate how to automatically generate engaging lifelike behavior with non-anthropomorphic robots. Motivated by the aforementioned goal and challenges, we work on creating simulation infrastructure, conducting a field study in an unconstrained environment, understanding and proposing DRL algorithms dealing with application challenges, identifying partial observability as a potential failure source in applying DRL to robot control, and proposing to learn to engage from human preference.

Specifically, this thesis makes the following contributions to the state of the art in applying DRL to LAS to enable automatic engaging behavior generation:

C1. Simulation Infrastructure: LAS Simulation Toolkit

Chapter 4 tackles the simulation infrastructure challenge to enable fast algorithm development and promote reproducible experiments. A simulation toolkit for applying DRL to LAS is described. The toolkit integrates necessary components to run a LAS simulation and provides a middle layer to allow a learning agent to communicate with the simulated LAS with configurable observation and action space. This toolkit also enables researchers to conduct research without a physical installation of LAS and to run multiple experiments simultaneously in a High Performance Computation system in order to reduce development time.

C2. Unconstrained Field Study Applying DRL to LAS

Chapter 5 addresses the challenge of ecological validation using manually designed engagement estimation, where we propose an approach to estimate engagement with low-cost sensors and learn a policy from the estimate of engagement within an unconstrained environment using reinforcement learning. Our results show that the proposed approach is able to generate more engaging behavior compared to prescripted behavior. The experiment is

conducted with a novel non-anthropomorphic robot and unique interaction style in a public museum setting, interacting with museum visitors, contributing an approach for crowd engagement estimation from ambient sensors and a first implementation of crowd-based interactive learning in a large-scale, long-term, and real-world study.

C3. Elaborating the Effect of Multi-step Methods on Overestimation in DRL

Chapter 6 addresses the challenge of improving the data efficiency of DRL algorithms, based on the concern that when applying DLR algorithms to a LAS, a meaningful behavior may be hard to learn within a limited time, especially for a non-permanent LAS. One possible cause of low data efficiency is *overestimation* [274]. This occurs when there is noise in the estimation of an action's value, causing an agent to overestimate the value of that action. Because of this overestimation, the resulting policy is not optimal to achieve the given goal. To help address this problem, this chapter proposes to combine multi-step methods with traditional DRL. Multi-step methods are a group of methods in RL where the multi-step immediate rewards are used to estimate an action's value. When these rewards are not optimal, the corresponding action value underestimates the true action value. In this chapter, we demonstrate experimentally that the introducing of multi-step methods helps to alleviate overestimation problem and consequently results in better final performance and learning speed.

C4. Proposing Memory-based DRL for POMDPs

Chapter 7 contributes to tackling the challenge of partial observability. In this chapter, we consider incorporating a memory component to DRL algorithms in order to solve Partial Observable Markov Decision Processes (POMDPs). Different from Markov Decision Processes (MDPs) where the observation of the environment is assumed to be a full representation of the state of the environment so that the next observation only depends on the current observation and action, in POMDPs the observation of the environment is a partial representation of the state of the environment which means past observations are needed to infer the underlying state. To enable the DRL algorithm to automatically infer the underlying state of the environment, this chapter proposes a neural network architecture to take both the past experiences and the current observation as the input to learn an action value function and infer a policy from the learned action value function. This chapter experiments with the proposed method on simulated tasks with different types of POMDPs, e.g., sensory data missing and sensory noise, etc., and demonstrates the advantages of proposed method in dealing with POMDPs.

C5. Identifying Partial Observability As A Potential Failure Source in Applying DRL to Robot Control

Chapter 8 is inspired by our work applying DRL algorithms to LAS, where we find the popular DRL algorithms have different performance compared to other standard benchmarks. In this chapter, we investigate partial observability as a potential failure source of applying DRL to robot control tasks, which can occur when researchers are not confident whether the observation space fully represents the underlying state. The motivation of this work is based on our observation that DRL algorithm that relies on multi-step immediate rewards performs better than DRL algorithms that only rely on one-step immediate reward on POMDPs, even though this is hard to observe on MDPs. Therefore, we hypothesize multi-step bootstrapping plays a role in transferring temporal information through reward signals. To validate this hypothesis, we investigate the effect of the step size on the performance of the DRL algorithms on POMDPs. Similar results can be reproduced on different tasks with various partial observabilities, and for most cases simply increasing the step size by 1 step can significantly increase the algorithm's performance on POMDPs.

C6. Generating Engaging Behavior From Human Preference

Chapter 9 aims to address the limitations of manually designed engagement metrics. In this chapter, we propose to generate engaging LAS behaviors from human preference by replacing the hand-crafted reward function with a reward function learned from human preference. We examine four DRL algorithms namely Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), Soft Actor-Critic (SAC) and Long-Short-Term-Memory based TD3 (LSTM-TD3) with simulated expert feedback. We then recruit three experts to provide unconstrained feedback. Our results show the preference-based reward continuously adapts to new preference labels and is able to predict the future expert preference significantly better than a randomly initialized reward model. The survey data is also analyzed in this chapter, and it is shown that experts have different perceptions of the usability of the system.

1.3 Structure

This thesis is structured as follows: We first review the related work in Chapter 2. Then, the necessary background is provided in Chapter 3. Chapter 4 introduces the testbeds utilized in this work. Chapter 5 presents the field study conducted in a public museum.

Chapter 6 shows the work elaborating the relationship between multi-step methods and overestimation problem. Chapter 7 proposes a memory-based DRL algorithm to deal with partial observable MDP. Chapter 8 identifies partial observability as a potential failure source of applying DRL to robot control tasks. Chapter 9 proposes a method to learn engaging behavior from human preference. Chapter 10 concludes with the contributions of this thesis and the future work.

Chapter 2

Related Work

This thesis considers embedding Reinforcement Learning (RL) into a large-scale interactive system to endow the system with the ability to engage occupants. At the interaction level, we want to understand how to automatically generate engaging behavior from the perspective of HRI, which relates our work to many works in HRI. At the implementation level, we consider an RL framework, but since there are still challenges successfully applying RL algorithms to real world robots, we also contribute RL algorithms that can be deployed to practical applications. In addition, considering the difficulty in manually designing a reward function, i.e., an estimate of engagement, we also exploit preference learning to transfer human preference into a reward estimator in order to reduce the reliance on expertise, which relates our work to preference learning as well.

2.1 Human-Robot Interaction

Adaptive Control Traditional robotic systems operating in environments where the system dynamics are unknown or varying have used adaptive control [15]. In the domain of human-robot interaction, [286] applied adaptive control to robot navigation, where a social proxemics potential field is constructed and used to design a robot motion controller which is able to adapt its desired trajectory smoothly and at the same time comply with the proxemics contraints. Nakazawa et al. [201] proposed a potential field imposing a repulsive fin to allow adaptive control of an accompanying robot, where the robot is able to adapt its relative position to the accompanied human in the presence of obstacles. Vitiello et al. [283] proposed a Neuro-Fuzzy-Bayesian approach for adaptive control of a robot's proxemics behavior, where recognized human activities and human personality acquired by

questionnaires are input into an Adaptive Neuro Fuzzy Inference Engine (ANFIS) to determine a robot's stopping distance. Adaptive control has also been applied to rehabilitation robots [300], robots driven by Series Elastic Actuators [163], and many other applications. However, adaptive control relies on a known structure of the dynamics model. In our case (i.e., the testbeds shown in Chapter 4), it is difficult to apply adaptive control because we do not have access to a good model of the environment dynamics, since the number of visitors is unpredictable and visitors' interaction style varies.

Learning with Human as Explicit Teacher Interactive Reinforcement Learning (IRL) studies RL algorithms in the context of HRI, where humans explicitly provide rewards or actions to guide an agent in RL. Isbell et al. [131] applied RL in a complex human online social environment, where a human interacts with a learning agent by providing a reward signal, and highlighted that many of the standard assumptions, such as stationary rewards, Markovian behavior, appropriateness of average reward, for RL are clearly violated. Thomaz [272, 271] proposed IRL for training human-centric assistive robots through natural interaction, where a human coach's feedback is used to shape the predefined reward. In addition, Thomaz [272, 271] introduced anticipatory cues to allow the human coach to predict the robot's action and provide timely feedback. In [263], IRL was first studied in real-world robotic systems, and showed that the positive effects of human guidance increase with state space size. Other works converting human feedback to reward include [148, 149]. Griffith et al. [102] proposed an algorithm (Advise) to formalize the meaning of human feedback as policy feedback, which is more effective and robust than using human feedback as extra reward. Krening et al. [150] studied the effect of different teaching methods with the same underlying RL algorithm on human teachers' experience in terms of frustration. This work emphasizes that high transparency will decrease frustration and increase perceived intelligence. Different from most of these works where the human plays the direct role of teacher, i.e., directly providing either a reward or a policy advice in an interactive way, in this thesis we aim to learn how to engage visitors without requiring them to consciously teach or train the interactive system. Therefore, in Chapter 5 the learning agent does not explicitly receive rewards from visitors, but uses a measure of engagement as reward, which is calculated based on observed visitor behaviors, without any constraints on the frequency and consistency of interaction on the visitor.

Learning with Human as Implicit Teacher RL algorithms have been deployed in social robotics to enable a robot to acquire socially acceptable skills, where a predefined reward function is employed to implicitly infer the reward signal from sensed human behavior. In [153], emotion recognition based on videos captured during participants' interaction with a robot is exploited as a reward within an RL framework to adapt the robot's behavior towards participants' personal preferences. Leite et al. [159] proposed to model the user's

affective state by combining facial expression recognition and game state in a supervised way. The predicted affective state is then used to calculate the reward in an RL framework to personalize response to users. Macharet et al. [173] investigated the effectiveness of using RL to learn socially acceptable approaching behavior for a mobile robot. Gordon et al. [101] studied affective personalization in an integrated intelligent tutoring system, where in a one-to-one interaction setting, facial expression based engagement and valence estimation are used as reward in a standard SARSA algorithm. These studies all work on relatively small state and action spaces without exploiting deep neural networks. Recently, **Deep Reinforcement Learning (DRL)** has also been investigated in social robotics. Qureshi et al. [222] proposed Multimodal Deep Q-network (MDQN) for a handshake robot, where a deep neural network was employed to approximate the Q-value function based on both image and depth information. Chen et al. [60] studied Social Aware Collision Avoidance Deep Reinforcement Learning (SA-CADRL), where the reward function was designed to penalize actions leading to states where a robot is too close to nearby pedestrians and a deep neural network is used to approximate the Q-value function. Different from these works where either DRL is not exploited or the action space is discrete, Chapter 5 applies continuous control DRL to an interactive social robot.

One-to-one HRI In Natural Settings Without Learning Interactions in natural settings, e.g., public spaces, are too complex to be simulated in controlled laboratory settings, so to understand natural HRI it is important to conduct field studies. Many HRI systems have been tested in public spaces such as hospitals [198], train stations [111], service points [137, 141], airports [134], shopping malls [245], hotels [66] and museums [246], among others. Although robots investigated in these papers are sometimes surrounded by a group of people in a public place, these robots typically interact with only one person at a specific time, i.e., one-to-one interaction. Unlike these works, the larger scale of LAS enables LAS to simultaneously accommodate interactions with multiple people. In addition, the behaviors of robots studied in most of these works are pre-designed by researchers without continuous learning and adaptability.

Interactive Artworks Without Learning Interactive artworks are outcomes of combining arts and engineering, and have brought a new research direction for understanding HRI, especially with non-anthropomorphic robots, not only from the robotics perspective but also from an artistic perspective. LAS in this work and previous installations [29] are examples of such immersive interactive systems that promote roboticists' and architects' understanding of life-like interactive behavior. Other examples include [160], who investigated visitors' interaction and observation behaviors with mobile pianos in a museum. [259, 260] studied flying cubes in multiple publicly accessible spaces where flying cubes play the role of living creatures. In [71], roboticists collaborated with a professional

dancer to design interactive dancing robots. [284] created spatial interactions with immersive virtual reality technology. Raffaello D'Andrea created one of the first interactive agents in art called "The Table" [73], with artist Max Dean. It interacts with humans in the wild, one-on-one setting and does not have any learning. [18] studied dancing quadcopters, without an interactive component. Close collaboration between artists and roboticists has been fostering the creation of richer modes of interaction and extending the scope of studies in HRI. Most works mentioned here rely heavily on the design of choreographers, while in Chapter 5 and Chapter 9 we add adaptability on top of choreography by learning engaging behavior based on the action space defined by choreographers.

Engagement Estimation in HRI As the intention of HRI is to engage human participants in interactions, detecting and measuring initial and ongoing engagement in HRI is critical for initiating and maintaining interaction [78, 96, 204]. Many approaches for measuring engagement rely on rich sensors, such as cameras, lasers, sonar and audio sensors, and on sophisticated facial expression, gesture, gaze, body movements, physiological signals and speech recognition technologies [151, 270, 11]. Sanghvi et al.[235] proposed to estimate engagement by extracting features of body motion from video by training an engagement prediction model in a supervised way, but the proposed method requires a lateral camera and users to wear a specific colored coat. Keizer et al. [141] trained an engagement classifier based on multimodal corpus in a supervised way, which classifies engagement into three levels i.e. *NotSeekingEngagement*, *SeekingEngagement* and *Engaged*. Michalowski et al. [189] studied a spatial model of engagement for a social robot and tested a robotic receptionist to engage visitors, and suggested that direction and speed of motion are more appropriate measures of engagement than location or distance alone. However, this work used distance to determine the engagement level and what pre-scripted behavior to take, rather than using the distance as a learning signal. Bohus et al. [36, 37] studied the engagement model for a multiparty open-world dialog system, but the system can only actively engage in at most one interaction even though it could simultaneously keep track of additional, suspended interactions. Engagement in [36, 37] is viewed as several states. Behaviors corresponding to each engagement state are pre-defined. Even though the authors claim the system is a multiparty interaction, the number of participants is limited because of the field of view of the camera and the interaction screen. Khamassi et al. [143] and Velentzas et al. [282] proposed to estimate engagement from human gaze and use it as a reward function in an RL framework to allow fast adaption to environment change, where experiments are conducted in a well controlled setting. Most works studying engagement in HRI focus on humanoid social robots, one-to-one interactions, and discrete measures of engagement. In this thesis we study a non-anthropomorphic robotic environment, group interaction, and measure engagement as a continuous value. Most importantly, we use a

measure of engagement as the extrinsic reward to drive learning, rather than using the measure of engagement as a contingent event to trigger a predefined behavior. In addition, since our environment has highly varying lighting conditions and the potential for severe occlusions, camera based methods may be less reliable.

2.2 Deep Reinforcement Learning

Applying RL methods to real-world robots is challenging. First, real world robots have continuous and large state-action spaces. Considering traditional RL methods, neither tabular methods for discrete state and action spaces nor approximate methods with manually designed features for continuous state and action space are suitable for large, continuous state and action spaces [265]. DRL methods for continuous state and action spaces are appealing, as they are end-to-end methods that could learn policy directly from raw input. However, DRL methods face the low data efficiency and incapacity to Partial Observable Markov Decision problem, etc.

2.2.1 Overcoming Overestimation Problem in DRL

DRL is criticized for its low data efficiency and instability, i.e., drastic fluctuations in accumulated reward increase rather than a smooth increase [117, 264, 130]. These disadvantages hinder DRL from broad use in applications where interactive data collection is time-consuming and smooth adaption of behavior is crucial to maintain engagement, e.g. interactive robots [186].

Low data efficiency corresponds to slow learning speed, assuming the learning algorithm is capable of learning an optimal policy given sufficient data, and can be caused by two reasons: **1)** lack of data, and **2)** lack of training. If slow learning is caused by lack of data, the environment is under-explored. In this case, an efficient exploration strategy, e.g. parameter space noise [219], or a complementary source of experiences, e.g. World Models [105], can be helpful for generating additional training data. On the other hand, if slow learning is caused by lack of training, since enough experiences have been collected but not coded into policy, a more effective way to use the collected data is necessary such as prioritized replay buffer [238, 120] and hindsight experience replay [9].

Instability is partially related to the catastrophic forgetting problem of Deep Learning (DL) [147] which is inherent in the continuously evolving nature of policy learning in Reinforcement Learning (RL). In addition, inaccurate estimation and continuous tuning of the

Q-value function might lead the learned policy in directions far away from optimal or cause it to fluctuate around a local optimum. The overestimation problem [274] in Q-learning is a typical example of inaccurate estimation in which the maximization of an inaccurate Q-value estimate induces a consistent overestimation. As the result of overestimation, the estimated Q-value of a given (state, action) pair might explode and drive the corresponding policy away from optimal. Therefore, DRL algorithms based on Deep Neural Networks (DNNs) [99] should strive to alleviate overestimation problem if it cannot be completely overcome.

Multi-step methods have been studied in traditional RL for both on- and off-policy learning considering both the forward view, i.e., updating each state by looking forward to future rewards and states, and the backward view, i.e., updating each state by combining the current Temporal Difference (TD) error with eligibility traces of past events [265]. Recently, a new multi-step action-value algorithm $Q(\sigma)$ was proposed to allow the degree of sampling performed by the algorithm at each step during its backup to be continuously varied, with Sarsa at one extreme, and Expected Sarsa at the other [75]. The results show that an intermediate value of σ performs better than either extreme. However, a systematic way to adjust σ still needs to be studied, and the learning tasks in [75] are relatively simple with small state and action spaces, avoiding the need for DL methods. Multi-step TD learning for non-linear function approximation was studied in [281] where forward TD(λ) was investigated on simple discrete control tasks. Although a neural network was used for function approximation in this work, only simple discrete control tasks were examined. Rainbow [118], an integrated learning agent combining many extensions including multi-step learning, found that multi-step not only helps speed up early learning but also improves final performance on Atari 2600 games. However, Rainbow is built on top of Deep Q-Network (DQN) [192] and only discrete action space tasks were examined. In our work, we focus on continuous control tasks. Multi-step methods have also been investigated in asynchronous methods [191], which rely on parallel actors employing different exploration policies in parallel instances of environments, to decorrelate consecutive updating experiences and to stabilize policy learning without using a replay buffer. However, such a parallel paradigm can only work practically in simulated environments, and is unfeasible for real applications where multiple instances of physical systems (e.g., robots) are too expensive.

The **Overestimation Problem** [274] in DRL is cited as the reason non-linear function approximation fails in RL. Based on Double Q-learning [109], Double DQN [280] was shown to be effective in alleviating this problem for discrete action spaces. Although Twin Delayed Deep Deterministic Policy Gradient (TD3) [89] proposed to take the minimum of two bootstrapped Q-values of a state-action pair which are separately estimated by two

critics to overcome the overestimation problem, the extra neural network for the second critic also introduces additional computation cost, especially when the state and action spaces are large. Model-based Value expansion (MVE) [83] is a multi-step method that expands multi-step on a learned environment model, whose performance tends to degrade in complex environments. As an improvement of MVE, STochastic Ensemble Value Expansion (STEVE) [47] expands various multi-steps on an ensemble of learned environment models, including transition dynamics and reward function, then adds these to an ensemble of Q-value functions. The final target Q-value is a weighted mean of these generated target Q-values. Both MVE and STEVE suffer from modeling error, and more importantly they both introduce vast extra computation, especially STEVE. Averaged-DQN [10] is proposed to reduce variance and stabilize learning by exploiting an average over a set of target Q-values calculated from a set of past Q-value functions. It is shown that Averaged-DQN also helps in alleviating the overestimation problem. Different from that, our method MMDDPG proposed in Chapter 6 takes the average over a set of target Q-values calculated with different step sizes, which is also shown to be more stable than MDDPG.

2.2.2 Approaches to Solving POMDPs

Most works in DRL focus on developing algorithms [192, 154, 164, 239, 241, 89, 106] for Markov Decision Processes (MDPs) with fully observable state spaces [79], i.e. the observation at each time step fully represents the state of the environment. Few works consider the more complex Partially Observable Markov Decision Process (POMDP) where the observation is just a partial representation of the underlying state. However, POMDPs are ubiquitous in real robotics applications [196], such as robot navigation [50], robotic manipulation [213], autonomous driving [268, 285], and planning under uncertainty [287, 61]. Partial observability may be due to limited sensing capability, or an incomplete system model resulting in uncertainty about full observability.

POMDPs have been tackled with the concept of belief state [230], which represents the agent's current belief about the possible physical states it might be in, given the sequence of actions and observations up to that point. These algorithms are designed to estimate the belief state, then the value function and/or the policy are learned based on the belief state [242]. However, these methods need to know the environment model and the state space and they only work on tasks with small state and action spaces.

POMDPs have also been addressed with DRL, for both discrete [110, 154, 302] and continuous [258, 285] control problems. Recurrent Neural Networks (RNN) have been exploited in DRL to solve POMDPs by considering both the current observation and action, and the history of the past observations and actions [114, 258, 301, 302, 154].

Incorporating Memory into DRL has been investigated in a number of previous works. Deep Recurrent Q-Learning (DRQN) [110] adds recurrency to the Deep Q-Network (DQN) [193] by replacing the first post-convolutional fully-connected layer with a recurrent Long-Short-Term-Memory (LSTM). The results on Atari 2600 games show DRQN significantly outperforms DQN on POMDPs, which validates the effectiveness of memory extracted by the LSTM for solving POMDPs. [154] investigated a similar idea but augmented the structure with an auxiliary game feature, e.g. presence of enemies, learning in 3D environments in first-person shooter games. The results show the proposed architecture substantially outperforms DRQN. These methods only consider past observations in the history. [302] proposed Action-specific Deep Recurrent Q-Network (ADRQN) to also consider past actions in the memory. However, these works are based on tasks with discrete action spaces, rather than on continuous control tasks. [114] extended Deterministic Policy Gradient (DPG) [250] to Recurrent DPG (RDPG) by adding LSTM and investigated it on continuous control tasks with partial observations. Dramatic performance improvement was observed with memory. However, even though the observation space was large for some tasks, the action space had relatively few dimensions for the investigated tasks. [258] investigated RDPG on bipedal locomotion tasks with both visual and sensory input, but only one task was examined. Different from directly optimizing RNN, [301] proposed to augment the observations and actions with the continuous memory states and use guided policy search to optimize a linear policy. The method shows better performance than other policy search methods. However, the guided policy search is less powerful and generalizable than non-linear policy. In Chapter 7, we consider continuous control tasks with large observation and action spaces and propose LSTM-TD3 within a recurrent actor-critic framework, which is a further improvement of RDPG by exploiting TD3 to reduce the overestimation problem.

Deep Reinforcement Learning (DRL) [193] has been making tremendous improvement in end-to-end learning in decision making problems by combining Deep Neural Networks (DNNs) [156] and Reinforcement Learning (RL) [265]. While most works [193, 164, 191, 117] are evaluated in simulated tasks, applications to real world robots [153, 222, 60, 104, 299, 136] have also achieved promising performance. From the perspective of embedded artificial intelligence, not only the robots' capability for completing a given task effectively and efficiently is important, but also the robots' capacity to engage human companions is crucial in order to embed robots into our daily life, which is one of the main research topics in Human Robot Interaction (HRI) [244, 107, 139, 278, 100, 303].

Previous works [30, 29] studied pre-scripted engaging behavior on an interactive system, based on expertise of architects. The demanding labour work of pre-scripted behavior motivates research on automatically generating engaging behavior by combining RL where

reward function is based on an estimated curiosity namely Curiosity-based Learning Agent (CBLA) [57]. Even though it is attractive to allow a learning agent to act based on its intrinsic motivation, the study is conducted in a lab environment without considering extrinsic motivation of engaging human companions.

2.2.3 Partial Observability during DRL

Many popular DRL algorithms are formulated for MDP problems. However, POMDP is common in novel and complex control tasks [53, 85, 205] due to lack of knowledge of the structure of the dynamics, sensor limitations, missing data, etc. In addition, most literature assumes that the key components of an environment, namely the action and observation space and the reward function, are given, which may not necessarily be true for complex systems. For DRL algorithms developed for handling POMDPs [110, 154, 302, 129, 182], researchers usually assume the tasks on hand are POMDPs, rather than first determining if the given task is POMDP or MDP. In real applications, these assumptions may not be satisfied. Therefore, researchers may encounter unexpected results that are different from those usually reported in literature with these assumptions. Moreover, researchers tend to present successes in applying DRL to robot control, but hide the failure stories behind these successes, the reason of these failures and how they detect them, which can be useful to make DRL approaches more useable in robotics.

Multi-step methods have been investigated in the literature for improving reward signal propagation [281, 191, 75, 118] and alleviating the over-estimation problem [181]. However, as far as we know, there is no work connecting multi-step methods to their potential effect on passing temporal information when solving POMDP. In Chapter 8, we empirically show that multi-step bootstrapping helps TD3 and SAC to perform better on POMDPs.

POMDPs have been investigated within both model-free [110, 154, 182] and model-based [129, 254] DRL, where the tasks are known POMDPs. In Chapter 8, we do not focus on proposing new algorithms for solving POMDPs, but empirically show that when unsuccessfully applying DRLs to a complex control task the source of the failure may be related to partial observability, which is applicable to applications where researchers are not confident whether the given task is MDP or POMDP.

[226] studies the environment design, including the state representations, initial state distributions, reward structure, control frequency, episode termination procedures, curriculum usage, the action space, and the torque limits, that matter when applying DRL. They empirically show these design choices can affect the final performance significantly. In addition, [128] focuses on investigating the challenges of training real robots with DRL rather

than simulated ones. [175] studies the factors that matter in learning from offline human demonstrations, where the observation space design is highlighted as one of the prominent aspects. These works aim to comprehensively cover broader topics in applying DRL, but in Chapter 8 we mainly focus on the partial observability problem during DRL for robot control. Moreover, we try to reproduce the problem encountered when applying DRL to novel robot on toy tasks that are accessible to everyone, as we understand many domain robots are not available to researchers who want to reproduce the experiment represented in many works.

2.3 Preference Learning in HRI

Preference Learning (PL) [90] is one of the popular methods to learn a reward from human teachers, which is especially useful when the manually engineered reward is extremely difficult due to large observation and action space and/or complicated dynamics, etc., and has been applied in video game play [127], autonomous car driving task [34], recommendation robot [296], entertainment robot [177], assembly robot [195], exoskeleton gait optimization [162], etc. Many of these related works consider *Single Agent Scenario* where a robot learns a policy based on the reward function induced from PL to complete a task without human involvement. Even though there are also applications that investigate *Multi-Agent Scenario* where the robot aims to interact with human based on human preference, it is common that only one human is involved in the interaction with the robot. In our case, we are interested in applying PL to LAS that aims to facilitate crowd human-robot interaction enabled by the large scale of LAS which can be seen from the testbeds introduced in Chapter 4. Moreover, the specific people in the group within the LAS will vary in the case of public LAS installations. [177] propose to use Preference Learning System to personalize human robot interaction during entertainment activities, but the action space is discrete and small. PL is also employed in Learning from Demonstration (LfD) where it is used to extrapolate the reward derived from demonstrations by comparing trajectories injected with different degrees of action noise [46], where the preference labels are automatically generated by assuming the trajectory with less action noise is preferred over the one with more action noise. However[59] shows that the automatically generated preference labels following Luce-Shepard rule [170] results in a counterproductive inductive bias. Nevertheless, LfD still needs user to provide demonstrations that are not random if not optimal, Providing demonstrations for LASs is difficult because the non-stationarity, resulting from visitor diversity among other things, means the demonstration environment will differ from the environment during policy execution.

Many approaches use engineered features to represent human preference. For example, the methods proposed to actively choose queries: Approximate Expected Utility of Selection [7], Volume Removal [234], Information Gain [35], Maximum Regret [292] rely on manually designed trajectory features, which are limited to tasks with intuitive and relatively short trajectories. [293] considers preferences over partial trajectory segments, which are thought to be more informative than single states and at the same time more differentiable than using whole trajectories, but the approach is still limited to simple tasks. In addition, when a reward model is learned, most works [234, 35, 292, 291] assume a linear reward model on the trajectory features. Even though [34] proposed to use Gaussian Process (GP) to fit a non-linear reward, the reward model still relies on hand-crafted features, which is much less flexible and impractical for tasks with very large observation spaces and complicated dynamics.

Humans may get impatient with the process of preference labelling if too many labels are required, so minimizing the required number of samples is a best practice. To improve sample efficiency, [157] proposed the unsupervised PrE-training and preference-Based learning via relaBeLing Experience (PEBBLE), which uses intrinsic reward-based pretraining of policy to increase behavior diversity, then uses SAC to learn policy to maximize the reward elicited from preference labels. The proposed method is examined on standard simulator benchmarks with relatively low-dimensional action and observation space. In Chapter 9, we investigated a variant of PEBBLE by using SAC but without reward ensemble and unsupervised pre-training. In addition, Chapter 9 investigates the performance of TD3, PPO, and LSTM-TD3 proposed in Chapter 7. In another ways, [115] proposes few-shot preference learning for RL by putting the PL into the scenario of meta-RL where a preference-based reward model is pretrained on a set of tasks and continuously adapts to a new task enabled by Model agnostic meta-learning [84]. [291] proposes to use scale feedback rather than binary preference choice to improve the data efficiency by employing a sliding bar to allow users to provide more nuanced information. Besides, [127, 35] propose to use multiple sources of human feedback to improve the data efficiency by combining preference and demonstration. Again, the tasks investigated in these works have much smaller observation and action space compared to that in LAS.

[158] proposed a benchmark for Preference-based Reinforcement Learning where both continuous robot locomotion and robot control are considered. Especially, teacher irrationality is considered in the simulated teacher preference. In Chapter 9, we borrow a few ideas, e.g., irrational preference model, from this work to set up a simulation environment to examine the application of PL to LAS.

Different from the works relying on manually designed trajectory features, [64] investigates end-to-end reward model learning from human preference on MuJoCo tasks [277, 43],

where human preference teachers are asked to provided preference according to a given criteria. It was shown that after hundreds of preference labels, the policy learned from the reward function induced from human preference is comparative to the hand-crafted reward. However, only one on-policy DRL algorithm, i.e., PPO, is investigated, and the simulated tasks have much smaller observation space than our case. Most importantly, the robots within these tasks are not intended to interact with human, i.e., single agent scenario. In Chapter 9, we investigate both on-policy, i.e., PPO, and off-policy, i.e., TD3, SAC, and LSTM-TD3, DRL algorithms on LAS with different types of preference, namely simulated preference, constrained human preference and unconstrained human preference.

Chapter 3

Background

3.1 Decision Making

Decision Making is the process of perceiving the environment and making an action choice accordingly to achieve some goals, and is broadly studied in Psychology [252], Social Science [252], Optimization Theory [124], Robotics [197], Artificial Intelligence [265, 230], Human Robot Interaction [72], Economics [253] and Manufacturing [103], etc. This thesis builds on top of a specific type of decision process called Markov Decision Process (MDP) and its generalization to Partially Observable Markov Decision Process (POMDP).

3.1.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a sequential decision process for a fully observable, stochastic environment with a Markovian transition model and additive rewards [33, 230]. Formally, MDP can be defined as a 4-tuple (S, A, P, R), where S is the state space, A is the action space, $P(s'|s,a) = p(s_{t+1} = s'|s_t = s, a_t = a)$ is the transition probability that action a in state s at time t will lead to a new state s' at time $t+1$, and $R(s,a,s') \in \mathbb{R}$ provides the immediate reward r indicating how good taking action a is in state s after transitioning to a new state s'. In MDP, it is assumed that the state transitions defined in P satisfy the Markov property, i.e. the next state s' only depends on the current state s and the action a. The goal for an agent in MDPs is to choose actions at each time step that maximize its expected future discounted return $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, where r_t is the immediate reward received at time t and $\gamma \in [0,1]$ is the discount factor that describes the preference of the agent for current rewards over future rewards.

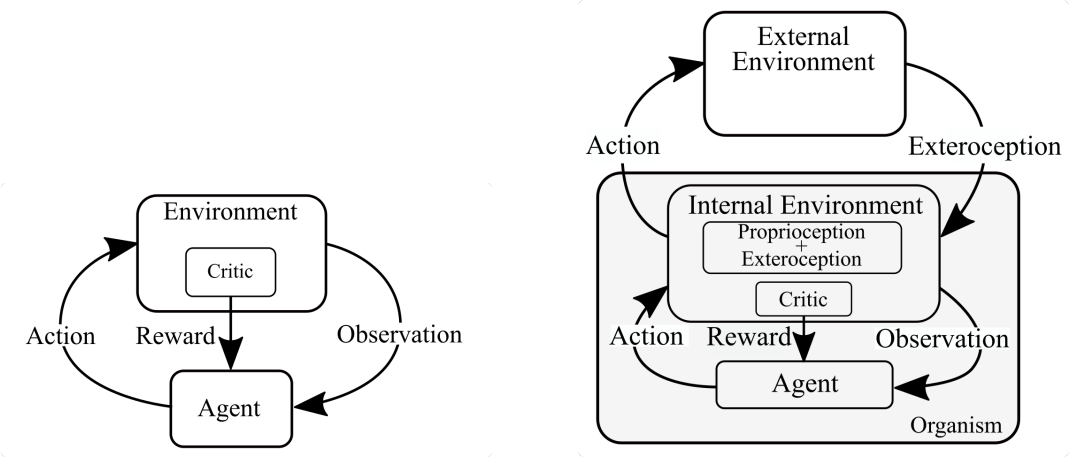
3.1.2 Partially Observable Markov Decision Process (POMDP)

Partially Observable Markov Decision Process (POMDP) [16, 48, 230] is a generalization of a MDP, which does not assume that the state is fully observable, and is defined as a 6-tuple (S, A, P, R, O, Ω), where S, A, P, and R are the same as that in MDP, with an additional observation space O and observation model Ω. Although the underlying state transition in a POMDP is the same as in an MDP, the agent cannot observe the underlying state, instead it receives an observation $o_{t+1} \in O$ when reaching the next state s_{t+1} with the probability $\Omega(o_{t+1} \mid s_{t+1})$. If the observation o_{t+1} can fully represent the current state s_{t+1}, a POMDP will reduce to a MDP.

For some cases, using a history of past observations and/or actions and/or rewards up to time t as a new observation can reduce a POMDP to MDP. For example, for tasks where the velocity is part of the state, using a history of past positions of a robot as the observation can make the POMDP, where only position is included in its observation, a MDP, as the velocity can be inferred from the two consecutive positions. For these cases, history aids with dealing with partial observability. The goal for an agent in POMDPs is the same as that in MDPs.

3.2 Reinforcement Learning (RL)

Reinforcement Learning (RL) [265] solves decision problems such as MDP and POMDP based on a learning paradigm where an agent learns to act by trial-and-error without knowing the underlying transition and reward model. Two agent and environment interaction models are commonly adopted as shown in Fig. 3.1, where Fig. 3.1a shows the classic view and Fig. 3.1b shows a more realistic view. From the classic view, an agent perceives the environment through its observation and takes action in the environment which will move to the next state represented by a new observation and give a reward signal, generated by the critic inside the environment, to the agent. Then, a new interaction will be repeated. The classic view assumes the reward is part of the environment, which is a simplified model of the reality where the reward signal may be internally generated by the brain of an organism. Therefore, [62, 255] expanded the classic view by differentiating the external environment from the internal environment as shown in Fig. 3.1b. Within the expanded view, the internal environment and the agent together can be seen as an organism that interacts with the external environment. The internal environment integrates the exteroception and proprioception to form an observation and send a reward signal to the agent.



(a) Classic View (b) Expanded View

Figure 3.1: Agent and Environment Interaction in RL [62]

Even though the agent and environment interaction models, i.e. classic view and expanded view, are different, the agent works in the same way. Specifically, at a discrete time t an agent interacts with the external/internal environment by taking action a_t according to either a stochastic policy $\pi(a_t \mid o_t)$ or a deterministic policy $a_t = \mu(o_t)$ when observing the representation o_t of the current state s_t. Then, the environment transitions to the next state s_{t+1} represented by a new observation o_{t+1} and returns an immediate reward signal r_t. By continuously interacting with the environment and receiving new observations and rewards, an agent learns an optimal policy to maximize the expected future return.

There are three functions commonly used in RL algorithms. For MDPs where the observation o fully represents the state s, the state-value function $V^\pi(s)$ of a state s under a policy π represents the expected return starting from s and following π thereafter, which is formally defined as follows:

$$V^\pi(s) \doteq \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \mid S_t = s \right], \tag{3.1}$$

where the γ is the discount factor. Similarly, the action-value function $Q^\pi(s, a)$ of taking action a in state s under a policy π is the expected return starting from s, taking a, and following π thereafter, which is defined as:

$$Q^\pi(s, a) \doteq \mathbb{E}_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \mid s_t = s, a_t = a \right]. \tag{3.2}$$

The advantage $A^\pi(s,a)$ of taking action a in state s is defined as

$$A^\pi(s,a) \doteq Q^\pi(s,a) - V^\pi(s), \tag{3.3}$$

when following a policy π. For POMDP, since the state s is not directly observable, the observation o is used in learning these value functions or policy. To simplify the notation, for the rest of this paper, we will use o to represent s.

3.2.1 Traditional RL Methods

Traditional RL methods employ either tabular representation or simple function approximation [265], with hand-crafted feature construction, to represent the value function. Even though these methods are well studied with comparatively clear properties and are more stable, they are not suitable to tasks with continuous and high-dimensional observation and action space which are ubiquitous in the real world. This work aims to apply RL techniques to large interactive systems where hundreds of sensors and actuators are embedded, which limits the applicability of traditional RL.

3.2.2 Multi-step Methods

Multi-step Methods (also called *n-step methods*) [265] refer to RL algorithms utilizing multi-step bootstrapping to facilitate fast propagation of knowledge about the outcomes of selected actions, by looking n steps forward. Formally, n-step discounted accumulated reward following policy π can be expressed as

$$R_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_{t+i}, \tag{3.4}$$

where n is the step size after which the bootstrapped value will be used. The bootstrapped value corresponds the value estimated based on other estimates. The idea of bootstrapping originates from Dynamic Programming [265] and corresponds to the operation where the estimates are updated on the basis of other estimates. Combining with the n-step discounted accumulated reward defined in Eq. 3.4, the n-step bootstrapping $\hat{Q}^{(n)}$ of observation and action pair (o_t, a_t) can be defined as

$$\hat{Q}^{(n)}(o_t, a_t) = R_t^{(n)} + \gamma^n Q(o_{t+n}, a), \tag{3.5}$$

where $a \sim \pi(a|o_{t+n})$ and Q is the approximation of the true state-action value function Q^π of policy π defined in Eq. 3.2. Note that when $n = 1$, Eq. 3.4 reduces to 1-step bootstrapping, which is commonly used in Temporal Deference (TD) based RL.

3.3 Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) [192] [164] learns the value function and policy in an end-to-end manner by recruiting deep neural networks. In effect, DRL uses Deep Neural Networks (DNN) for approximating the policy and value functions by reformulating RL as supervised learning. However, collecting data through exploration results in data that is not Independent and Identically Distributed (IID), which violates some key assumptions in supervised learning. The *replay buffer* employed in DRL helps address this problem by randomly sampling a batch of experiences from a buffer with millions of experiences, which breaks the time dependence between consecutive experiences. Learning both the policy and value networks by bootstrapping the value with the current policy and value networks may cause very unstable learning and may not converge at all, as the current policy and value networks continuously change. To stabilize the learning, DRL employs *target neural networks*, which are slowly updated towards the current policy and value networks. Since the tasks investigated in this work are all continuous control tasks, more interest will be given to DRL methods that are applicable to tasks with continuous action space, which will be detailed in the rest of this chapter.

3.3.1 DRL Foundations

Two types of deep neural networks, namely fully connected Multi-Layer Perceptron (MLP) and Long-Short-Term-Memory (LSTM), are employed in this work. In particularly, MLP is a powerful function approximator, whereas LSTM is especially beneficial to incorporate memory information in POMDPs.

Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) is a fully connected multi-layer feed-forward neural network. Its structure is shown in Fig. 3.2. MLP has powerful function approximation capacity known as Universal Function Approximation Theorem [121]. When MLP is used to approximate an actor, i.e., a policy, it takes an observation as input and generates an action as output for deterministic policy or the mean and standard deviation as outputs for stochastic policy. When MLP is used to approximate a critic, if the critic is a action-value function, it takes an observation and an action as input and generates a real value as an output, whereas if the critic is a state/observation-value function, it takes a state/observation as input and generates the value as an output.

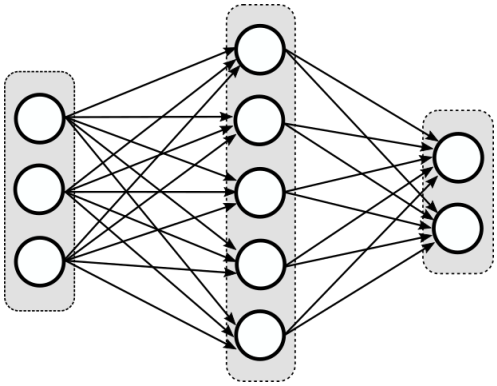


Figure 3.2: MLP Diagram

Long-Short-Term-Memory (LSTM)

Long-Short-Term-Memory (LSTM) [119] is a type of Recurrent Neural Network (RNN) [99] that has an outer recurrence from the outputs to the inputs of the hidden layer and also an internal recurrence between LSTM-Cells. As shown in the circuit diagram in Fig. 3.3, within a LSTM-Cell, a system of gating units controls the flow of information, (Fig. 3.3) and enables the remembering and forgetting of information given a sequence of inputs. To ease understanding, Fig. 3.3 also shows the unfolded diagram. The forward calculation

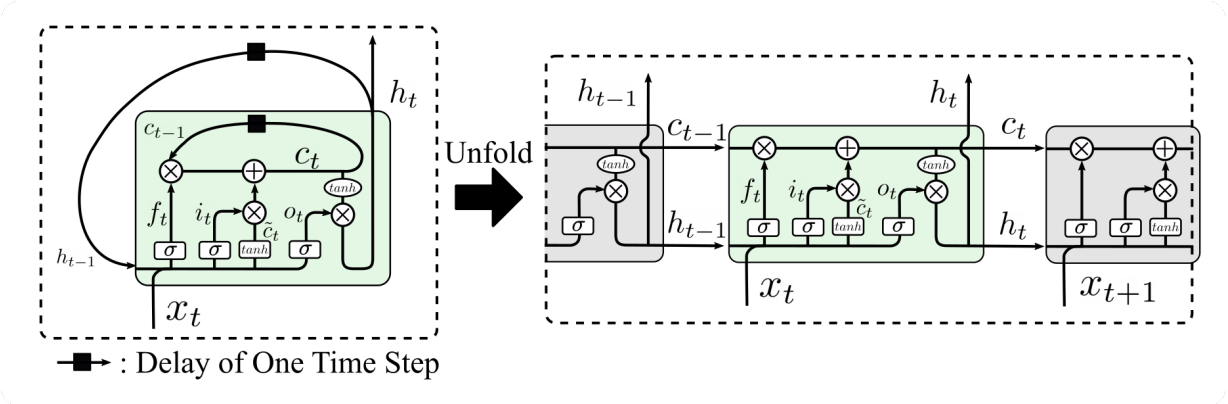


Figure 3.3: Circuit and Unfolded Diagram of LSTM-Cell

of the LSTM-Cell is defined in Eq. 3.6

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
\tilde{c}_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
\tag{3.6}
$$

where c_t and h_{t-1} are the cell and hidden states at time $t-1$, x_t is the input at time t, and i_t, f_t, $\tilde{c}_t$ and o_t are the input, forget, cell and output gates, respectively. σ is the sigmoid activation function, and $\odot$ is the Hadamard, i.e. element-wise, product. c_t and h_t are the cell and hidden states at time t. The trainable parameters in the LSTM-cell are the weights W_{ii}, W_{hi}, W_{if}, W_{hf}, W_{ig}, W_{hg}, W_{io}, W_{ho} and the biases b_{ii}, b_{hi}, b_{if}, b_{hf}, b_{ig}, b_{hg}, b_{io}, b_{ho}, where the subscripts indicate the source and the destination of the neural connections. The new cell state c_t is a combination of forgetting something in the previous cell state c_{t-1} and remembering something new coded in cell gate $\tilde{c}_t$, and the output of the hidden layer is further controlled by the output gate o_t. In this way, LSTM is able to learn both the short-term and long-term dependencies.

3.3.2 Overestimation Problem

The *Overestimation Problem* [274] is inevitable when function approximation is employed in Q-Learning based value estimation. Generally speaking, the generalization error induced by function approximation can lead to systematic overestimation of the value estimations because of the max operation used in bootstrapping the target Q value in Q-learning, which will be formally described in Eq. 3.7 - 3.10. As a consequence of this overestimation, as well as the large discount factor typically used in RL, the overestimation will lead to failure to learn an optimal policy based on the estimated value function. Formally, assume the implicit true value function Q^{true} is approximated by a function approximator Q^{approx}, e.g. a neural network, that introduces some noise $Y_{o'}^{a'}$ to the estimates of Q^{true} as

$$
Q^{approx}(o', a') = Q^{true} + Y_{o'}^{a'}, \tag{3.7}
$$

where $Y_{o'}^{a'}$ is modeled by a zero-mean random variable, and Q-learning is used to update Q following

$$
Q(o, a) \leftarrow r + \gamma \max_{a'} Q(o', a'), \tag{3.8}
$$

where Q indicates the value function and can be either the approximator Q^{approx} or the true value function Q^{true}. Then, the error $Z_{o,a}$ between the estimated and the true Q in (o, a) is defined as

$$\begin{aligned} Z_{o,a} &\doteq r + \gamma \max_{a'} Q^{approx}(o', a') - \left[r + \gamma \max_{a'} Q^{true}(o', a') \right] \\ &= \gamma \left[\max_{a'} Q^{approx}(o', a') - \max_{a'} Q^{true}(o', a') \right]. \end{aligned} \tag{3.9}$$

According to [274], zero-mean noise $Y_{o'}^{a'}$ may easily result in $Z_{o,a}$ with positive mean, i.e.,

$$\mathbb{E}\left[Y_{o'}^{a'} \right] = 0 \ \forall a' \overset{\text{often}}{\Longrightarrow} \mathbb{E}\left[Z_{o,a} \right] > 0. \tag{3.10}$$

Intuitively, due to the random noise $Y_{o'}^{a'}$, there might be some actions corresponding to $Q^{approx} > Q^{true}$ and some others corresponding to $Q^{approx} < Q^{true}$. However, the max operator always picks the largest one, making it prone to overestimation. Fig. 3.4 illustrates the informal view of overestimation. At the beginning (left panel) there is overestimation bias in the approximation. Then, because of the bootstrapping and max operation used to update the approximation Q, the overestimation bias is also bootstrapped and exacerbated (middle and right panel), which finally causes the wrong policy to be derived from the approximation Q.

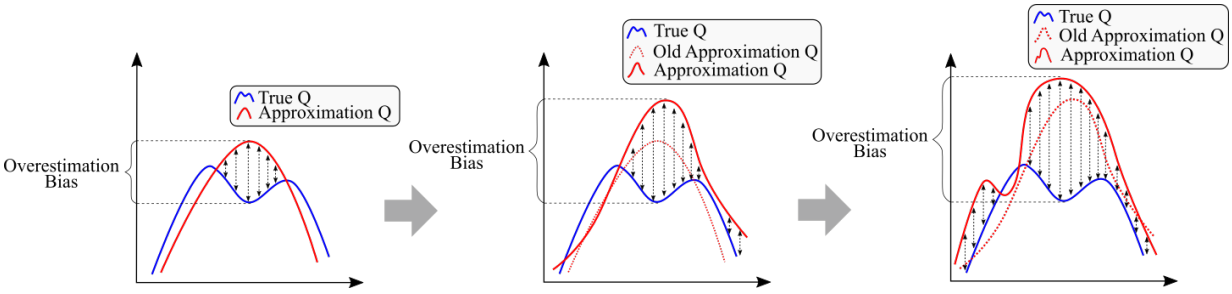


Figure 3.4: Informal View of Overestimation, where the blue lines indicate the true Q, the dotted red lines are the approximated Q before the current approximated Q in solid red lines, and the black arrows indicate the overestimation bias between the approximation and the true Q.

3.3.3 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) [164] adapts the ideas, i.e. replay buffer and target networks, underlying of the success of the Deep Q-Learning [192] to continuous

control within the Actor-Critic framework. Formally, the critic Q, i.e. the Q-value function, is approximated by a neural network parameterized by θ^Q, and the actor μ, i.e. the deterministic policy, is also approximated by a neural network parameterized by θ^μ. Then, the critic Q is optimized by minimizing the expected mean square error between the target $\hat{Q}$ and the predicted Q-value Q, as follows:

$$\min_{\theta^Q} \mathbb{E}_{\{(o_t,a_t,r_t,o_{t+1},d_t)\}_{i=1}^N} (Q(o_t,a_t) - \hat{Q}(o_t,a_t))^2, \tag{3.11}$$

where $\{(o_t,a_t,r_t,o_{t+1},d_t)\}_{i=1}^N$ is a mini-batch of N experiences randomly sampled from the replay buffer D, and the target $\hat{Q}$ is defined as:

$$\hat{Q}(o_t,a_t) = r_t + \gamma(1-d_t)Q^-(o_{t+1},a^-), \tag{3.12}$$

where r_t is the immediate reward after taking action a_t in observation o_t, γ is the discount factor, d_t is the boolean indicator of if o_{t+1} corresponds to the termination state, Q^- is the target critic, and $a^- = \mu^-(o_{t+1})$ based on target actor μ^-. The target actor-critic are updated periodically to the online actor-critic according to

$$\theta^{\mu-} \leftarrow \rho\theta^{\mu-} + (1-\rho)\theta^\mu \qquad \text{and} \qquad \theta^{Q-} \leftarrow \rho\theta^{Q-} + (1-\rho)\theta^Q. \tag{3.13}$$

The actor μ is optimized to maximize the approximated Q-value $Q(o_t,\mu(o_t)$ with respect to the parameter θ^μ of the actor, as follows:

$$\max_{\theta^\mu} \mathbb{E}_{\{(o_t)_i\}_{i=1}^N} Q(o_t,\mu(o_t)), \tag{3.14}$$

where $\{(o_t)_i\}_{i=1}^N$ are N observations sampled from the replay buffer.

3.3.4 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) [89] is a variant of DDPG to address the function approximation error in Actor-Critic methods, especially the overestimation problem [274], in Actor-Critic methods. Specifically, TD3 employs two critics Q_1 and Q_2, parameterized by θ^{Q1} and θ^{Q2} respectively, and uses the minimum of the predicted optimal future return in observation o_{t+1} to bootstrap the Q-value of the current observation o_t and action a_t, as follows:

$$\hat{Q}(o_t,a_t) = r_t + \gamma(1-d_t)\min_{j=1,2} Q_j^-(o_{t+1},a^-), \tag{3.15}$$

where r_t is the immediate reward after taking action a_t in observation o_t, γ is the discount factor, d_t is the boolean indicator of if o_{t+1} corresponds to the termination state, Q_j^- is

the target critic, and $a^- = \mu^-(o_{t+1}) + \epsilon$ where $\epsilon \sim \text{clip}(\mathbb{N}(0, \sigma), -c, c)$ is the clipped target action noise with mean 0, standard deviation σ and boundary c, and μ^- is the target actor.

With the bootstrapped target Q-value $\hat{Q}(o_t, a_t)$, critics Q_1 and Q_2 are optimized to minimize the mean square error between the target and the predicted Q-value, as follows:

$$\min_{\theta^{Q_j}} \mathbb{E}_{\{(o_t, a_t, r_t, o_{t+1}, d_t)\}_{i=1}^N} (Q_j - \hat{Q}(o_t, a_t))^2, \tag{3.16}$$

where $\{(o_t, a_t, r_t, o_{t+1}, d_t)\}_{i=1}^N$ is the randomly sampled mini-batch of size N from the replay buffer D.

The actor μ is optimized to maximize the approximated Q-value $Q_j(o_t, \mu(o_t)$ with respect to the parameter θ^μ of the actor, as follows:

$$\max_{\theta^\mu} \mathbb{E}_{\{(o_t)_i\}_{i=1}^N} Q(o_t, \mu(o_t)), \tag{3.17}$$

where $j = 1$ or 2, and $\{(o_t)_i\}_{i=1}^N$ are states sampled from the replay buffer. The target networks update in TD3 is similar to that in DDPG as defined in Eq. 3.13.

3.3.5 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) [106] is also an off-policy actor-critic DRL and employs two neural networks to approximate two versions of the state-action value function (named *critics*) $Q_{i=1,2}$ parameterized by $\theta_{i=1,2}$. Learning two versions of the critic is used to address the function approximation error by taking the minimum over the bootstrapped Q-values in the next observation o_{t+1}. Different from TD3, SAC gives a bonus reward to an agent at each time step, proportional to the entropy of the policy at that timestep. Different from TD3 learning a deterministic policy μ, SAC learns a stochastic policy π_ψ parameterized by ψ. Given a mini-batch of experiences (o_t, a_t, r_t, o_{t+1}) uniformly sampled from the replay buffer D, the target bootstrapped Q-value $\hat{Q}(o_t, a_t)$ of taking action a_t in observation o_t can be defined with the target networks as follows:

$$\hat{Q}(o_t, a_t) = r_t + \gamma \left[\min_{i=1,2} Q_{\theta_i^-} \left(o_{t+1}, a^- \right) + \alpha H(\pi(\cdot|o_{t+1})) \right], \tag{3.18}$$

where the target Q-value functions are parameterized by θ_i^-, $a^- \sim \pi_{\psi^-}(a|o_{t+1})$ with target policy π_{ψ^-}, and $\alpha \geq 0$ balances the maximization of the accumulated reward and entropy. Then, Q_i can be optimized by minimizing the expected difference between the prediction and the bootstrapped value with respect to parameters θ_i, following

$$\min_{\theta_i} \mathbb{E}_{(o_t, a_t, r_t, o_{t+1}) \sim D} \left[Q_{\theta_i}(o_t, a_t) - \hat{Q}(o_t, a_t) \right]^2, \tag{3.19}$$

where (o_t, a_t, r_t, o_{t+1}) are experiences sampled from replay buffer D. The policy is updated to maximize the expected Q-value on o_t, a where a is sampled from policy $\pi_\psi(\cdot|o_t)$ and the expected entropy of π in observation o_t as follows:

$$\max_{\psi} \mathbb{E}_{o_t \sim D} \left[\min_{i=1,2} Q_{\omega_i}(o_t, a) \mid_{a \sim \pi_\psi(a|o_t)} + \alpha H(\pi_\psi(\cdot, o_t)) \right]. \tag{3.20}$$

The target networks update in SAC is similar to that in DDPG as defined in Eq. 3.13.

3.3.6 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [241] optimizes a policy by taking the biggest possible improvement step using the data collected by the current policy, but at the same time limiting the step size to avoid performance collapse. A common way to achieve this is to attenuate policy adaptation. Formally, for a set of observation and action pairs (o_t, a_t) collected from the environment based on the current policy π_{φ_k}, the new policy π_φ is obtained by maximizing the expectation over the loss function $L(o_t, a_t, \varphi_k, \varphi)$ with respect to the policy parameter φ as $\max_\varphi \mathbb{E}_{(o_t,a_t)\sim\pi_{\varphi_k}} L(o_t, a_t, \varphi_k, \varphi)$. The loss function L is defined as

$$L(o_t, a_t, \varphi_k, \varphi) = \min \left(\frac{\pi_\varphi(a_t|o_t)}{\pi_{\varphi_k}(a_t|o_t)} A^{\pi_{\varphi_k}}(o_t, a_t), \text{clip}(\frac{\pi_\varphi(a_t|o_t)}{\pi_{\varphi_k}(a_t|o_t)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\varphi_k}}(o_t, a_t) \right), \tag{3.21}$$

where ϵ is a small hyperparameter that roughly says how far away the new policy is allowed to go from the current one, and $A^{\pi_{\varphi_k}}(o_t, a_t)$ is the advantage value. A common way to estimate the advantage is called generalized advantage estimator GAE(λ) [240], based on the λ-return and the estimated state-value $V(o_t)$ in observation o_t as $A^{\pi_{\varphi_k}}(o_t, a_t) = R_t(\lambda) - V_v(o_t)$. The $R_t(\lambda)$ is defined by

$$R_t(\lambda) = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t-1} R_t^{(T-t)}, \tag{3.22}$$

where $\lambda \in [0, 1]$ balances the weights of different multi-step returns and the summation of the coefficients satisfies $1 = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} + \lambda^{T-t-1}$. Particularly, $R_t^{(n)}$ is defined as $R_t^{(n)} = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_t + \gamma^n V_v(o_{t+n})$ that is bootstrapped by state-value $V_v(o_{t+n})$ in observation o_{t+n}. The state-value function V_v is optimized to minimize the mean-square-error between the predicted state value $V_v(o_t)$ and the Monte-Carlo return $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_t$ based on the experiences collected by the current policy, as $\min_v \mathbb{E}_{o_t} [V_v(o_t) - R_t]^2$.

3.4 Preference Learning

Preference Learning (PL) [90, 65, 97, 122] is essentially a supervised learning of a ranking function which maps an input to a real number as an indicator of the order of the input. As the learned ranking function can be interpreted as the reward function in RL, PL has also being studied in RL [6, 7, 293, 295] and Deep RL [64, 294, 127] as a way to infer the underlying reward function from the provided ranking information, which is particularly useful for tasks,e.g. tasks with large action space, where either manually specifying the reward function or providing demonstration is difficult. On the contrary, ranking information, especially preference, is easy to be specified by users. Therefore, this work chooses PL as the way to learn the underlying reward function from user preferences in order to eliminate the dependency of manually designed reward function.

When PL is introduced into the RL setting, indicated as *PL+RL*, there are three key differences compared to the standard RL setting. ***Firstly***, as a type of Interactive Reinforcement Learning (IntRL), PL+RL involves two types of interaction. In particular, in addition to the interaction between the agent and environment in the standard RL, there is another interaction between the preference teacher and preference learner, where the preference teacher provides preferences to the learner and the later infers the underlying reward function of the teacher. ***Secondly***, in PL+RL the reward function is normally not assumed to be provided by the environment but learned from the human preference, even though it is still possible to have a handcrafted reward function within the environment and mix it with the reward function learned by PL. ***Thirdly***, the reward function provided by PL is non-stationary, because the collection of the preference labels is a recursive process and the reward function is also periodically trained on the collected preference labels. Fig. 3.5 shows the aforementioned interactions and the generation of the reward signal in PL+RL. In PL+RL, the goal of PL is to learn the reward function approximating the underlying reward function of the human, as well as the policy that maximizes the learned reward function.

Formally, PL+RL can be defined as a MDP, but does not assume the reward function r exists in the environment. Instead, it assumes that there is a preference teacher who can express preferences between trajectory segments. A trajectory segment is a sequence of k observations and actions pairs, $\sigma = ((o_t, a_t), (o_{t+1}, a_{t+1}), \cdots, (o_{t+k-1}, a_{t+k-1})) \in (O \times A)^k$, collected during the interaction between an agent and the environment. Given a pair of segments (σ^0, σ^1), a preference teacher indicates which segment is preferred, i.e., $y = 1$ if σ^1 is preferred over σ^0 indicated as $\sigma^0 \prec \sigma^1$, or $y = 0$ if σ^0 is preferred over σ^1 indicated as $\sigma^0 \succ \sigma^1$, or $y = -1$ if σ^0 and σ^1 are equally preferred indicated as $\sigma^0 \sim \sigma^1$. By

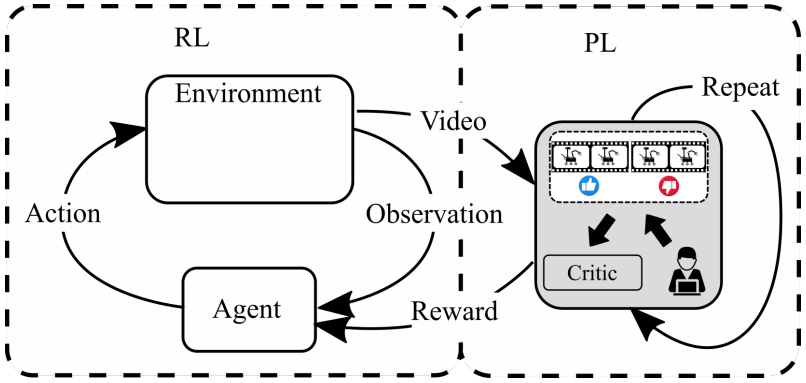


Figure 3.5: Agent and Environment Interaction and Preference Teacher and Preference Learner Interaction in PL+RL

interacting with the preference teacher, a set $\left\{(\sigma^0, \sigma^1, y)^i\right\}_{i=1}^{M}$ of M preference labels are collected. Given a reward function $\hat{r}(o, a) \in \mathbb{R}$ as a preference-predictor, if we view $\hat{r}$ as an approximator of a latent factor r determining the preference teachers' judgement and assume the preference teachers' probability of preferring one segment over another of a segment pair depends exponentially on the summation over the values of the latent reward within the segment, then we have

$$\hat{P}\left[\sigma^0 \succ \sigma^1\right] = \frac{e^{\sum_{i=0}^{k-1} \hat{r}\left(o_{t+i}^0, a_{t+i}^0\right)}}{e^{\sum_{i=0}^{k-1} \hat{r}\left(o_{t+i}^0, a_{t+i}^0\right)} + e^{\sum_{i=0}^{k-1} \hat{r}\left(o_{t+i}^1, a_{t+i}^1\right)}}, \tag{3.23}$$

and $\hat{r}$ can be optimized by minimizing the cross-entropy loss between the predictions $\hat{P}$ and the labels

$$L\left(\hat{r}\right) = -\sum_{(\sigma^0, \sigma^1, y) \in D} (1-y)\hat{P}\left[\sigma^0 \succ \sigma^1\right] + y\hat{P}\left[\sigma^0 \prec \sigma^1\right], \tag{3.24}$$

where $D = \left\{(\sigma^0, \sigma^1, y) \mid \left\{(\sigma^0, \sigma^1, y)^i\right\}_{i=1}^{M} \text{ and } y \neq -1\right\}$ which excludes the segment pairs that are equally preferred.

In practice, PL is more challenging than the simplified theoretic representation. For example, preference labels are collected periodically and then used to estimate the reward function, which causes the reward signals used for policy learning to be non-stationary. In addition, the segment generation and pairing are nontrivial, as they affect the effectiveness and efficiency of preference labelling. Besides, the observation and action are normally

represented as numeric vectors for most applications which are not friendly for preference teachers to assess. Therefore, in reality video clips corresponding to the sequence of the observation and action pairs are used for preference labelling. These practical challenges will be discussed in Chapter 9.

Chapter 4

Experiment Testbeds

The two physical testbeds, namely Aegis Canopy and Meander introduced in this chapter, are developed by Philip Beesley Studio Inc. (https://www.philipbeesleystudioinc.com). For the LAS Simulation Toolkit, the LAS-Behavior-Engine is developed by the engineering team of Philip Beesley Studio Inc. and their collaborators. The author of this thesis solely developed the rest of the simulation toolkit.

In this chapter, we will introduce two physical testbeds and one simulation toolkit. The first physical testbed *Aegis Canopy* was installed in a museum and exhibited for few months. Even though the interaction between the testbed and the visitors is unconstrained, the visitors need to pay for a ticket to access to it. However, the second physical testbed *Meander* is a long-term installation at an event venue where part of the installation is publicly accessible. Therefore, the second testbed is completely uncontrolled and closer to a wild study. The physical testbeds are indispensable to our understanding on how to engage people with an interactive system in a uncontrolled environment, but the development with physical testbeds is too expensive especially when dozens of design choices are involved. For this sake, we developed a simulation toolkit to simulate the physical testbed and visitors. The physical testbeds support ecological validation in real-world environments and with diverse users, while development of the simulation toolkit helps to address the slow algorithm development and poor experiment reproducibility challenges when endowing interactive systems with engaging behavior, as identified in Chapter 1.

4.1 Testbed 1: *Aegis Canopy*

In this section, we describe the physical system used as the testbed in Chapter 5, and the design of *Pre-scripted behaviors (PBs)* that drive the interactive behavior of the system. The PBs, which are designed by expert architects and interactive system designers, are the baseline we use to compare to the learning system described in Section 5.1 below.

The testbed *Aegis Canopy* was part of the exhibition Transforming Space, which consists of two sculptures, i.e. the *Aegis Canopy* and *Noosphere*, as shown in the top-left sub-figure in Fig. 4.1. The exhibition was exhibited at the Royal Ontario Museum (ROM) in Toronto, Canada from June 2 to October 8, 2018 (https://www.rom.on.ca/en/philip) and was publicly accessible to over 60,000 visitors who had toured the exhibit.

Since this research mainly used the *Aegis Canopy* part of the installation, we will describe the design of the *Aegis Canopy* in detail, and subsequently refer to the *Aegis Canopy* part of the installation as the Living Architecture System (LAS).

4.1.1 Physical Installation

The LAS hangs overhead within the *Aegis Canopy* space, with an approximate height of 1.8 meters. Fig. 4.1 shows the front view of the LAS. The active part of the system is composed of eight speakers and 24 nodes.

Each node in the LAS consists of six Fronds, one Moth, and one high-power LED as its actuated systems and one infrared (IR) sensor, as shown in Fig. 4.2. Each Frond includes a shape memory alloy (SMA) wire which contracts when voltage is applied, pulling a cord attached to a flexible co-polyester sheet, as illustrated in Fig. 4.2a. The contraction generates a smooth and gentle movement, and when the applied voltage is removed the SMA slowly relaxes to its original shape.[1] The Moth consists of wing-like flexible flaps attached to a small DC vibration motor that vibrates when activated, making the moth appear to be flapping its wings. The Moth also houses two small LED lights which illuminate during vibration. The single high-power LED located beneath the central flask can be faded to illuminate the colored liquid in the flask. The IR sensor senses the proximity of visitors, and generates a continuous reading proportional to the distance between any part of the body of a visitor and the sensor location in the *Aegis Canopy*.

[1] A video illustrating the SMA assembly contracting and relaxing can be found in https://youtu.be/YcreirXrRF4.

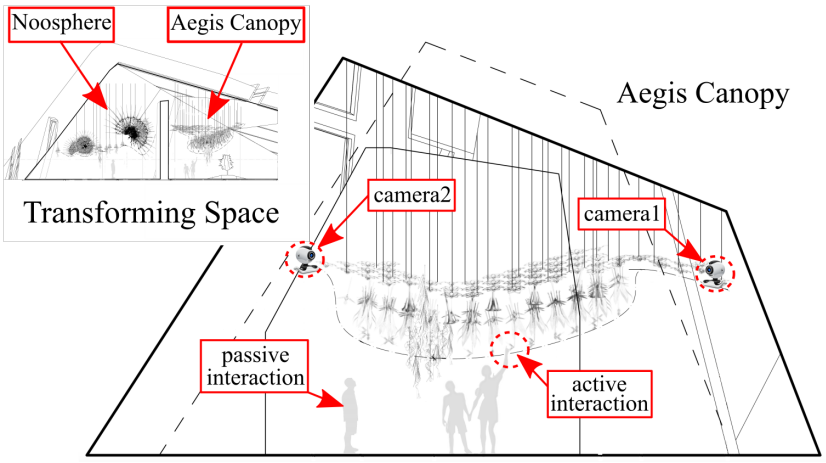
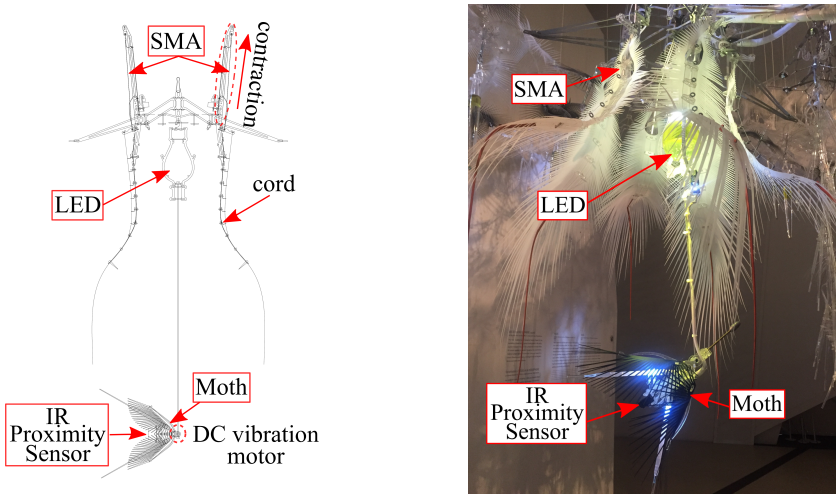


Figure 4.1: Installation Diagram and Interaction Types



(a) Node Diagram (b) Fully Assembled Node

Figure 4.2: Diagram of Node in the LAS

There are eight speakers distributed throughout the LAS. These speakers play two types of sound samples. The first sound is a background sound played on a continuous loop. The second sound is triggered by the IR sensors. These speakers are independently controlled by specialized software, so here we treat them as background behaviors.

The arrangement of the speakers and nodes is illustrated in Fig. 4.3, where the 24 nodes are highlighted by red circles. A photo of the physical LAS is shown in the bottom-left of

Fig. 4.3. The 24 nodes are at varying height levels. Specifically, nodes at the left and right edges are slightly higher than those in the middle of the LAS. This spatial arrangement distinguishes three types of visitor engagement with the system. When visitors observe the LAS but are not underneath the LAS, no IR sensor is activated, i.e., visitors are observing the LAS but cannot be observed by the LAS sensors. As shown in Fig. 4.1, when visitors walk or stand underneath the LAS, which we name *Passive Interaction*, the IR sensors above them are activated, but the distance between the visitor and the system is still large, corresponding to a small reading of the IR sensor. Visitors engaging in *Active Interaction* might also reach their hand upwards to interact with the LAS, resulting in a higher activation value of the closest IR sensor.

Two web cameras (labeled Camera1 and Camera2 in Fig. 4.1 and Fig. 4.3) are mainly used to record video during our experiment and to calibrate sensory data; they are not used by the Deep Reinforcement Learning (DRL) algorithm. These two web cameras are mounted on the wall in the front-right and back-left corners of the LAS space.

4.1.2 Pre-scripted Behavior

Pre-scripted behavior (PB) is the interactive behavior manually designed by the architects, and it is also the baseline used for comparison with adaptive behaviors we will describe in Section 5.1. Within the PB mode, the system can be in two types of states: active and background, in which behaviors are mainly controlled by 17 parameters (shown in Table 4.1) specified by the architects. The values in the Default and Range columns are used in the PB and Parameterized Learning Agent (PLA), which will be introduced in Chapter 5, modes, respectively.

The active state is entered if any of the IR sensors is triggered. In this state, the node corresponding to the triggered sensor will first activate its *local reflex behavior*. In the local reflex behavior, the Moth, the LED and six SMAs attached to the same node as the triggered IR sensor will be activated. When a Moth is activated, it will ramp up the vibration to its maximum I_{max} over time T^m_{ru}, hold there for a period of time T^m_{ho}, then ramp down over (T^m_{rd}). After a waiting period (T^m_{gap}) following the sensor trigger, the LED on the same node is activated. It ramps up over time period (T^l_{ru}) to its maximum brightness (I_{max}), holds for a period of time (T^l_{ho}) and then gradually dims over (T^l_{rd}). At the same time, the SMAs are activated one after another separated by (T^{sma}_{gap}). A step voltage is applied to contract the SMA, after which a cooling-down time is started during which this SMA will not be activated again. The activation profile of the SMA wires is fixed in order to protect them from overheating, so these are not included in the parameterization shown

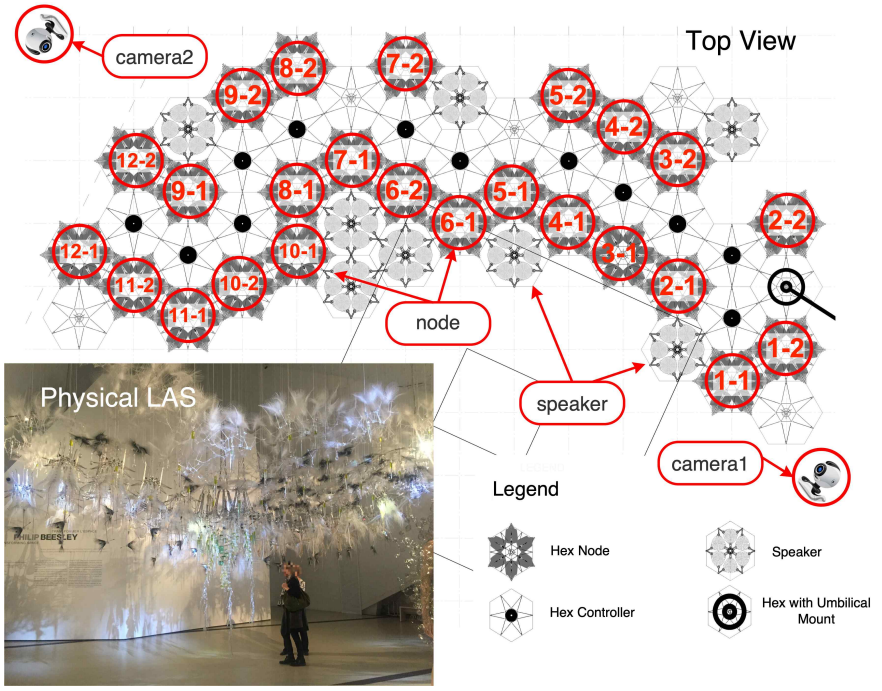


Figure 4.3: LAS: *Aegis Canopy*

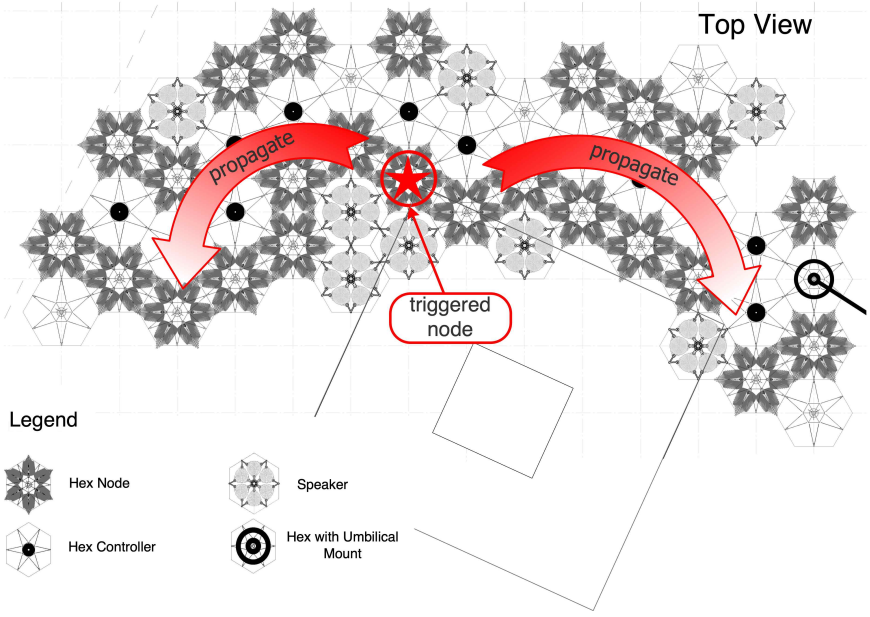


Figure 4.4: Pre-scripted Behavior

Table 4.1: Pre-scripted Behavior Parameters

Parameters	Meaning	Default	Range
T^m_{ru}, T^l_{ru}	ramp up time: the time it takes for the Moths or LEDs to increase up to their maximum value	1.5	$[0, 5]$
T^m_{ho}, T^l_{ho}	hold time: the time that Moths and LEDs are held at their maximum value	1	$[0, 5]$
T^m_{rd}, T^l_{rd}	ramp down time: the time it takes for Moths and LEDs to fade down to 0	2.5	$[0, 5]$
I_{max}	maximum percentage of duty cycle per PWM period	78	$[0, 100]$
T^m_{gap}	the time gap between the Moth starting to ramp up and the LED starting to ramp up	1.5	$[0, 5]$
T^{sma}_{gap}	the time gap between activation of each SMA armon the nodes	0.3	$[0, 5]$
T^n_{gap}	the time gap between activation of each node during neighbour behavior	1.8	$[0, 5]$
T^{min}_{bg}	minimum time to wait before activating background behavior	45	$[15, 60]$
T^{max}_{bg}	maximum time to wait before activating background behavior	90	$[60, 100]$
T_w	time to wait before trying to pick a moth or LED	5	$[0, 10]$
P	probability of successfully choosing an actuator during background behavior	0.4	$[0, 1]$
T_{sma}	time between choosing SMAs to actuate	0.7	$[1, 5]$
T^{min}_{sw}	minimum time to wait before performing sweep	120	$[5, 200]$
T^{max}_{sw}	maximum time to wait before performing sweep	240	$[200, 400]$

The unit of all time parameters is seconds, except I_{max} is percentage and P is probability.

in Table 4.1. After the local reflex behavior is triggered, the IR-detected event will be propagated from the triggered node to neighbouring nodes after a delay (T^n_{gap}), until the edge nodes of the LAS are reached (shown in Fig. 4.4), causing a cascade of local reflex behaviors at each node.

If no IR sensor triggering happens for a random time within $\left[T^{min}_{bg}, T^{max}_{bg}\right]$, the system goes into the background state. In this state, the LEDs and moths will randomly activate

their local reflex behaviors with probability P every amount of time T_w. The SMAs are also activated independently with the same probability P every T_{sma}.

In either state, a sweep of LEDs in either direction along the longer axis of the installation happens at random time intervals within $[T_{sw}^{min}, T_{sw}^{max}]$. During the sweep, each LED activates local reflex behavior and propagates in the direction of the sweep.

4.2 LAS Simulation Toolkit (LAS-Sim-Tkt)

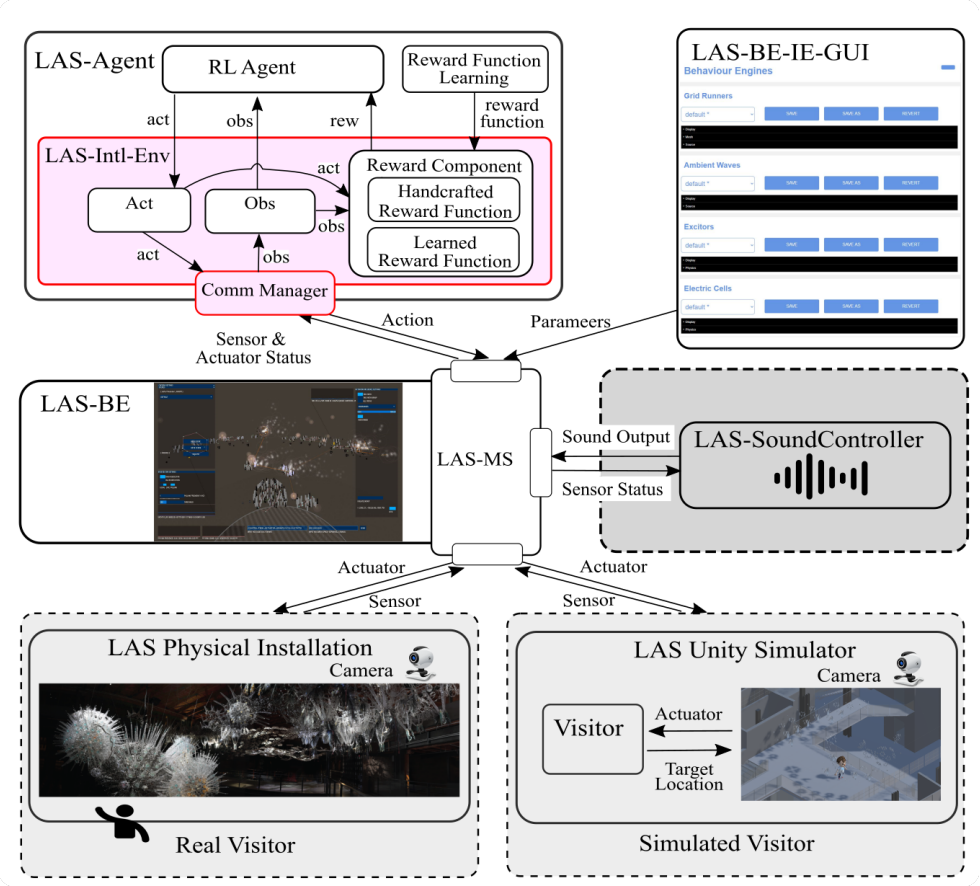


Figure 4.5: LAS-Sim-Tkt Diagram.

To facilitate algorithm development and validation, we developed a LAS Simulation Toolkit (LAS-Sim-Tkt). As shown in Fig. 4.5, LAS-Sim-Tkt is a combination of LAS-

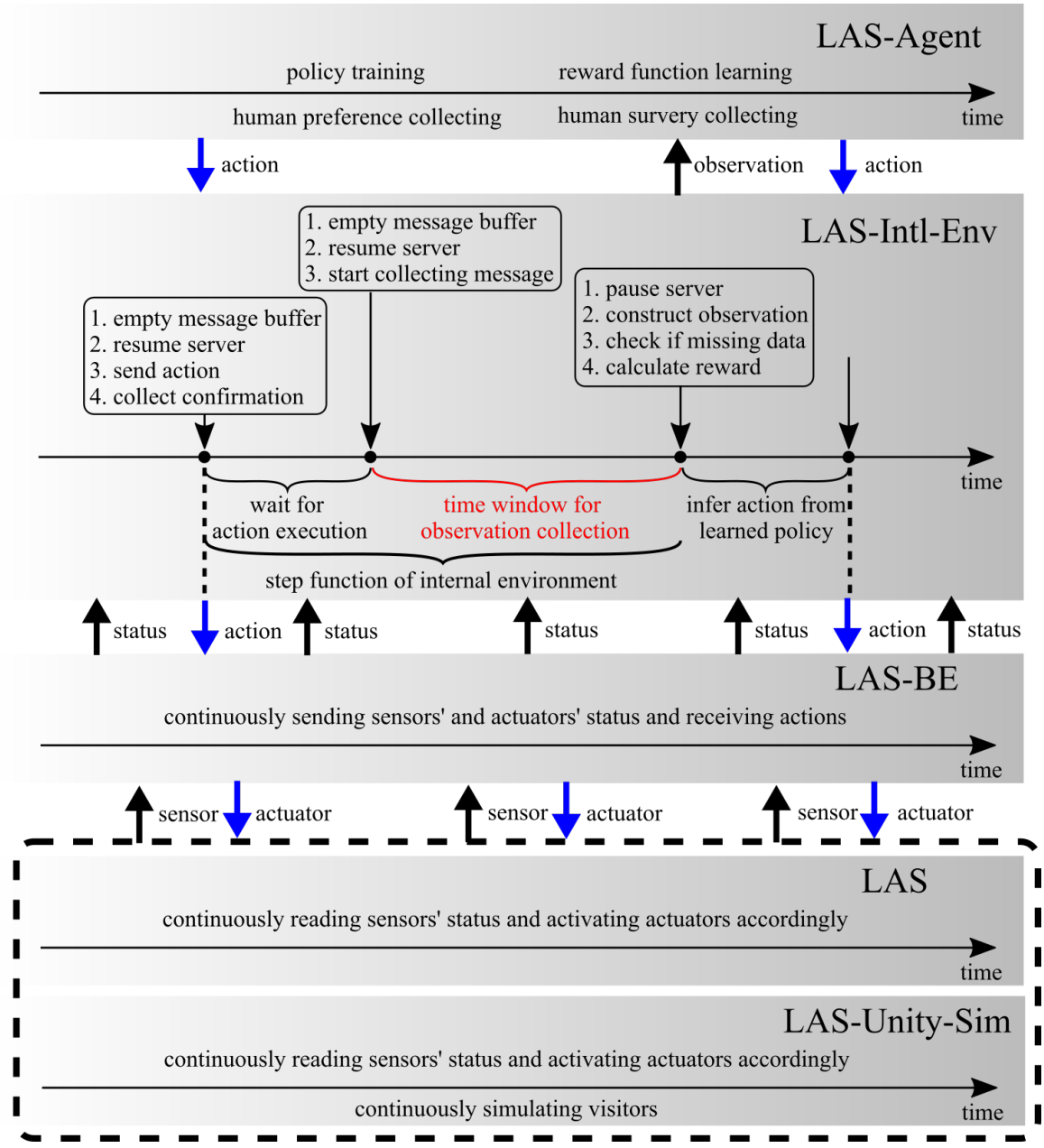


Figure 4.6: Timeline of Modules in LAS-Sim-Tkt, where all modules are running in parallel, and the components within a module might either follow a strict sequential sequence or run in parallel depending on the logical structure of the components.

Behavior-Engine (LAS-BE), LAS-Message-Server (LAS-MS), LAS-BE-Influence-Engine-GUI (LAS-BE-IE-GUI), LAS-Unity-Simulator (LAS-Uni-Sim), and LAS-Agent-Internal-Environment (LAS-Intl-Env)[2]. The communication among the modules is through the LAS-MS, where the LAS-BE, the LAS-BE-IE-GUI, the LAS-Uni-Sim and the LAS-Intl-Env communicate by passing Open Sound Control (OSC)[3] messages, while the LAS-BE and the LAS communicate via User Datagram Protocol (UDP). As indicated with dashed boxes in Fig. 4.5, there is no need to have a LAS physical installation and/or LAS-Uni-Sim in order to run a simulation. Fig. 4.6 illustrates the parallel running of the modules in the LAS-Sim-Tkt along a timeline, where the components within a module might either follow a strict sequential sequence or run in parallel depending on the logical structure of the components, which will be elaborated in the following subsections. In particular, from the top to the bottom of Fig. 4.6, the action generated by LAS-Agent is passed to LAS-Intl-Env to be interpreted. Then, it will be passed to LAS-BE to be executed by sending commands to the actuators in LAS and/or LAS-Unity-Sim, where LAS, i.e. the physical installation, is not needed for simulation and LAS-Unity-Sim is used for better visualization and for simulating visitors. From the bottom to the top of Fig. 4.6, sensory readings are transmitted to LAS-BE. After collected by LAS-BE, sensory readings will be sent to LAS-Intl-Env for constructing observation which will be used by LAS-Agent to learn policy and infer next action, etc.

LAS-Sim-Tkt is used for initial algorithm development and validation, especially when the accessibility of a physical installation is very limited. An additional function of the toolkit is to pretrain a learning agent within the simulator and transfer the learned policy to the physical system. The toolkit has been built into a Singularity (https://sylabs.io) container and opensourced on https://github.com/LinghengMeng/las_sim_tkt.

In this section, we will describe each module of the toolkit. The toolkit provides the infrastructure for our experiments, which necessitates a detailed description.

4.2.1 LAS-Behavior-Engine (LAS-BE)

LAS-Behavior-Engine (LAS-BE) is developed by the engineering team at the Philip Beesley Studio Inc., which is a core middle layer module based on Processing (a flexible software sketchbook and a language for learning how to code within the context of the visual arts)[4],

[2]The LAS-SoundController in Fig. 4.5 only works with the physical LAS rather than the simulated LAS.

[3]https://en.wikipedia.org/wiki/Open_Sound_Control

[4]https://processing.org/

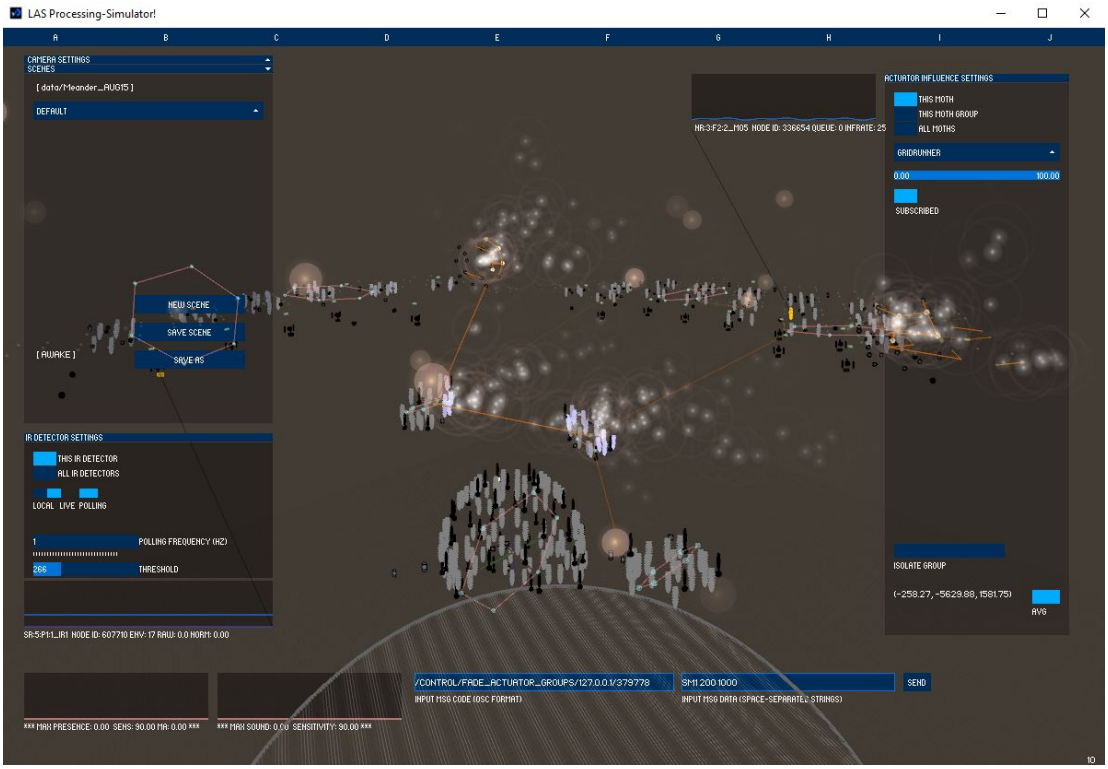


Figure 4.7: Screenshot of LAS-BE.

and is responsible for interpreting high-level behavior design into control commands for hardware. Fig. 4.7 shows a screenshot of the LAS-BE. The LAS-BE is used to choreograph the LAS's interactive behavior, which is challenging due to the large number of actuators (up to hundreds of actuators within a typical installation), the wide area distribution of sensors and actuators (e.g. spread over a whole building), and the complexity of social factors during HRI (e.g. many-to-many interaction). LAS-BE eases the design of the pre-scripted behaviors for architects and enables a parameterized action space for a learning agent. Although a detailed design of the LAS-BE is out of the scope of this work, we would like to briefly introduce the high-level concepts invented for LAS-BE to highlight the complexity of designing pre-scripted behaviors and to emphasize the necessity for introducing AI technology to this application field.

At a high level, the behaviors of the actuators are determined by two concepts namely the *Influence Engine*, and the *Influence Map*. Specifically, the influence map defines which actuators are subscribed to which influence engine, and the influence engine is the al-

gorithm(s) that generates the influence on the subscribed actuators. In addition to the influence engine, an actuator could also be influenced by a sensor and/or other predefined sources. Each influence engine is characterized by a set of parameters which defines its dynamics, and hence its influence on subscribed actuators. With the great flexibility provided by the influence engine and influence map mechanism of the LAS-BE, the architect can specify new, overlapping influence engines of different types, and subscribe actuators to each engine individually or in groups. In this section we describe the idea of each influence engine at the high level and introduce a typical set of parameters that are most related to our work, rather than an exhaustive list of all the possible engines and parameters.

Influence Map: A Subscription of Influence Source

An actuator can subscribe to one or more of the following influence sources: influence engines including GridRunner, AmbientWave, Excitor and ElectricCell, nearby sensors including IR sensor, sound detector, grideye presence (processed sensor data that indicates the presence of humans)[5], random noise, and sample behaviors hardcoded by the engineers. When an actuator is subscribed to multiple influence sources, its response is determined by the combination of all subscribed influence sources by superposing the intensities induced by the various subscription sources. Formally, if the intensity range of an actuator is in $[0, \iota_{max}]$ and it is under the influence of N_ι influence sources each with induced intensity ι_i on it, then the final intensity ι of the actuator is as follows:

$$\iota = \min \left[\sum_{i=1}^{N_\iota} \iota_i, \iota_{max} \right]. \tag{4.1}$$

Influence Engine: GridRunner

A key underlying structure in the LAS scaffold is a 3D hexagonal grid, with actuators populated throughout. The GridRunner influence engine considers each vertex of this grid as a potential source of (virtual) particles. Every time a sources is activated, it will emit particles in a specific direction, and these particles steer on the predefined 3D grid, activating any (physical) actuators as they pass by. Specifically, the steering path of the particles follow the edges of the hexagons with different speeds. Each particle has a sphere

[5]Grideye sensors and IR sensors are sensing different areas of the LAS. In addition, grideye sensors are able to detect a larger area than IR sensors and roughly infer human gestures.

Table 4.2: Parameters of GridRunner Influence Engine

ParamName	Description	Min	Max
nParticles	Determine how many particles can be generated in total	5	2000
sourceRotation	Rotation of the source of the burst of the particles	0	2
sourceSpread	Spread of the source of the burst of the particles	0	6.28
sourceHeading	Heading of the source of the burst of the particles	0	6.28
burstInterval	Frequency of the burst	10	5000
burstQty	Quantity of the burst	0	250
yVelocity	Verticle speed of particles	0	1
influenceSize	The size of the influence	0	1500
influenceIntensity	Intensity of the influence	0	1
maxSpeed	The maximum speed of the particles	0.5	10

influence range, where actuators within the sphere will be activated. The sources can be one or more of the following source type:

- Mouse Source: the sources that are activated when the mouse passes by the predefined grid vertices. This is designed for manually activating sources within the LAS-BE.
- Sensor Source: the sources located near a sensor that can either be activated manually by clicking the checkbox in the LAS-BE-IE-GUI (which will be introduced in section 4.2.1) or be activated by the nearby sensor. For example, if the IR sensor's reading is above its threshold, the source nearby the sensor will be activated. This type of source is mainly to improve the interactivity, as the sensor reading can only be changed by a visitor who is trying to interact with the sculpture.

Within a LAS, there can be as many sources as wished by engineers for each type of source. The dynamics of the GridRunner is defined by a set of parameters, described in Table 4.2 and illustrated in Fig. 4.8. In Table 4.2, *nParticles* controls how many particles can coexist within the 3D space of a sculpture, the following 5 parameters apply to any source defined in the space, and the last 4 parameters apply to any particle that can be generated by a source. Fig. 4.8a illustrates the overall effect of this influence engine on the actuators within a grid world, where the actuators will be activated with the intensity of the particles emitted by a source. 4.8b shows that both the influence size and intensity

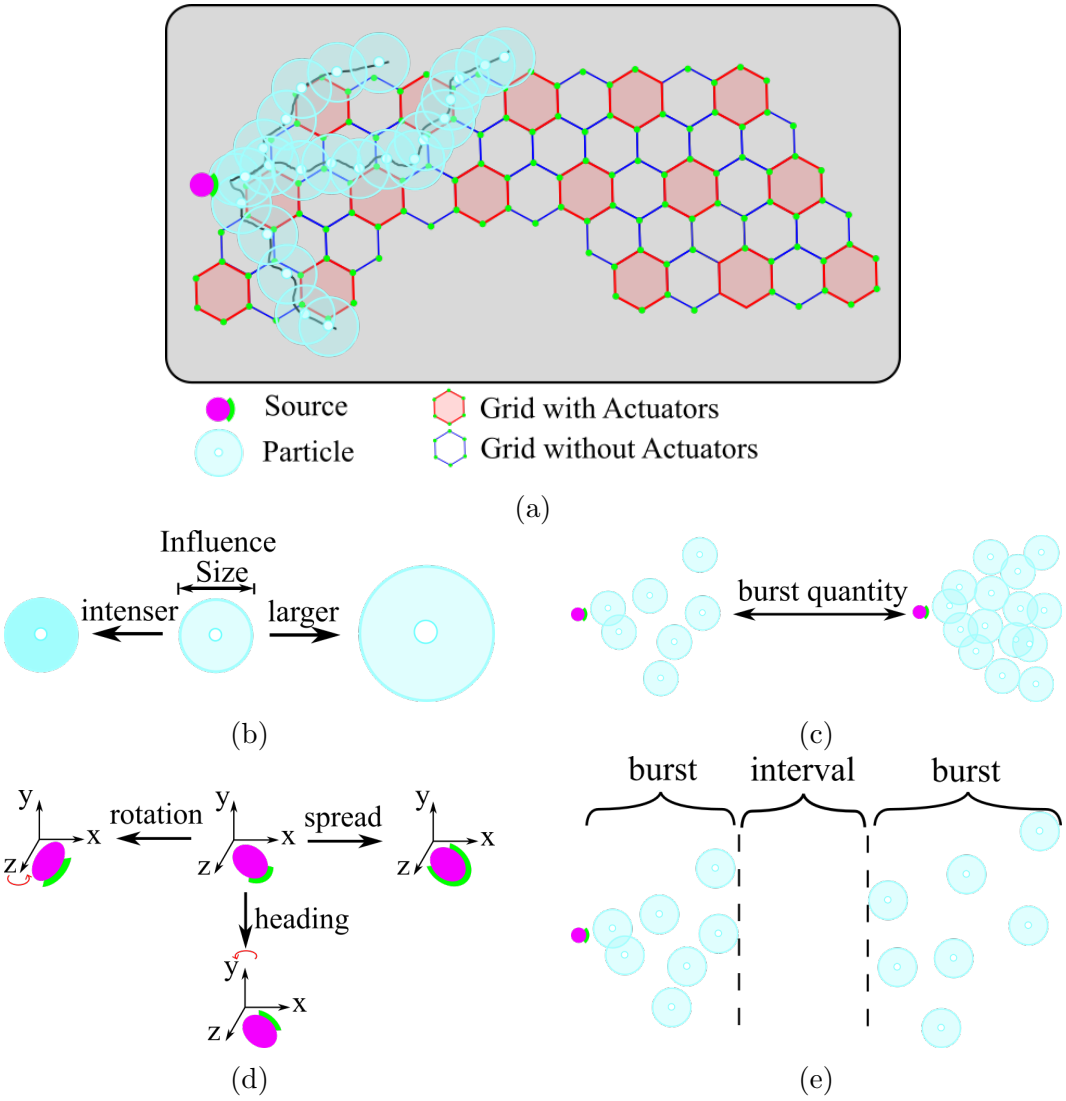


Figure 4.8: GridRunner Diagrams, where (a) shows the overall effect of the influence engine, (b) shows the influence size and intensity change of a particle, (c) depicts the effect of the burst quantity, (d) shows the effect of source spread, heading and rotation, and (e) shows the burst interval.

can be tuned. The *burstQty* is used to change the burst quantity as shown in Fig. 4.8c. As shown in Fig. 4.8d, the *sourceRotation* rotates the source along the z-axis, while the *sourceHeading* turns the source heading along the y-axis. The *spread* controls how wide

the particles can be emitted from the source. Between two consecutive bursts, there is a pause during which no particles are emitted, which is determined by the *burstInterval* as shown in Fig. 4.8e.

Table 4.3: Parameters of AmbientWave Influence Engine

ParamName	Description	Min	Max
waveActive	Indicate whether activate a wave influence	Flase	True
velocity	The propagation velocity of a wave	0	2
period	The width of a wave	0	1
angle	The propagation direction of a wave	0	6.283
amplitude	The intensity of the activation when an actuator is touched by a wave	0	1

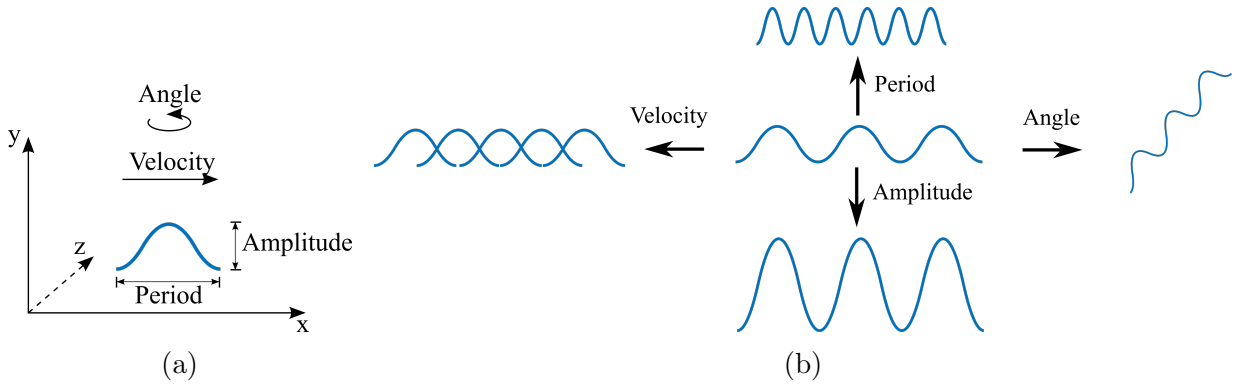


Figure 4.9: AmbientWave Diagram, where (a) illustrates the parameter change dimensions and (b) shows the outcomes when turning these parameters.

Influence Engine: AmbientWave

The AmbientWave influence engine provides dynamic global patterns imitating (virtual) waves, which flow throughout the LAS and aim to influence a large portion of (physical) actuators as the waves wash over their location. The waves generated by AmbientWave will determine the intensity to which an actuator will be activated. Table 4.3 illustrates the involved parameters and their possible value range, while Fig. 4.9 depicts the change

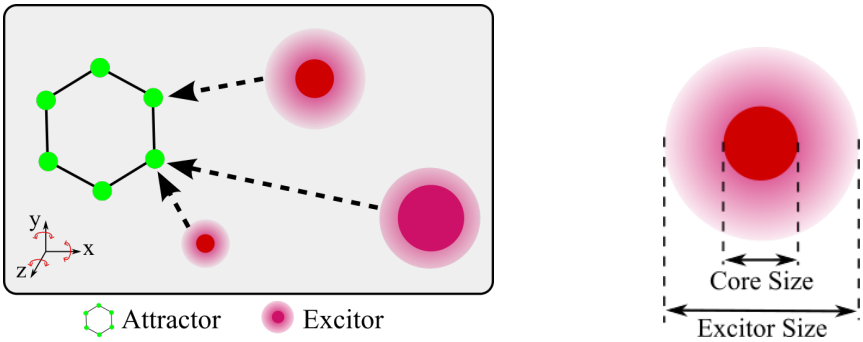
dimensions related to the AmbientWave. When waves are generated consecutively, the outcomes of these parameters are illustrated in Fig. 4.9b, which clearly shows the diverse effects created by this influence engine. Even though AmbientWave is designed for a global pattern, it can also be customized for a local pattern by specifying a small group of actuators that can be influenced by an AmbientWave influence engine.

Table 4.4: Parameters of Excitor Influence Engine

ParamName	Description	Min	Max
excitorSize	The size of an excitor	40	2000
excitorCoreSize	The portion of the core of an excitor	0	1
excitorLifespan	The lifespan of an excitor	500	20000
excitorMasterIntensity	The intensity of the activation of an actuator triggered with an excitor	0	1
excitorSpeedLimit	The maximum speed of an excitor.	0	1
attractorAngleSpeed	The angular speed of an attractor.	0	0.25
attractorForceScalar	The force scalar of the attractor to attract an excitor	0	5
maxExcitorAmount	The maximum number of excitors that can coexist within the space	1	35
bgHowOften	How often the background excitors are generated	250	1000

Influence Engine: Excitor

The Excitor influence engine implements a (virtual) spherical object moving through the LAS towards an attractor, activating any (physical) subscribed actuators it passes on its route. The attractor, specified by architects, plays the role of a hotspot that attracts excitors to fly to it. Depending on the source of an excitor, four types of excitor are designed: Sound Excitor, Motion Excitor, Presence Excitor and Background Excitor, which are generated by a sound detector, the movement of human detected by a grideye sensor, the presence of humans detected by a grideye sensor, and randomly picked locations, respectively. The parameters related to excitor are listed in Table 4.4, and the diagram in Fig. 4.10. In Fig. 4.10, (a) shows excitors are attracted to attractors and the the 3D axes



(a) (b)

Figure 4.10: Excitor Diagram, where (a) shows excitors are attracted to attractors and the the 3D axes indicate the movement are in 3D, and (b) depicts the relationship between the core size and the excitor size.

indicate the movement are in 3D, and (b) depicts the relationship between the core size and the excitor size. Also note that, the rotation of the attractor and the excitor motion are all in 3D space.

Table 4.5: Parameters of ElectricCell Influence Engine

ParamName	**Description**	**Min**	**Max**
active	Indicate whether the ElectricCells influence engine is active	Flase	True
masterIntensity	Intensity of the trigger of an actuator	0	1
neighbourRange	The distance range from the current cell within which the next actuator is randomly selected and activated	1	10
cellCount	The amount of electric cells flowing from actuator to actuator	1	8
rate	The rate at which the cells jump between actuators	10	100
triggerChange	The probability that an actuator can be triggered by the cell it encountered	0	1

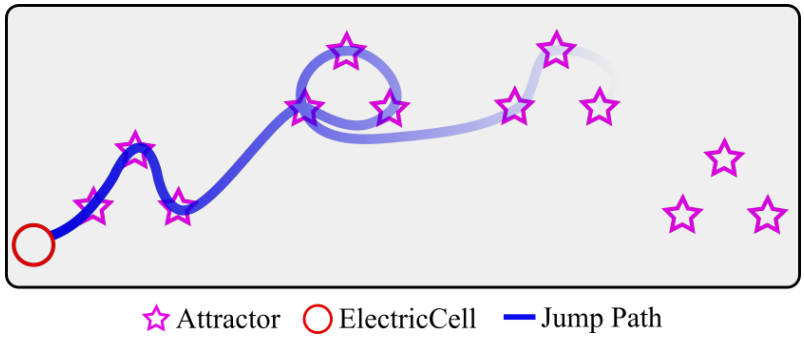


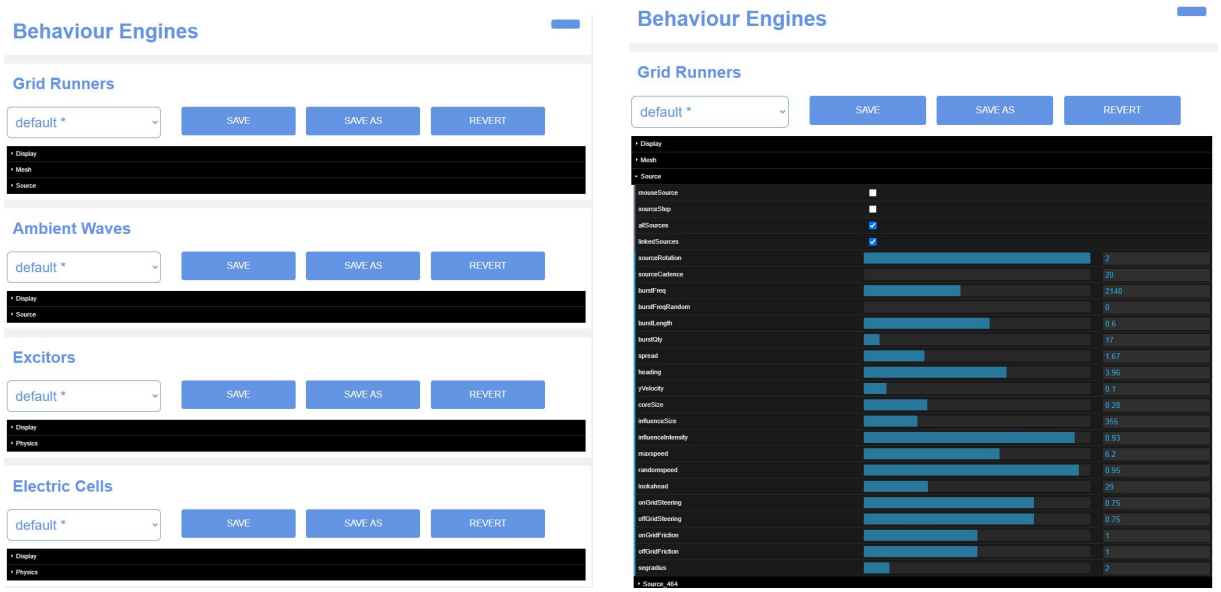
Figure 4.11: ElectricCell Diagram

Influence Engine: ElectricCells

The ElectricCell influence engine creates fast patterns by simulating a (virtual) "lightning bolt" that jumps from actuator to actuator within a given range (nearest neighbours), at a given rate (ms), activating the (physical) actuators as it passes by. The purpose is to generate semi-random paths that look like electric shocks are moving through the sculpture. Table 4.5 describes the parameters related to this influence engine, and Fig. 4.11 illustrates an ElectricCell jumps among actuators along the path until the maximum neighbour range and/or the maximum cell count is reached.

LAS-Message-Server (LAS-MS)

The LAS-Message-Server (LAS-MS) connects the LAS-BE with other modules i.e. the LAS-Unity-Simulator (LAS-Uni-Sim) 4.2.2 and the LAS-Agent-Internal-Environment (LAS-Intl-Env) 4.2.3 by sending messages through the high speed network. Within the LAS-Sim-Tkt, modules can run on separate computers depending on the computation resources requirements and the required resilience of the whole system. In particular, when applying Machine Learning (ML) techniques within the LAS-Agent shown in Fig. 4.5 to control the LAS, large CPU and GPU resources may be required. Therefore, separately running the resource-consuming ML related computation on a powerful computer and passing either the learned policy or the action based on the policy through LAS-MS can make the control of the LAS smoother and faster. In addition, separating the LAS-BE and policy learning with distributed computers meanwhile saving a copy of the learned policy on the machine running LAS-BE improves the resilience of the whole system, as the shutdown of the machine running ML related computation will not interrupt the control.



(a) Folded (b) Unfolded

Figure 4.12: Screenshot of LAS-BE-IE-GUI, where (a) and (b) corresponds to folded and unfolded GUI panel respectively.

LAS-BE-Influence-Engine-GUI (LAS-BE-IE-GUI)

The GUI of the LAS-BE shown in the Fig. 4.12a is mainly used for designing the subscription relationship between actuators and influences and visualizing the choreographed behavior of a LAS, rather than tuning the parameters related to various influence engines. To complement the design of the influence engines, a web-based LAS-BE-Influence-Engine-GUI (LAS-BE-IE-GUI) is developed to choreograph the various influence engines. By cooperating with the LAS-BE, the outcomes of the tuning of the parameters of the influence engines within the LAS-BE-IE-GUI can be visualized immediately and restored for future use. Fig. 4.12a is the screenshot of the LAS-BE-IE-GUI. Unfolding the panels under each influence engine, the parameters will show up and be ready to be tuned by dragging the value bar as shown in Fig. 4.12b.

4.2.2 LAS-Unity-Simulator (LAS-Uni-Sim)

To facilitate evaluation of design strategies before deploying to physical installations and conducting field studies, we developed a simulator called LAS-Unity-Simulator (*LAS-Uni-*

Sim) based on Unity (https://unity.com/). The simulator can not only simulate sensors and actuators in *Meander* but also the visitors with either manually designed or autonomous policies.

One of LAS-BE's functionalities is to allow architects and engineers to collaboratively choreograph the LAS's interactive behavior even before deploying the designed behavior into the physical system, based on simulated sensory inputs. However, the visual fidelity is limited due to the visual art development platform Processing [6] used by LAS-BE, and the LAS-BE lacks a collision detection mechanism and path planning capacity which are essential to simulate visitors. Since our algorithms will be finally deployed to the LAS whose visual appearance of its interactive behavior will be perceived by both visitors and by teachers who provide their preferences, both the visual fidelity and simulated visitors are necessary for our initial research before moving to field study. Therefore, we further developed a simulator based on Unity to enable the direct visual assessment and the flexible simulation of visitors' behavior. We call this simulator the *LAS Unity-Simulator (LAS-Uni-Sim)*.

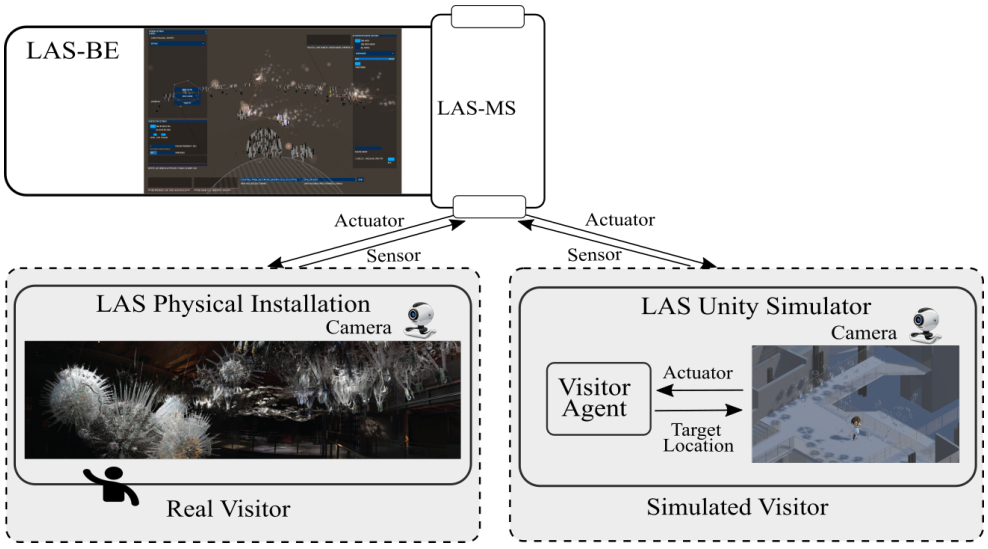
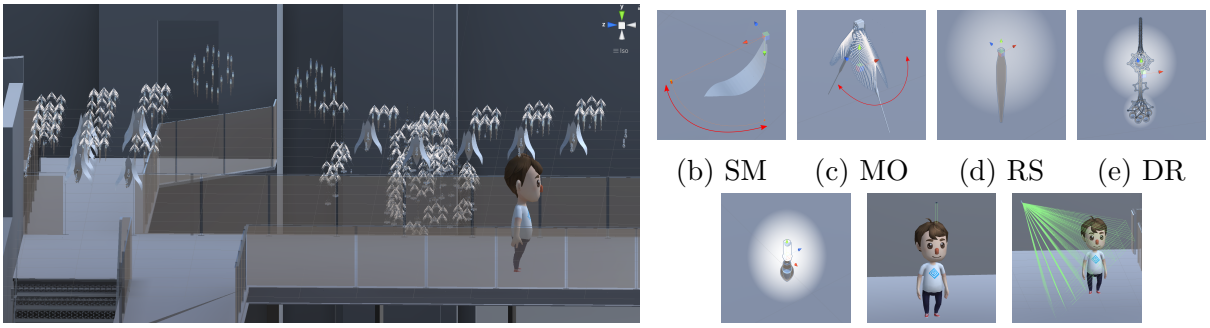


Figure 4.13: Communication Diagram Among LAS, LAS-BE, and LAS-Uni-Sim.

The communication among LAS, LAS-BE, and LAS-Uni-Sim is shown in Fig. 4.5, but for convenience we reproduced the related portion in Fig. 4.13. The main purpose of having LAS-Uni-Sim is to simulate sensors detecting simulated visitors. With LAS-Uni-Sim, we are able to validate our design of learning algorithms prior to recruiting

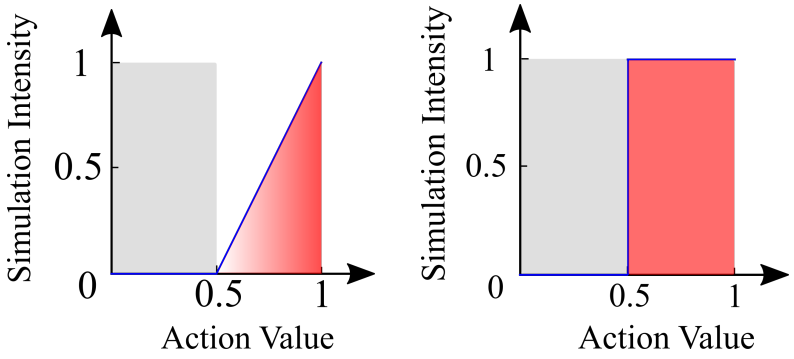
[6] https://processing.org/

participants. Specifically, simulated sensor signals are sent from LAS-Uni-Sim to LAS-BE, then LAS-BE generates raw actuator commands based on pre-scripted high-level behavior. After that, the raw actuator commands are either sent to the physical system or to LAS-Uni-Sim for execution.



(a) *Meander* in LAS-Uni-Sim (b) SM (c) MO (d) RS (e) DR (f) PC (g) IR (h) GE

Figure 4.14: LAS-Uni-Sim, where (a) shows *Meander* (the name of the physical LAS that will be introduced in Section 4.3) in LAS-Uni-Sim, and (b)-(h) are implemented sensors and actuators in LAS-Uni-Sim. Specifically, the sensors are Infrared sensor (IR), and GridEye Infrared array sensor (GE), while actuators are Moth (MO), Rebel Star (RS), Double Rebel Star (DR), Protocell (PC), and Shape-Memory Alloy (SM).



(a) Continuous Activation (b) Threshold Activation

Figure 4.15: Balanced Activation, where "balanced" means that the probability of the off and on state of an actuator is 0.5. Specifically, (a) has continuous activation when the action value is greater than 0.5, and the activation of (b) will jump to 1 for any action value greater than 0.5.

Fig. 4.14 shows the simulated testbed *Meander* (the name of the physical LAS that

will be introduced in Section 4.3) in LAS-Uni-Sim and the simulated sensors, i.e., Infrared sensor (IR) and GridEye Infrared array sensor (GE), and actuators, i.e., Moth (MO), Rebel Star (RS), Double Rebel Star (DR), Protocell (PC), and Shape-Memory Alloy (SM). Note that sound related sensors and actuators, i.e. Sound Detector (SD) and Open Sound Speaker (OS), are not implemented. Fig. 4.15 illustrates the two types of activation function for different actuators, where Fig. 4.15b is only used for SM and Fig. 4.15a is used for the rest of the actuator types. For an action value in $[0, 1]$ issued on an actuator, we want to make the on and off state of the actuator to be equal, i.e. a value in $[0, 0.5)$ corresponds to off and a value in $[0.5, 1]$ corresponds to on. Even though SM only has two states, i.e. either on or off, other actuators, e.g. light, can have different intensities when they are on, so for the action value in $[0.5, 1]$ the intensity of the actuator grows from $[0, 1]$. The reason for this consideration is to ease the learning algorithm to equally explore to the off and on state of an actuator which are the most obvious visual difference.

4.2.3 LAS-Agent-Internal-Environment (LAS-Intl-Env)

In order to align with the classical environment interface exploited in the OpenAI Gym[7] and to separate the development of learning algorithms from the different choices of control and sensing abilities of a learning agent, e.g. the choice of the action and observation space, we designed a LAS Machine Learning Agent Internal Environment (LAS-Intl-Env) which is the middle layer between the environment, including both the external (i.e. exteroception) and the internal (i.e. proprioception) environment of a LAS, and the learning agent. Fig. 4.5 highlights the functionality of the LAS-Intl-Env in the whole control system when the learning capability is enabled, where for convenience the connection between the LAS-BE and the LAS-Intl-Env is reproduced in Fig. 4.16.

As shown in Fig. 4.16, at the high-level the LAS-Intl-Env is composed of four components namely Action Execution Component, Observation Construction Component, Reward Component, and Communication Manager. Specifically, the Action Execution Component involves the definition of the action space which could be either raw actuator space or parameterized action space. The Observation Construction Component specifies the observation space which may include both the exteroception, i.e., the perception of the external world of the LAS (e.g. the existence of the visitors), and the propriception, i.e., the perception of the status of the LAS (e.g. the status of an actuator). By mixing the exteroception and the proprioception together, an agent can get a better understanding of the state of the environment and make an appropriate action decision to maximize the

[7] https://gym.openai.com

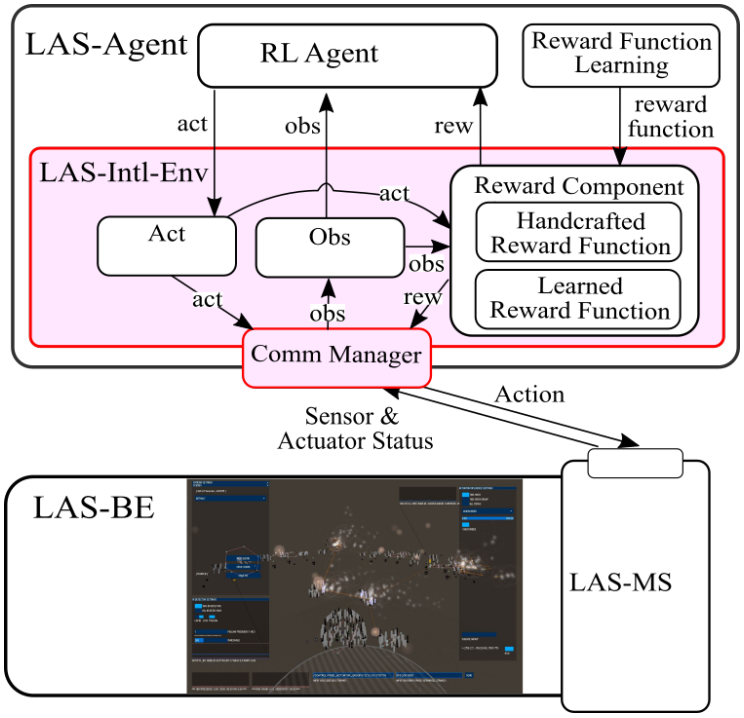


Figure 4.16: LAS-Intl-Env Diagram.

goal defined by the Reward Component. The Reward Component specifies the reward function at time step t, indicating the goodness/badness of taking an action. The Communication Manager is responsible for the communication between the other components in the LAS-Inter-Env and the LAS-BE. In this section, the high-level cooperation among the components will be introduced first, followed by the details about each component and the summary about the various choices and challenges involved in the design of these components.

In order to better simulate the real world interaction between the LAS and the visitors, which cannot be paused to wait for the rendering of the state of the environment, the LAS-Intl-Env asynchronously receives the sensors' and actuators' status from and sends action to the LAS-BE while the interaction is continuously happening as shown in Fig. 4.6. Even though the parallel running of the modules makes the learning task harder because of the potential data loss and inconsistency, it can make the behavior smoother because there is no pause to synchronize the outcome of the taking of an action. Therefore, to keep the simulation as realistic as possible, the LAS-Sim-Tkt keeps the asynchronous communication mechanism among the various modules.

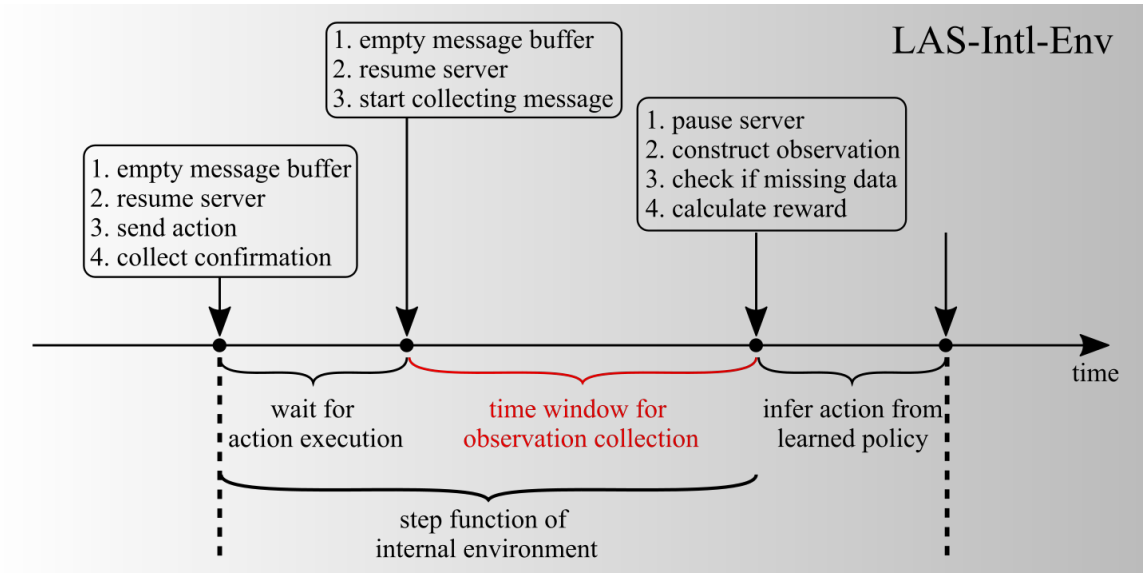


Figure 4.17: LAS-Intl-Env Step Timeline.

ALGORITHM 1: Pseudo Code for the Interaction between LAS-Intl-Env and Learning Agent

```
Initialize: intl_env, agent
while not done do
    /* Infer and execute action, and return observation and reward */
    act ← agent.infer_action(obs)
    new_obs, rew, done, info ← intl_env.step(act)
    /* Store experience into replay buffer if not missing data */
    if info.missing_data == True then
        obs, info ← intl_env.get_obs()
    else
        replay_buffer.store(obs, act, rew, new_obs, done, info)
        obs ← new_obs
    end
end
```

The interaction between an agent and the LAS-BE is in discrete time and realized by calling the step function of the LAS-Intl-Env, as with the other popular benchmarks in the RL community. By using a similar interface to other RL benchmarks, the state of the art RL algorithms can be easily adopted to LAS. Alg. 1 illustrates the pseudo code

for the interaction between the LAS-Intl-Env and a learning agent. Inside Alg. 1, the *intl_env.step*(*obs*) involves the collaboration among the components within LAS-Intl-Env, which will be introduced in the rest of this section and detailed in Alg. 2. Specifically, an agent infers the next action according to its policy when observing an observation. Then, the LAS-Intl-Env sends the action to LAS-BE for execution, and waits to receive the new observation, the reward signal, the task terminal indicator, and the diagnostic information (e.g. missing data information) after taking the action. This procedure continues until reaching the terminal state or the maximum episode length.

The time interval between two consecutive action executions is considered as one time step. Within one time step, the collaboration among the components of the LAS-Intl-Env is illustrated in Fig. 4.17. As shown in Fig. 4.17, a time step starts with emptying the message buffer for action-received confirmation, resuming message server to receive action-received confirmation from LAS-BE, sending the inferred action to LAS-BE, and collecting the action-received confirmation from LAS-BE. Then, LAS-Intl-Env will wait for T_{aw} seconds for the action receiving and execution, where T_{aw} is actually the minimum time to execute an action and observe its immediate/short-term effect[8]. In other words, if T_{aw} is equal to 0, it assumes that the effect of taking the action on the environment is immediately observable, whereas if T_{aw} is greater than 0, it assumes that it takes some time for the action to take effect or the action will last for a while. In many control problems in robotics, it is quite common that the actuators run at a very high frequency so that the execution time is negligible, which means the sensory reading immediately after the sending of an action command can be thought as the new observation of the environment. Or in other cases, the action execution and observation construction run in a synchronized way that the observation is constructed only after the execution of the action. However, these are not applicable to LAS. Firstly, the action execution time in LAS cannot be ignored both for the raw or parameterized action space because of the nature of the actuators (e.g. SMA which gently contracts and relax) and the nature of the parameterized action (e.g. the excitor takes time to fly through the actuators to activate them). Secondly, the action execution and the observation construction are asynchronized as shown in Fig. 4.6 that the action sent from LAS-Intl-Env is executed in LAS-BE and cannot be paused when collecting observations. Therefore, T_{aw} is especially useful when the immediate effect of taking an action takes longer time and including the effects of the actions before that harms the decision making. After waiting for action execution, there is a time window T_{ow} for

[8]The immediate/short-term effect is to differentiate delayed/long-term effect, where the first one corresponds to the earliest effect credited to the action execution and the later one is the delayed effect of the action execution after a while, which may be intertwined with the effects of the other actions taken during the time.

collecting the observation. The action execution and observation time window combined together determine the time interval used for collecting sensor readings and actuator status in order to construct the new observation. Even though T_{aw} can be set to 0 as the effect of the action execution on the actuator status can be observed immediately, T_{ow} is rarely set to 0, because to observe the effect of the execution of an action on the sensors perceiving the human robot interaction typically takes some time. Once the observation time window is passed, the message server will be paused, followed by the observation construction, the missing data checking and the reward calculation. With this information, a new action based on the new observation will be inferred from the agent's policy.

As illustrated in Fig. 4.17 and the description in the previous paragraph, the interaction frequency f can be calculated by Eq. 4.2:

$$f = \frac{1}{T_{aw} + T_{ow} + T_{ai}}, \tag{4.2}$$

where T_{ai} is the time used for action inference. It can be seen that the larger the $T_{aw} + T_{ow} + T_{ai}$ is , the lower the interaction frequency will be. It is worth to emphasize that the T_{ow} cannot be as small as in most of HRI applications, including LAS, as that for other robotics control tasks, because the human in the loop cannot interact with a robot with very high frequency. For LAS, this is more prominent, because the architectural scale of the sculpture makes the behavior of the sculpture harder to perceive and makes the human's reaction slower, e.g. moving from one part to another part of the sculpture. It is not hard to imagine that if f is very high, the sensor readings within t_ow at time step t may be the same as that at time step $t-1$.

Communication Manager

Communication Manager (CM) is responsible for managing the communication between the LAS-BE and the other components in the LAS-Intl-Env. Specifically, CM starts the server for (1) receiving sensor readings and actuator status from the LAS-BE and distributing them to the Observation Construction Component of LAS-Intl-Env, and (2) receiving the action received confirmation from the LAS-BE to confirm an action is received. In addition, CM sends interpreted actions to the LAS-BE.

Action Execution Component

Action Execution Component (AEC) defines the action space for the learning agent.The learning agent does not need to know the meaning of each dimension of the action space.

It is the AEC's responsibility to interpret the actions received from the learning agent and send the interpreted action to the LAS-BE with the help of CM. Formally, the action space $\boldsymbol{a} \in [-1, 1]^d$ is a vector specifying all possible actions, where the dimension of $\boldsymbol{a}$ is d, and each dimension $a^{(i)}$ (where $i = \{1, \cdots, d\}$) has the value range $[-1, 1]$. The value range for each dimension is constrained to [-1, 1], whereas the value range of the original action $a'^{(i)}$ is $[a'^{(i)}_{max}, a'^{(i)}_{min}]$, where the i is the corresponding dimension index and the maximum $a'^{(i)}_{max}$ and the minimum $a'^{(i)}_{min}$ value for each dimension may be different depending on the nature of the action dimension. The interpretation of the action $\boldsymbol{a}$ to the original action $\boldsymbol{a}_{orig}$ depends on the data type as follows:

$$a'^{(i)} = \begin{cases} \frac{(a^{(i)}+1)}{2} \times \left(a'^{(i)}_{max} - a'^{(i)}_{min}\right) + a'^{(i)}_{min}, & \text{if } a'^{(i)} \text{ is float,} \\ \begin{cases} False, & \text{if } a^{(i)} < 0 \\ True, & \text{Otherwise.} \end{cases}, & \text{if } a'^{(i)} \text{ is bool,} \\ \left\lfloor \frac{(a^{(i)}+1)}{2} \times \left(a'^{(i)}_{max} - a'^{(i)}_{min}\right) + a'^{(i)}_{min} + 0.5 \right\rfloor, & \text{If } a'^{(i)} \text{ is integer,} \end{cases} \tag{4.3}$$

where i is the dimension index of the action $\boldsymbol{a}$ and $\lfloor \cdot + 0.5 \rfloor$ rounds a value $\cdot$ up to the nearest integer. Eq. 4.3 shows how the action $\boldsymbol{a}$ in the action space is converted into the original action $\boldsymbol{a}'$ that can be understood and executed by the LAS-BE.

The original action can be in either the raw actuator space (i.e. the direct control of physical actuators) or the parameterized action space (i.e. control the primitive parameters, e.g. the parameters related to the influence engines introduced in Section 4.2.1, which will subsequently affect the behavior of raw actuators).

Observation Construction Component

Observation Construction Component (OCC) defines the observation space for the learning agent. Specifically, it retrieves the sensory readings and actuator status from CM, preprocesses and constructs the observation according to the definition of the observation space. Formally, assume within the observation window T_{ow}, $o'_{n_{s_i}}$ is the n_{s_i}th of the N_{s_i} sensory readings of the sensor $s_i \in \{1, \cdots, S\}$ with the value range $[o'_{min_{s_i}}, o'_{max_{s_i}}]$ and $o'_{n_{a_i}}$ is the n_{a_i}th of the N_{a_i} actuator status of the actuator $a_i \in \{1, \cdots, A\}$ with the value range $[o'_{min_{a_i}}, o'_{max_{a_i}}]$, received by MC from the LAS-BE, the $o'_{s_i n_{s_i}}$ and $o'_{a_i n_{a_i}}$ will be converted to the range $[0, 1]$ as follows:

$$o_{n_{s_i}} = \frac{o'_{n_{s_i}} - o'_{min_{s_i}}}{o'_{max_{s_i}} - o'_{min_{s_i}}} \qquad \text{and} \qquad o_{n_{a_i}} = \frac{o'_{n_{a_i}} - o'_{min_{a_i}}}{o'_{max_{a_i}} - o'_{min_{a_i}}}. \tag{4.4}$$

With the processed sensory reading $o_{n_{s_i}}$ and the processed actuator status $o_{n_{a_i}}$, the aggregated sensory reading o_{s_i} for sensor s_i and the aggregated actuator status o_{a_i} for actuator a_i can be calculated either by averaging the processed data within the observation time window as follows:

$$o_{s_i} = \frac{1}{N_{s_i}} \sum_{n_{s_i}=1}^{N_{s_i}} o_{n_{s_i}} \qquad \text{and} \qquad o_{a_i} = \frac{1}{N_{a_i}} \sum_{n_{a_i}=1}^{N_{a_i}} o_{n_{a_i}}, \tag{4.5}$$

or by concatenating the processed data within the observation time window as follows:

$$o_{s_i} = \bigcup_{n_{s_i}=1}^{N_{s_i}} o_{n_{s_i}} \qquad \text{and} \qquad o_{a_i} = \bigcup_{n_{a_i}=1}^{N_{a_i}} o_{n_{a_i}}, \tag{4.6}$$

where $\bigcup$ is the concatenation operator[9]. After that, then the observation can be divided into two types (i.e. the exteroception and the proprioception) and calculated as follows:

$$o_s = \bigcup_{s_i=1}^{S} o_{s_i} \qquad \text{and} \qquad o_a = \bigcup_{a_i=1}^{A} o_{a_i}, \tag{4.7}$$

where S and A are the total number of sensors and actuators involved in the LAS. Based on the choice of perception, the final observation o can be formed as follows:

$$o = \begin{cases} o_s, & \text{if only use exteroception,} \\ o_a, & \text{if only use proprioception,} \\ o_s \bigcup o_a, & \text{if use both.} \end{cases} \tag{4.8}$$

Reward Component

Reward Component (RC) is computed based on the action and observation from AEC and OCC, respectively, as shown in Fig. 4.16. The RC provides a reward function to evaluate the goodness of taking an action when observing an observation. This reward function can be handcrafted by engineers, or learned by some algorithms, e.g. Inverse Reinforcement Learning, or a combination of them. The reward r_t at time step t can be a function of observation and/or action variables as follows:

$$r_t = R(o_t, a_t) \quad \text{or} \quad r_t = R(o_{t+1}) \quad \text{or} \quad r_t = R(o_t, a_t, o_{t+1}) \quad \text{or} \quad r_t = R(o_{t+1}|h_t^l) \tag{4.9}$$

[9]Scalar value is essentially a vector with only one dimension, so in this section we will use lowercase letter to represent both the scalar and vector.

where o_{t+1} is the new observation after taking the action a_t when observing o_t, and h_t^l is the observation history sequence at time step t with the length l with the following format:

$$h_t^l = \begin{cases} o_{t-l+1}, a_{t-l+1}, \cdots, o_t, a_t & \text{if t>l} \\ o_1, a_1, \cdots, o_t, a_t & \text{otherwise.} \end{cases} \tag{4.10}$$

Summary

Alg. 2 summarizes the pseudo code for the step function of LAS-Intl-Env which involves the collaboration among the components as depicted in Fig. 4.16. For a clear understanding of the complexity of the control task, the following list summarizes the design choices in each component:

- CM: Communication Manager
  - T_{aw}: Action window determines when to start observation window within one time step.
  - T_{ow}: Observation window determines how many sensory readings and actuator status are received for one time step.
- AEC: Action Execution Component
  - $\boldsymbol{a}$: action space
    - raw actuator space with hundreds of actuators.
    - parameterized action space with dozens of parameters.
- OCC: Observation Construction Component
  - S: Sensors in the observation space.
  - A: Actuators in observation space.
  - o_{s_i}: aggregated observation on sensors over the received sensory readings within the observation window.
    - Average (Eq. 4.5)
    - Concatenate (Eq. 4.6)
  - o_{a_i}: aggregated observation on actuators over the received actuator status within the observation window

ALGORITHM 2: Pseudo Code for Step Function of LAS-Intl-Env

Input: a_t

Output: o_{t+1}, r_t, d_t, $info$

Initialize: time window size for waiting for action execution T_{aw}, time window size for collecting observation T_{ow}, observation construction type *obs_construct*, reward calculation type *rew_type*

`/* Execute action */`

convert a_t to original action command a'_t

send a'_t to LAS-BE for execution

`/* Wait for action execution */`

sleep(T_{aw})

`/* Collect and construct observation */`

empty message buffer

resume message server

sleep(T_{ow})

pause message server

collect original sensory readings $\left\{\left\{o'_{n_{s_i}}\right\}_{n_{s_i}=1}^{N_{s_i}}\right\}_{s_i=1}^{S}$ and actuator status $\left\{\left\{a'_{n_{a_i}}\right\}_{n_{a_i}=1}^{N_{a_i}}\right\}_{a_i=1}^{A}$ from message buffer

$missing\_data \leftarrow$ check if missing data

$\left\{\left\{o_{n_{s_i}}\right\}_{n_{s_i}=1}^{N_{s_i}}\right\}_{s_i=1}^{S}$, $\left\{\left\{a_{n_{a_i}}\right\}_{n_{a_i}=1}^{N_{a_i}}\right\}_{a_i=1}^{A}$ convert original sensory readings and actuator status to $[0, 1]$ according to Eq. 4.4

$\{o_{s_i}\}_{s_i=1}^{S}$, $\{o_{a_i}\}_{a_i=1}^{A} \leftarrow \leftarrow$ aggregate processed sensory readings and actuator status according to Eq.Eq.4.6 and 4.6

o_s, $o_a \leftarrow$ aggregate exteroception and proprioception according to Eq. 4.7

$o_{t+1} \leftarrow$ construct final observation according to Eq. 4.8

`/* Calculate reward */`

$r_t \leftarrow$ calculate reward according to Eq, 4.9 and 4.10

`/* Determine if task is done */`

$d_t \leftarrow$ determine_terminal_state($new\_observation$)

`/* Set diagnosing information */`

$info.missing\_data \leftarrow missing\_data$

    * Average (Eq. 4.5)
    * Concatenate (Eq. 4.6)
- type

    - o_s: exteroception only.
    - o_a: proprioception only.
    - $o_s \bigcup o_a$: both exteroception and proprioception.

- RC: Reward Component
  - format
    - $r_t = R(o_t, a_t)$ depends on o_t and a_t. When the environment dynamics is not deterministic or nonstationary, o_t and a_t may be inefficient to capture enough information to determine the goodness of taking an action.
    - $r_t = R(o_{t+1})$ depends on o_{t+1}. When the environment dynamics is not deterministic or nonstationary, o_t and a_t may be inefficient to capture enough information to determine the goodness of taking an action.
    - $r_t = R(o_t, a_t, o_{t+1})$ depends on o_{t+1}. Even though o_t, a_t, and o_{t+1} working together are providing enough information when the underlying decision process is one-degree MDP, i.e. the underlying current state is all needed to make a decision, for decision process that relies on previous states this format is not enough.
    - $r_t = R(o_{t+1}|h_t^l)$ depends on o_{t+1} and the past history h_t^l. This format tries to include as much information as possible, but this requires a very powerful learning method and much more computation resources.

4.3 Testbed 2: *Meander*

Meander is the testbed used to evaluate the approaches proposed in Chapter 9. It is a long-term installation at Tapestry Hall in Cambridge, Canada (https://www.tapestryhall.ca/meander/), where parts of the installation are accessible to the public and some are private. The component actuators and sensors of *Meander* are similar to those of *Aegis Canopy* introduced in 4.1. However, the middle layer control algorithm LAS-BE introduced in Section 4.2 is different from that in Section 4.1.

4.3.1 Physical Installation

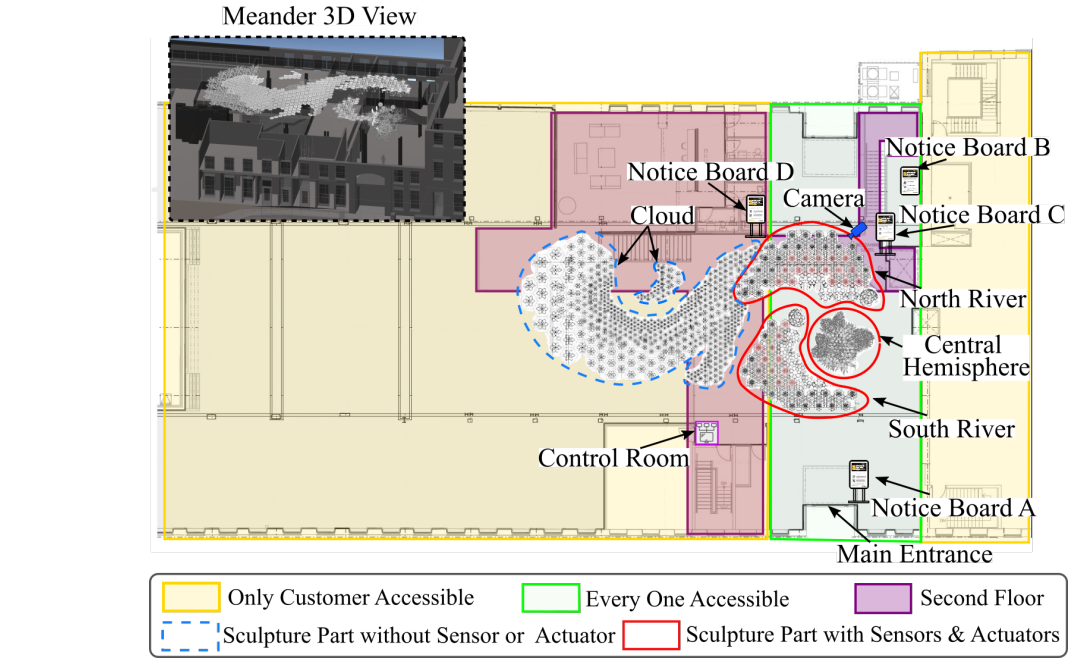
Meander is a large-scale testbed environment constructed within Tapestry Hall (https://www.tapestryhall.ca), a historic warehouse building at the centre of a residential highrise

development in Cambridge, Ontario. Sensors embedded within the environment signal the presence of visitors, and send ripples of light, motion and sound through the system in response, under the control of LAS-BE with either pre-scripted behavior choreographed by experts or learned behavior based on ML techniques.

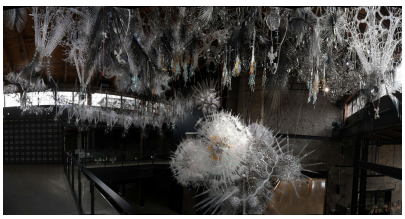
Fig. 4.18 shows a schematic diagram and photographs of *Meander*. Specifically, Fig. 4.18a illustrates the composition of *Meander*, namely Cloud, North River (NR), South River (SR) and Central Hemisphere. Cloud does not have sensors and actuators for interaction, while the other three parts all have the capacity to sense and interact with visitors. The spatial position of *Meander* in the building can be better viewed in the top-left 3D view panel, from which it can be seen that the North River, South River and Cloud are hanging above the hallway on the second floor (pink and purple in Fig. 4.18a) and accessible to visitors on the second floor, while the Central Hemisphere is hanging above the ground on the first floor (green area) in the lobby and is accessible to visitors in the lobby. Fig. 4.18b, 4.18c and 4.18d are images of *Meander* taken on the second floor. The sculpture parts in the green and purple areas, i.e. the NR and Central Hemisphere, are accessible to public visitors, whereas access to the sculpture part in the pink area, i.e. the SR, is restricted. There are gates between the public and private spaces. Fig. 4.18g shows one of the gates on the second floor. By accessible, we mean the capacity to physically interact with the sculpture, i.e. the ability to be detected by a sensor. However, the majority of the sculpture is within the view of the visitors in the public accessible space. There is a camera (Fig. 4.18f) onsite to collect videos for further study and for maintenance. The Control Room is locked and hidden in the corner, where all control facilities are located as shown in Fig. 4.18e. To inform the visitors that there is a study ongoing, we will locate four posters on notice boards in visible locations, e.g. the main entrance and locations near the sculpture on both floors (as indicated in Fig. 4.18a). The posters have a brief introduction of the study and the QR code leads potential participants to the website with more details about the experiment. Fig. 4.18h shows a visitor who is interacting with the North River part of *Meander*.

Sensors and actuators are embedded in *Meander* to endow it with perception and reaction ability. Specifically, the sensors employed are Infrared sensor (IR), GridEye Infrared array sensor (GE), and Sound Detector (SD), while actuators employed are Moth (MO), Rebel Star (RS), Double Rebel Star (DR), Protocell (PC), Shape-Memory Alloy (SM), and Open Sound speaker (OS).

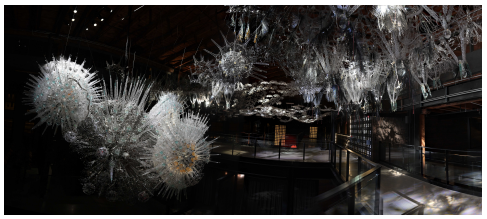
The IR sensor senses the proximity of visitors, and generates a continuous reading proportional to the distance between any part of the body of a visitor and the sensor location. The GE is a state-of-the-art thermopile sensor that features 64 thermopile elements in an 8×8 grid format, detecting a low-resolution heat scan of its field of view. The SD provides



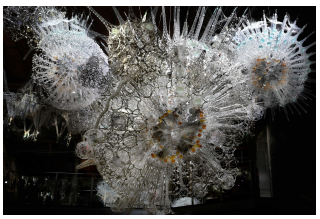
(a) *Meander* Sculpture Composition



(b) South River

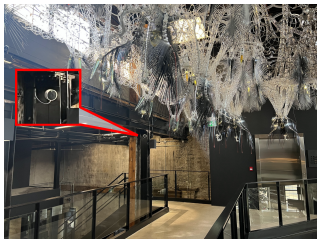


(c) North River



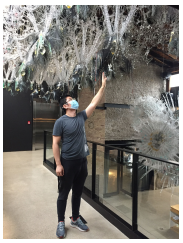
(d) Central Hemisphere

(e) Control Room



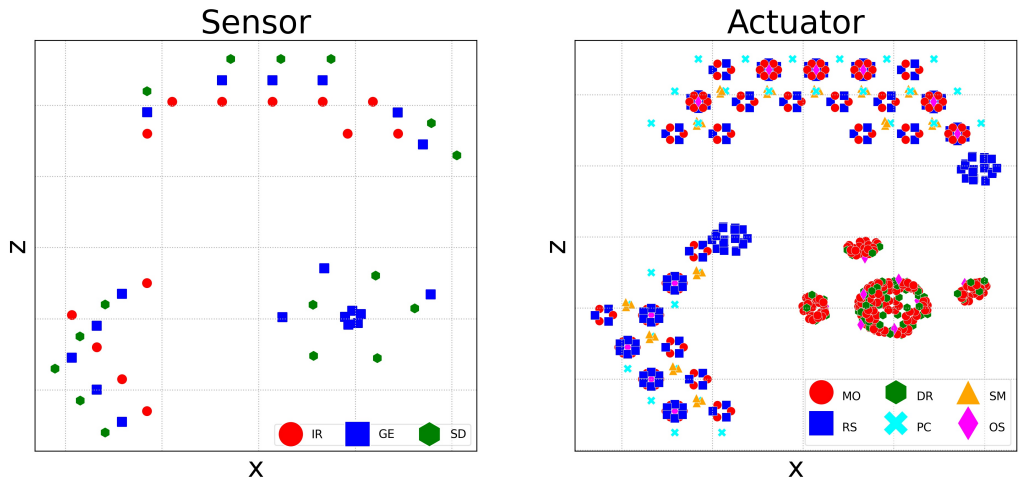
(f) Camera

(g) Gate Access

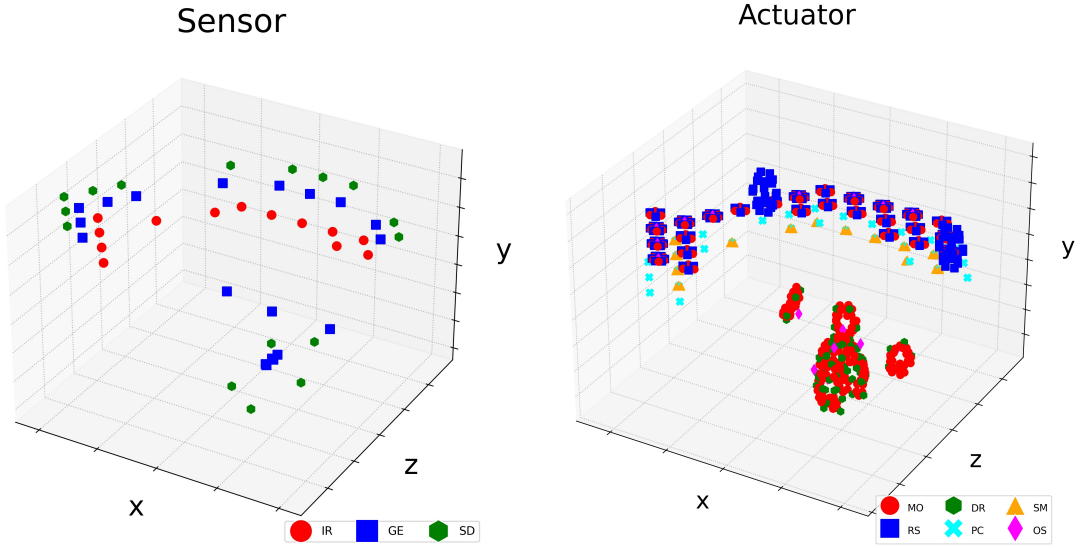


(h) Visitor

Figure 4.18: Physical *Meander*, where (a) shows the overall composition of the sculpture, namely the South River, the North River and the Central Hemisphere; (b), (c) and (d) are images of the three parts of *Meander* (courtesy of Philip Beesley Studio Inc.); (e) is the control room; (f) is the camera used to collect video recordings; (g) is the gate access between NR and SR; and (h) is a visitor is interacting with *Meander*.



(a) 2D



(b) 3D

Figure 4.19: Sensor and Actuator Type and Location, where (a) is plotted in 2D from top view to emphasize sensors and actuators 2D arrangement, and (b) is plotted in 3D to emphasize sensors' and actuators' variation in height.

a binary indication of the presence of sound, and an analog representation of its amplitude.

The Moth consists of wing-like flexible flaps attached to a small DC vibration motor

Table 4.6: Summary of Sensors and Actuators

		North River	South River	Central Hemisphere	Cloud
Sensors	**IR**	8	5	0	0
	GE	6	5	8	0
	SD	6	5	5	0
	Sum	20	15	13	0
Actuators	**MO**	66	45	150	0
	RS	86	65	0	0
	DR	0	0	60	0
	PC	18	11	0	0
	SM	24	15	0	0
	OS	6	5	8	0
	Sum	200	141	218	0

that vibrates when activated, making the moth appear to be flapping its wings. The Moth also houses two small LED lights which illuminate during vibration. The RS is a single high-power LED located beneath the central flask, which can be faded to illuminate the colored liquid in the flask. The DR is actually two high-power LEDs mounted on the top and bottom of a plastic bar. The PC is also a single high-power LED, but located beneath a central flask with a different shape from that for RS. SM contracts when voltage is applied, pulling a cord attached to a flexible co-polyester sheet, as illustrated in Fig. 4.2a. The contraction generates a smooth and gentle movement, and when the applied voltage is removed the SM slowly relaxes to its original shape.[10] The OS is essentially a speaker controlled by sophisticated 4D sound software to play immersive, spatialized multi-channel sound.

For clarity, Fig. 4.19 shows the position of sensors and actuators embedded in *Meander*, and Table 4.6 summarizes the number of sensors and actuators in different parts of *Meander*. From Fig. 4.19 and Table 4.6, it can be summarized that:

- Sensors and actuators are spread over the whole space (Fig. 4.19a) at various heights (Fig. 4.19b) to enable the LAS to simultaneously accommodate multiple interactions happening among visitors and the LAS.
- The diversity of sensors and actuators enhances the LAS's ability to perceive various

[10] A video illustrating the SM assembly contracting and relaxing can be found at https://youtu.be/YcreirXrRF4.

sources of information happening in the space and to react in different ways.

4.3.2 Pre-scripted Behavior

The pre-scripted behavior of *Meander* is defined by the influence maps among actuators, sensors and the parameters of the influence engines within the LAS-BE as introduced in section 4.2.1. For the dynamics of the LAS-BE, please refer to section 4.2.1, and in this section we will only describe the setting of the parameter values of various influence engines employed in the pre-scripted behavior, which will be the base of the parameterized action space for the baseline of the learning based approaches proposed in section 9. The influence maps are fixed not only for the pre-scripted behavior but also for the parameterized action space of the learning agent introduced in Chapter 9, so the details of them are out of the scope of this thesis and will be omitted. Since in Chapter 9 the parameters of the influence engines of the pre-scripted behavior will be used as the action space of a learning agent, we will elaborate them in the following section. However, we should note that for the pre-scripted behavior, the influence engine settings are fixed, which greatly limits its flexibility and adaptability.

Influence Engine Settings

Table 4.7 shows the the parameter values for each influence engine in the default pre-scripted behavior followed by its value range. Once the pre-scripted behavior is set, these parameter values will not change during the course of the running of the given pre-scripted behavior, even though another pre-scripted behavior can be loaded to replace the current one. Therefore, the pre-scripted behavior is carefully designed by architects based on their knowledge.

4.4 Summary

In this chapter, we introduced two physical testbeds namely *Aegis Canopy* and *Meander*, and the LAS simulation toolkit. For the two testbeds, we first described the sensor and actuator composition, then illustrated their pre-scripted behavior. Different from the Aegis, which is installed in a museum and accessible to visitors who paid a ticket, the *Meander* is installed in a building where part of the sculpture is publicly accessible. However, the physical testbeds are not always accessible to researchers and visitors. Therefore, we

Table 4.7: Parameters of the Influence Engines of the Pre-scripted Behavior

Influence Engine	ParamName	Pre-scripted Value	Min	Max	Count
GridRunner	nParticles	1915	5	2000	11
	sourceRotation	2	0	2	
	sourceSpread	1.67	0	6.28	
	sourceHeading	3.96	0	6.28	
	burstInterval	2140	10	5000	
	burstQty	17	0	250	
	yVelocity	0.1	0	1	
	influenceSize	355	0	1500	
	influenceIntensity	0.93	0	1	
	maxSpeed	6.2	0.5	10	
Ambient Wave	waveActive	True	Flase	True	5
	velocity	0.1	0	2	
	period	0.2	0	1	
	angle	1.4	0	6.283	
	amplitude	0.01	0	1	
Excitor	excitorSize	40	40	2000	9
	excitorCoreSize	0.95	0	1	
	excitorLifespan	20000	500	20000	
	excitorMasterIntensity	1	0	1	
	excitorSpeedLimit	0.6	0	1	
	attractorAngleSpeed	0.2	0	0.25	
	attractorForceScalar	1	0	5	
	maxExcitorAmount	16	1	35	
	bgHowOften	250	250	1000	
ElectricCell	active	True	Flase	True	6
	masterIntensity	1	0	1	
	neighbourRange	3	1	10	
	cellCount	2	1	8	
	rate	75	10	100	
	triggerChange	0.005	0	1	
					31

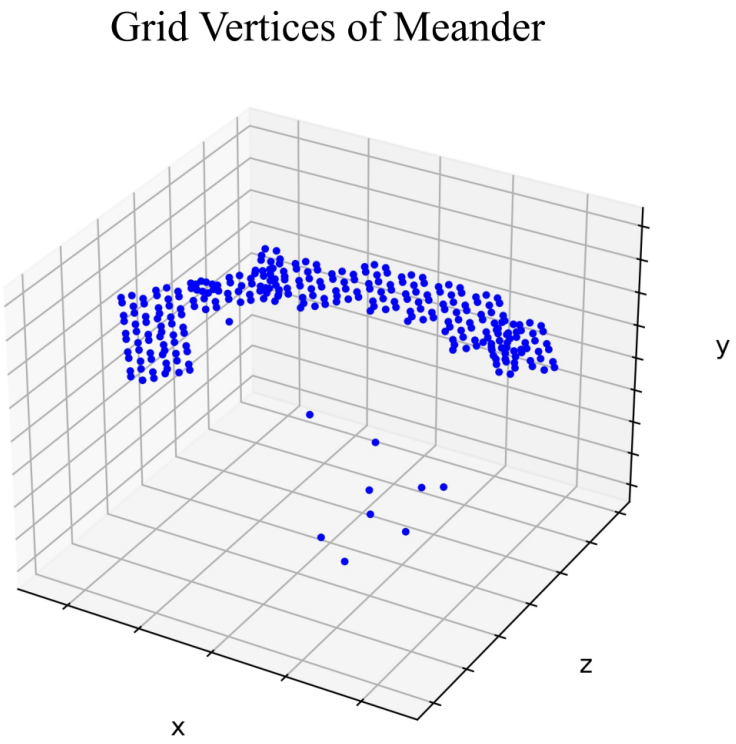


Figure 4.20: Grid Vertices of *Meander*

developed a simulation toolkit to allow researchers to more easily develop and evaluate machine learning techniques for such challenging interactive systems.

Chapter 5

Learning to Engage with Interactive Systems: A Field Study on Deep Reinforcement Learning in a Public Museum

The research in this chapter was conducted in conjunction with a MASc. student of the University of Waterloo, Daiwei Lin[1]*, and was published in the ACM Transactions on Human-Robot Interaction (THRI) in 2020 [187].*

In this chapter, we investigate extrinsically motivated learning in natural settings with group interaction. Specifically, we use sensors embedded in the interactive system to estimate the overall occupancy and engagement of the visitors in a LAS, and use this estimate as the extrinsic reward for learning. We investigate whether we can exploit the designer's pre-scripted behaviors to bootstrap learning. We use the Deep Deterministic Policy Gradient (DDPG) algorithm to train an Actor-Critic agent which acts in the designer-parameterized action space, which we call the Parameterized Learning Agent (PLA). Both fixed pre-scripted and learning behaviors are evaluated over three weeks at a public exhibition at the Royal Ontario Museum (ROM), and all data is collected from museum visitors. We evaluate how engaging the behaviors are based on a quantitative analysis of

[1]This research was conducted at the Royal Ontario Museum by Daiwei Lin and Lingheng Meng under the supervision of Dr. Dana Kuli´c. Daiwei Lin implemented Parameterized Learning Agent (PLA), while Lingheng Meng implemented Single Agent Raw Action Space (SARA) and Agent Community Raw Action Space (SARA) and contributed to the data analysis.

estimated engagement and the number of active interactions, and on qualitative analysis of human survey results. This chapter contributes to solving the engagement estimation challenge with the proximity sensors embedded within the LAS. With this engagement estimate, Reinforcement Learning (RL) can be exploited to enable autonomous engaging behavior generation for the LAS. In addition, this chapter addresses the challenge of ecological validation by conducting a field study in a museum environment and crowd setting with diverse visitors.

5.1 Proposed Approach

In this section, we describe how the *Parameterized Learning Agent (PLA)* is designed to automatically generate interactive actions.

Fig. 5.1 illustrates the software architecture connecting the physical hardware to the learning agents. The Middle Layer coordinates the sensor reading and action execution among the physical LAS and agents. Only one agent type controls the LAS at any given time. PB only takes and stores IR readings without issuing actions because its action policy is fixed, while PLA is formulated as a standard RL framework where the Observation Constructor constructs the observation, the Reward Critic generates a reward that estimates the engagement of visitors, and Actor-Critic is the learned policy and Q-value function.

5.1.1 Parameterized Learning Agent: Learning on Top of Prescripted Behavior

PLA is designed to learn on top of PB, i.e., parameterized action space, where the PB introduced in section 4.1, Chapter 4 is designed by expert architects and interactive system designers and is the baseline used to compare to learning agent. The motivation for this approach is to bootstrap learning by exploiting the designer's knowledge of engaging behavior, where we hypothesize the designer already has a good idea about what types of actions might be engaging to visitors and this can form a helpful starting point for the learner.

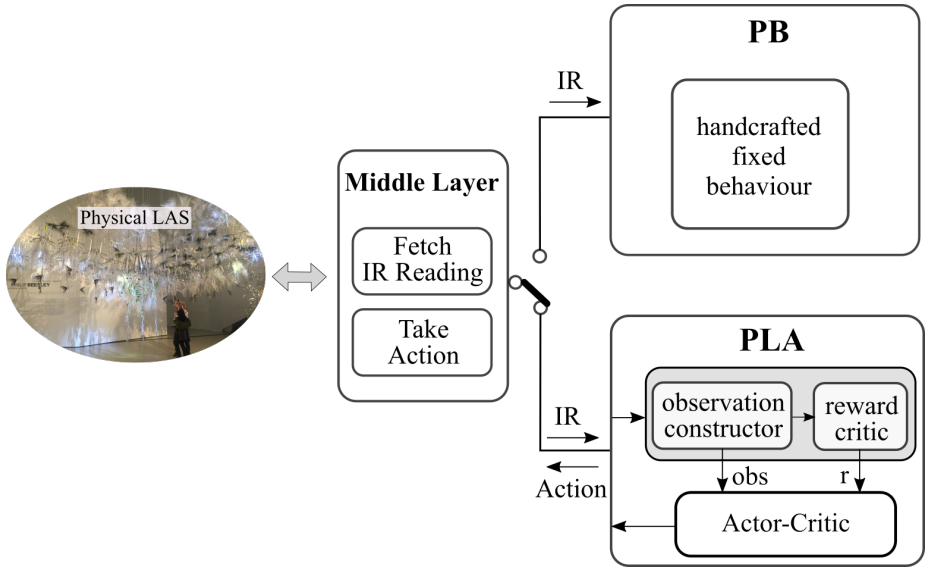
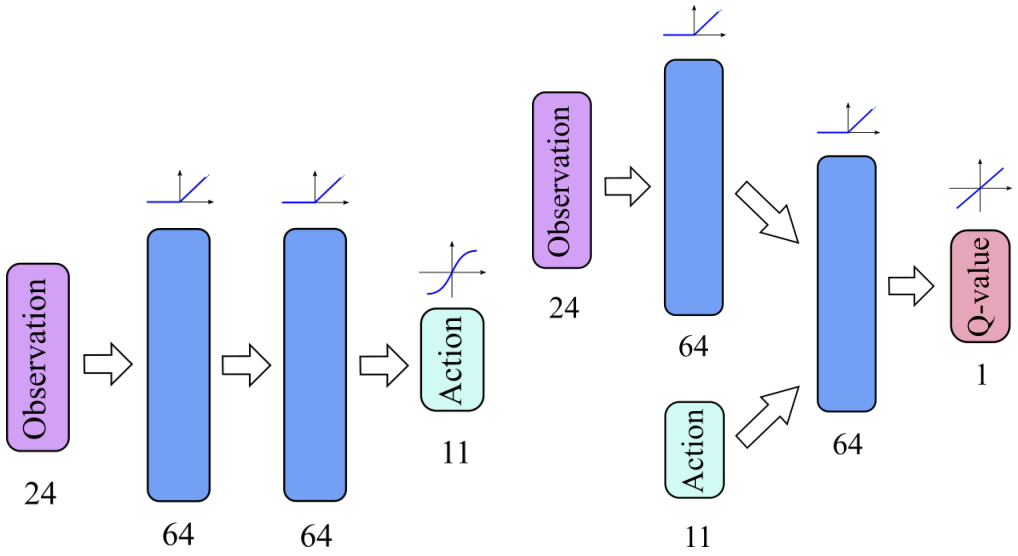


Figure 5.1: Interface Between LAS and PLA



(a) Actor (b) Critic

Figure 5.2: Actor-Critic of PLA, where layer size and activation function are indicated under and above each layer respectively.

Observation and Action Space Construction

For PLA, we select 11 parameters from Table 4.1 as the action space, i.e., the dimension of the action vector is 11. This is because some parameters do not take effect until a

subsequent trigger or until the current propagation finishes. This could lead to obtaining an observation which is based on both previous and updated parameters. To avoid this issue, we exclude T_{bg}^{min}, T_{bg}^{max}, T_w, P, T_{sw}^{min} and T_{sw}^{max} from the action space. In this way, we make sure every observation is only related to the latest action. To attenuate IR sensor noise, the observation for PLA is an average over 20 IR readings as defined in Eq. 5.1:

$$\boldsymbol{obs}^{(t)} = \frac{1}{20} \cdot \sum_{i=0}^{19} \boldsymbol{ir}^{(t-i\cdot\Delta t)}, \tag{5.1}$$

where $\boldsymbol{ir}^{(t-i\cdot\Delta t)}$ is the value array of 24 IR sensor readings at time $(t - i \cdot \Delta t)$, $\sum$ is element-wise summation, and $\Delta t \approx 0.1s$ is the time to retrieve one set of 24 IR values from the physical LAS. Thus the dimension of the observation vector is 24 and each observation vector represents the average IR readings over 2 seconds, which means an agent generates an action every 2 seconds. Note that this delay between an agent's two consecutive actions is in the parameterized action space (i.e., an update to the PB parameters), which means that this will not cause a delay in the LAS's response to human visitors. The 2s time delay between every two consecutive parameterized actions generated by PLA is chosen to allow an action to complete before initiating another action. As described in Section 5.2.3, IR values are converted and scaled so that 0 corresponds to the maximum distance reading (i.e., there is nothing in front of the sensor), and 1 corresponds to the minimum distance reading (i.e., there is something very close to the sensor).

Estimating and Using Engagement as a Reward for Learning

A key feature of our approach is the formulation of the reward function: we wish to learn and reward behaviors which foster visitor engagement. Specifically, the extrinsic reward is computed by summing over the IR observations, which can be regarded as a rough estimate of occupancy and engagement, because: 1) more activated IRs means more people are standing under the LAS, thus indicating higher occupancy; 2) closer distance between visitors and IR sensors implies more active interaction, e.g., looking very closely or raising hands, which are higher engagement behaviors. Therefore, higher occupancy and more active interaction will cause higher extrinsic reward. Formally, given a new observation at time $t+1$, $\boldsymbol{obs}^{(t+1)} = (obs_1^{(t+1)}, obs_2^{(t+1)}, \cdots, obs_n^{(t+1)})$[2], the reward $r^{(t)}$ for taking action $\boldsymbol{a}^{(t)}$

[2]In this chapter, we will use normal lowercase for scalar and bold lowercase for vector.

while observing $\boldsymbol{obs}^{(t)}$ can be expressed as Eq. 5.2:

$$r^{(t)} = \sum_{i=1}^{n} obs_i^{(t+1)}, \tag{5.2}$$

where the value of n is the number of dimensions of the observation vector, e.g. n is 24 for PLA.

Low cost embedded IR sensors are employed to estimate engagement for three reasons: First, the varying lighting conditions in the exhibition area do not allow us to use the webcam footage for reliable pose or facial expression analysis. Because of occlusions and the uncontrolled nature of the space and the varying number of occupants, an array of inexpensive IR sensors that can be precisely located within the space is more effective than lower numbers of more sophisticated equipment. Finally, due to privacy concerns we could not guarantee that we would be permitted to use camera data.

Implementation

Given the observation and extrinsic reward, the optimal policy can be learned by an RL algorithm[3]. In this paper, learning is implemented using the Deep Deterministic Policy Gradient (DDPG) algorithm [164], a variant of Deterministic Policy Gradient [250], where both the Actor and Critic are approximated with deep neural networks.

PLA uses the DDPG agent from OpenAI's Baselines package [77], following the hyper-parameters specified in Table 5.1 (the detailed pseudo-code can be found in Alg. 4 Appendix A). The structure of the neural network is shown in Fig. 5.2. All layers are dense layers, with layer-norm applied, where the activation function and neurons for each layer are indicated above and under each layer in Fig. 5.2, respectively. At the start of the field study, the actor-critic networks of PLA were randomly initialized. On all subsequent deployment days, the learned actor-critic from the previous day was loaded and training continued from the previous day's network parameters. There is a replay buffer $\mathcal{D}$ of size 10^6 that stores all experiences collected thus far on each day. An experience is a tuple of $(obs^{(t)}, a^{(t)}, r^{(t)}, obs^{(t+1)}, done^{(t+1)})$ where $done^{(t+1)}$ is always 0. Every 10 steps, we will sample 20 mini-batches from the replay buffer. The system does not have a terminal state, so we manually set the maximum length of an episode to 100 steps, which corresponds to about 200s, and after each episode the start observation is the observation encountered at the end of the last episode. The reason for not having a terminal state for interaction with

[3]However, note that the interaction frequency is limited by the physical constraints of the LAS.

Table 5.1: Hyper-parameters of DDPG-based PLA

Hyper-parameters	value
actor learning rate	10^{-4}
critic learning rate	10^{-3}
discount rate *gamma*	0.99
batch size N	64
replay buffer size $\|\mathcal{D}\|$	10^6
train_interval	10
train_times	20
episode_length	100
parameter noise α	1.01
parameter noise δ	0.1
initial parameter noise σ	0.1

a single visitor is related to the sensor capabilities, the distributed nature of the sensors, and the fact that multiple visitors move through the space simultaneously. Since we cannot track individual visitors, we do not know when one leaves one sensor space and moves into another, or when leaves the LAS and another arrives.

The maximum episode length is arbitrarily set to 100 steps only for logging statistic summaries, e.g. accumulated reward, estimated Q-values, etc., regularly. It does not affect the training when the maximum episode length is reached, because there is no reset of the environment.

The actor-critic is trained 20 times on a randomly sampled mini-batch every 10 interactions, corresponding to approximately 20s. At every step, the exploratory action is generated by the current actor perturbed with parameter noise, i.e. Parameter Space Noise [219].

All hyper-parameters used in this paper were empirically chosen via experiments on a simplified simulator where the agent directly controls all actuators and maximizes the reward based on IR readings, and visitors are simulated to approach LEDs with the highest intensity. Our tests on the simplified simulator showed that DDPG worked well with the chosen hyper-parameters and was able to smoothly converge to the optimal policy in 100 episodes where the maximum length of each episode is 1000. Because the simulator is an extremely simplified model of the actual environment, we mainly used it to make sure our code is bug free rather than to pre-train a policy.

5.1.2 Learning in Raw Action Space

During the field study, we also tested two additional learning systems designed to act in raw action space, i.e. directly control actuators, rather than acting in parameterized action space. The *Single Agent Raw Act (SARA)* directly controls the 192 raw actuators of the LAS using a single agent, while the *Agent Community Raw Act (ACRA)* controls the raw actuators in a decentralized way, where a distributed multi-agent learning system replaces the single large learner. Unfortunately, we were not able to allocate enough time for examining SARA and ACRA in the field study, so their performance is not analysed in the results section.

5.2 Experiments

5.2.1 Experimental Procedure

Our experiment was conducted for two weeks from Sept. 14 to Oct. 03, 2018, at the ROM. We were permitted by the ROM to collect data from 1 p.m. to 4 p.m. every day on weekdays. In addition, we conducted in-person surveys on Sept. 18, 20, and 27. During the entire experiment period, visitors were free to visit and interact with the installation without any interference from researchers.

For each day of the experiment, the following procedure was followed:

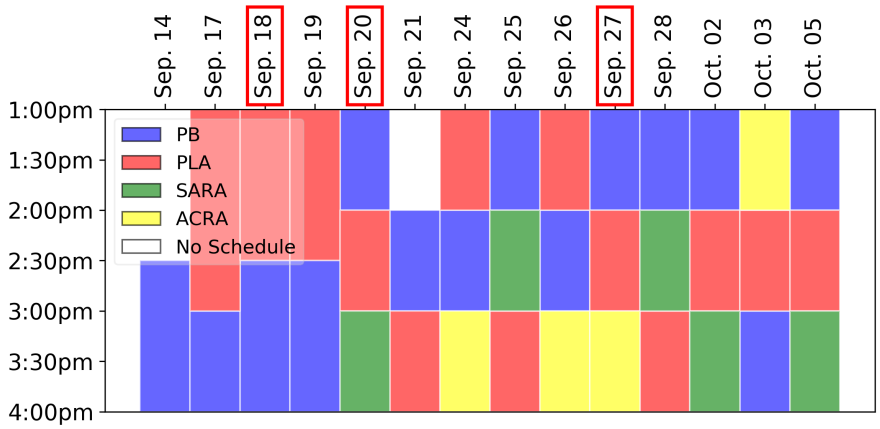
1. Randomly schedule the different agent conditions into 1 or 1.5 hour time slots as shown in Fig. 5.3. PB and PLA were scheduled on each day, while only one of SARA and ACRA were scheduled per day.

2. Automatically run scheduled behavior at each time slot, and save interaction data and learned models and videos at the end of each behavior.

During days where no visitor surveys were collected, researchers were not present in the environment. During the three survey days, researchers were present, but did not provide any additional instructions to visitors. Researchers observed which visitors interacted with the LAS, passively or actively, within a specific behavior period. When visitors were finished with their visit, researchers unobtrusively approached randomly selected visitors who had interacted with the system, and asked them if they were willing to participate in a survey. If a visitor agreed to do the survey, they were guided to a table located around

a corner and were provided with a tablet with a questionnaire (see Section 5.2.2). The researchers also recorded which mode the visitor had interacted with. We only recruited visitors who had interacted with only one behavior mode.

The overall experiment schedule is shown in Fig. 5.3, where red, blue, green, yellow and white areas correspond to PB, PLA, SARA, ACRA and no schedule respectively. A summary of the experiment schedule and collected data is shown in Table 5.2.



Video was not available on Sep. 14.

Figure 5.3: Experiment Schedule

Table 5.2: Summary of Experiment Schedule and Data

Behavior	**Days**	**Hours**	**Survey Participants**
PB	14	15.5	14
PLA	13	15	15
SARA	4	4	4
ACRA	4	4	3

Video was not available on Sep. 14.

5.2.2 Data Collection

The data collected comes in four types: sensor readings, learning agent logs, human survey data and video data from the two web-cams.

Every raw sensor reading is logged. In addition to the raw sensor data, each agent also logs its own learning algorithm data collected during the course of learning.

For human survey data, 14, 15, 4, and 3 participants completed surveys in the PB, PLA, SARA and ACRA modes respectively, as summarized in Table 5.2. The questionnaire used in our experiment is a standardized measurement tool for HRI: the Godspeed questionnaire [24]. In addition to the 24 Godspeed questions, we asked participants about their interests and background, and their general feedback and comments.

The Questionnaire consists of four types of questions:

1. Participants' interests and background (multiple-select multiple choice);
2. Participants prior knowledge about interactive architecture and machine learning, including "How familiar are you with interactive architecture?" and "How familiar are you with machine learning algorithms?";
3. 24 Godspeed questions namely Godspeed I: Anthropomorphism, Godspeed II: Animacy, Godspeed III: Likeability, Godspeed IV: Perceived Intelligence and Godspeed V: Perceived Safety [24];
4. Participants' general feedback, i.e., "Any additional comments regarding your experience?" and "Any overall feedback?".

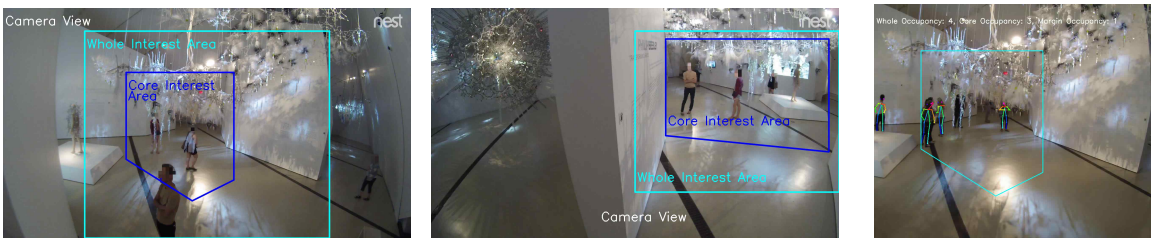
Video data is collected to calibrate sensory readings and validate occupancy estimates, which will be discussed in detail in Section 5.2.4. Video data is available for all the experiments except for Sep. 14.

5.2.3 Data Preprocessing

The IR data is scaled so that all sensory observations for the learning agent are within $[0, 1]$ and all actions that learning agent can take are within $[-1, 1]$. The IR readings are scaled into $[0, 1]$ corresponding to the nearest object being at a detected distance of 80cm or more (no nearby humans detected), to the nearest object being 10cm (very close human detected). The action values for all raw actuators are scaled to $[-1, 1]$. For SMAs, values in the range $[-1, 0)$ means off and values in the range $[0, 1]$ means on, while for LED and Moths the continuous values from -1 to 1 are interpreted into 0 to 255, where 0 indicates off and 255 indicates the brightest light for LEDs, and the highest intensity vibration for the Moths. The 11 parameterized actions are scaled to $[-1, 1]$, where -1 corresponds to minimum and 1 corresponds to maximum, and their corresponding original values are shown in Table 4.1 Chapter 4.

5.2.4 Data Analysis

The camera view includes regions outside of the LAS itself. To only focus on areas directly related to the LAS, we define three parts of the whole camera view (as shown in Fig. 5.4a and Fig. 5.4b for Camera1 and Camera2 respectively). In Fig. 5.4, each camera view is divided into Camera View, Whole Interest Area and Core Interest Area. For both IR Data Calibration (see Section 5.2.4) and Occupancy Estimation (see Section 5.2.4), we only consider the Whole Interest Area. Any visitors outside this interest area will be ignored for the purposes of occupancy estimation. The Core Interest Area approximates the space directly underneath the LAS.



(a) Camera1 (b) Camera2 (c) Sample of Estimated Occupancy from Camera1

Figure 5.4: Interest Area Used to Estimate Occupancy

IR Data Calibration

To enable comparison between different behavior modes, the sensor data must be pre-processed to ensure consistency between conditions. Since visitors can physically interact with the system, it is possible that a visitor changes the direction of the IR sensor thus changing its field of view and subsequent readings. To calibrate the IR data, two calibration steps are taken: 1) IR sensors, whose value is relatively constant and effectively not responding to occupants (e.g., due to obstructions), are removed, 2) the baseline reading for each sensor is shifted to zero. Note that the calibration is only done for analysis, during the learning uncalibrated readings are used. To identify blocked IR and baseline shifts, we visually checked the videos recorded by the two web-cams, and selected a time period when there is no visitor within the whole interest area. Then, we find the IR data corresponding to the no-visitor time. Using the no-visitor time, we determine the thresholds for noise

removal and blocked IR detection for the IR data. We use these thresholds to calibrate the raw IR data.

Occupancy Estimation

We also use the camera data to generate a second estimate of occupancy. We estimate the number of people occupying the space during a one minute interval, using OpenPose[4] [52] based on the videos recorded by one of the web-cams[5]. When estimating occupancy, we only considered the Whole Interest Area.

Non-visitor Period Examination

We also used the camera data to determine whether there are significant periods when no visitors are present. To identify the time periods with no visitors in Whole Interest Area, we manually labelled the time periods when no person is under either camera in the Whole Interest Area. If a person's body is partially visible in Whole Interest Area, we consider it as a person being in the area. The total amount of non-visitor time throughout the experiment is 1 hour, only 2.5% of total experiment time. Therefore, we use the whole time period for analysis without removing any non-visitor intervals.

5.2.5 Quantitative Evaluation

Two metrics, i.e. estimated engagement level and active interaction count, will be used to quantitatively evaluate the ability of PB and PLA to engage visitors. Although these two metrics both depend on IR readings, they emphasize different aspects of engagement. Specifically, estimated engagement level does not differentiate passive and active interaction (illustrated in Fig. 4.1), while active interaction count focuses on measuring active interaction.

We report the average estimated engagement and the average active interaction rather than accumulated reward commonly used in RL, because of the non-episodic, i.e. no termination, and non-stationarity, i.e. transition dynamics $T(s'|s,a)$ changes with different visitors, nature of the test environment. Specifically, in the natural setting of LAS, the

[4] https://github.com/CMU-Perceptual-Computing-Lab/openpose

[5] Videos from Camera2 are highly affected by the changing light of the projector as shown in Fig. 5.4b, so for occupancy estimation we only used videos from Camera1.

number of visitors varies at different time periods and is highly irregular, which makes the evaluation of learned policy on a separate run unfeasible since the same environment will never be encountered twice. Similarly, comparing accumulated reward within a fixed-length episode is also unfair, because if we compare an episode from PLA during which there are no visitors with an episode from PB during which there are several visitors, we can not say PLA is worse than PB and vice versa, because no matter what PLA does there is no reward at all. Therefore, we regard the whole experiment as continuous learning and compare PLA and PB in terms of average estimated engagement and average active interaction.

Estimated Engagement Level

We use raw IR readings recorded during each behavior and Eq. 5.3 to calculate an estimated engagement for comparison among behaviors. Specifically, given M IR readings received within 1 minute (typically sampled at 10Hz) $\left\{\boldsymbol{ir}^{(1)}, \boldsymbol{ir}^{(2)}, ..., \boldsymbol{ir}^{(M)}\right\}$ where each IR reading $\boldsymbol{ir}^{(i)}$ is a vector of 24 IR values, the estimated engagement level e is defined by Eq. 5.3:

$$e = \frac{1}{M}\frac{1}{24}\sum_{m=1}^{M}\sum_{i=1}^{24} ir_i^{(m)}, \tag{5.3}$$

where $ir_i^{(m)}$ is the ith IR sensor in the mth IR reading. The estimated engagement is in the range [0,1], where the maximum 1 corresponds to a maximally engaging state, where all IR sensors are receiving maximum readings during the entire 1 minute window, while the minimum 0 corresponds to fully non-engaging state, where all IR sensors are receiving minimum readings for the duration of the one-minute window.

Active Interaction Count Analysis

In addition to the estimate of engagement, we separately estimate the level of active interaction. To capture active interactions, we count the number of IR readings having value $>= 0.25$, which corresponds to a proximity of 35cm or less from an IR sensor, within 1 minute. Formally, given M IR readings received within 1 minute (typically sampled at frequency $F = 10Hz$) $\left\{\boldsymbol{ir}^{(1)}, \boldsymbol{ir}^{(2)}, ..., \boldsymbol{ir}^{(M)}\right\}$ where each IR reading $\boldsymbol{ir}^{(i)}$ is a vector of 24 IR values, the number of active interactions N_{active} is defined by Eq. 5.4:

$$N_{active} = \frac{1}{F}\sum_{m=1}^{M}\sum_{i=1}^{24} \mathbb{1}\left\{ir_i^{(m)} \geq 0.25\right\}, \tag{5.4}$$

where $ir_i^{(m)}$ is the ith IR sensor in the mth IR reading, and $\mathbb{1}\{\cdot\}$ is a indicator function. Therefore, N_{active} is the total detected active interactions within 1 minute.

5.3 Results

In this section, we first quantitatively compare the performance of PB and PLA based on evaluation metrics introduced in Section 5.2.5. After that, we analyze the human survey data.

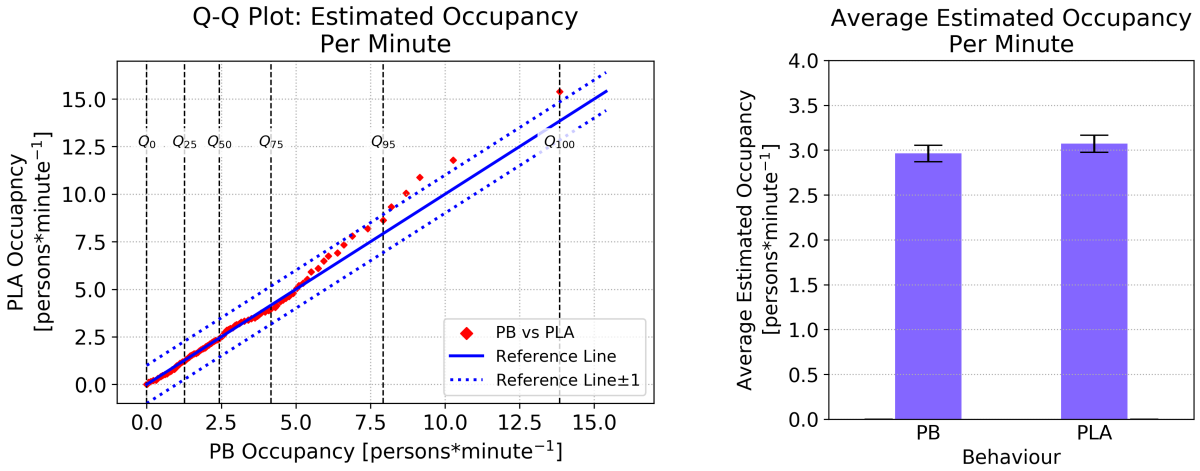
5.3.1 Quantitative Comparison Between PB and PLA

In this section, we quantitatively compare the performance of the two behavior modes based on sensory data collected during the interaction between visitors and the LAS. We use two ways to quantitatively compare the two behaviors' performance: 1) comparing the estimated engagement level, as described in Section 5.2.5, and 2) comparing the number of active interactions, as introduced in Section 5.2.5.

Our experiment is run in a natural setting, i.e., a publicly accessible museum, so it is possible that there are different occupancy levels in the space due to factors not related to the behavior mode. To check whether there are different occupancy levels between conditions (which might be caused either by some behaviors being more attractive to visitors, or factors not related to system behaviors), we analyze the overall occupancy level for PB and PLA, as described in Section 5.2.4. Fig. 5.5 shows a comparison of the estimated occupancy between PB and PLA, where (a) shows that, in only about 5% of data, PLA has approximately 1 more visitor than PB, and (b) shows that the average occupancy between PB and PLA is very similar. A Mann-Whitney U test indicates that there is no significant difference between PB and PLA in terms of occupancy level, $U = 239030.5$, $p = 0.92$ (two-sided).

Estimated Engagement Level Comparison

Fig. 5.6 compares the distributions of estimated engagement (see Section 5.2.5) between PB and PLA. From Fig. 5.6, we can observe that for the first 75% of data there is no noticeable difference between PB and PLA, while for the last 25% of data PLA has larger estimated engagement than PB. Fig. 5.8 shows the average estimated engagement, which shows PLA achieves higher average engagement than PB.



(a) Comparison of Estimated Occupancy Distributions.

(b) Average Estimated Occupancy

Figure 5.5: Estimated Occupancy Comparison. (a) is a Q-Q (100-quantiles-100-quantiles) plot of estimated per-minute occupancy, using the method introduced in Section 5.2.4, for PB and PLA, where the coordinate (x, y) of the q-th point from bottom-left to up-right corresponds to the estimated occupancy of (PB, PLA) for the q-th percentile, i.e. $Q_q, q = 0, 1, ..., 100$, and the reference line indicates a perfect match of distribution between PB and PLA. For example, the point $(4.3, 4)$ for PB vs PLA at the Q_{75} means that 75% of observations for PB and PLA are less than 4.3 and 4, respectively. (b) shows the average estimated per-minute occupancy and its standard error for PB and PLA.

Active Interaction Comparison

Fig. 5.7 compares the active interaction count based on Eq. 5.4 between PB and PLA. From this figure, we can see that for about 50% of observations, PLA achieves higher active interaction than PB. Fig. 5.9 compares PB with PLA in terms of average raw active interaction count. As shown in the figure, PLA almost doubles the PB average active interaction count.

Evolution of Average Estimated Engagement and Active Interaction

To analyse how performance evolved over the 3 week experiment, we plot the daily average engagement and active interaction over the whole experiment. Fig. 5.10 shows daily

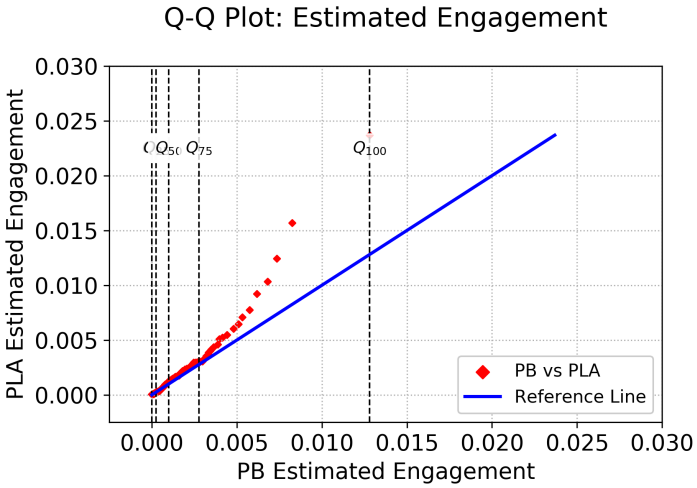


Figure 5.6: Estimated Engagement Distribution Comparison. The Q-Q (100-quantiles-100-quantiles) plot between PB and PLA is based on average estimated engagement, where the coordinate (x, y) of the q-th point from bottom-left to up-right corresponds to the estimated engagement level of (PB, PLA) for the q-th percentile, i.e. $Q_q, q = 0, 1, ..., 100$, and the reference line indicates a perfect match of distributions between PB and PLA.

average metrics of PB and PLA. In terms of daily estimated engagement, Fig. 5.10a shows that during the first two days, PB outperforms PLA, while after Sep. 25 PLA overtakes PB for the remainder of the experiment. A similar trend can be seen in terms of daily active interaction as shown in Fig. 5.10b. PLA receives more active interaction than PB from the very beginning and keeps expanding the gap between PLA and PB. Even though it seems PLA is improving, due to the uncontrolled experimental setting, we cannot be certain whether this is caused by continuous adapting of PLA, or due to factors independent from the interactive action of the LAS.

5.3.2 Analysis of Actions Automatically Generated by PLA

In this section, we analyze actions automatically generated by PLA. Since the dimensionality of the action space is 11 and each dimension is continuous, visualisation and analysis of the action trajectory is difficult. For ease of visualisation, we first cluster actions into 6 clusters using K-Means [12], then use t-SNE [172] to embed actions generated by PLA into a 2 dimensional space, where actions which are close in high dimensional space are

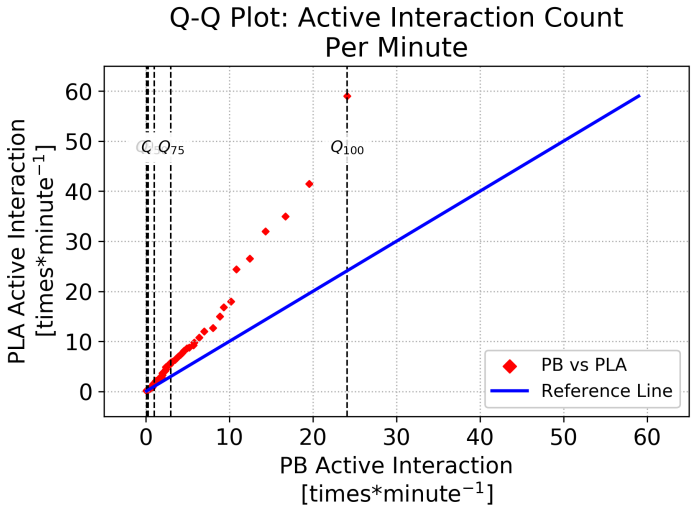


Figure 5.7: Active Interaction Count Comparison. The Q-Q Plot is based on active interaction count per minute obtained using Eq. 5.4, where every (x, y) corresponds to the active interaction count of PB in one percentile, and the line indicates a perfect match of distributions between PB and PLA.

modeled by nearby points and dissimilar actions are modeled by distant points with high probability. The resulting visualisation allows us to compare actions over all days or between specific days. Note that the clustering is done in the 11-dimensional action space of PLA rather than the 2-dimensional embedding space, and the centroid of each cluster is an 11-dimensional data point which can be thought of as the multi-dimensional average of the points in the cluster. Fig. 5.11a and 5.11b show embedded actions generated by PLA viewed in different colors for different days and different clusters, respectively, where PB is also embedded for comparison and is indicated by the red star. Fig. 5.11c shows the actions taken by PLA separately by day, with the different clusters labeled.

Fig. 5.12 depicts, for each PLA action cluster, the difference between each of the 11 dimensions of the cluster centroid and the corresponding default value of PB. From Fig. 5.12, we can observe that: (1) most action dimensions of the centroids of clusters 2, 3, 4 and 5 are greater than the default values of PB; (2) cluster 1 is very similar to PB on many action dimensions; and (3) the centroid of cluster 6 has more action dimensions with values lower than the default value of PB. Taking cluster 4 of PLA as a concrete example, most dimensions of the centroid are larger than the default value of PB, indicating actions in cluster 4 are slower and smoother than PB since Moths and LEDs take more time to ramp

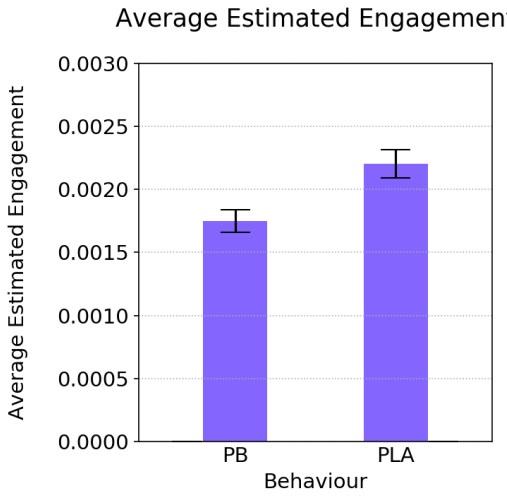


Figure 5.8: Average Estimated Engagement Comparison, where blue bars with standard errors show the average estimated engagement and its corresponding standard error.

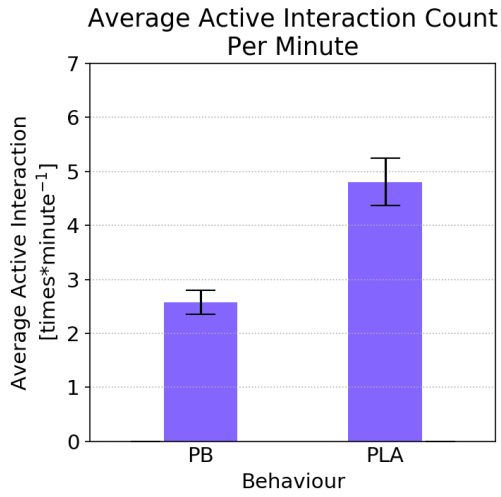
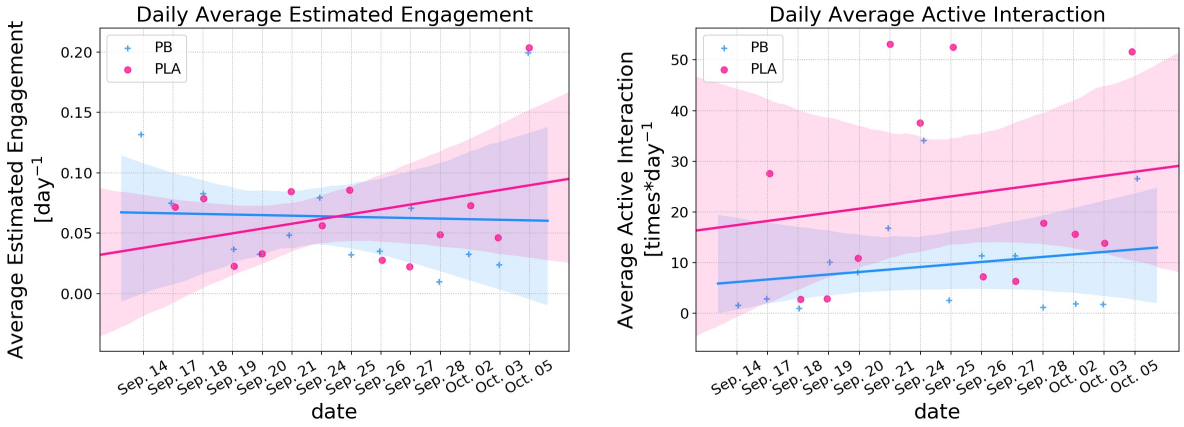


Figure 5.9: Average Active Interaction Count Comparison, where blue bars with standard errors show the average number of active interactions per minute for each behavior (the counting is based on 1Hz interaction frequency).

up (T^m_{ru}, T^l_{ru}), hold on (T^m_{ho}, T^l_{ho}) and ramp down (T^m_{rd}, T^l_{rd}), and the time gaps between activating Moths and LEDs and neighbour nodes are longer (T^m_{gap}, T^n_{gap}, T_{sma}). Cluster 4 seems to be preferred on Sep. 26th, Oct. 3rd and Oct. 5th as shown in Fig. 5.11c, as many actions in cluster 1 are taken. Cluster 1 shares more similarity with PB, as seen from the small differences in most dimensions shown in the 4th panel in Fig. 5.12. Actions in cluster 1 are taken densely on Sep. 17th, 19th, 20th, 26th and Oct. 5th, as shown in Fig. 5.12. As an interesting and slightly contrary example to cluster 4, cluster 6 of PLA shows abrupt action where Moths and LEDs take less time than PB to ramp up (T^m_{ru}, T^l_{ru}), which seems to be preferred on Sep. 24th. Note that here we arbitrarily set the total cluster number to 6 for a relatively clear visualization. As the clusters still have considerable within-cluster variance, the analysis of centroids only provides an approximate analysis of the diversity of actions taken by PLA.

Combining Fig. 5.11a, 5.11c and 5.12, we do not see specific types of action which are dominant. Nevertheless, it seems that PLA continuously adapts, because actions generated by PLA on each day show slightly different coverage as shown in Fig. 5.11c and



(a) Estimated Engagement

(b) Active Interaction

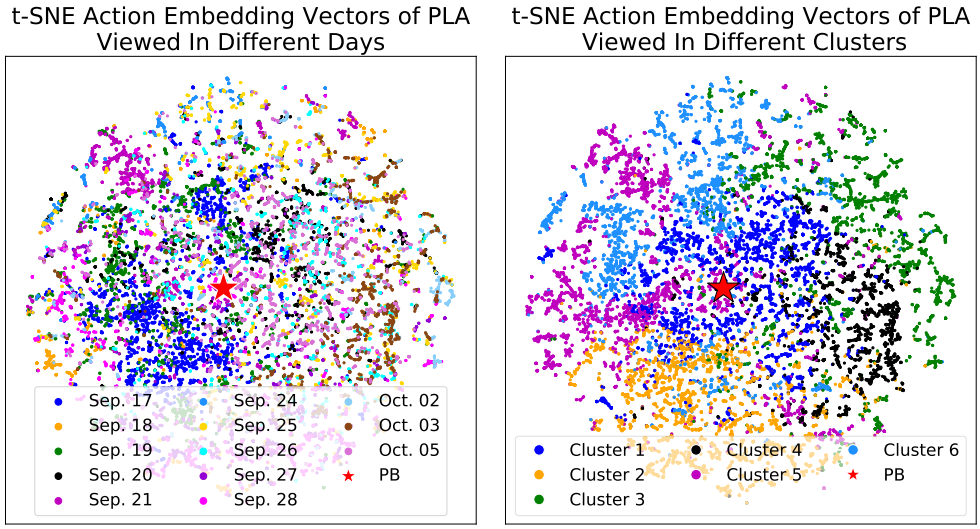
Figure 5.10: Trajectory of Daily Average Metrics. (a) Daily Average Estimated Engagement and (b) Daily Average Active Interaction, where each data point is the corresponding average on each day, the lines are linear regressions of these data and the translucent bands around the regression line are the 95% confidence intervals for the regression estimate.

demonstrated in the previous paragraph. Overall, we can observe that PLA covered a wide range of actions on each day and no obvious dominant actions were reached by the end of the experiment.

5.3.3 Human Survey Results

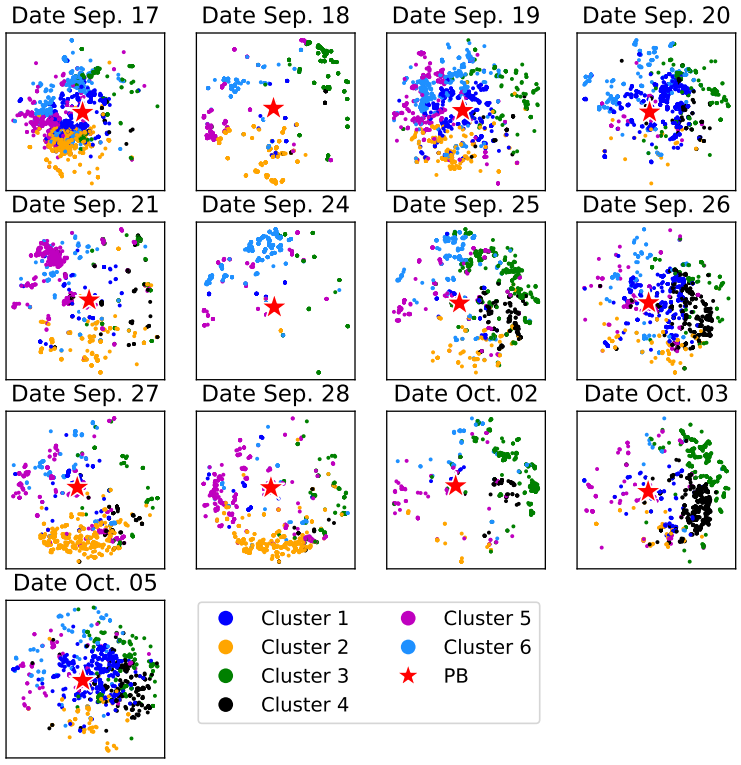
In this section, we analyze the visitor responses to the survey for PB and PLA. We first examine if there are any differences in the population characteristics between the participants who engaged with the system in PB or PLA behavior modes. Then, we compare the PB and PLA responses for each Godspeed category. Finally, we compare PB and PLA for each question in Godspeed Likeability category individually.

We first analyze whether there are population differences between conditions. In Section 5.2.4, we confirmed that there were no significant differences between PB and PLA in terms of estimated occupancy. To test for differences in participant background and interest, we performed a χ^2-test on participants' background and interests based on the first two questions in our questionnaire (see Section 5.2.2), and found no statistically significant differences between the two groups.



(a) Viewed in Days

(b) Viewed in Clusters



(c) On Each Day in 6 Clusters

Figure 5.11: Visualizing Actions Taken by PLA Embedded in a Two Dimensional Space.

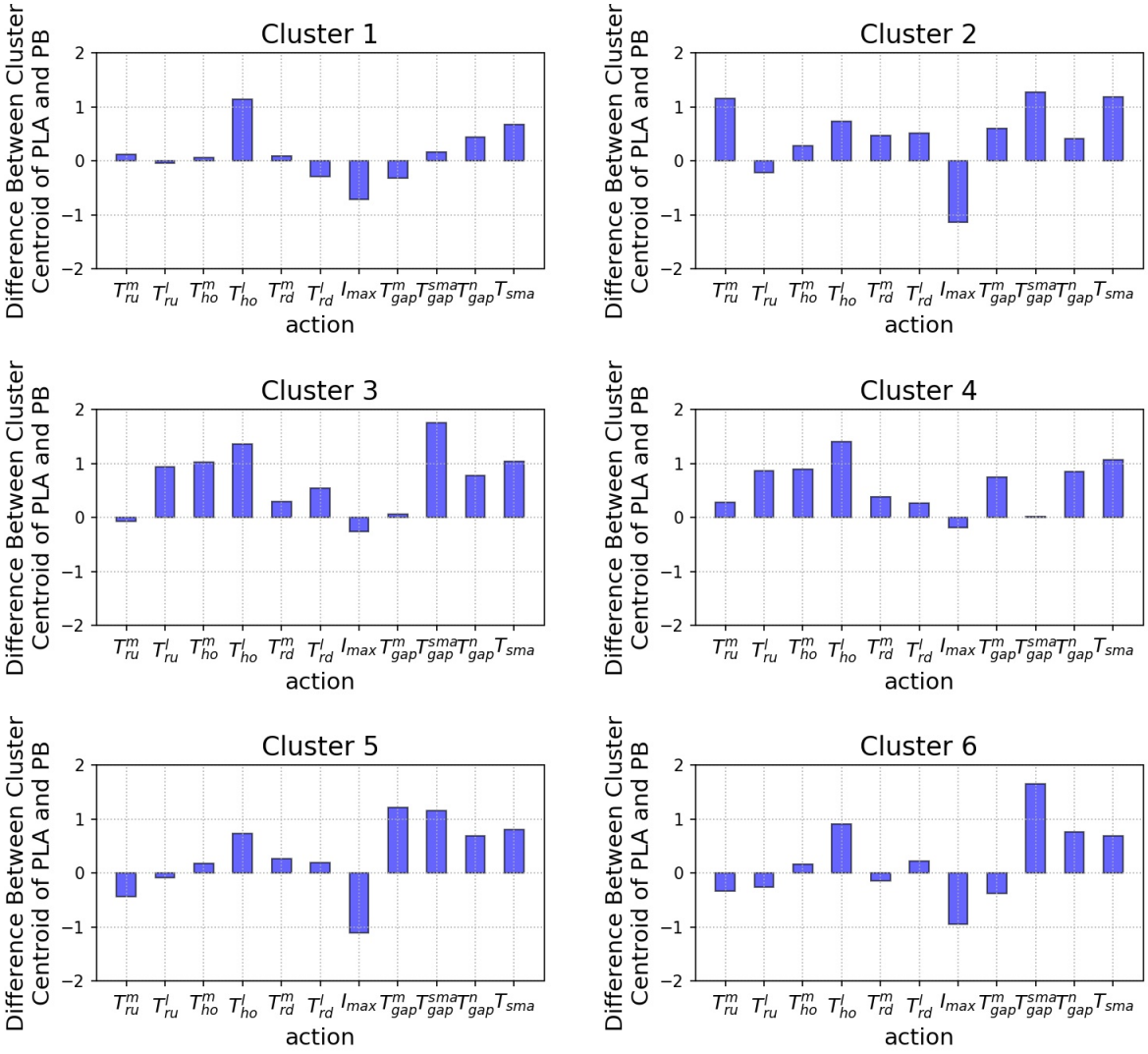


Figure 5.12: Difference between the Cluster Centroid of PLA and PB. Each panel shows the difference $a_{PLA} - a_{PB}$ between the cluster centroids of PLA a_{PLA} and the default value of PB a_{PB} for each dimension of the action space elaborated in Table 4.1, where a positive value means that the corresponding dimension of the cluster centroid of PLA is greater than the default value of PB, while a negative value means the contrary.

Cronbach's α-test was conducted on each category of Godspeed for both PB and PLA to examine the reliability of participants' responses, results are shown in Table 5.3. Although

α on Anthropomorphism and Perceived Safety is low, α on others is in the acceptable range, especially for Likeability $\alpha \geq 0.85$.

Table 5.3: Cronbach's α on Godspeed for PB and PLA

	Anthropomorphism	**Animacy**	**Likeability**	**Perceived Intelligence**	**Perceived Safety**
PB	0.74	0.77	0.85	0.89	0.52
PLA	0.64	0.80	0.93	0.85	0.27

A commonly accepted rule [76]: $0.9 \leq \alpha$: Excellent; $0.8 \leq \alpha < 0.9$: Good; $0.7 \leq \alpha < 0.8$: Acceptable; $0.6 \leq \alpha < 0.7$: Questionable; $0.5 \leq \alpha < 0.6$: Poor; $\alpha < 0.5$: Unacceptable.

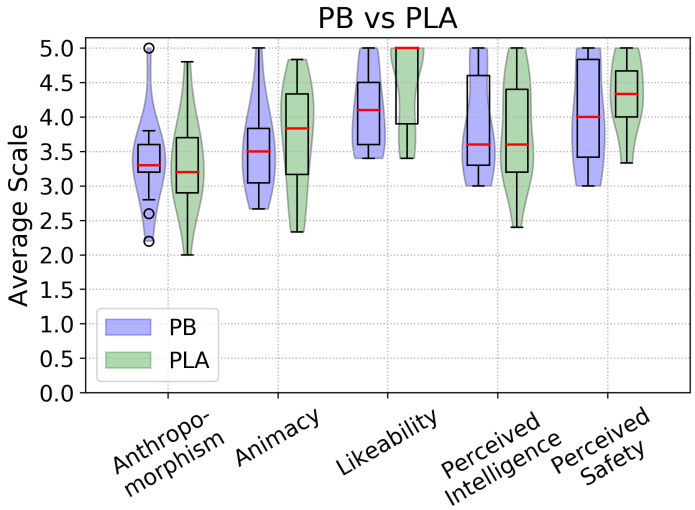


Figure 5.13: Boxplot and Violinplot of Average Scale of each Godspeed Category over Participants within PB or PLA.

Fig. 5.13 shows the Box-plot and Violin-plot of the calculated average scale over each Godspeed category for PB and PLA. Within the five Godspeed categories, only *Likeability* has a relatively large gap between the medians of PB and PLA. In addition, Likeability has a relatively small variance, whereas other categories have large variance. A normality test was conducted for the Likeability category for participants from PB and PLA, respectively. Shapiro-Wilk Test [243] indicates PB ($p = 0.12$) is normally distributed, while PLA ($p = 0.0008$) is not. Therefore, non-parametric test Mann−Whitney U test [176] was conducted and finds that the distributions of PB (*Median=4.10*, *M=4.09*, *SD=0.56*)

and PLA (*Median=5.00, M=4.48, SD=0.64*) differ significantly (*Mann−Whitney U = 67, p=0.044 ¡ 0.05*) in terms of Likeability, whereas for other categories there is no significant difference between PB and PLA.

In summary, PLA is rated higher than PB by the participants in terms of Likeability, while there are no significant differences between PB and PLA in the other Godspeed categories.

5.4 Discussion

In this paper, we investigated how an interactive system can learn to engage with visitors in a natural setting, where no constraints are imposed on visitors and group interaction is accommodated. Relying on the standard RL framework and a novel measure of engagement as reward, an adaptive behavior *PLA* was compared to a pre-scripted behavior *PB*. Our results show that PLA outperforms PB in terms of estimated engagement level, active interaction count, and the Likeability in human survey data. We hypothesize that the PLA configuration outperforms PB because it benefits from both human expert input, such as parameterized action space and manual reward function, and learning. During the design process, architects design PB based on their expertise. We exploit this human expertise to effectively restrict the parameterized action space of PLA into a region where we know good solutions can be found, and at the same time limit the dimension of the PLA action space. Therefore, PB and PLA both incorporate human intelligence, but compared with PB, PLA is endowed with adaptability by applying RL to its parameterized action space.

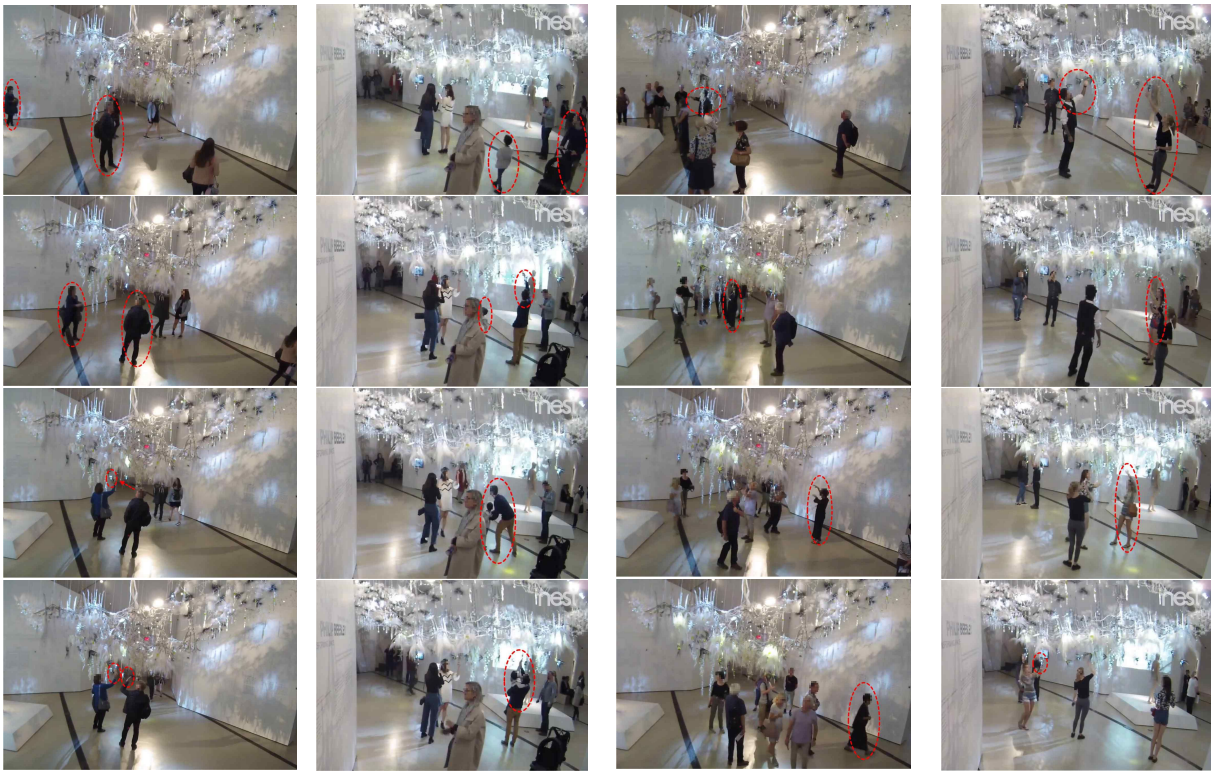
Our approach uses low-cost IR sensors for engagement estimation. Compared to other social HRI work with rich sensing such as cameras and microphones, we were still able to estimate engagement with limited sensing and generate engaging behaviors accordingly. This might be helpful to other large scale interactive systems where having sophisticated measurement may be unfeasible. In summary, this work provides two useful generalizable guidelines for designing engaging behavior for long-term interaction: (1) in a group interaction setting, group engagement from low-cost ambient sensors can be used either standalone, or as a complement of individual engagement measures, and (2) such a measure of engagement can be used as a reward signal to generate customised and evolving behavior.

Creating engaging behaviors for LAS requires learning algorithms that continue to adapt rather than optimizing to a single best policy. First, the Markov Decision Process

and Stationary Environment Dynamics assumptions are broken because of the complicated interaction environment. In addition, interaction time cannot be assumed to have a constant length, and the interaction speed is bound to physical interaction and cannot be sped up. In addition, since LAS is an architectural scale interactive system, perception of the environment is more complex and may need to include both proprioception and exteroception. Therefore, although we exploit a RL framework in our work, the role it plays is different from that of standard testbeds such as OpenAI Gym [43] or Arcade Learning Environment (ALE) [32]. In this work, RL is used to introduce adaptability, but there is no guarantee that the learning leads to optimal policy. This is illustrated by the observation that compared with PB, PLA shows very flexible action patterns, and some are very different from PB. Specifically, one observed behavior generated by PLA is LEDs turned on and propagated quickly from one node to another back and forth multiple times accompanying activated SMAs and Moths (see https://youtu.be/2tICanYEpoo) for the video comparing PB and PLA, which makes the LAS look like a thunderstorm. This novel behavior illustrates that the sculpture has taken the primitives composed by the designers and evolved engaging and interesting behavior from those.

The group setting presents a challenging environment for learning. During the entire experiment, we found some scenarios that highlight the complexity of using RL in LAS, such as interactions between visitors and the possibility that the LAS could be physically changed by touching as shown in Fig. 5.14. In addition, there are many other examples of complicated environment dynamics that present challenges to a learning algorithm. Examples include visitors who use alternate interaction strategies, change their interaction strategy over time or because they are influenced by other visitors, as well as visitors who raise their hands for reasons other than interaction. These observations illustrate the non-stationarity of the environment, and the influence of human-human interaction in group scenarios during HRI. They also emphasize the importance of developing and testing these algorithms "in the wild."

Selecting RL algorithms and hyper-parameters with real experiments is challenging, since we do not have a "validation set" that allows us to do multiple runs and we cannot accelerate learning. Due to the non-stationary nature of the investigated environment in our case, this is also impossible to do with a simulator, because unlike some fields in robotics where a realistic simulator can be devised, in our case the diversity of visitors' interaction styles, social influence of human-human interaction and variations in visitor numbers during different time periods, etc. make it impossible to devise such a good simulator. Therefore, in this work we only used a simplified simulator to make an initial hyperparameter selection and confirm our code is bug free rather than to pre-train a policy or fine-tune hyper-parameters. Finally, although DDPG was selected as the learning



(a) Learn how to interact from other visitor

(b) Parent lifts child

(c) Group visit lead by a guide

(d) Visitors taught by security guard but misunderstood touch IR sensors

Figure 5.14: Sample Interesting Scenario. (a) shows how visitors can affect each other. Before the woman in blue arrives, the man in black had spent a while just looking around and did not know how to interact with the LAS. Then coincidentally he saw how the woman raised her hand and how the LAS responded. After that he copied her action to interact with the LAS. (b) shows a parent helping his child experience the interaction after he explored how to interact. Since the child could not reach the IR sensor, the parent lifted his child. (c) shows a group visit where a group of visitors is led by a guide. (d) shows a more complex scenario about misunderstanding shared information. In this case, three girls were taking photos without interacting. Then a security guard taught one of the three girls how to interact. At the same time, the second girl saw and joined them. After that the third girl learned this from her friends. However, there was a misunderstanding of the security guard's instruction, so they directly touched the IR sensors rather than just waving hand closely.

algorithm as it is easy to implement and works on continuous action space, the best choice of RL algorithm requires further investigation. DDPG may not be sample efficient and can be susceptible to overestimation and sensitive to hyper-parameters [117]. More advanced continuous control algorithms, such as Soft Actor-Critic (SAC) [106], Twin Delayed DDPG (TD3) [89], and Multi-step DDPG (MDDPG) [180], should be investigated in the future.

5.5 Summary

In this chapter, we developed and evaluated algorithms for generating interactive behaviors in group environments. Specifically, we provide a way to estimate engagement during group interaction based on multiple IR sensors, where both individual engagement, passive and active interaction, and group engagement, i.e. occupancy, are taken into account. PB and PLA were examined to evaluate how the use of human knowledge influences interaction. By analyzing interaction and human survey data, we found that learned interactive behaviors, i.e. PLA, result in higher engagement and perceived likeability than pre-scripted behavior, i.e., PB.

In this chapter, we used DDPG as the learning algorithm. However, similar to other DRL algorithms, DDPG faces the low data efficiency problem. To improve the data efficiency of DDPG, we propose two variants of DDPG by combining it with multi-step methods in the next chapter. Specifically, we study the effect of multi-step methods on the overestimation problem in DRL.

Chapter 6

The Effect of Multi-step Methods on Overestimation in Deep Reinforcement Learning

A version of this chapter was published in the proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR) [184].

Applying Deep Reinforcement Learning (DRL) [193] to a novel robot, such as a Living Architecture System (LAS), is not always straightforward, because many DRL algorithms are mainly investigated on simulation tasks with the focus on theoretical, rather than engineering, aspects and do not prioritise data efficiency. This chapter aims to improve the data efficiency of the DRL algorithm used in Chapter 5 to reduce the barrier of applying DRL to Human Robot Interaction. In addition, to ease the comparison this chapter utilizes the standard benchmarks intensively investigated in DRL community rather than our special interactive systems, i.e., LASs, but the knowledge developed in this chapter is still applicable to LASs.

DRL incorporates the powerful representation capacity of nonlinear Deep Neural Networks (DNNs) into classic Reinforcement Learning (RL), but also results in low data efficiency and instability, i.e., drastic fluctuations in accumulated reward increase rather than a smooth increase, [117, 264, 130]. Deep Q-Networks (DQNs) [193] represent the Q-value function with DNNs, but is only designed for discrete action spaces Deep Deterministic Policy Gradient (DDPG) [164] was proposed to tackle continuous control tasks, where the actor and critic are both represented with DNNs. The move from low dimensional state and action spaces to high-dimensional state and action spaces by employing DNNs comes

with the cost of low data efficiency and non-monotonic learning progress, where policy diverges from optimal due to inaccurate approximation of the value function. These disadvantages hinder DRL from broad use in applications where interactive data collection is time-consuming and smooth adaption of behavior is crucial to maintain engagement, e.g. interactive robots [187].

Low data efficiency corresponds to slow learning speed, assuming the learning algorithm is capable of learning an optimal policy given sufficient data, and can be caused by two reasons: **1)** lack of data, and **2)** lack of training. If slow learning is caused by lack of data, the environment is under-explored. In this case, an efficient exploration strategy, e.g. parameter space noise [219], or a complementary source of experiences, e.g. World Models [105], can be helpful for generating additional training data. On the other hand, if slow learning is caused by lack of training, since enough experiences have been collected but not coded into policy, a more effective way to use the collected data is necessary such as prioritized replay buffer [238, 120] and hindsight experience replay [9].

Instability is partially related to the catastrophic forgetting problem of Deep Learning (DL) [147] which is inherent in the continuously evolving nature of policy learning in Reinforcement Learning (RL). In addition, inaccurate estimation and continuous tuning of the Q-value function might lead the learned policy in directions far away from optimal or cause it to fluctuate around a local optimum. The overestimation problem [274] in Q-learning is a typical example of inaccurate estimation in which the maximization of an inaccurate Q-value estimate induces a consistent overestimation.

Building on top of DDPG [164], we experiment with Multi-step DDPG (MDDPG), where different step sizes are manually set, and with a variant called Mixed Multi-step DDPG (MMDDPG) where a mixture of different multi-step backups is used as target Q-value. We first experimentally show that MDDPG and MMDDPG outperform DDPG, in terms of final performance and learning speed, mainly because of their effect helps alleviate the overestimation problem. Then, we compare MMDDPG with other state-of-the-art approaches to show that the proposed method can achieve comparable performance to TD3 which is dedicated to addressing overestimation problem. After that, we discuss the underestimation and overestimation underlying offline multi-step method. At the end, we conclude this work and provide prospects for future research.

6.1 Proposed Methods

In this section, we will progressively introduce MDDPG and MMDDPG.

6.1.1 Multi-step DDPG (MDDPG)

MDDPG is a variant of DDPG where multi-step experiences sampled from the replay buffer are used to calculate the direct accumulated reward, which is then added to the bootstrapped Q-value after these experiences.

Based on the n-step discounted accumulated reward in Eq. 3.4, we can easily realize n-step bootstrapped return using consecutively stored experiences in the replay buffer. Assuming that past experiences of an agent are consecutively stored in the replay buffer D and the experience at time step t is sampled into a training mini-batch, the n consecutive experiences from t to $t+n-1$ are treated as a single multi-time step sample. Then, for each sample in the n-step mini-batch $\left\{(s_t, a_t, r_t, \cdots, s_{t+n}, d_{t+n})^{(i)}\right\}_{i=1}^{N}$ with size N, the n-step bootstrapped estimated action value function can be defined as Eq. 6.1:

$$\hat{Q}_t^{(n)} = \begin{cases} \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_a Q_{\theta^{Q^-}}(s_{t+n}, a), \\ \qquad if\ \forall\ k \in [1, \cdots, n]\ \ and\ d_{t+k} \neq 1; \\ \sum_{i=0}^{k-1} \gamma^i r_{t+i}, \\ \qquad if\ \exists\ k \in [1, \cdots, n]\ \ and\ d_{t+k} = 1, \end{cases} \tag{6.1}$$

with $d_{t+k} = 1$ if the episode is done, otherwise $d_{t+k} = 0$.

The value function Q of MDDPG is updated by minimizing the objective given in Eq. 6.2.

$$L_{\theta^Q} = \mathbb{E}_{(s_t, a_t, r_t, \cdots, s_{t+n}, d_{t+n}) \sim U(D)} \left[\left(\hat{Q}_t^{(n)} - Q_{\theta^Q}(s_t, a_t) \right)^2 \right], \tag{6.2}$$

where $\hat{Q}_t^{(n)}$ is Eq. 6.1, while the policy update remains the same as DDPG. For convenience, in this chapter we will denote MDDPG with step size n as MDDPG(n). Specifically, when $n = 1$, MDDPG(1) is equivalent to DDPG.

6.1.2 Mixed Multi-step DDPG (MMDDPG)

MMDDPG is a variant of MDDPG based on the observation that for different tasks the best choice of step size n may differ. MMDDPG mixes target Q-values calculated with different step sizes. This also helps to reduce the bias of the target Q-value by mixing a small set of target Q-values. The mixture can be an average over target Q-values with different step sizes from 1 to n as $\hat{Q}_t^{(n_{avg})}$ in Eq. 6.3, or the minimum of such a set of target Q-values as $\hat{Q}_t^{(n_{min})}$ in Eq. 6.3. Or considering $n = 1$ is the most prone to overestimation,

MMDDPG could take the average over target Q-values with step size from 2 to n, as $\hat{Q}_t^{(n_{avg-1})}$ in Eq. 6.3.

$$\hat{Q}_t^{(n_{avg})} = \frac{1}{n}\sum_{i=1}^{n}\hat{Q}_t^{(i)} \quad \text{or} \quad \hat{Q}_t^{(n_{min})} = \min_{i\sim[1,n]}\hat{Q}_t^{(i)}$$
$$\text{or} \quad \hat{Q}_t^{(n_{avg-1})} = \frac{1}{n-1}\sum_{i=2}^{n}\hat{Q}_t^{(i)} \tag{6.3}$$

Similar to MDDPG(n), we will denote MMDDPG with different mixture methods introduced in Eq. 6.3 as MMDDPG(n-avg), MMDDPG(n-min), and MMDDPG(n-avg-1), respectively.

6.2 Experiments

Tasks used for evaluation are selected from PyBulletGym[1]. We compare MDDPG and MMDDPG with vanilla DDPG, TD3, SAC [106], MVE and STEVE.[2] All algorithms use policy and value functions with 2 hidden layers and each layer has 300 hidden units. Other hyper-parameters are set to the default values. All experiments are run five times for five different random seeds.

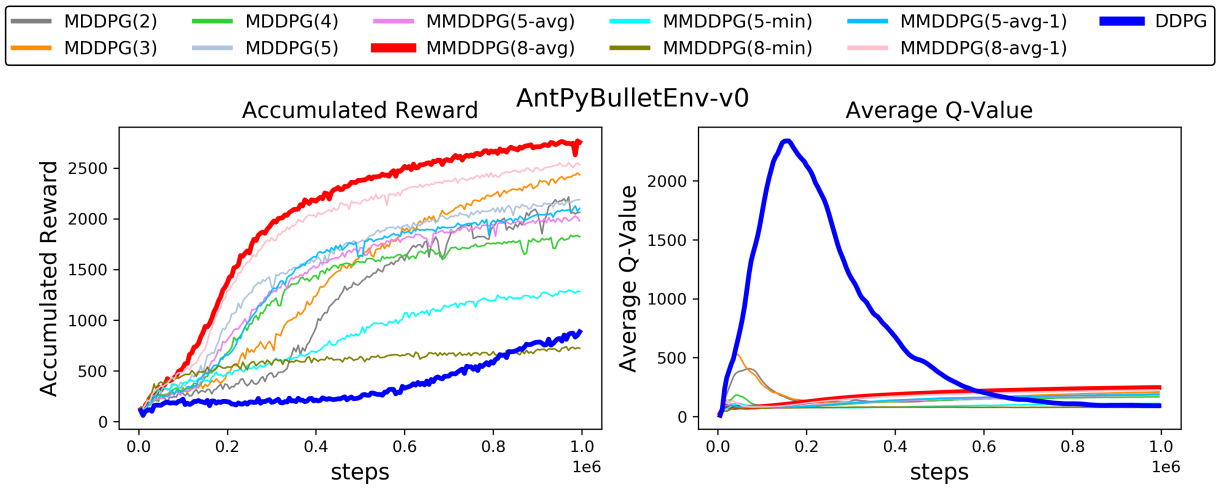
6.2.1 Experimental Evidence of Multi-step Methods' Effect on Alleviating Overestimation

Fig. 6.0 compares DDPG with its variants MDDPG and MMDDPG with different step size n. This figure illustrates that all MDDPG(n) with $n > 1$ outperform DDPG, and especially for MMDDPG(8-avg) the improvement, in terms of final performance and learning speed, is significant as highlighted with the red line. To illustrate the underlying relationship between the performance and learned Q-value, the average Q-value is shown in parallel, from which we can see that the bad performance of DDPG always corresponds to an extremely large Q-value. Note that even though multi-step methods help to relieve the

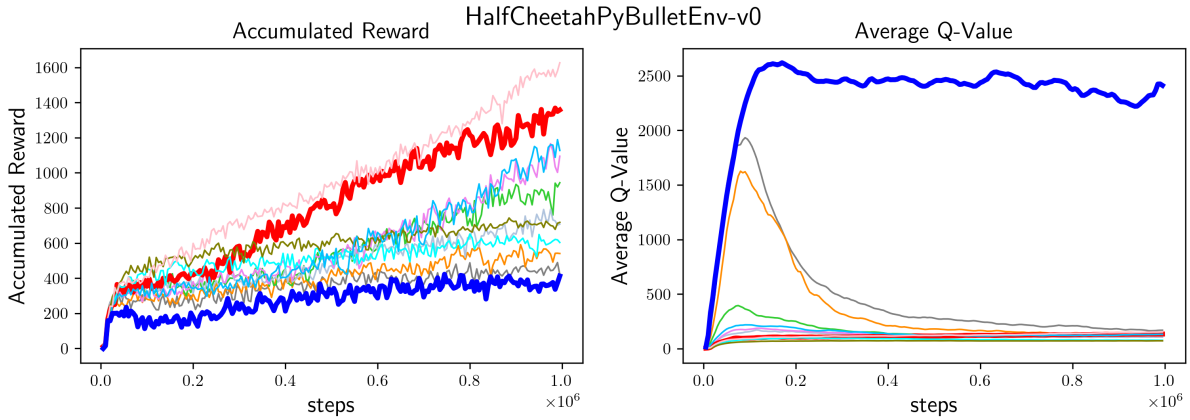
[1] https://github.com/benelot/pybullet-gym

[2] DDPG, TD3 and SAC use implementations in https://github.com/openai/spinningup, and MVE and STEVE use the implementations in https://github.com/tensorflow/models/tree/master/research/steve

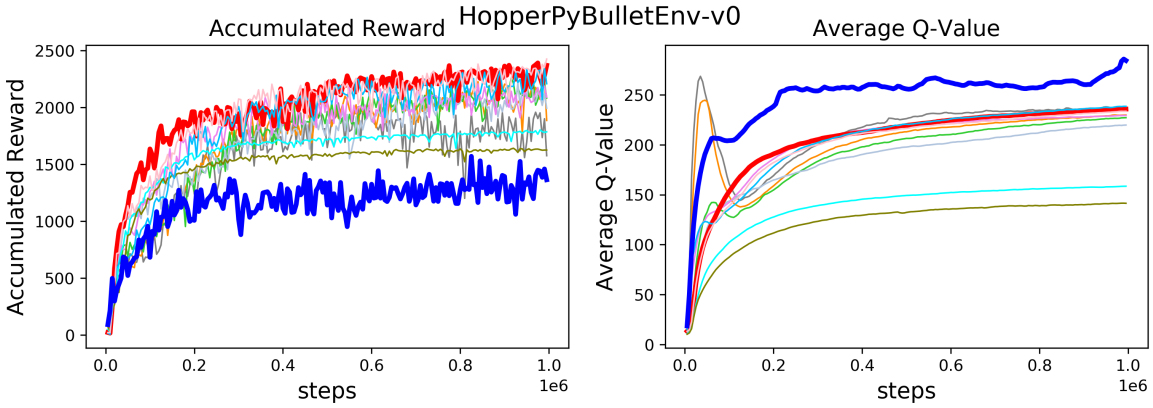
overestimation problem, they cannot completely overcome this problem, as shown by the drastic increase and followed by the sharp decrease in Q-value within the first few epochs in Fig. 6.0, whereas DDPG takes more time to decrease its Q-value, and in some cases never does. The initial overestimation is caused by approximation error on most (state, action) pairs, because at the beginning stage of the learning only a small set of (state, action) pairs are encountered, causing a high error in (state, action) pairs without training data. However, as more and more experiences are collected in the replay buffer, the approximation error is reduced. From the average Q-values in Fig. 6.0, we can see that none of the examined approaches avoid this initial explosion in the average Q-values.



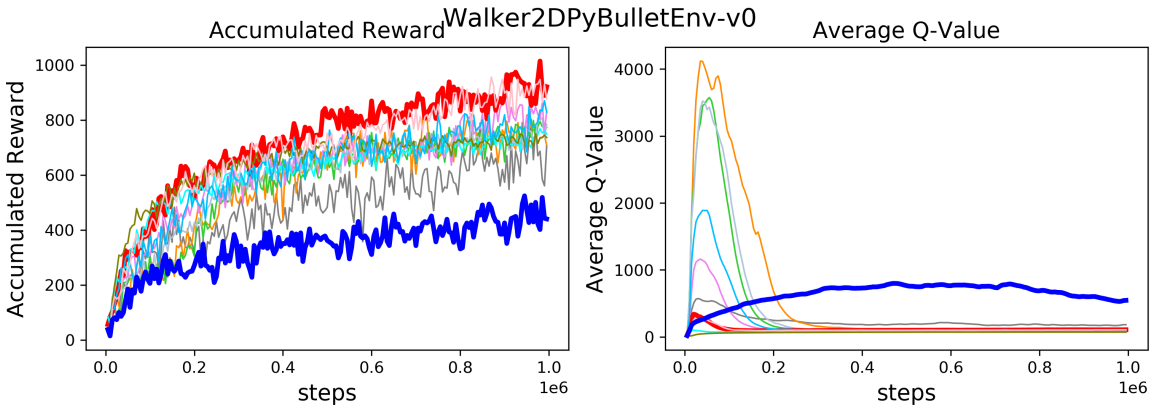
(a) AntPyBulletEnv-v0



(b) HalfCheetahPyBulletEnv-v0



(c) HopperPyBulletEnv-v0



(d) Walker2DPyBulletEnv-v0

Figure 6.0: Comparison Among MDDPG, MMDDGP and DDPG, where for each task accumulated reward and average Q-value are shown side-by-side correspondingly to demonstrate the relationship between the overestimation of Q-value and performance.

To investigate why multi-step methods help to alleviate the overestimation problem, we record backups of sampled experiences $\hat{Q}_t^{(1)}$, $\hat{Q}_t^{(2)}$, $\hat{Q}_t^{(3)}$, $\hat{Q}_t^{(4)}$, $\hat{Q}_t^{(5)}$, $\hat{Q}_t^{(n_{avg})}$, and $\hat{Q}_t^{(n_{min})}$ for DDPG, MDDPG(n) and MMDDPG(n-avg) to depict the gap between 1-step and multi-step backups. For example, $\hat{Q}_t^{(1)} - \hat{Q}_t^{(2)}$, indicated as *"TQ 1Step- TQ 2Step"* in Fig. 6.1 shows the difference between 1-step and 2-step backups. Four key characteristics can be observed in this figure: **(1)** within a specific algorithm all gaps are positive which means multi-step methods provide smaller estimated target Q-values than that of the 1-step method; **(2)** the larger the step, the smaller the corresponding estimated target

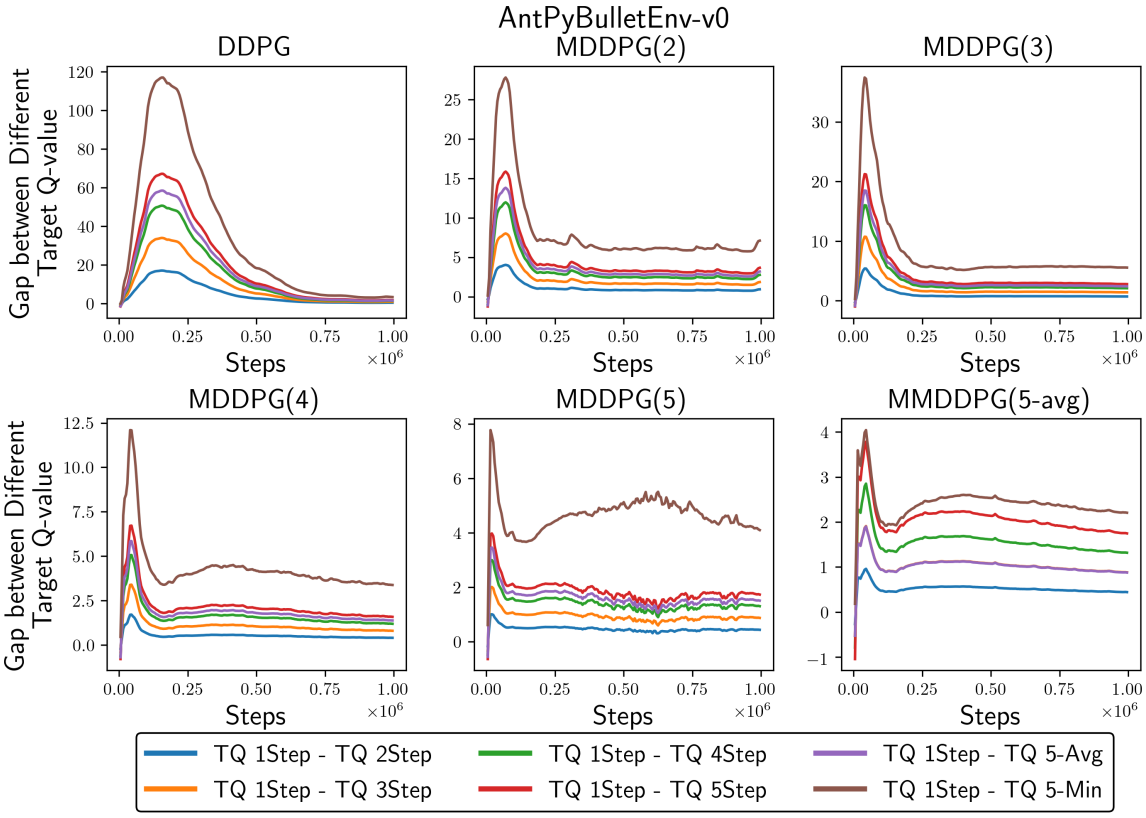
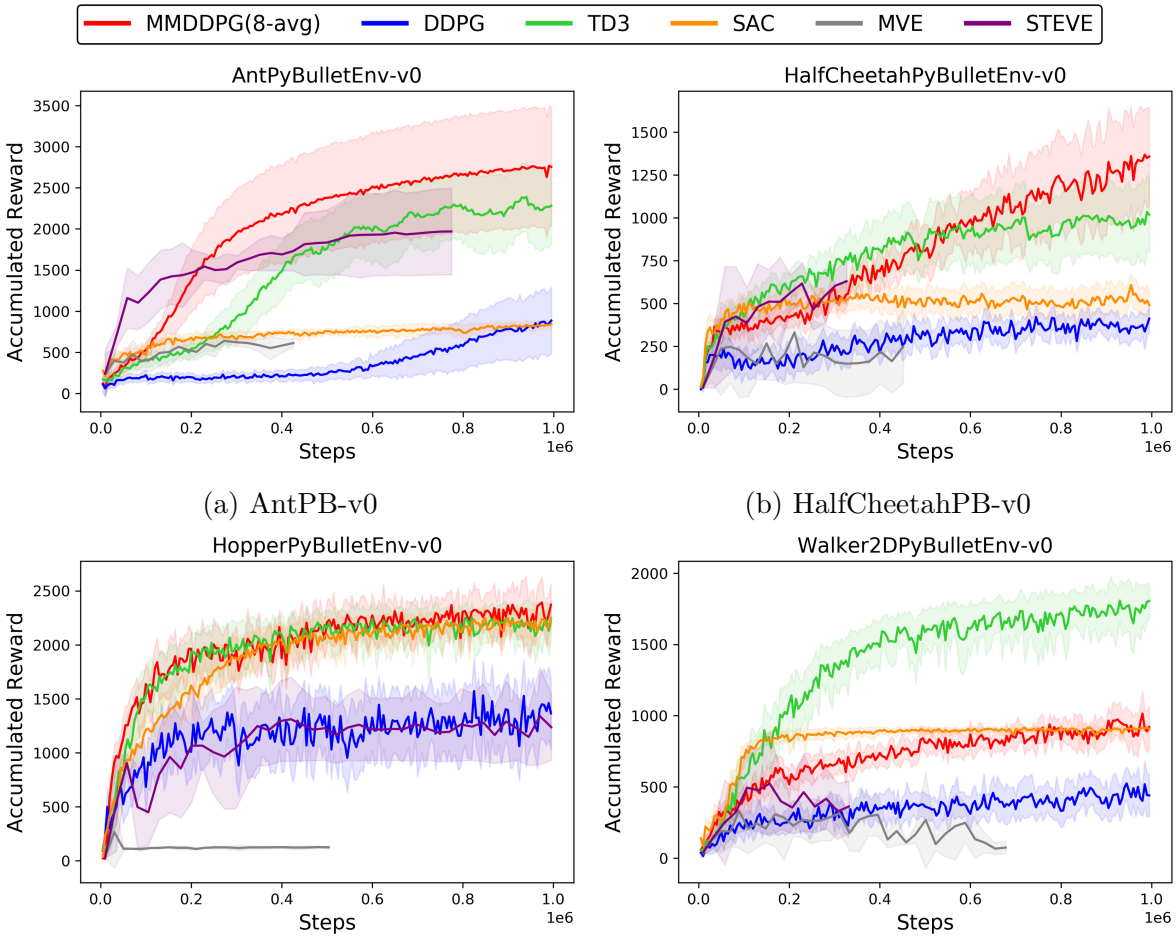


Figure 6.1: The Difference in Estimated Target Q-values Between 1-step and Multi-step Methods, where the larger the value, the bigger the difference. (MDDPG(n) is a multi-step DDPG with step size n, and MMDDPG(n-avg) is a mixture of 1- to n-step DDPG.)

Q-value, e.g. the blue line underneath the yellow line in each sub-figure; **(3)** the difference becomes smaller with increased interactions; and **(4)** among the different algorithms, the magnitude of the estimated Q-value decreases as the step size n increases. These findings provide insight into multi-step methods' effect on alleviating the overestimation problem.

6.2.2 Performance Comparison

This section focuses on comparing MMDDPG(8-avg) with other baselines namely DDPG, TD3, SAC, MVE and STEVE. Special attention is given to MVE and STEVE, because these two algorithms are very similar to MDDPG and MMDDPG with the difference that they expand multi-steps in a learned environment model. Fig. 7.4 shows the learning curves



(a) AntPB-v0

(b) HalfCheetahPB-v0

(c) HopperPB-v0

(d) Walker2DPB-v0

Figure 6.2: Learning Curves for PyBulletGym Tasks. The shaded area shows half of standard deviation of the average accumulated return over 10 trails. MEV and STEVE are not run for full 1 million steps as they take more than 58 hours even on a machine with 2 x NVIDIA P100 Pascal and 3 CPUs @ 2.1GHz.

of these algorithms on various tasks, and Table 6.1 summarizes the maximum average return. Obviously, MMDDPG(8-avg) significantly outperforms DDPG on all examined tasks. Surprisingly, MMDDPG(8-avg) performs comparably and even better on some tasks than TD3 which is currently one of the state-of-the-art approaches, specifically designed to address function approximation error in DDPG. Considering all examined tasks in this work

Table 6.1: Maximum Average Return over 10 Trials of 1 million Steps of MMDDPG(8-avg), DDPG, TD3, SAC, MVE and STEVE. The maximum value for each task is **bolded**.

Algos	AntPB	HalfCheetahPB	Walker2dPB	HopperPB
MMDDPG (8-avg)	**2767.1±1461.4**	**1368.9±527.3**	1014.0 ±316.3	**2391.9±473.3**
DDPG	885.4±811.7	422.2±137.6	524.0±227.5	1570.7±626.8
TD3	2388.0±832.6	1033.9±429.6	**1806.8±270.0**	2253.9±295.2
SAC	845.5±103.7	608.3±131.1	918.9±33.4	2249.9±207.9
MVE	639.5±33.9	331.9±290.6	332.4±261.8	263.4±332.4
STEVE	1969.0±525.7	630.5±132.2	522.7±368.3	1338.6±449.6
Algos	**AntMJC**	**HalfCheetahMJC**	**Walker2dMJC**	
MMDDPG (8-avg)	3042.8±1038.5	**2242.2±338.6**	1365.6±409.9	
DDPG	2014.8±1371.6	1311.8±1367.6	844.7±521.1	
TD3	**3495.2±725.9**	2201.1±692.8	**1583.3±670.1**	
SAC	1680.5±414.6	1977.7±180.4	779.1±178.7	

have a dense reward signal, where multi-step methods' effect on enabling fast propagation of reward will be less important, we speculate that multi-step plays a similar role in alleviating the overestimation problem as does TD3, but using a different mechanism. Compared with MDDPG and MMDDPG, the disadvantage of TD3 is it introduces more computation for training its critics, because it maintains two separate critics and at each training step these two critics are updated to the minimum estimated Q-value of their target critic networks. Detailed comparison in terms of computation cost among DDPG, MDDPG, MMDDPG and TD3 will be discussed in Section 6.3.

Counterintuitively, SAC performs worse than TD3 on tasks from PyBulletGym, and for some tasks SAC is even worse than DDPG. This is unexpected, as it is shown in [106] that SAC outperforms TD3 on some difficult continuous control tasks from OpenAi gym which use the MuJoCo [277] physics engine. One possible explanation is that PyBulletGym uses Bullet physics [69], at the same time environments in PyBulletGym are ported from Roboschool environments which are harder than MuJoCo gym, as the robot's body is heavier than that in MuJoCo tasks and termination states are added if robot flips over. This needs further investigation. Especially if SAC is going to be employed in a real robot, the belief that SAC is the best choice might be misleading.

MVE performs the worst on most tasks. STEVE is shown to be sample efficient on AntPyBulletEnv-v0 and is better than MVE, which is consistent with the results in [47].

However, STEVE is worse than MMDDPG(8-avg) and TD3.

6.3 Discussion

In this section, we discuss the advantages and disadvantages of different ways to do multi-step expansion, and expose the tradeoff between overestimation and underestimation that underlies offline multi-step methods. Then, we compare the computation resource consumption between TD3 and our proposed methods, since they show comparable final performance and learning speed.

Comparison of Multi-step Expansion Methods

As shown in Eq. 6.4, there are three ways to calculate $\hat{Q}_t^{(n)}(s_t, a_t)$ depending on how the $n-1$ experiences after (s_t, a_t, r_t, d_t) are acquired: **(1) offline expansion**, sampled from the replay buffer, e.g. MDDPG and MMDDPG; **(2) online expansion**, sampled from the environment according to an online policy, e.g. $Q(\sigma)$ [75]; **(3) model-based expansion**, sampled from a learned environment model according to an online policy, e.g. MVE and STEVE.

$$\hat{Q}_t^{(n)}(s_t, a_t) = r_t + \overbrace{\underbrace{\sum_{k=1}^{n-1} \gamma^k r_{t+k}}_{underestimation\ prone}}^{offline,\ online,\ model} + \gamma^n \underbrace{\max_a Q_{\theta^{Q^-}}(s_{t+n}, a)}_{overestimation\ prone} \tag{6.4}$$

Theoretically, online expansion is the best as the multi-step experiences are directly sampled from the environment. However, this is unrealistic, because expanding multi-step for each experience (s_t, a_t, r_t, s_{t+1}) in a mini-batch is time-consuming, especially when running multiple parallel environments (e.g. in simulation) is impossible.

A compromise is learning an environment model, then doing multi-step expansion on the learned environment model, as is done in MVE and STEVE. The challenge with this approach is that learning an environment, including transition dynamics and reward function, might be as hard or even harder than learning a policy, even without considering the extra cost for computation resources. It is also not clear to what extent the error introduced by the learned environment model will harm the learning of a policy. As shown

in Fig. 7.4 and Table 6.1, MVE and STEVE do not provide significant benefit, compared with MMDDPG(8-avg) and TD3.

Offline expansion is a solution somewhat in between. On one hand, it does not need to learn an environment model or to expand according to current policy on the environment, but uses past experiences after (s_t, a_t, r_t, s_{t+1}) as an expansion of the current online policy on the environment, with only negligible extra computation required. On the other hand, it is not an exact expansion of online policy, which introduces error in the estimated target Q-value and tends to be an underestimation of Q-value following current online policy. Nevertheless, offline expansion gradually approaches online expansion as the replay buffer fills with experiences following a stable optimal policy. This is seen in Fig. 6.3 where the initial gap between online and offline multi-step expansion is big, indicating large underestimation, but gradually decreases with the increase of interactions.

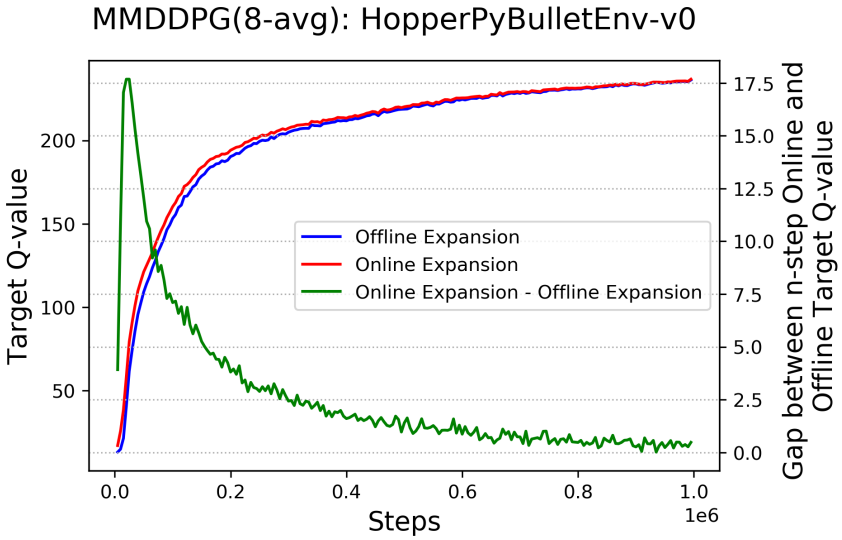


Figure 6.3: Comparison between Online and Offline Multi-step Expansion, where the blue and the red line correspond to average of offline and online multi-step expansion over a mini-batch sampled from replay buffer, and the green line is the gap between them.

Obviously, multi-step expansion cannot completely overcome the overestimation problem, because the bootstrapped Q-value after n-steps is still prone to overestimation as shown in Eq. 6.4. But since $n > 1$, the bootstrapped part is weighted less than in the 1-step method. Overall, offline multi-step expansion tends to be an underestimation of the online multi-step expansion, while the bootstrapped Q-value after n-step tends to be an

overestimation of the value in state s_{t+n}. Therefore, the step size n balances the tradeoff between overestimation and underestimation.

Computation Resource Consumption Comparison with TD3

Similar to MVE and STEVE, MDDPG and MMDDPG proposed in this paper employ multi-step expansion to provide a more accurate target Q-value estimation for the critic in DDPG. As discussed in Section 6.2.2 and 6.3, MMDDPG outperforms MVE and STEVE in terms of learning speed, final performance and computation resource consumption. However, unlike multi-step expansion, TD3 takes the minimum of target Q-values estimated from two critics as the final target Q-value to update these two critics, to avoid value approximation error. MDDPG and MMDDPG show comparable final performance and learning speed as TD3 on most tasks. Here we focus on comparing the computation resource consumption of these three methods.

Table 6.2: Comparison of Forward and Backward Propagation on a Mini-batch for Updating the Critic

	DDPG	TD3	MDDPG(n)	MMDDPG(n)
FP	1	2	1	n
BP	1	2	1	1
Total	2	4	2	n+1

FP: Forward Propagation, BP: Backward Propagation.

Table 6.2 summarizes the time of forward and backward propagation needed for training the critic on a mini-batch of transitions. As shown in the table, DDPG needs 1 forward propagation to estimate the target Q-value and 1 backward propagation to update the critic. TD3 performs 2 forward and 2 backwards propagations, one for each critic. MDDPG with a specific step size n does not introduce extra propagations compared with DDPG. However, the number of forward propagations needed for MMDDPG with a specific choice of step size n is n, while only 1 backward propagation is needed for the updating critic. Therefore, MDDPG consumes less computation resource than TD3, while MMDDPG consumes more computational resource than TD3 only when $n \geq 4$, assuming the forward and backward propagation are equally demanding in terms of computational resources.

6.4 Summary

In this chapter, we empirically revealed multi-step methods' effect on alleviating overestimation in DRL, by proposing MDDPG and MMDDPG which are a combination of DDPG and multi-step methods, and discussed the underlying underestimation and overestimation tradeoff. Results show that employing multi-step methods in DRL helps to alleviate the overestimation problem by exploiting bootstrapping. This chapter also discussed the advantages and disadvantages of three ways to implement multi-step methods from the point of view of extra computation cost and modeling error.

However, a principled way for choosing step size n is still needed. Perhaps dynamically tuning n during the course of learning is more suitable as at different stages of learning the trade-off between overestimation and underestimation needs to be balanced differently. The most important future direction arising from this work is to find a more effective way to overcome overestimation since this is key to improving DRL algorithms' sample efficiency, while still retaining a simple exploration method in order to limit computational needs.

Chapter 7

Memory-based Deep Reinforcement Learning for POMDPs

A version of this chapter was published in the proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) [185].

Unlike the commonly used standard benchmarks where the observation space of the control task is well-engineered by researchers to fully capture the underlying state of an agent, the state space of Living Architecture Systems may not be fully observable due to the architecture scale, limited sensing, and the challenge of perceiving the internal state of human occupants, as introduced in Chapter 4. Therefore, this chapter aims to address the partial observability challenge by proposing a memory-based DRL algorithm and examining the algorithm on standard benchmarks with various types of partial observability.

DRL [192, 164]has been intensively studied in simulated environments, such as games [192] and simulated robots [164], as well as in real-world studies, such as robotics control [104, 299, 267] and human-robot interaction [222, 60, 187]. DRL enables end-to-end policy learning on tasks with high-dimensional state and action spaces, without relying on labour-consuming feature engineering. However, most works focus on developing algorithms [192, 154, 164, 239, 241, 89, 106] for Markov Decision Processes (MDPs) with fully observable state spaces [79], i.e. the observation at each time step fully represents the state of the environment. Few works consider the more complex Partially Observable Markov Decision Process (POMDP) where the observation is just a partial representation of the underlying state. However, POMDPs are ubiquitous in real robotics applications [196], such as robot navigation [50], robotic manipulation [213], autonomous driving [268, 285], and planning under uncertainty [287, 61]. Partial observability may be due to limited sensing capability,

or an incomplete system model resulting in uncertainty about full observability.

POMDPs have been tackled with the concept of belief state [230], which represents the agent's current belief about the possible physical states it might be in, given the sequence of actions and observations up to that point. These algorithms are designed to estimate the belief state, then the value function and/or the policy are learned based on the belief state [242]. However, these methods need to know the environment model and the state space and they only work on tasks with small state and action spaces.

POMDPs have also been addressed with DRL, for both discrete [110, 154, 302] and continuous [258, 285] control problems. Recurrent Neural Networks (RNN) have been exploited in DRL to solve POMDPs by considering both the current observation and action, and the history of the past observations and actions [114, 258, 301, 302, 154].

In this chapter, we propose a memory-based DRL, called *Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3)*, for continuous robot control. We provide a comparison study with other DRL algorithms where both MDP and POMDP versions of the tasks are investigated to demonstrate how observeability properties influence performance on both POMDPs and MDPs. Compared to other DRL algorithms, results show that LSTM-TD3 improves performance significantly on POMDPs where the observation space of the underlying MDPs is disturbed to reduce the observability. We will also provide an ablation study to show the contribution of each design component, and discuss the advantages and disadvantages of the proposed method.

7.1 Proposed Approach

In this chapter, we propose a memory-based DRL algorithm named LSTM-TD3 within a recurrent actor-critic framework, where both the actor and the critic employ recurrent neural networks, as illustrated in Fig. 7.1. In this section, we will first introduce the proposed recurrent actor-critic framework, then present the optimization method for the actor-critic.

In the proposed approach, a mini-batch of N experiences $\left\{(h_t^l, o_t, a_t, r_t, o_{t+1}, d_t)_i\right\}_{i=1}^N$ is sampled from the replay buffer D of experiences $(o_t, a_t, r_t, o_{t+1}, d_t)$, where d_t indicates whether the terminal state is reached after observing o_{t+1} and for each sample the past history h_t^l with length l until observation o_t at time t is defined as:

$$h_t^l = \begin{cases} o_{t-l}, a_{t-l}, \cdots, o_{t-1}, a_{t-1} & \text{if } l, t \geq 1, \\ o^0, a^0 & \text{otherwise,} \end{cases} \tag{7.1}$$

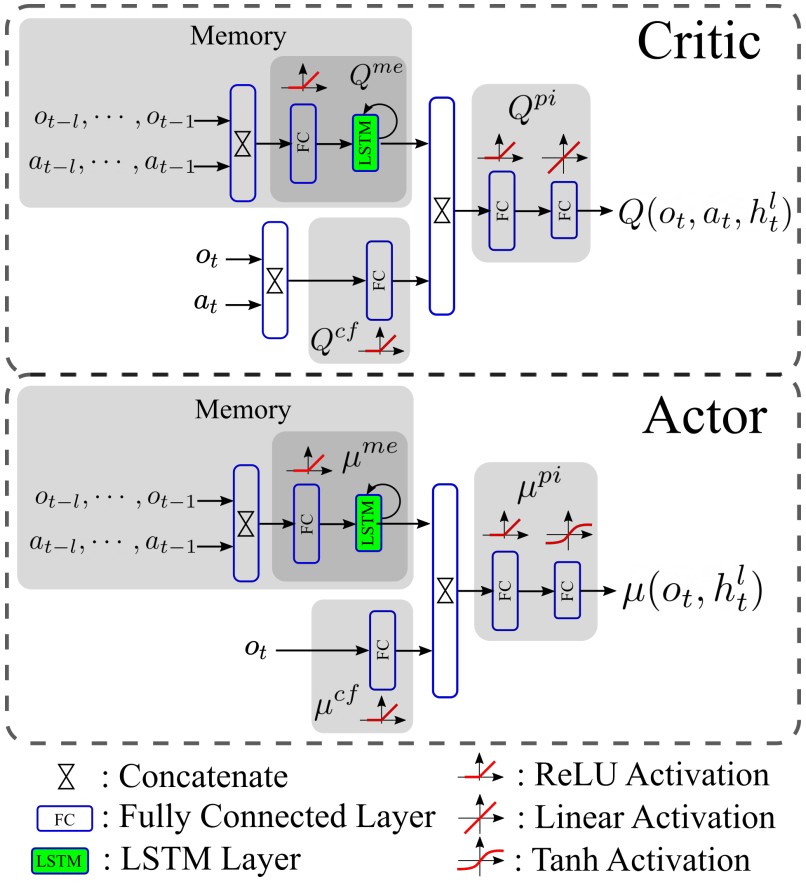


Figure 7.1: Recurrent Actor-Critic Framework

where o^0 and a^0 are the zero-valued dummy observation and action vectors with the same dimensions as those of the normal observation and action. The reason for adding the zero-valued dummy observation and action is that the recurrent actor-critic framework employs a memory component to separately extract memory from history, which means we have to feed a history into the memory component and cannot leave it empty. As defined in Eq. 7.1, if history length $l \geq 1$ and time step $t \geq 1$, the history h_t^l at time t is defined as the past l (observation, action) pairs, otherwise no history is used and zero-valued dummy vectors (observation, action) are used as input to the memory component.

7.1.1 Recurrent Actor-Critic Framework

The structure of the proposed recurrent actor-critic framework is illustrated in Fig. 7.1, where Long-Short-Term-Memory is introduced to extract information beneficial to the actor and critic from past history. The proposed framework can handle history of any length.

Formally, given a mini-batch sample of experiences, the memory-based critic Q, as illustrated in Fig. 7.1, can be seen as a compound function of the memory extraction Q^{me}, the current feature extraction Q^{cf}, and the perception integration Q^{pi} components, following Eq. 7.2

$$\begin{aligned} Q(o_t, a_t, h_t^l) &= Q^{me} \circ Q^{cf} \circ Q^{pi} \\ &= Q^{pi}(Q^{me}(h_t^l) \bowtie Q^{cf}(o_t, a_t)), \end{aligned} \tag{7.2}$$

where $\circ$ indicates function composition, $\bowtie$ indicates the concatenation operation, Q^{me} is the extracted memory based on history h_t^l, and Q^{cf} is the extracted current feature based on current observation o_t and action a_t.

Similarly, the memory-based actor μ is also a compound function of the memory extraction μ^{me}, the current feature extraction μ^{cf}, and the perception integration μ^{pi} components, defined as follows:

$$\begin{aligned} \mu(o_t, h_t^l) &= \mu^{me} \circ \mu^{cf} \circ \mu^{pi} \\ &= \mu^{pi}(\mu^{me}(h_t^l) \bowtie \mu^{cf}(o_t)), \end{aligned} \tag{7.3}$$

where $\circ$ indicates function composition, $\bowtie$ indicates the concatenation operation, μ^{me} is the extracted memory based on history h_t^l, and μ^{cf} is the extracted current feature based on current observation o_t.

7.1.2 Optimization of the Recurrent Actor-Critic

The optimization of the proposed recurrent actor-critic framework follows that of TD3. Specifically, each critic $Q_{j\in\{1,2\}}$ is optimized to minimize the mean-square-error between the predicted Q_j and the estimated target $\hat{Q}$ with respect to the parameters θ^{Q_j} of the critic Q_j, as follows:

$$\min_{\theta^{Q_j}} \mathbb{E}_{\{(h_t^l, o_t, a_t, r_t, o_{t+1}, d_t)_i\}_{i=1}^N} (Q_j - \hat{Q})^2, \tag{7.4}$$

where given the definition of memory-based critic (Eq. 7.2) and actor (Eq. 7.3), the target Q-value $\hat{Q}$ based on the target actor μ^- and critic Q_j^- is defined as follows

$$\hat{Q} = r_t + \gamma * (1 - d_t) * \min_{j=1,2} Q_j^-(o_{t+1}, a^-, h_{t+1}^l), \tag{7.5}$$

where $a^- = \mu^-(o_{t+1}, h_{t+1}^l) + \epsilon$ with $\epsilon \sim \text{clip}(\mathbb{N}(0, \sigma), -c, c)$ and c is the boundary of target action noise, $h_{t+1}^l = (h_t^l - (o_{t-l}, a_{t-l})) \cup (o_t, a_t)$ is the l observation and action pairs before o_{t+1}, and the minimum of the estimated optimal Q-values of the two target critics in (o_{t+1}, h_{t+1}^l) is taken to bootstrap the target Q-value of (o_t, a_t, h_t^l).

For the actor, its parameters θ^μ are optimized to maximize the approximated Q-value in observation (o_t, h_t^l) and the corresponding estimated optimal action $\mu(o_t, h_t^l)$ with respect to the parameters of the actor, as follows:

$$\max_{\theta^\mu} \mathbb{E}_{\{(h_t^l, o_t)_i\}_{i=1}^N} Q(o_t, \mu(o_t, h_t^l), h_t^l), \tag{7.6}$$

where the Q could be either of the two critics Q_1 and Q_2, as in TD3. The pseudo-code for optimizing the recurrent actor-critic can be found in Alg. 3 and the actor optimization is depicted in Fig. 7.2 where the parameters θ^{Q_1} of the critic Q_1 is fixed while the parameters θ^μ of the actor is optimized according to Eq. 7.6..

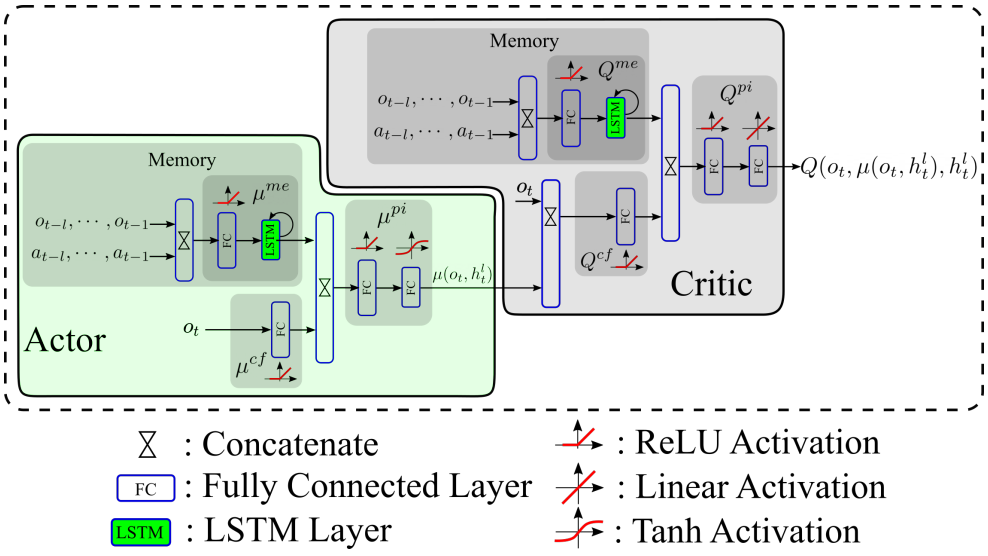


Figure 7.2: Actor Optimization

ALGORITHM 3: Pseudo-code for LSTM-TD3

Input: History length L

Initialize critics $Q_{\theta^{Q_1}}$, $Q_{\theta^{Q_2}}$, and actor μ_{θ^μ} with random parameters θ^{Q_1}, θ^{Q_2} and θ^μ

Initialize target networks $\theta^{Q_1^-} \leftarrow \theta^{Q_1}$, $\theta^{Q_2^-} \leftarrow \theta^{Q_2}$ and $\theta^{\mu^-} \leftarrow \theta^\mu$

Initialize environment $o_1 = \text{env.reset}()$, past history $h_1^l \leftarrow \mathbf{0}$, and replay buffer D

for $t = 1$ **to** T **do**

`/* Interacting */`

Select action with exploration noise $a_t \sim \mu_{\theta^\mu}(o_t, h_t^l) + \epsilon$, $\epsilon \sim \mathbb{N}(0, \sigma)$

Interact and observe new observation, reward, and done flag: o_{t+1}, r_t, $d_t = \text{env.step}(a_t)$

Store experience tuple $(o_t, a_t, r_t, o_{t+1}, d_t)$ in D

if d **then**

Reset environment $o_{t+1} = \text{env.reset}()$ and history $h_{t+1}^l \leftarrow \mathbf{0}$

else

`/* Update` h_{t+1}^l `*/`

$h_{t+1}^l = (h_t^l - (o_{t-l}, a_{t-l})) \cup (o_t, a_t)$

end

`/* Learning */`

Sample mini-batch of N experiences with their corresponding histories $\{(h_t^l, o_t, a_t, r_t, o_{t+1}, d_t)_i\}_{i=1}^N$ from D

Optimize Q_j according to Eq. 7.4

Optimize μ according to Eq. 7.6

Update target actor-critic

end

7.2 Experiment Settings

The tasks (Fig. 7.3 where (a) HalfCheetahPyBulletEnv-v0, (b) AntPyBulletEnv-v0, (c) Walker2DPyBulletEnv-v0, (d) HopperPyBulletEnv-v0, and (e) InvertedDoublePendulumPyBulletEnv-v0) tested in this work come from PyBullet-Gym[1], an open-source implementation of the OpenAI Gym MuJoCo environment based on BulletPhysics[2]. In this work, an MDP-version and 4 POMDP-versions of each task are investigated, described in Table 8.1. The MDP-version is the original task, as it has a fully observeable state-space, while the 4 POMDP-versions simulate different scenarios that potentially cause partial observability in real applications. Specifically, the POMDP-RemoveVelocity (POMDP-RV) is designed to simulate the scenario where the observation

[1] https://github.com/benelot/pybullet-gym

[2] https://github.com/bulletphysics/bullet3

(a) (b) (c) (d) (e)

Figure 7.3: Example PyBulletGym Tasks.

space is not well-designed, which is applicable to a novel control task that is not well-understood by researchers and therefore the designed observation may not fully capture the underlying state of the robot. The POMDP-Flickering (POMDP-FLK) models the case where remote sensor data is lost during long-distance data transmission. The case when a subset of the sensors are lost is simulated in POMDP-RandomSensorMissing (POMDP-RSM). Sensor noise is simulated in POMDP-RandomNoise (POMDP-RN).

Table 7.1: MDP- and POMDP-version of Tasks

Name	**Description**	**Hyper-parameter**
MDP	Original task	–
POMDP-RV	Remove all velocity-related entries in the observation space.	–
POMDP-FLK	Reset the whole observation to 0 with probability p_{flk}.	p_{flk}
POMDP-RN	Add random noise $\epsilon \sim \mathbb{N}(0, \sigma_{rn})$ to each entry of the observation.	σ_{rn}
POMDP-RSM	Reset an entry of the observation to 0 with probability p_{rsm}.	p_{rsm}

The baselines used to compare with the proposed LSTM-TD3 are the DDPG [164], SAC [106], TD3 [89], TD3 with Observation-Window (TD3-OW) where the o_t is simply concatenated with the observations within the history window h_t^l to form an observation as input, and TD3 with Observation-Window-AddPastAct (TD3-OW-AddPastAct) where o_t is concatenated with the observations and the actions within the history window h_t^l. The hyperparameters for the baseline algorithms were always the defaults provided in OpenAISpinningUp[3]. For the proposed algorithm, hyperparameters were empirically set to that for TD3, and the network structures of the LSTM-TD3 were chosen to have a similar number of parameters to the networks in TD3. All reported results are averaged over 10 evaluation episodes based on 4 different random seeds. The code used for this work can

[3]https://spinningup.openai.com

be found in https://github.com/LinghengMeng/LSTM-TD3. All hyperparameter testing and additional results (e.g. LSTM-TD3 in POMDPs with lower observability and larger history length than that reported here) are reported in the Appendix B.

7.3 Results

7.3.1 Performance Comparison

The rows of Fig. 7.4 show the learning curves of the three sampled tasks, where the fist column shows the performance on MDP, while the following 4 columns show results on POMDPs. The results on MDP show that the proposed method has competitive performance to the baselines. The results on the POMDPs highlight the advantage of having memory when solving partially observable tasks. On all types of POMDP, LSTM-TD3 outperforms all baselines, except on POMDP-RV of HalfCheetahPyBulletEnv-v0, where LSTM-TD3(5) shows slightly worse performance than TD3. Although TD3-OW shows better performance than DDPG, TD3, and SAC on POMDPs for most tasks, it still fails for some POMDPs, such as the POMDP-FLK version of most tasks. This reveals that simply concatenating observations is not a good choice, compared to having a LSTM-based memory extraction component as that in LSTM-TD3. LSTM-TD3(0) seems sensitive to random seeds (1st panel of the 3rd row of Fig. 7.4) as it achieves lower performance compared to that of TD3 and LSTM-TD3 with history length larger than 0. To explain this, even though we set the history for it to zero, it may still predict nonzero for the Q^{me} (introduced in Eq. 7.2), because the gradients with respect to the randomly initialized weights of Q^{me} may be nonzero and back-propagated, which could influence the agent during learning.

Particularly, a significant performance gap can be observed on POMDP-FLK for all tasks (the 3rd column in Fig. 7.4), where the baselines basically fail while LSTM-TD3 achieves comparative performance to that on MDP. This is especially promising for tasks where whole sensor data may be lost, either caused by hardware failure or by temporary occlusion, etc. Similar, dramatic performance improvement can be seen on POMDP-RN and POMDP-RSM.

Surprisingly, comparing LSTM-TD3 and TD3, memory does not always help for POMDP-RV (the 2nd column in Fig. 7.4) of HalfCheetah and Ant. Intuitively, if the velocity is important to learn a task, there is no way to infer such information without past observations i.e. memory. However, if previous observations are available, the velocity

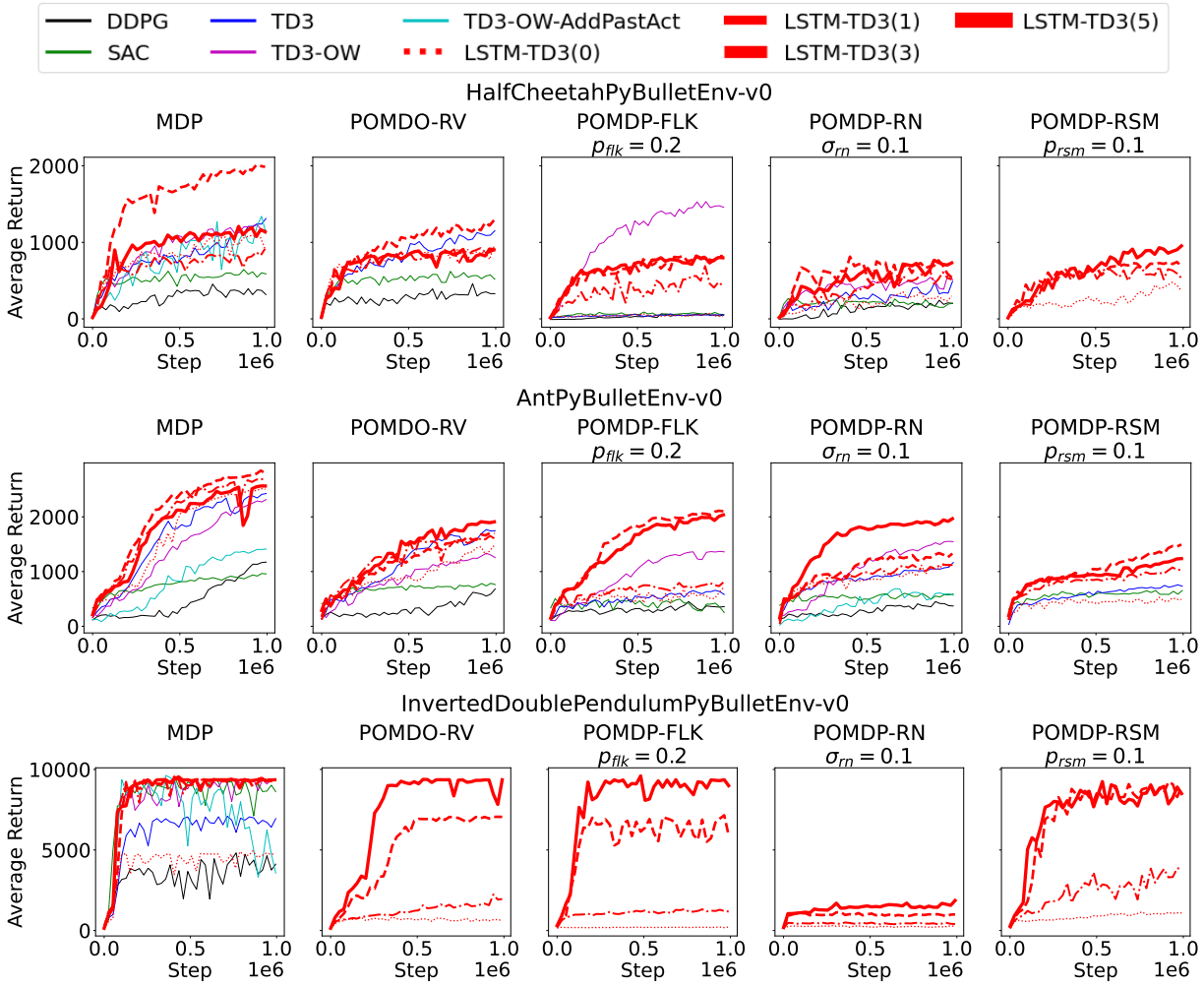


Figure 7.4: Learning Curves for PyBulletGym Tasks, where to ease the comparison only average values are plotted. In the legend, the value in the bracket of LSTM-TD3 indicate the length of the history, e.g. LSTM-TD3(5) uses history length 5.

can be inferred by differences in position between consecutive steps. This intuition can be clearly observed on the POMDP-RV of Hopper and InvertedDoublePendulum, where the performance of LSTM-TD3 is significantly better than that of TD3. For the HalfCheetah and Ant, if we compare the performance on MDP and POMDP-RV, we can still see a noticeable gap, which means velocity does contribute to learn a good policy. We hypothesize that LSTM-TD3(5) does not outperform TD3 on the POMDP-RV version of HalfCheetah and Ant due to the fact that within the history window all speeds are very similar and velocity cannot be accurately inferred, which may be caused by relatively high sampling rate.

Interestingly, by comparing the results of TD3-OW and TD3-OW-AddPastAct, we found that adding past actions consistently harms the performance compared to TD3-OW, which does not have past actions in its observation window. Even though TD3-OW-AddPastAct still outperforms TD3 on POMDP, it performs worse than TD3 on MDP, which is undesirable if we have no prior knowledge of whether the current design of the observation space is partially or fully observable. Ideally, even if the past action-related information does not provide anything new beyond the past observation, it can be safely ignored and should not harm the performance. We think this is related to the simple construction method where actions are concatenated with observations to form a single observation that includes history information. In this way, the observation dimension is expanded, which makes the learning harder. In addition, this simple construction method treats all observations equally instead of prioritizing the most recent observation, which is normally more valuable in decision-making than earlier observations. This observation based on TD3-OW and TD3-OW-AddPastAct in fact supports our idea to structurally separate the memory extraction and current feature extraction in the recurrent actor-critic framework (Fig. 7.1) designed for LSTM-TD3, then combine them together to further learn a presentation of the critic and the actor. In section 7.4.3, we will further investigate if adding past actions is beneficial for LSTM-TD3.

7.3.2 Policy Generalization

To better understand the generalization of the learned policy using LSTM-TD3, we evaluated the learned policy on a different version of a given task, i.e., if the policy is learned on the MDP-version of a task, and evaluated on the POMDP-versions of the task. This is valuable for real applications where the environment may be non-stationary. Fig. 7.5 shows the cross evaluation results on AntPyBulletEnv-v0. POMDP-RV is not included as it has a different observation dimension which corresponds to a different input shape for the neural networks. From the first panel (i.e. policies trained on MDP), TD3, TD3-OW,

and LSTM-TD3 significantly outperform DDPG, SAC, and TD3-OW-AddPastPact. When evaluating on POMDPs, there is always a decrease for TD3, TD3-OW, and LSTM-TD3, but LSTM-TD3 is the most robust and achieves better performance on these evaluation environments than TD3 and TD3-OW. As for the last three panels, even though LSTM-TD3 still outperforms TD3-OW significantly when evaluated on a different environment, for each algorithm there is not much change in performance. Actually, when trained on POMDP-FLK and evaluated on MDP, LSTM-TD3 achieves a better performance than evaluated on POMDP-FLK.

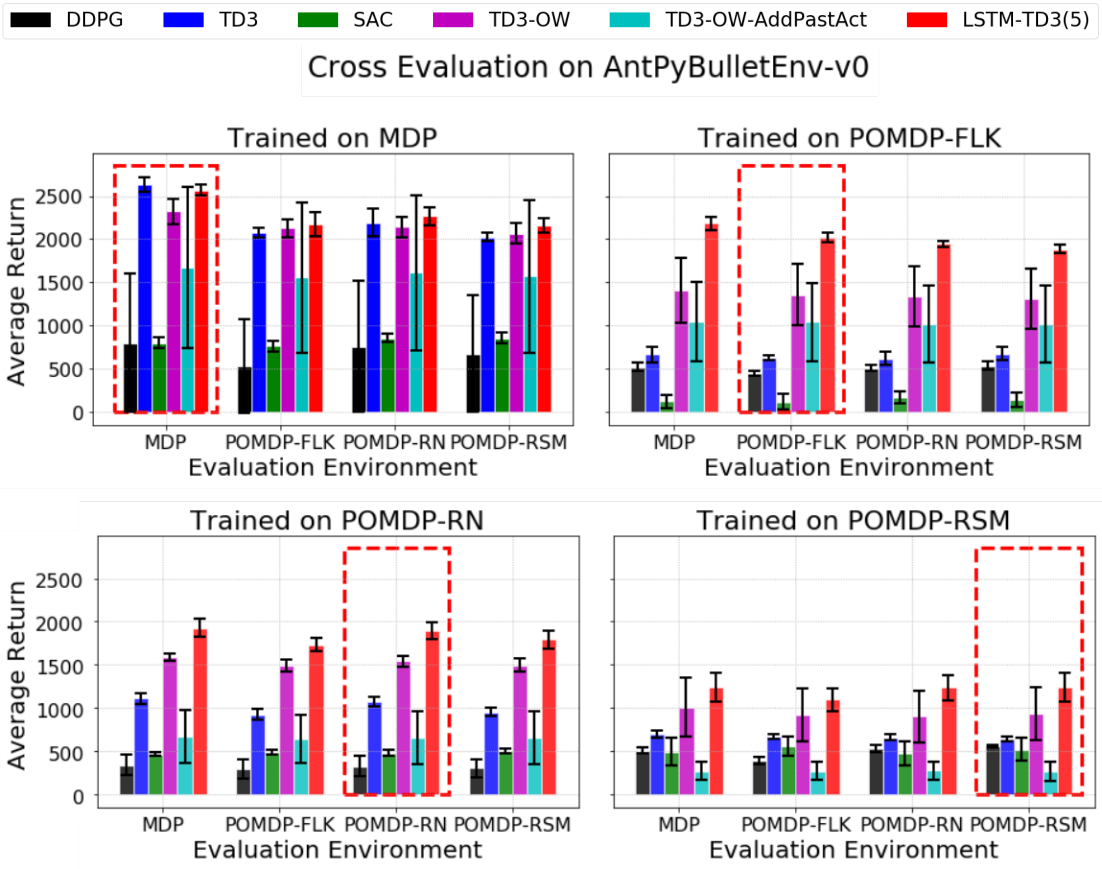
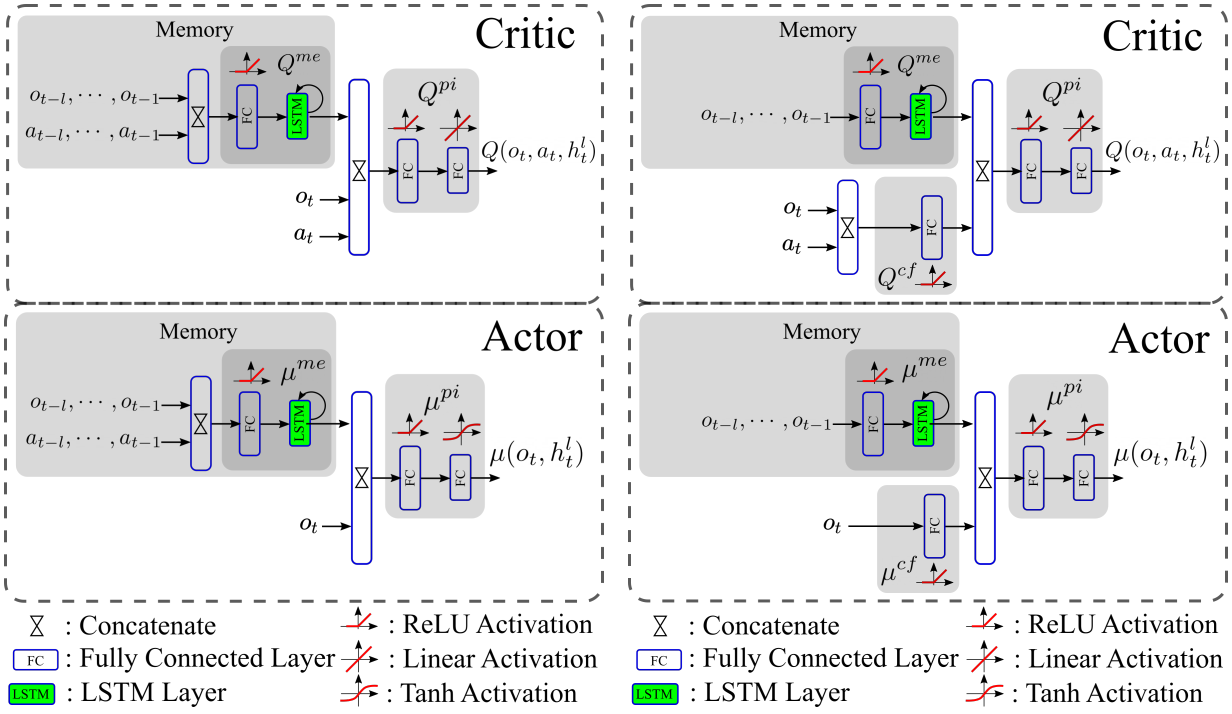


Figure 7.5: Cross Evaluation. In each panel, the x-axis indicates the evaluation environment (proposed in Table 8.1) and the y-axis is the average return, where the bars highlighted with red dashed box correspond to performances evaluated on the same environment where the policies are trained, and errorbar on the bar tips indicates the standard deviation of the performance.



(a) Full−CFE (b) Full−PA

Figure 7.6: Diagram of Full−CFE and Full−PA, where for the Full−CFE the extracted memory is directly concatenated with the current observation for the actor and with the current observation and action for the critic; and for the Full−PA past actions are exclued from the history.

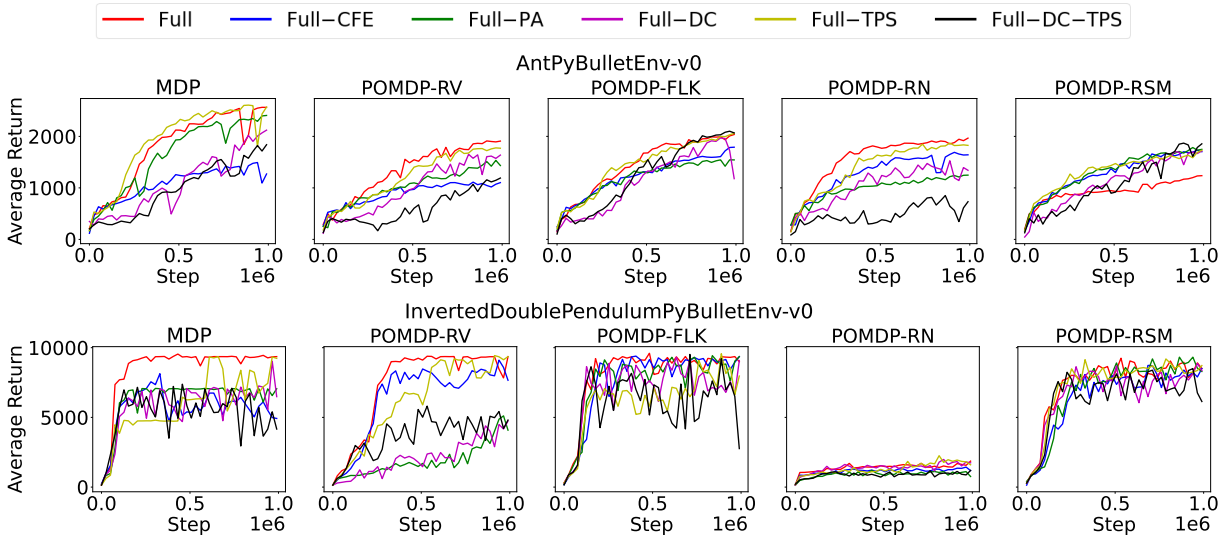


Figure 7.7: Learning Curves of Ablation Study, where to ease the comparison only average values over 10 evaluation episodes based on 4 different random seeds are plotted. In the legend, Full, Full−CFE, Full−PA, Full−DC, Full−TPS, and Full−DC−TPS correspond to LSTM-TD3 with full components, removing current feature extraction, excluding past action, not using double critics, not using target policy smoothing, and simultaneously not using double critics and target policy smoothing.

Table 7.2: Comparing DDPG and LSTM-DDPG in Terms of Maximum Average Return, where ± indicates a single standard deviation. The bolded value of LSTM-DDPG indicates the performance of LSTM-DDPT on a specific version of a task is better than the performance of DDPG on MDP-version of the task.

Task		**Algorithms**	
Name	Version	**DDPG**	**LSTM-DDPG**
HalfChePB	MDP	487.6 ± 6.1	$\mathbf{517.4 \pm 102.0}$
	POMDP-RV	508.4 ± 23.9	$\mathbf{552.0 \pm 1.4}$
	POMDP-FLK	84.8 ± 20.4	$\mathbf{690.8 \pm 0.0}$
	POMDP-RN	268.7 ± 70.2	$\mathbf{731.1 \pm 330.9}$
	POMDP-RSM	283.7 ± 27.0	$\mathbf{606.3 \pm 63.0}$
AntPB	MDP	1210.8 ± 226.1	$\mathbf{1855.8 \pm 494.2}$
	POMDP-RV	683.5 ± 101.4	1068.6 ± 363.0
	POMDP-FLK	449.0 ± 93.3	$\mathbf{2145.1 \pm 107.2}$
	POMDP-RN	449.6 ± 18.5	879.3 ± 446.9
	POMDP-RSM	465.2 ± 51.0	$\mathbf{1831.7 \pm 33.9}$

$p_{flk} = 0.2$, $\sigma_{rn=0.1}$, $p_{rsm} = 0.1$.

7.4 Ablation Study

To further understand the effect of each component of the proposed LSTM-TD3, in this section we perform an ablation study. Specifically, we examine the effects of the following components: (1) double critics (DC), (2) target policy smoothing (TPS), (3) current feature extraction (CFE), and (4) including past actions (PA) in the history. Fig. 7.7 shows the learning curves of ablated versions of LSTM-TD3, each removing a different component, while Table B.3 reports the maximum average return of the investigated ablated algorithms.

7.4.1 Effect of Double Critics and Target Policy Smoothing

As shown in Fig. 7.7, Full−DC shows a significant decrease in performance compared to Full on all MDPs and most POMDPs, whereas TPS seems less important to the best performance of Full. When simultaneously removing DC and TPS, the performance significantly decreases. Note that without DC and the TPS, the LSTM-TD3 is in fact reduced to LSTM-DDPG, similar to RDPG proposed in [114]. To ease the comparison, we summarized the results of DDPG and LSTM-DDPG, i.e. Full−DC−TPS, in Table 7.2. When

compared on the same version of a task, LSTM-DDPG always outperforms DDPG, which can be observed by comparing the results in each row. Remarkably, LSTM-DDPG even achieves significantly better performance on POMDPs than DDPG on MDP.

7.4.2 Effect of Current Feature Extraction

In this chapter, we intentionally separate the memory extraction and the current feature extraction, then combine them together (Fig. 7.1), in order to differentiate the current and the past and to reduce the interference from useless information in the memory. Alternatively, we can directly combine the current observation for the action (or the current observation and action for the critic) with the extracted memory, i.e. removing the CFE (Full−CFE) (Fig. 7.6a). As shown in Fig. 7.7, Full−CFE performs much worse, compared to Full, especially for MDP-version tasks. Recall that one scenario for devising the LSTM-TD3 is the situation where engineers are not sure if the design of the observation space is appropriate to capture the state of the agent, if the designed observation space properly captures the state of the agent and there is no CFE, poor performance will be achieved. Therefore, CFE is important for such scenarios.

7.4.3 Including Past Action Sequence in Memory

Fig. 7.6b illustrates Full−PA, where past actions are excluded from the history. As shown in Fig. 7.7, removing PA causes a decrease in performance, where a remarkable decrease can be observed on the POMDP-RV version of InvertedDoublePendulumPyBulletEnv-v0 (the 2nd panel in the last row in Fig. 7.7), which is contrary to the observation in Section B.2.1 that TD3-OW-AddPastAct performs significantly worse than TD3-OW by adding past actions in the history. This means LSTM-TD3 is more robust than OW-TD3. This is desirable especially when designers have no prior about whether observation of past actions is needed to infer the current state of an agent for an unknown task.

7.5 Summary

In this chapter, we proposed a memory-based DRL algorithm called LSTM-TD3 by combining a recurrent actor-critic framework with TD3. The proposed LSTM-TD3 was compared to standard DRL algorithms on both the MDP- and POMDP-versions of continuous control tasks. Our results show that LSTM-TD3 not only achieves significantly better

performance on POMDPs than the baselines, but also retains the state-of-art performance on MDP. Our ablation study shows that all components are essential to the success of the LSTM-TD3 where DC and TPS help in stabilizing learning, CFE is especially important to retain the good performance in MDP, and PA is beneficial for tasks where past actions provide information about the current state of the agent.

Chapter 8

Partial Observability during DRL for Robot Control

A version [183] of this chapter is under review at the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2023).

This chapter builds on the analysis in Chapter 7 to identify the source of the failure caused by partial observability when applying Deep Reinforcement Learning (DRL) to Living Architecture Systems. The work in this chapter is inspired by the counter-intuitive observation that applying DRL algorithms to a simulated LAS produces different results compared to applying DRL on standard benchmarks. Based on the observed behavior in the simulated LAS, this chapter reproduces the same counter-intuitive observations on standard benchmarks and hypothesizes that partial observability may cause the failure of the application of DRL algorithms.

DRL has made tremendous advances in both simulated and real-world robot control tasks in recent years. Nevertheless, applying DRL to novel robot control tasks is still challenging, especially when researchers have to design the action and observation space and the reward function. In this chapter, we investigate partial observability as a potential failure source of applying DRL to robot control tasks, which can occur when researchers are not confident whether the observation space fully represents the underlying state. We compare the performance of three common DRL algorithms, TD3, SAC and PPO under various partial observability conditions. We find that TD3 and SAC become easily stuck in local optima and underperform PPO. We propose multi-step versions of the vanilla TD3 and SAC to improve robustness to partial observability based on one-step bootstrapping.

This chapter aims to fill the gap between directly assuming a novel system to be con-

trolled is MDP or POMDP. Particularly, we first introduce an exemplar robot control problem caused by partial observability on a novel complex system, then discuss the potential effect of multi-step bootstrapping on passing temporal information. After that, we propose two hypotheses and the corresponding algorithms to verify these hypotheses. In the experiments, we reproduce the counter-intuitive results observed from the complex system on toy control tasks to confirm that the problem is caused by partial observability. In addition, the results to verify our hypotheses are presented. Then, we discuss the limitations of this work.

8.1 Exemplar Robot Control Problem

(a) Physical LAS (b) Simulated LAS

Figure 8.1: LAS installation *Meander*, where (a) is an image of the physical installation *Meander* (courtesy of Philip Beesley Studio Inc.), and (b) shows the simulated LAS.

In this section, we will provide a motivating example highlighting the challenge of applying DRL to complex robotic control tasks. The Living Architecture System (LAS) [28, 188], an architectural-scale interactive system with hundreds of actuators such as lights, shape memory alloys, DC motors and speakers, etc., and sensors such as infrared sensors and microphones, etc., is a robot system that is: (1) novel, i.e., not a commonly used test-bed in DRL, requiring researchers to design the action/observation space and reward function; and (2) complex, i.e., the dynamics of the robot itself and its external environment is unknown. LAS is designed by a collection of architects, artists, psychologists, roboticists, computer scientists and engineers, aiming to engage occupants in a long-term interaction. A classic scenario in LAS is multiple visitors wandering around a LAS and trying to interact with it, where the long-term goal is to engage the visitors and hold their interest. Fig.

8.1 shows the LAS installation *Meander* [1], where Fig. 8.1a shows a photo of the physical installation and Fig. 8.1b shows the simulated *Meander* within LAS-Behavior-Engine, a simulator and a behavior controller. Within the LAS, there are over 500 actuators and about 50 sensors spread over the whole space of the installation. With such a large set of actuators and sensors and the complexity of human factors, it is extremely hard to handcraft engaging behavior by direct control of the actuators. Therefore, a middle layer is designed to add a set of dynamics to induce different activation intensities in the actuators either when observing changes in sensors or when in background behavior mode. The parameters involved in controlling the dynamics can be used by either human designers or learning agents to generate engaging behaviors in the LAS. In other words, the large raw action space is transferred by a complex mapping into a simplified and easy-to-understand parameterized action space. For example, in Fig. 8.1b the excitors (yellow spheres) are randomly positioned and attracted to move to attractors (pink hexagons) and activate actuators they pass by as they do so, which are controlled by parameters such as the size, the speed and the maximum number of excitors, etc.

To apply DRL to LAS, researchers need to design the key RL components, namely the observation and action space and the reward function. However, the design of these components is not trivial. In a first attempt to learn an effective control policy, we designed the three components of LAS as: **Observation space**: the status of actuators and sensory readings in $[0, 1]$ within a time window. Specifically, for a 1 second time window and 1Hz data reading, the observation space has 724 dimensions, composed of 124 dimensions of sensory readings and 600 dimensions of actuator status; **Action space**: the 9 dimensional parameterized action space in $[-1, 1]$ where each dimension corresponds to one parameter involved in the excitor dynamics and applies to all excitors within the system; and **Reward function**: the average over the actuator intensities included in the observation space, which means the reward function encourages actions that maximally activate the actuators.

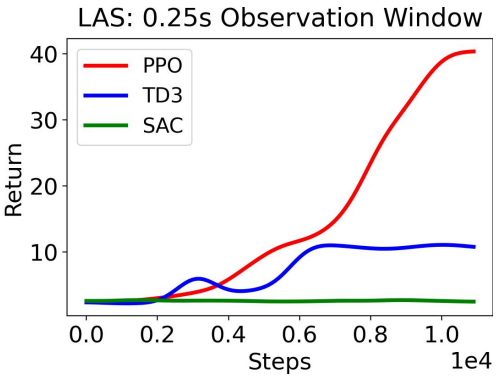


Figure 8.2: Unexpected Results on LAS, where PPO is better than TD3 and SAC.

Three state-of-the-art DRL algorithms, i.e. TD3, SAC, and PPO, introduced in Section 3.3, were tested on the environment. Fig. 8.2 shows the learning curves of these algorithms

[1] More images and demonstrative videos of *Meander* can be found in https://youtu.be/SVTc7xOSBrg.

on LAS with the observation, action and reward formulated as described above. Surprisingly, TD3 and SAC perform much worse than PPO, where TD3 is slightly better than random and SAC is about the same as random, contrary to reports that the performance of both TD3 and SAC are much better than PPO on tasks provided in OpenAI Gym [89, 106].

After some investigation, we hypothesized that the problem shown in Fig. 8.2 could be caused by the ***partial observability*** of the observation space[2]. The intuition behind this is that temporal information is unavailable through the short 0.25s observation window, but somehow PPO seems to be able to incorporate some temporal information while TD3 and SAC fail. To be more concrete, when the observation window is short, there is no information about the change rate of the actuator status and the sensory readings, which may be important for solving the problem. For example, knowing the increase or decrease of the activation intensity of an actuator caused by an action is beneficial to learn a policy that encourages active behavior. However, intuitively, TD3, SAC and PPO are all general DRL algorithms without special consideration for handling POMDP. In addition, considering the special characteristics of LAS, we also suspect the problem in Fig. 8.2 could be related to ***observation delay*** [40], where the observation o_t received at time step t is the representation of the state $s_{t-\Delta_d}$ that Δ_d is the time an observation is delayed. This is inevitable when the observations are communicated through UDP protocal. Further, ***action transformation*** [140, 8] is also a possible cause, considering the parameterized action space will experience a complex transformation into the raw action space for a robot to execute. To interpret these results we identify variations on benchmark OpenAI gym tasks that replicate the algorithms' performance.

8.2 The Potential Effect of Multi-step Bootstrapping on Passing Temporal Information

To understand why PPO is better than TD3 and SAC in terms of handling POMDP, we revisit their policy and value function optimization introduced in Section 3.3. One prominent difference among them is that PPO uses multi-step bootstrapping with $n > 1$ while TD3 and SAC use 1-step bootstrapping. Specifically, PPO uses λ-return defined in Eq. 3.22, which is a weighed average of n-step returns where $n \in [1, T-t-1]$, to calculate

[2]Along with environment related hyper-parameters such as observation and action space and reward function design, we did try to reduce the depth of the neural networks employed in TD3 and SAC, but did not find an obvious difference in the performance.

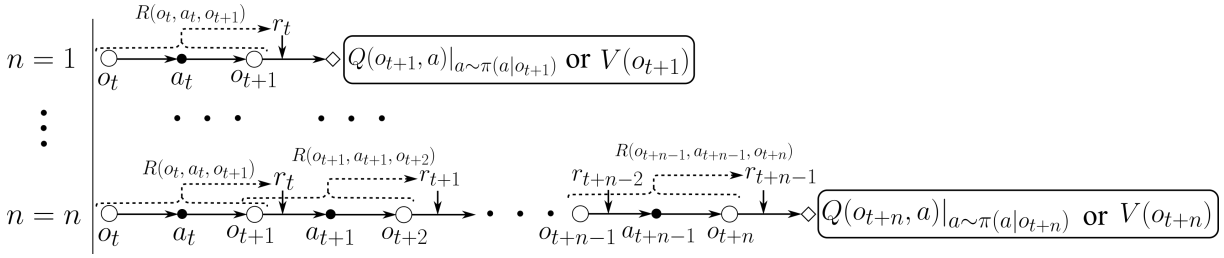


Figure 8.3: Information Incorporated in n-step Bootstrapping, where the n immediate rewards and the bootstrapped value, whose calculation depends on what value function is available, thereafter are included.

the advantage $A^{\pi_{\varphi_k}}(o_t, a_t)$ of taking action a_t in observation o_t. However, TD3 and SAC only use 1-step bootstrapping to calculate their target Q-value as defined in Eq. **??**. Fig. 8.3 illustrates n-step bootstrapping with different n values, where the n immediate rewards and the bootstrapped value are discounted [3] and added together.

The reward signal $r = R(o, a, o')$ can be seen as a one-dimensional state-transition abstraction of (o, a, o'). If the reward signal is dense, it is possible that each underlying state can be uniquely represented by a reward signal (This may sound extreme, but considering the most MuJoCo tasks from OpenAI Gym where the reward is a function of the action, robot status and moving direction and velocity, a reward may uniquely represent the underlying state). Then, for the case where $R(o, a_1, o'_1) < R(o, a_2, o'_2)$, a_1 can be thought to encode less information than a_2. Therefore, the goal of an agent can be interpreted to maximize accumulated information encoding. With n-step bootstrapping where $n > 1$, n consecutive state-transition abstractions are combined through weighted summation. By combining consecutive state-transition abstractions, some temporal information is also incorporated. One may argue that even for $n = 1$ it is also possible to incorporate temporal information that exists in the value function. However, the value function is approximate, which is not as effective as that calculated directly from the n consecutive state-transition abstractions. Once the n-step bootstrapping has incorporated temporal information, it will be passed to the value function. When the policy is optimized based on the value function, it will be further passed to the policy as well.

[3] For clarity, we did not draw discount factor γ in the figure.

8.3 Hypotheses Verification

Based on the aforementioned unexpected results and the analysis, we make four hypotheses:

Hypothesis 1 *If observation delay caused the unexpected results in Section 8.1, then similar results should be reproducible on benchmark tasks with various degrees of observation delay.*

Hypothesis 2 *If action transformation caused the unexpected results in Section 8.1, then similar results should be reproducible on benchmark tasks with different types of action transformation.*

Hypothesis 3 *If partial observability caused the unexpected results in Section 8.1, then similar results should be reproducible on MDP vs. POMDP versions of benchmark tasks.*

Hypothesis 4 *The λ-return (Eq. 3.22) based on n-step bootstrapping employed in PPO leads to robustness to POMDP, therefore (1) n-step versions of TD3 and SAC with $n > 1$ should also improve robustness to POMDP compared to their vanilla versions, and (2) replacing λ-return with 1-step bootstrapping should cause PPO's performance to decrease when moving from MDP to POMDP.*

To empirically verify the **Hypothesis 3** and **4**, we investigate the performance of the multi-step (also known as n-step) variants of vanilla TD3 and SAC, namely Multi-step TD3 (MTD3) [269] and Multi-step SAC (MSAC) [20], on various tasks. Specifically, instead of sampling a mini-batch $\left\{(o_t, a_t, r_t, o_{t+1})^{(k)}\right\}_{k=1}^{K}$ of K 1-step experiences, we sample a mini-batch $\left\{(o_t, a_t, r_t, o_{t+1}, \cdots, o_{t+n-1}, a_{t+n-1}, r_{t+n-1}, o_{t+n})^{(k)}\right\}_{k=1}^{K}$ of K n-step experiences. Then, we replace the target Q-value calculation defined in Eq. 3.15 and Eq. 3.18 with the n-step bootstrapping defined as $\hat{Q}^{(n)}(o_t, a_t) = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_t + \gamma^n \left[\min_{i=1,2} Q_{\theta_i^-}(o_{t+n}, a) + \alpha H(\pi(\cdot|o_{t+n}))\right]$, where $a = \mu_{\phi^-}(o_{t+n})$ and $a \sim \pi_{\psi^-}(a|o_{t+n})$ for MTD3 and MSAC, respectively, $\alpha = 0$ for MTD3, and the policy update is the same as that for TD3 and SAC.

8.4 Experiments

Fig. 8.4 shows the benchmark MuJoCo tasks (Ant-v2, HalfCheetah-v2, Hopper-v2, Walker2d-v2) from OpenAI Gym that we will use to verify hypotheses 1-4. We investigate three types

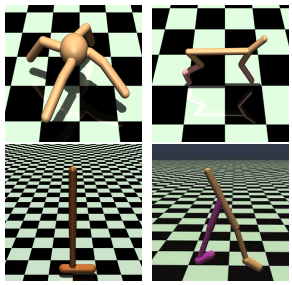


Figure 8.4: Benchmark Tasks.

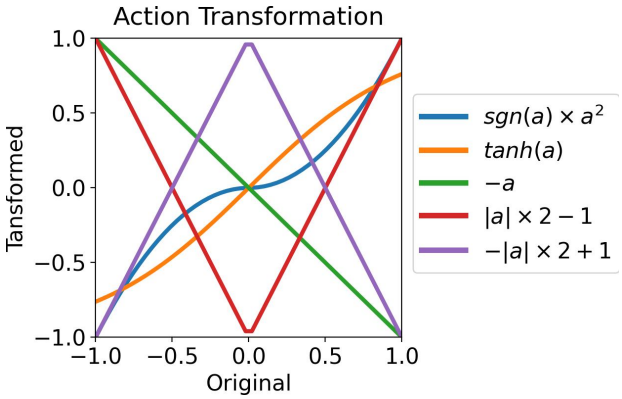
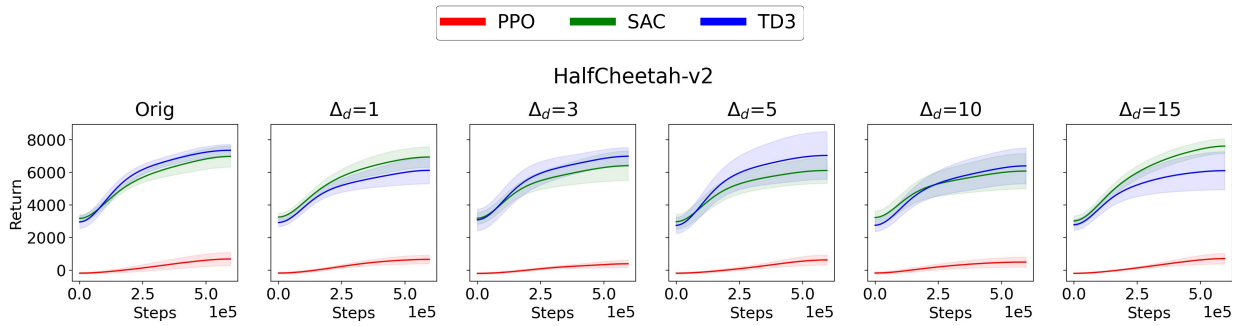


Figure 8.5: Action Transformation Functions

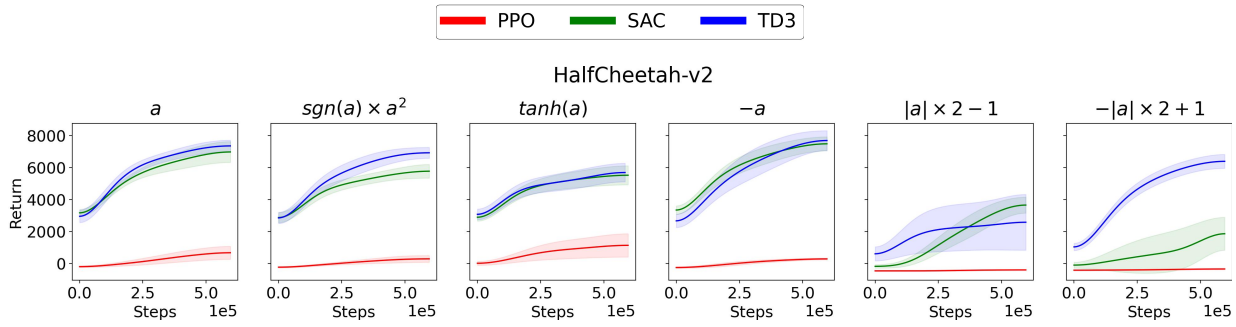
Table 8.1: MDP- and POMDP-version of Benchmark Tasks

Name	**Description**	**Hyper-param**
MDP	Original task	–
POMDP-RV	Remove all velocity-related entries in the observation space.	–
POMDP-FLK	Reset the whole observation to 0 with probability p_{flk}.	$p_{flk} = 0.2$
POMDP-RN	Add random noise $\epsilon \sim \mathbb{N}(0, \sigma_{rn})$ to each entry of the observation.	$\sigma_{rn} = 0.1$
POMDP-RSM	Reset an entry of the observation to 0 with probability p_{rsm}.	$p_{rsm} = 0.1$

of potential causality, namely observation delay, action transformation, and partial observability. For observation delay, we assume at time step t the agent can only receive the observation $o_{t-\Delta_d}$, that is Δ_d step delayed, and for $t < \Delta_d$ the o_0 is used. Fig. 8.5 shows the 5 action transformations investigated, where the original action given by an agent is first transformed accordingly before execution, and $sgn(\cdot)$ indicates the sign of a value. Table 8.1 shows the POMDP-versions of the original tasks proposed in [182]. Note that the POMDP-version tasks only transform the observation space of the original task, but leave the reward signal unchanged, which means the reward signal is still based on the original observations. This enables fair comparison among the performances of an agent



(a) Observation Delay



(b) Action Transformation

Figure 8.6: Results on Benchmark Tasks with Action Transformation and Observation Delay

on MDP and POMDP.

The neural network structures and hyper-parameters of PPO, TD3, SAC, MTD3(n), MSAC(n) are the same as the implementation in OpenAI Spinning Up (https://spinningup.openai.com), and the source code of this paper can be found in https://github.com/LinghengMeng/m_rl_pomdp.[4] The n in the bracket indicates the step size in multi-step bootstrapping. LSTM-TD3(l) [182] is also compared to see how the MTD3(n) and MSAC(n) perform compared to an algorithm specifically designed for dealing with POMDP, where l indicates the memory length. The results shown in this section are based on three random seeds. For better visualization, learning curves are smoothed by 1-D Gaussian filter with $\sigma = 20$.

[4]Additional implementation details and complementary results can be found in https://arxiv.org/pdf/2209.04999.pdf.

Table 8.2: The Maximum of Average Return over 5 evaluation episodes within 2 million steps based on 3 different random seeds. If TD3 or SAC perform worse than PPO, they are gray-colored. If MTD3(5) or MSAC(5) outperform the corresponding TD3 or SAC, they are red-colored. The maximum value of all evaluated algorithms for each task is bolded.

Task		Algorithms					
Name	Version	**PPO**	**TD3**	**SAC**	**MTD3(5)**	**MSAC(5)**	**LSTM-TD3(5)**
Ant	MDP	1315.61	5976.49	**6106.42**	4174.73	5474.10	4745.36
	POMDP-FLK	1087.93	1339.88	972.37	2154.18	**4205.45**	3420.69
	POMDP-RN	587.87	1684.48	1431.37	1315.31	**3185.56**	1130.96
	POMDP-RSM	836.34	1737.50	931.68	2819.81	**4204.35**	1459.67
	POMDP-RV	3412.95	1870.12	1102.99	3123.41	**4160.47**	1958.36
HalfCheetah	MDP	3770.88	11345.21	**11887.53**	7001.51	8694.46	10086.52
	POMDP-FLK	2183.27	1377.18	248.49	1289.14	**4803.40**	1678.92
	POMDP-RN	3975.56	5306.12	4651.56	5503.61	**5865.68**	4395.61
	POMDP-RSM	3338.03	1395.06	123.70	3314.96	**5847.08**	1467.68
	POMDP-RV	4120.39	2937.80	498.51	3955.83	4017.05	**4406.33**
Hopper	MDP	3604.01	3823.88	3993.51	3700.19	**4065.26**	3677.02
	POMDP-FLK	2657.15	1043.63	1047.01	1219.17	1048.73	**3587.02**
	POMDP-RN	**3469.32**	2125.03	1026.52	3022.95	3318.88	3426.28
	POMDP-RSM	3107.39	2506.26	1003.19	**3187.08**	3184.48	1169.30
	POMDP-RV	**2613.43**	1023.38	1119.84	1000.27	1144.11	592.39
Walker2d	MDP	4230.38	5762.66	6097.24	**7181.57**	5615.33	5189.87
	POMDP-FLK	2723.41	999.17	1003.27	1243.08	1412.80	**4219.05**
	POMDP-RN	4160.27	1220.65	1060.09	3959.90	3977.56	**4191.02**
	POMDP-RSM	3295.34	2178.42	2009.92	4197.03	**4778.27**	4083.16
	POMDP-RV	3531.14	1443.01	2199.88	2670.89	3397.12	**4356.57**

8.4.1 Results on Benchmark Tasks with Observation Delay and Action Transformation

Fig. 8.6 shows the results of PPO, TD3 and SAC on benchmark tasks with observation delay (Fig. 8.6a) and action transformation (Fig. 8.6b), where the first column corresponds to the original task. From this figure, we can see that no tested observation delay or action transformation causes PPO to outperform TD3 and/or SAC. These results reject **Hypotheses 1** and **2**.

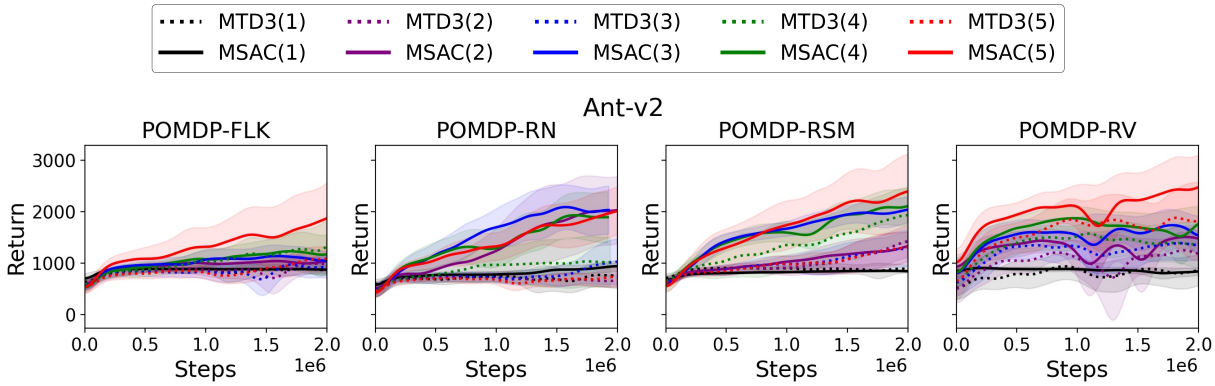


Figure 8.7: Effect of Multi-step Size on The Performance of MTD3 and MSAC, where the average learning curves correspond to MTD3(n) and MSAC(n) with different multi-step sizes n and the shaded area shows half of standard deviation of the average accumulated return over 3 random seeds.

8.4.2 Results on Benchmark Tasks with Partial Observability

Table 8.2 shows the maximum of average return of PPO, TD3, SAC, MTD(5), MSAC(5) and LSTM-TD3(5) calculated over 5 episodes. Firstly, to verify **Hypothesis 3** and comparing PPO, TD3 and SAC, we found that: **(1)** TD3 and SAC significantly outperform PPO on all MDP tasks; **(2)** TD3 and SAC are much worse than PPO on most POMDP tasks. In addition, **(3)** TD3 and SAC experience dramatic drops in performance when moving from MDP to POMDP, while PPO does not experience too much change in performance. These observations match perfectly the unexpected results described in Section 8.1, indicating that the problem comes from the partial observability.

Interestingly, in some HalfCheetah (POMDP-RV and POMDP-RN) and Ant (POMDP-RV) environments, PPO performs better than in the standard MDP environment. While in case of Cheetah the difference is small and it could be argued that it is purely statistical noise and is thus insignificant, in Ant the difference is really large. From the hypothesis that PPO could incorporate temporal information, a potential explanation is that velocity is indirectly incorporated and probably additionally adding it to the observation can only make the observation space larger and introduce difficulty in finding an optimal policy.

When we compare TD3 and SAC to their multi-step versions MTD3(5) and MSAC(5) on POMDPs in Table 8.2, it can be seen that MTD3(5) and MSAC(5) outperform their vanilla versions on most POMDPs (highlighted in red in Table 8.2), which verifies **Hypothesis 4 (1)**. Even though MTD3(5) and MSAC(5) show performance increase compared

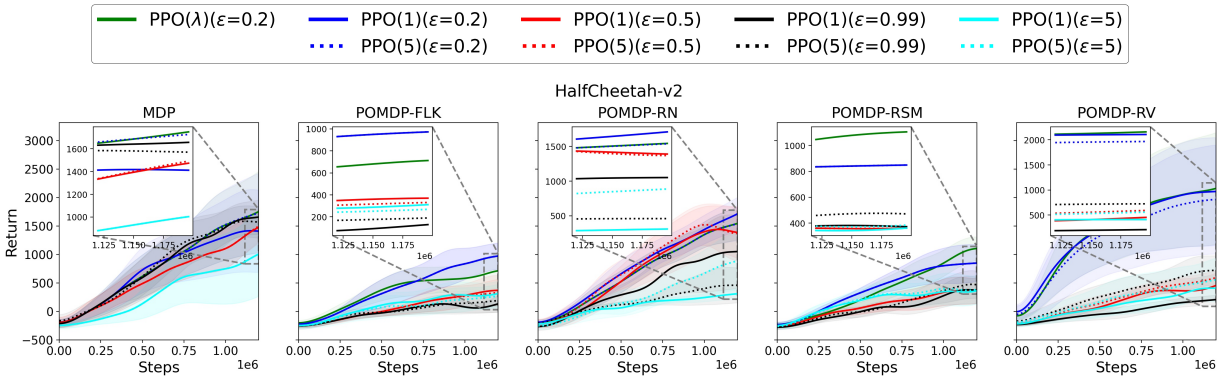


Figure 8.8: Effect of Multi-step Size on The Performance of PPO, where the learning curves correspond to PPO(λ) with λ-return and PPO(n) with simple n-step bootstrapping (the shaded area shows half of standard deviation of the average accumulated return over 3 random seeds). The ϵ indicates how far away the new policy is allowed to go from the current one.

to TD3 and SAC and comparable or better performance than PPO on POMDPs, LSTM-TD3(5) exhibits dramatically better performance on some tasks, e.g. POMDP-FLK of Walker2d-v2 and Hopper-v2. This indicates that for some cases directly learning a good representation of the underlying state from a short experience trajectory is more effective than relying on multi-step bootstrapping to pass some temporal information. Fig. 8.7 shows the average learning curves of MTD3(n) and MSAC(n) with different multi-step sizes $n = \{1, 2, 3, 4, 5\}$, when $n = 1$ they reduce to TD3 and SAC. It can be seen from Fig. 8.7 that simply increasing n by a few steps makes their performance dramatically better than $n = 1$ with a little extra computation cost. For the n we tested, $n = 5$ shows the best performance on most tasks.

Fig. 8.8 compares PPO(λ) with λ-return and PPO(n) with simple n-step bootstrapping. From this figure, we observe that when λ-return is replaced with 1-step bootstrapping, the performance of PPO does not change much, which rejects **Hypothesis 4 (2)**. Compared to the return estimation, when the clip ratio ϵ is increased, PPO's performance experiences a significant decrease. In summary, the results shown in this section support **Hypothesis 3**, but for **Hypothesis 4** only the first part is supported by the results.

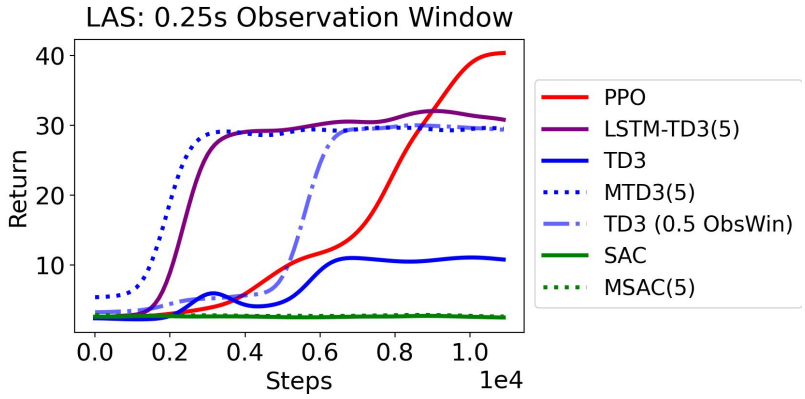


Figure 8.9: Revisit LAS Results

8.4.3 Revisit LAS

After observing similar relative performance on the POMDP-versions of the benchmark tasks, we revisit the LAS task and test the MTD3, MSAC and LSTM-TD3 on LAS as well as TD3 on LAS with longer observation windows. As shown in Fig. 8.9, TD3 on LAS with longer observation window achieves better performance compared to TD3 with shorter observation window. In addition, both MTD3(5) and LSTM-TD3(5) achieve better performance than vanilla TD3. Unfortunately, similar to SAC, both MSAC(5) and SAC on LAS with longer observation window (not plotted because it overlaps with the line of SAC) failed. If only looking at the results of TD3, we may conclude that on LAS, TD3 under-performs PPO because of partial observability as we are able to reproduce the similar results on standard benchmark tasks and to increase the performance of vanilla TD3 by techniques used to tackle POMDP. However, this is not applicable to SAC. There may be differences between TD3 and SAC that leads to SAC's failure on LAS, which is beyond the scope of this work and will be left for the future. For instance, in terms of exploration strategy, TD3 adds fixed action noise to explore, while SAC always encourages broader exploration because of the maximizing policy entropy in its policy update. On the contrary, PPO is more conservative than TD3 and SAC, as it constrains the policy change. This may also explain why MTD3(5), LSTM-TD3(5), and TD3 with longer observation window learn faster at the initial stage but achieves worse final performance than PPO.

8.5 Summary

In this chapter, we first highlight the counter-intuitive observation found when applying DRLs to a novel complex robot, that PPO outperforms TD3 and SAC. We hypothesize that this degradation in performance is potentially caused by observation delay, action transformation, or partial observability. Then, we provide a potential explanation about why the multi-step bootstrapping employed in PPO makes it more robust to partial observability compared to TD3 and SAC, which only rely on 1-step bootstrapping. Based on the hypotheses on partial observability, we use multi-step versions of vanilla TD3 and SAC, i.e., MTD3 and MSAC, to verify our hypotheses on MDP- and POMDP-version of benchmark tasks. Even though the hypotheses on observation delay and action transformation are rejected on the benchmark tasks, the same counter-intuitive observation can be reproduced on the POMDP-versions of the benchmark tasks. The results of MTD3 and MSAC with multi-step size $n = 5$ show that simply increasing the step size from $n = 1$ to $n = 5$ can significantly increase the performance of vanilla TD3 and SAC on POMDPs. After that, we revisit the LAS task and find that TD3 with longer observation window, MTD3(5) and LSTM-TD3 are able to achieve better performance than vanilla TD3, but MSAC(5) still fails. We provide a potential explanation that this may be related to the exploration strategies employed by different DRL algorithms.

Chapter 9

Engaging Behavior Generation from Human Preferences In A Large Scale Interactive System: A Simulation Experiment

This chapter proposes to use Preference Learning (PL) to learn the engagement estimate from human preferences and to avoid the disadvantages of manually designed engagement estimates, as introduced in Chapter 1. By combining PL and Reinforcement Learning (RL), more flexibility and adaptability is introduced in the approach to autonomous engaging behavior generation for crowd and long-term interaction. Particularly, in chapter 5, we showed that the behavior of an interactive system, controlled by a learning agent acting in parameterized action space, outperforms pre-scripted behavior choreographed by expert architects, by maximizing a manually designed reward function estimating passive and active engagement. However, the manually designed reward function relies on designers' expertise, needs to be manually adapted to different sensors, and might lead to undesired behavior if the designed reward function is not correctly formulated [82].

To reduce the reliance on manually designed reward functions, techniques such as Inverse Reinforcement Learning [1, 203, 88] and Human Preference learning [7, 292, 64] have been proposed. This chapter is inspired by [64], where a reward function is learned from human preferences over two video clips of an agent's past behavior. As shown in [64], novel complex behaviors can be learned with the learned reward function elicited from human preferences, which is hard to manually design by engineers and architects. In addition,

for applications where users' preferences might change over time, continuously eliciting a reward function from human preferences is necessary. Therefore, interactively learning a reward function is more appealing than using a fixed reward function.

Different from the continuous control tasks investigated in [64], in this thesis we investigate whether a reward function could be learned from the preferences provided by users so that engaging interactive behavior could be learned from this reward function. The testbed is a non-anthropomorphic robot aiming to engage its occupants, whose interaction might be affected by personal and social factors, e.g. aesthetic preferences, social distancing, different backgrounds and personalities, etc., which means the external environment of the robot is highly non-deterministic.

In this chapter, we will describe our proposed approach to investigate whether a reward function learned from human preferences over short video segments can be used to learn a policy to maximise interaction engagement.

9.1 Methodology

We propose to combine Preference Learning (PL) with Reinforcement Learning (RL), called PL+RL, to learn engaging behavior from users' preferences, by replacing the hand-crafted reward function with a reward function induced from user preferences. In this section, we will first introduce the overall framework of PL+RL and then elaborate the components in detail.

9.1.1 Overall Framework

The overall framework is shown in Fig. 9.1, which is an implementation of the simplified version of PL+RL in Fig. 3.5. The interactive system is the simulated Living Architecture System (LAS) named *Meander* introduced in Section 4.3. The RL-based agent *LAS-Agent* interacts with its internal environment *LAS-Intl-Env* which directly interacts with its external environment, i.e., *Meander*, simulated by LAS-Behavior-Engine (LAS-BE) (described in Section 4.2.1). Within *LAS-Agent*, a reward function learned by Preference Learning can be used to replace the hand-crafted reward function. To enable preference learning, segments are generated from trajectories by the *Segment Generation* module, and the *User Preference Interface* is used to collect human preferences which will be used to induce the underlying reward function through Preference Learning.

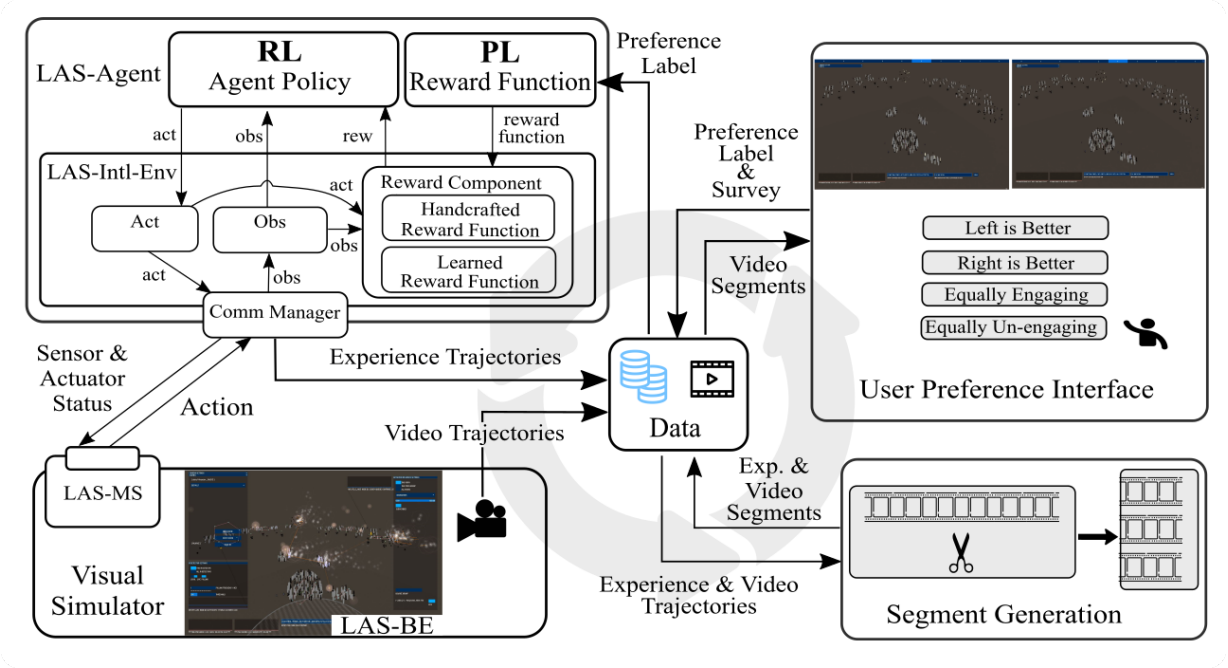


Figure 9.1: Overall Framework of PL+RL Setting

9.1.2 Preference Learning of Reward Function

In this section we will introduce the preference learning (PL) of the reward function R^{pb}, built on top of the method [64] introduced in section 3.4. The basic idea of PL is to show a user a pair of video clips of the interaction between an agent and its environment and ask for feedback on which one is preferred. With these collected preference labels, PL learns a reward function which then drives the policy of the agent. This process involves trajectory generation, segment generation, segment pair sampling, preference query scheduling, and reward function learning. In this section, we will focus on introducing reward function learning and leave the rest to the appendix.

Preference-based Reward Function Learning

According to [64], given an experience (o_t, a_t, o_{t+1}, d_t) where an agent takes action a_t in observation o_t then observes a new observation o_{t+1}, and terminates the trajectory or not, indicated by d_t, the preference-based reward r_t can be defined as

$$r_t = R^{pb}\left(o_t, a_t, o_{t+1} \mid \theta_{pbr}\right), \tag{9.1}$$

where R^{pb} is the preference-based reward model parameterized by θ_{pbr}

Formally, we use $\sigma = (\sigma_e, \sigma_v)$ to represent a segment composed of experience segment σ_e and video segment σ_v that are sampled from the trajectory $l = (l_e, l_v)$ composed of experience trajectory l_e and video trajectory l_v collected during the interaction of an agent with its environment. When two segments σ^0 and σ^1 are sampled for preference labelling, the corresponding video segments are shown to the preference teacher to ask for preference label $y \in \{0, 1, -1\}$, where $y = 0$ or 1 indicates either segment 0 or 1 is preferred and $y = -1$ indicates the two segments are equally engaging or unengaging, while the experience segments are used to train the preference-based reward model. Given a preference data point $(\sigma^0 = (\sigma_e^0, \sigma_v^0), \sigma^1 = (\sigma_e^1, \sigma_v^1), y)$, the preference prediction can be calculated following

$$\hat{P}\left[\sigma^0 \succ \sigma^1\right] = \hat{P}\left[\sigma_e^0 \succ \sigma_e^1\right] = \frac{e^{\sum_{i=0}^{K-1} R^{pb}\left(o_{t+i}^0, a_{t+i}^0, o_{t+i+1}^0 | \theta_{pbr}\right)}}{e^{\sum_{i=0}^{K-1} R^{pb}\left(o_{t+i}^0, a_{t+i}^0, o_{t+i+1}^0 | \theta_{pbr}\right)} + e^{\sum_{i=0}^{K-1} R^{pb}\left(o_{t+i}^1, a_{t+i}^1, o_{t+i+1}^1 | \theta_{pbr}\right)}}, \tag{9.2}$$

where the experience segment $\sigma_e^{0/1} = \left\langle \left(o_{t+k-1}^{0/1}, a_{t+k-1}, r_{t+k-1}^{0/1}, o_{t+k}^{0/1}\right) \right\rangle_{k=1}^{K}$ are the K consecutive experiences of segment 0 or 1 starting from time step t to $t + K - 1$.

Given a preference training dataset $D_{pl}^{train} = \left\{(\sigma^0 = (\sigma_e^0, \sigma_v^0), \sigma^1 = (\sigma_e^1, \sigma_v^1), y)^n\right\}_{n=1}^{\left|D_{pl}^{train}\right|}$ with $\left|D_{pl}^{train}\right|$ data points, the reward function R^{pb} is updated by minimizing the cross-entropy loss between the true and the predicted preference label with respect to the parameters θ_{pbr} of the reward function, as follows:

$$\min_{\theta_{pbr}} L = -\sum_{(\sigma^0, \sigma^1, y) \in D} (1 - y)\hat{P}\left[\sigma_e^0 \succ \sigma_e^1\right] + y\hat{P}\left[\sigma_e^0 \prec \sigma_e^1\right], \tag{9.3}$$

where $\hat{P}\left[\sigma_e^0 \prec \sigma_e^1\right] = 1 - \hat{P}\left[\sigma_e^0 \succ \sigma_e^1\right]$ and $D = \left\{(\sigma^0, \sigma^1, y) \mid (\sigma^0, \sigma^1, y) \in D_{pl}^{train} \text{ and } y \neq -1\right\} \subseteq D_{pl}^{train}$ that excludes the equally rated cased, i.e., $y = -1$.

Each time a new set of preference labels is received, R^{pb} will be fitted to the whole training dataset D_{pl}^{train} following the pseudo-code shown in Alg. 5 in Appendix D. After that, the new R^{pb} will be used to drive the learning of an agent before the next new set of preference labels is collected.

9.1.3 Policy Learning from Preference-based Reward Function

The goal of PL is to infer a reward function that drives the policy learning in RL to generate the behavior matching the human preference. A key challenge for any RL algorithm

used in the PL+RL setting is the non-stationary reward function problem. As both the reward and the policy are being learned simultaneously, the RL algorithm faces a changing reward function during learning. Therefore, to get a comprehensive comparison of the performance of DRL, both off-policy and on-policy RL algorithms are investigated in this chapter. Off-policy RL algorithms TD3 [89], SAC [106], and LSTM-TD3 [185] are more data efficient than on-policy algorithms, because they can reuse the experiences in a replay buffer. However, on-policy RL algorithms such as PPO [241] may be more stable due to its limitation on the step size to avoid performance collapse. Another key difference between on- and off- policy algorithms is that the latter needs to recalculate the reward during training on the experiences from replay buffer.

Using PB-Rew in Off-policy RL

The off-policy RL algorithms investigated in this thesis are TD3, SAC, and LSTM-TD3 which are introduced in Section 3.3 and Section 7.1. Different from the standard RL setting, in PL+RL during the training the reward r_t in an experience $(o_t, a_t, r_t, o_{t+1}, d_{t+1})$ sampled from the replay buffer D needs to be recalculated before being used for calculating the bootstrapping Q-value, because the reward r_t is changed along with the reward function during PL. Formally, assume the current PB-Rew function is R^{pb} and for an experience $(o_t, a_t, r_t, o_{t+1}, d_{t+1})$, the target Q-value $\hat{Q}(o_t, a_t)$ of (o_t, a_t) for TD3, SAC and LSTM-TD3 can be calculated as Eq. 9.4, Eq. 9.5 and Eq. 9.6 by replacing the r_t in Eq. 3.15, Eq. 3.18 and Eq. 7.5 with $R^{pb}(o_t, a_t, o_{t+1})$ as follows:

$$\hat{Q}(o_t, a_t) = R^{pb}(o_t, a_t, o_{t+1}) + \gamma(1 - d_t)\min_{j=1,2} Q_j^-(o_{t+1}, a^-), \tag{9.4}$$

$$\hat{Q}(o_t, a_t) = R^{pb}(o_t, a_t, o_{t+1}) + \gamma \left[\min_{i=1,2} Q_{\theta_i^-}\left(o_{t+1}, a^-\right) + \alpha H(\pi(\cdot|o_{t+1})) \right], \tag{9.5}$$

$$\hat{Q}(o_t, a_t) = R^{pb}(o_t, a_t, o_{t+1}) + \gamma * (1 - d_t) * \min_{j=1,2} Q_j^-(o_{t+1}, a^-, h_{t+1}^l). \tag{9.6}$$

Using PB-Rew in On-policy RL

The on-policy RL algorithm investigated in this thesis is PPO. Using PB-Rew in PPO is implemented simply by replacing the HC-Rew with PB-Rew.

9.2 Individual Preference vs. Aggregate Preference

Preference Learning can be applied to either *individual preference* or *aggregate preference*. Individual preference is more applicable to domains where individuals are distinct and personalized responses are more desirable. In such cases, preference labels should be collected from each individual and individual reward functions estimated from each individual separately. Aggregate preference is more applicable to fields where a group of people share very similar preferences, and appropriate responses to this entire group are to be learned. In this case, preference labels collected from the group can be used together, reducing the demand on each individual to provide preference labels. With these considerations, in the user study conducted in this chapter, we choose the aggregate preference in order to provide more preference labels to a preference-based reward model and reduce the demand of preference labels from each user. As a result, this choice introduces a key assumption that the group of users share similar preferences. This assumption makes sense in the application of LAS, considering that the pre-scripted behaviors are normally designed by experts who assume their design of the behaviors can engage most visitors, i.e., satisfying the aggregate preference of the visitors. Therefore, we believe this assumption is appropriate, even though we believe the visitors' engagement may be enhanced by also considering individual preference.

9.3 Experiment Settings

In this section, we introduce the control task, three types of preference teachers, and the preference selection procedure for human preference teachers. More details about the experiment settings and implementation for each condition can be found in Appendix D.3 and D.4.

9.3.1 Control Task Description

The task is to generate engaging behavior in the simulated testbed *Meander* introduced in section 4.3 Chapter 4. The simulation is realized by employing the LAS Simulation Toolkit (LAS-Sim-Tkt) introduced in section 4.2 Chapter 4. The composition of the simulated *Meander* is the same as that in the physical sculpture.

Observation Space

The observation space O is composed of exteroception, i.e., the readings from the sensors that are mainly used to sense the external environment of *Meander*, and proprioception, i.e., the status of the actuators that represents the internal environment of *Meander*. In practice, at a specific time step the observation is constructed from the retrieved N_{sa} status of all sensors and actuators embedded in *Meander* at the frequency $f_o = \frac{N_{sa}}{T_{ow}}$ within a specific time window T_{ow} by either averaging or concatenating the processed data, which is done by the Observation Construction Component (OCC) in the LAS-Agent-Internal-Environment (LAS-Intl-Env) introduced in Section 11. Concretely, except OS[1], the status of all devices listed in Table 4.6 are included in the observation space in order to maximize the agent's perceptual capacity. Table 9.1 summarizes the composition of the observation space, where $f_o = 1$, $T_{ow} = 1$, and the original value range of each device is converted to range $[0, 1]$ according to Eq. 4.4-4.8. To summarize, for $f_o = 1$ and $T_{ow} = 1$[2], the dimension of the observation space is 724, where 124 dimensions are exteroception and 600 dimensions are proprioception. If using concatenating method, the dimension of the observation space will be $724 \times f_o \times T_{ow}$, while if using averaging method, it will be always 724. In this thesis, if not specified, we use $f_o = 1$ and $T_{ow} = 1$.

Table 9.1: Observation Space Composition

	Exteroception (Sensors)			**Proprioception (Actuators)**				
Device Type	IR	GE	SD	MO	RS	DR	PC	SM
Orig Value Range	$[0, 750]$	$[0, 2]$	$[0, 1024]$	$[0, 1]$				
Obs Value Range	$[0, 1]$							
Device Number	13	19	16	261	151	60	29	39
Each Device Data Dimension	1	5	1	1	1	2	1	1
Each Device Type Dimension	13	95	16	261	151	120	29	39
Each Obs Type Dimension	124			600				
Obs Dimension	724							

[1]Because OS is controlled by a 3rd party controller, called 4D Sound, which is not used in the simulated experiment, so we exclude it from the proprioception.

[2]When $f_o = 1$ and $T_{ow} = 1$, the results of concatenating and averaging method are the same, because there only one set of status of the sensors and actuators.

Table 9.2: Parameterized Action Space

Influence Map	ParamName	Pre-scripted Value	Orig Value Range	Act Value Range	Count
Excitor	excitorSize	40	$[40, 2000]$	$[-1, 1]$	9
	excitorCoreSize	0.95	$[0, 1]$		
	excitorLifespan	20000	$[500, 20000]$		
	excitorMasterIntensity	1	$[0, 1]$		
	excitorSpeedLimit	0.6	$[0, 1]$		
	attractorAngleSpeed	0.2	$[0, 0.25]$		
	attractorForceScalar	1	$[0, 5]$		
	bgHowOften	250	$[250, 1000]$		
	maxExcitorAmount	16	$[1, 35]$		

Action Space

The action space A is parameterized with the parameters within LAS-BE to leverage the expert knowledge. As introduced in Table **??** Section 4.3, there are 31 parameters of the influence maps defined in LAS-BE involved in the pre-scripted behavior of *Meander*. Even though having all of them can maximize the flexibility and the diversity of the action space, it will dramatically increase the exploration space for the agent's policy leading to the demand for a very large number of interaction samples which is undesirable to applications in HRI. Therefore, we need to balance the trade-off between having a smaller action space and maximizing the flexibility and the diversity of the possible behavior. By analyzing the effect of the parameters in Table **??**, we decided to only keep the 9 parameters related to the Excitor influence map as listed in Table 9.2 where the Orig Min and Orig Max indicate the minimum and maximum value in the original value range and the Act Min and Act Max indicate the minimum and maximum value in the action space. The Excitor influence map can generate very active behavior by having larger action values and generate very calm behavior by having small action values. Each dimension $a^{(i)}$ of the action $a \in \mathbb{R}^9$ is in $[-1, 1]$, which will be converted to its original value range by the Action Execution Component (AEC) in the LAS-Agent-Internal-Environmen (LAS-Intl-Env) during execution according to Eq. 4.3 in Section 4.2.3.

Reward Function

Two types of reward function R are investigated to generate engaging behavior. We compare a Hand-Crafted Reward (HC-Rew) function with three types of Preference-based Reward (PB-Rew): (1) a reward inferred from simulated preference based on HC-Rew, (2) a reward induced from human preferences that try to match the HC-Rew, and (3) a reward inferred from expert preference teachers who indicate their preference based on their own understanding about what behavior is more engaging.

Hand-crafted Reward (HC-Rew) Function The Hand-crafted Reward (HC-Rew), R^{hc}, is defined to encourage active behavior in *Meander*, which assumes active is more engaging to visitors. Formally, given an observation $o \doteq o_a \bigcup o_s$ concatenating the exteroception o_s and the proprioception o_a, the reward is defined as

$$r = R^{hc}(o) \doteq \frac{1}{|o_a|} \sum_{j=1}^{|o_a|} o_a^{(j)}, \tag{9.7}$$

where $|o_a|$ is the dimensionality of o_a.

Preference-based Reward (PB-Rew) Function Preference-based Reward (PB-Rew) function, indicated as R^{pb}, refers to the reward function derived from the human preferences. Unlike HC-Rew that is stationary during the learning of a policy in RL, PB-Rew is inevitably non-stationary. First, PB-Rew is non-stationary because the preference labels are collected online. Second, the underlying function that determines human preference[3] may change due to various reasons, such as the preference teachers may be a different group of people, or the teachers changed their mind during the course of preference providing.

Control Task Summary

Table 9.2 briefly summarizes the dimension and the value range of the action and observation space and the reward function of the task investigated, where the observation frequency $f_o = 1$ and observation window size $T_{ow} = 1$. Given the control task, the goal of an agent is to maximize the accumulated reward, either hand-crafted or preference-based, by observing the environment and taking an action according to the learned policy.

[3]Note that when we say human preference, without specification, we assume the preference labels are provided by a group people rather than a single person.

Figure 9.2: Control Task Summary

	Act Space	Obs Space	Reward Function	
			HC-Rew	PB-Rew
Dimension	9	724	1	1
Value Range	[-1,1]	[0,1]	[0,1]	[-1,1]

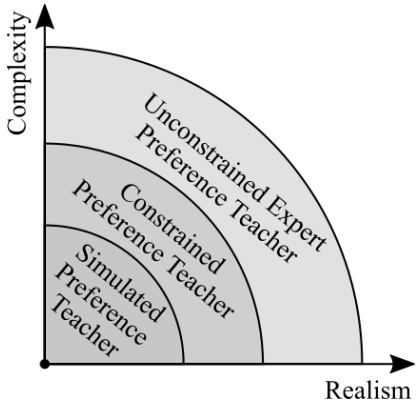


Figure 9.3: Hierarchy of Preference Teacher

9.3.2 Preference Teacher

In this thesis, three types of preference teacher will be investigated, as shown in Fig. 9.3.

Simulated Preference Teacher For the simulated preference teacher, there is no human user, so there is no need to show video segments and the web-based interface is not engaged. Formally, assume a segment pair $(\sigma^0 = (\sigma_e^0, \sigma_v^0), \sigma^1 = (\sigma_e^1, \sigma_v^1))$ is sampled for preference labeling and by definition $\sigma_e^0 = \left\langle \left(o_{t+k-1}^0, a_{t+k-1}^0, r_{t+k-1}^0, o_{t+k}^0, d_{t+k-1}^0\right)\right\rangle_{k=1}^K$ and $\sigma_e^1 = \left\langle \left(o_{t+k-1}^1, a_{t+k-1}^1, r_{t+k-1}^1, o_{t+k}^1, d_{t+k-1}^1\right)\right\rangle_{k=1}^K$, we define the simulated preference index as the average of hand-crafted reward introduced in 9.3.1 over a segment:

$$\rho^0 = \frac{1}{K}\sum_{k=1}^{K} R^{hc}(o_{t+k-1}^0) \quad \text{and} \quad \rho^1 = \frac{1}{K}\sum_{k=1}^{K} R^{hc}(o_{t+k-1}^1), \tag{9.8}$$

where $K = |\sigma_e|$ is the number of experiences within the experience segment σ_e of a segment σ, and the average operation is to make a fair comparison when $|\sigma_e^0| \neq |\sigma_e^1|$. With the simulated preference index, the simulated preference label can be calculated as:

$$y \doteq \begin{cases} 0, & \text{if } \rho^0 > \rho^1 \\ 1, & \text{if } \rho^0 < \rho^1 \\ -1, & \text{if } \rho^0 = \rho^1. \end{cases} \tag{9.9}$$

In Eq. 9.9, a perfect simulated preference teacher is presented. A non-perfect simulated preference teacher, also called ϵ-irrational preference teacher, can be realized by randomly

sampling a label from $0, 1, -1$ with a small irrational probability ϵ_{ip}. Formally, given a perfect synthetic preference label y generated by Eq. 9.9, and assuming a value p is uniformly sampled from $[0, 1, -1]$, then the new preference label y' given by an ϵ-irrational preference teacher with irrational probability ϵ_{ip} can be defined as

$$y' = \begin{cases} y, & \text{if } p \geq \epsilon_{ip}, \\ U_{\{0,1,-1\}}, & \text{otherwise}, \end{cases} \tag{9.10}$$

where $U_{\{0,1,-1\}}$ indicates a number uniformly sampled from set $\{0, 1, -1\}$.

Constrained Human Preference Teacher Constrained human preference teacher replaces the simulated preference teacher, but he/she is constrained to provide preference labels based on the criteria set by the hand-crafted reward, i.e., constrained to prefer the video segment that is visually more active.

Unconstrained Expert Preference Teacher For real applications, the underlying function of human preference is unknown. The criteria of human preference may be diverse, making it hard to hand-craft a reward function. Therefore, experiments with unconstrained preference teachers are necessary to determine if unconstrained human preference can be transferred to a reward function.

9.3.3 Participating Procedure

Participants are asked to participate in multiple sessions as shown in Fig. 9.4, where each session follows the same procedure for collecting preferences. In each session, we ask a preference teacher to provide at least 20 distinct preference labels following the instructions on the web-based interface. If we use N_c, $N_c^{distinct}$, $N_c^{eq\_eng}$ and $N_c^{eq\_uneng}$ to indicate the number of total, distinct (either left or right is better), equally engaging and equally unengaging preference responses provided by a user in one session, then their relationship is $N_c = N_c^{distinct} + N_c^{eq\_eng} + N_c^{eq\_uneng}$. The user is permitted to move to the next step once $N_c^{distinct} \geq 20$.

9.4 Experiment Results On Simulated and Constrained Human Preference

In this section, we will first show results on hand-crafted rewards and preference-based rewards induced from simulated or constrained human preference labels. Additional results and ablation studies for this section can be found in Appendix D.5.1, D.5.2, and D.5.3.

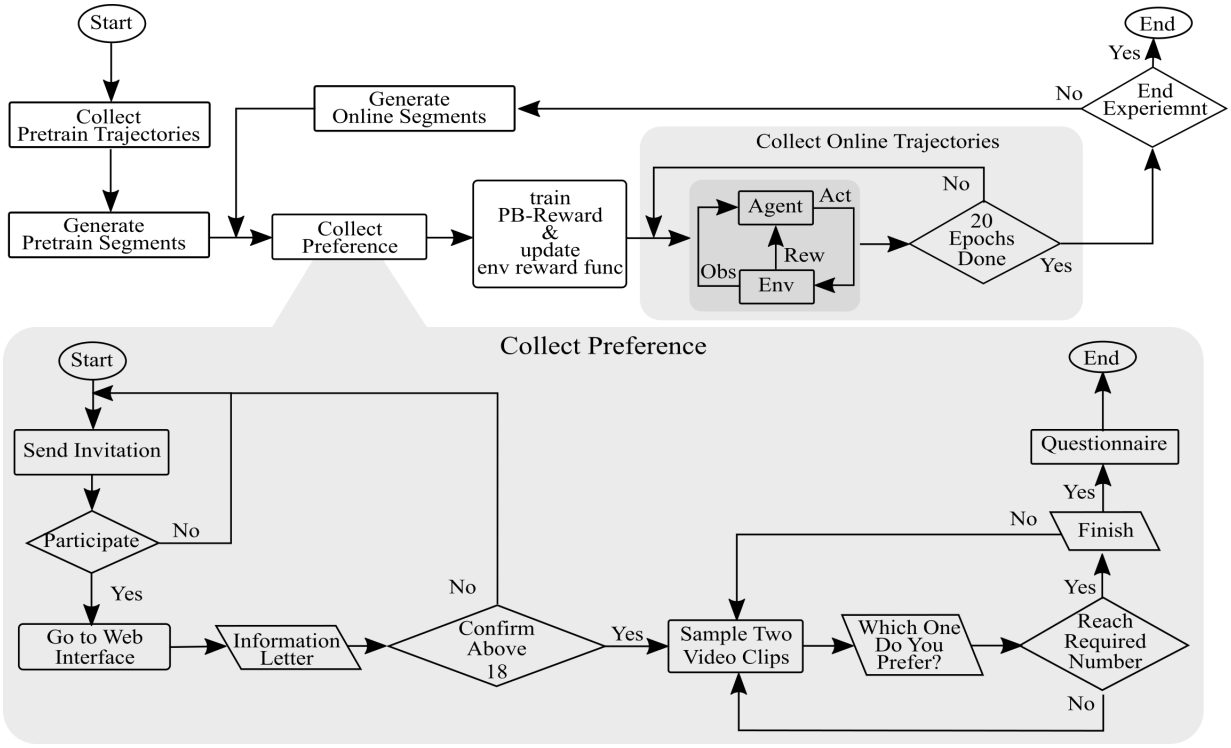
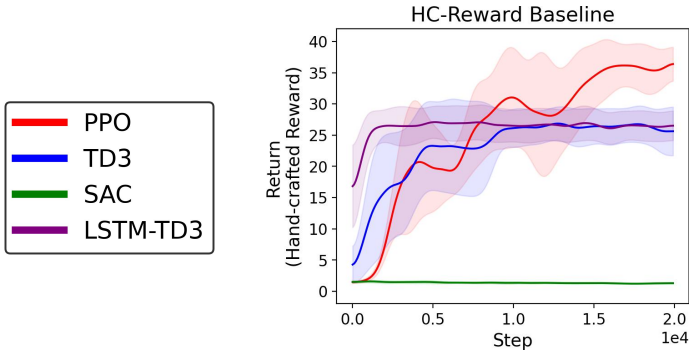


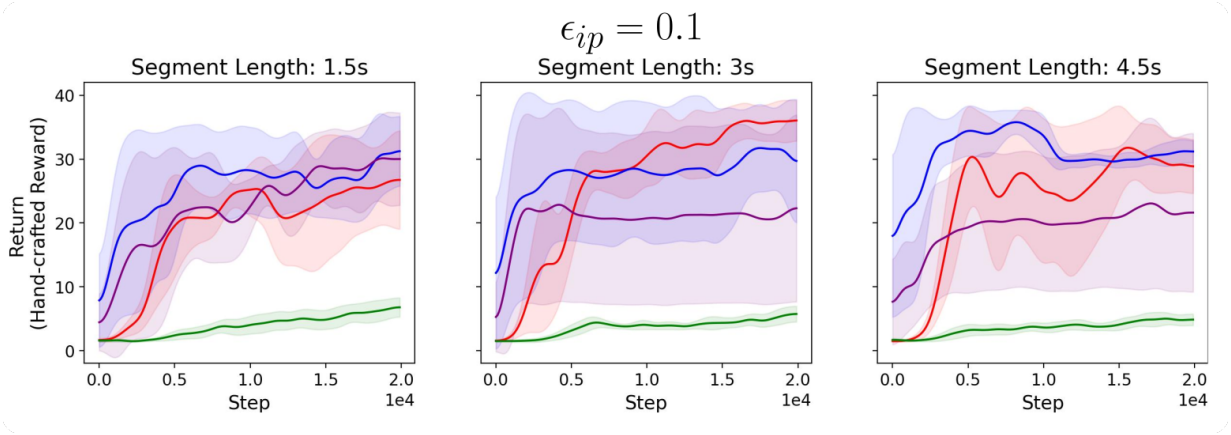
Figure 9.4: Study Participating Procedure

Fig. 9.5 shows the simulated and constrained preference results[4], measured in terms of the hand crafted reward. Fig. 9.5a shows the baseline results with the hand-crafted reward. We can see that TD3 and LSTM-TD3 learn faster than PPO at the beginning, but achieve worse final performance. The first part of this observation is not surprising, but the second part is very uncommon compared to the results reported on MuJoCo tasks [89], where TD3 usually outperforms PPO significantly in terms of final performance. Counter-intuitively, SAC performs worst among the four algorithms. This is very unexpected because on MuJoCo tasks SAC is always the best or comparable to TD3 as reported in [106]. To exclude the implementation bugs, we tested our implementation of the algorithms on MuJoCo tasks, and the results are similar to the results reported in [89, 106]. Therefore, we suspect this is related to the exploration strategy employed by different algorithms. For instance, TD3 adds fixed action noise to explore, while SAC always encourages broader exploration

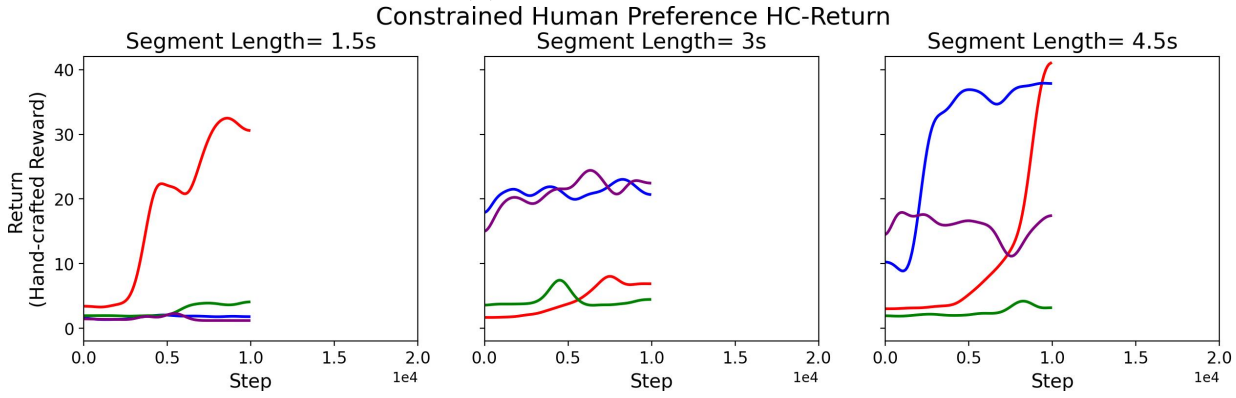
[4]The learning curves reported in this chapter are averaged over three random seeds and smoothed by 1-D Gaussian filter (Gaussian kernel standard deviation $\sigma = 5$). The shaded area corresponds to the standard deviation over the three random seeds.



(a) Hand-crafted Reward



(b) Simulated Preference



(c) Constrained Human Preference

Figure 9.5: Preliminary Results

because of the maximizing policy entropy in its policy update. On the contrary, PPO is more conservative than TD3 and SAC, as it constrains the policy change. This may also explain why LSTM-TD3 and TD3 learn faster initially but achieve worse final performance than PPO.

Fig. 9.5b shows the results with simulated preferences, where the simulated human preference is generated according to Eq. 9.8 - 9.10 with different segment lengths $l = 1.5$s, $l = 3$s, and $l = 4.5$s and irrational probability $\epsilon_{ip} = 0.1$. The performance is measured in hand-crafted reward (HC-Rew). Because the preference-based reward signal that is used by RL agent is induced from simulated preferences and changes between and during different runs, it is more consistent to compare the performance by the hand-crafted reward, which is fixed. We can see that overall the performance of each algorithm trained on PB-Rew with different segment lengths is comparable to their performance on HC-Rew, and for some cases the performance of PB-Rew is even better than that of HC-Rew, e.g., TD3 with segment length 1.5s gets better performance than on HC-Rew. These observations indicate that a good PB-Rew approximation is derived from the simulated preference labels. Besides, it is particularly interesting that SAC on PB-Rew gets better performance on all cases than it on HC-Rew, even though SAC is still worse than other algorithms.

Fig. 9.5c shows results on constrained human preference where the human teacher is constrained to prefer the video segment that is appears visually more active. To reduce the total time for this experiment, we only run RL agent for 10000 steps for each case, but to make the comparison easier, the x-axis in Fig. 9.5c extends to the same range as that in Fig. 9.5a and Fig. 9.5b. From Fig. 9.5c, we can see that TD3 is more sensitive to segment length for constrained human preference, compared to simulated preference, where TD3 achieves the best performance with segment length 4.5s. This indicates that human preference labels may not be able to accurately capture subtle differences between two video segments, while the simulated preference can easily achieve that because of the access to the underlying reward function. This finding reminds us that when using human preference the video segment length should be carefully selected in order that it contain sufficient content for the human to make an informed discrimination. For this work, we found that 4.5s segment length results in PPO and TD3 performance with human preference that is comparable to that of simulated preference, so for the following unconstrained expert preference we will use a segment length of 4.5s.

9.5 Experiment Results on Unconstrained Expert Preference

We recruited three experts, designers or engineers highly knowledgeable about installing and configuring LAS, and asked them to remotely participate in our study for 2 sessions following the procedure depicted in Fig. 9.4. In this experiment, only segment length $K = 4.5$s is tested, while other settings are the same as that for constrained human preference. In addition, the preferences collected from different experts are used together to train a preference-based model, i.e., learning preferences of "aggregate expert" as apposed to learning separate reward models for each expert. The reason for this is that separately collecting preferences from each expert needs more preference labels from each expert and is more time consuming. However, it is worth to mention that this choice assumes experts share common preferences, as discussed in Section 9.2.

9.5.1 Expert Data Summary

In this section, we first address the following questions at the high level:

Question 1 *During/after learning, does the policy become better at generating videos preferred by the experts?*

Question 2 *During/after learning, does the reward model become better at predicting the preferences of the experts?*

To answer **Question** 1, Fig. 9.6 illustrates the proportion of each preference choice to the total number of preference choices of each preference request[5]. Of interest is how the equally engaging and equally unengaging responses change over time (note that the bars from left to right in each preference choice correspond to the sequence that the preference choices are collected. It can be seen that for the first four preference requests the proportion of equally engaging increased while the proportion of equally unengaging decreased, which indicates more and more segments that are engaging are added into the segment pool, implying that the policy of the RL agent is improving. However, the fifth and sixth preference request shows the opposite trend, that the proportion of equally engaging clips decreased while the proportion of equally unengaging clips increased, which indicates more

[5]For the distribution of the segment pairs, please refer to Fig. D.19 in Appendix D.

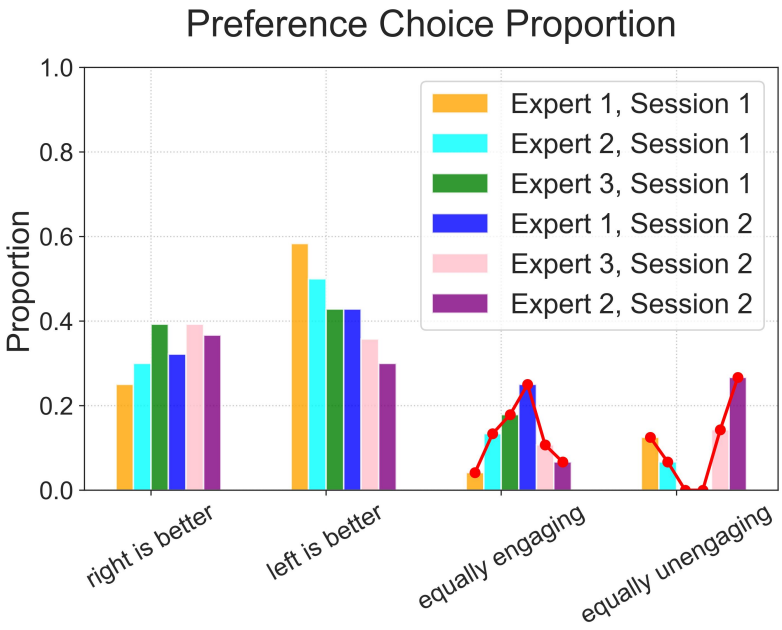


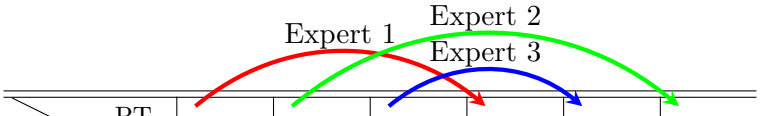
Figure 9.6: Proportion of Preference Choice for Each Request

segments that are unengaging are added into the segment pool. This is surprising as we expect the policy to improve following each interaction with the teacher. We investigated the policy and find that the policy is stuck in a local optimum, which causes the video clips generated later to be very similar. Note that the segment pool is initialized with segments sampled from the trajectories generated by a random policy in order to ensure the diversity of the initial segments. Based on these observations, we try to answer **Question** 1 as follows:

Answer to Question 1 *The policy derived from the human preferences does become better in term of generating videos preferred by the experts for the first few sessions compared to the initial policy. However, once the policy gets stuck in a local optimum, experts tend to rate similar video segments "equally unengaging" rather than "equally engaging", potentially because of the lack of diversity.*

To answer **Question** 2, Table 9.7 shows the preference prediction accuracy of a PB-Rew checkpoint on the preference labels, where the prediction accuracy above 0.5 is in bold. A simplified Table 9.8 shows the average prediction accuracy for each expert. Specifically, CP0 and CP1-CP6 are the PB-Rew checkpoints created after the initialization and the 6

Figure 9.7: Preference Prediction Accuracy of PB-Reward Checkpoints



CP \ PT	PT1	PT2	PT3	PT4	PT5	PT6
CP0 (init)	0.40	0.25	0.26	0.33	0.48	0.30
CP1	**0.95**	**0.58**	**0.57**	**0.76**	**0.57**	**0.60**
CP2	**0.95**	**0.58**	**0.57**	**0.76**	**0.57**	**0.60**
CP3	**0.95**	**0.63**	**0.52**	**0.71**	**0.57**	**0.60**
CP4	**0.95**	**0.67**	**0.61**	**0.86**	**0.52**	**0.60**
CP5	**0.90**	**0.71**	**0.61**	**0.91**	**0.57**	0.50
CP6	**0.90**	**0.71**	**0.57**	**0.95**	**0.62**	0.50

CP: PB-Rew Checkpoint

PT: Preference Teaching Session

Figure 9.8: Average Preference Prediction Accuracy of PB-Reward Checkpoints

CP \ PT	Expert 1	Expert 2	Expert 3	Average
CP0(init)	0.37	0.28	0.37	0.34
CP6	**0.93**	0.61	0.60	0.71

CP: PB-Rew Checkpoint, PT: Preference Teaching Session.

teaching sessions, respectively, and PT1-PT6 are the preference labels collected during the corresponding preference request, where the colored curves connect the preference labels from the same expert but different sessions. The white background of each cell indicates the row PB-Rew checkpoint has not seen the labels from a column preference request. The red background of the diagonal cells indicates the row checkpoint is created immediately after seeing the column preference labels for the first time. The gray background of the lower triangle cells indicates the row checkpoint has seen the column preference labels more than once, and the darker the color is, the more time the column preference labels are seen by the checkpoint. For CP0, because it is randomly initialized, its predictions are all less than 0.5[6]. After trained with preference labels from PT1, CP1 has prediction accuracy 0.95 on PT1 and has prediction accuracy greater than 0.5 for PT2-PT6, which indicates the

[6]One may expect the prediction accuracy of the randomly initialized reward model closer to 50%, but the number of "right is better" vs "left is better" is unbalanced for each request as shown in Fig. 9.6, so it makes sense to have the prediction accuracy less than 50% for randomly initialized reward model.

experts indeed share some common preference. In addition, as more and more preference labels are added to the training dataset and the PB-Rew is incrementally trained on the dataset, the preference prediction accuracy on the past (in gray color) is well contained, which means there is no catastrophic forgetting. It can be seen that all PB-Rew checkpoints after seeing expert preferences, i.e., CP1-6, outperform the randomly initialized PB-Rew, i.e., CP0, in term of the expert preference prediction accuracy. Therefore, based on these findings the answer to **Question** 2 is as follows:

Answer to Question 2 *The preference based reward model becomes significantly better at predicting the preferences of the experts than the randomly initialized reward model, and the more the reward model is trained on a set of preference labels, the higher the prediction accuracy of the reward model will be on that set of preference labels.*

In addition, it is also worth to note that all the models are good at predicting expert 1(above 90%), but they are much worse at predicting expert 2, and barely above 50% chance at predicting expert 3. This is likely because there is a difference between expert preferences, meanwhile because expert 1's preference is the first seen by the model and the continuous training is adopted, the model is more dominated by expert 1's preference. Or perhaps it is because expert 1 was more predictable and the other 2 were more volatile. A detailed analysis of the reward and policy function evolution is provided in Appendix D.6.2.

9.5.2 Expert Teacher Survey Data

The survey is composed of three parts, i.e., (1) questions adapted from the System Usability Scale (SUS) [44, 23, 45] and used to measure the usability of our web-based interface, which are listed in Table D.12, (2) questions adapted from the Robot Incentives Scale (RIS) [228, 113] and used to measure the user's perception of his/her emotion, utility, and social connection of using the preference teaching system, which are listed in Table D.13, and (3) open questions. In this study, we have three experts and each of them participated into two sessions. The survey is done after each session, but Expert 1 skipped the survey in the second session. Due to the very limited user samples, we are not able to conduct statistical analysis. Therefore, we will report all raw responses from the participants.

Usability of The Web-based Interface

Table 9.3 shows experts' response on system usability, where Q2, Q4, Q6, Q8 and Q10 are gray because they are questions negatively related to the score of system usability. Fig. 9.9

Table 9.3: Expert Response on System Usability Scale

SUS Question / Pref Request	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
Exp. 1, Ses. 1	3	2	4	1	3	1	5	1	4	2	**80.0**
Exp. 2, Ses. 1	1	4	4	2	3	2	2	4	3	2	47.5
Exp. 2, Ses. 2	1	4	2	1	4	3	4	4	2	2	47.5
Exp. 3, Ses. 1	4	1	5	1	4	1	5	2	5	1	**92.5**
Exp. 3, Ses. 2	4	1	4	1	5	1	5	2	4	1	**90.0**
Average	2.6	2.4	3.8	1.2	3.8	1.6	4.2	2.6	3.6	1.6	**71.5**

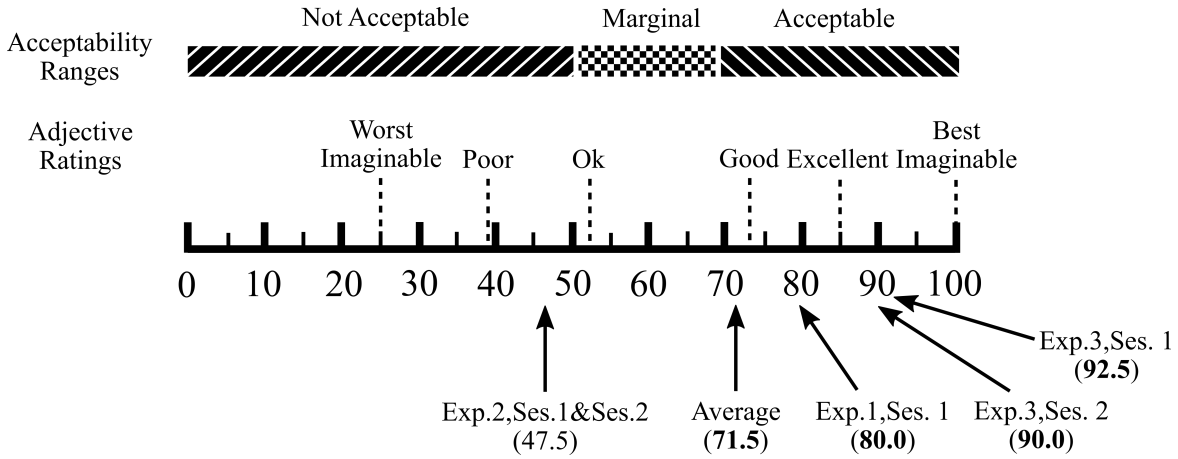


Figure 9.9: SUS Score Interpretation (adapted from [45])

shows the interpretation of SUS score along with the score of each survey data. Specifically, from the perspective of acceptability, if the SUS score is below 50, it is considered not acceptable, while if the score is above 70, it is considered acceptable. From the perspective of adjective rating, the interpretation for the score is indicated by some key points, such as worst imaginable (25), Poor (39.17), OK (52.01), Good (72.75), Excellent (85.58), and Best Imaginable (100) as shown in Fig. 9.9. Overall, expert 1 and 3 give high SUS score indicating the system is acceptable, while expert 2 does not like the interface and only gives score 47.5 which is interpreted as not acceptable. In addition, from the adjective rating perspective, expert 3 thinks the system is excellent, expert 1 thinks it is good, whereas expert 2 thinks the system is between poor and ok. Expert 1 and 3 have very

Table 9.4: What do you think is the best way to teach an interactive system engaging behavior?

Expert	Response
Expert 1	• Demonstrating actions for the sculpture
Expert 2	• Correcting the sculpture's Actions • Demonstrating actions for the sculpture • Programming the sculpture • "*Demonstrating and correcting given some context i.e. how many people are there and where are they, what is the desired outcome?*" • "*Enabling parts of the sculpture to respond to other parts of the sculpture, creating emergent feedback loops of constructive and destructive reinforcement*"
Expert 3	• Selecting from alternatives (as in this system) • Correcting the sculpture's Actions

different opinions from that of expert 2. This reminds us that experts may not share the same attitude on how to incorporate human knowledge into an interactive system to make it more engaging or on how to interpret the usability of an interface. It is also worth noting that expert 2 is a User Experience (UX) design professional, while the other two experts are more system control engineers. The different attitude among experts is also illustrated by their feedback on the open question: "*What do you think is the best way to teach an interactive system engaging behavior?*" with options as follows: (1) Selecting from alternatives (as in this system), (2) Correcting the sculpture's Actions, (3) Demonstrating actions for the sculpture, (4) Programming the sculpture, and (5) Other, please comment in detail.

Table 9.4 shows the responses from experts on this question. Among all of the options, expert 2 selected all options but not Selecting from alternatives. This indicates expert 2's strong opinion that "Selecting from alternatives" is not an ideal way to teach an interactive system, and that leads to their low ratings on SUS. Even though expert 1 thinks "Demonstrating actions for the sculpture" is the best way to teach an interactive system engaging behavior, he/she still gives a good rating on the usability of the "Selecting from alternatives". Expert 3 believes both "Selecting from alternatives" and "Correcting the

Table 9.5: Expert Response on Custom Preference Teaching Questions

Pref Request \ Question	Emotion						
	Q1	Q2	Q3	10-Q4	10-Q5	10-Q6	Score
Exp. 1, Ses. 1	2	2	1	4	1	2	20.0
Exp. 2, Ses. 1	3	3	6	3	3	8	43.3
Exp. 2, Ses. 2	2	2	3	3	7	8	41.6
Exp. 3, Ses. 1	8	7	5	7	3	3	**55.0**
Exp. 3, Ses. 2	9	8	7	9	2	7	**70.0**

Pref Request \ Question	Utility					Social Connection				
	Q1	Q2	Q3	Q4	Score	Q1	Q2	Q3	Q4	Score
Exp. 1, Ses. 1	5	5	5	7	**55.0**	7	9	6	5	**67.5**
Exp. 2, Ses. 1	4	3	5	3	37.5	3	7	8	1	47.5
Exp. 2, Ses. 2	3	5	2	4	35.0	3	4	7	1	37.5
Exp. 3, Ses. 1	9	9	9	8	**87.5**	10	10	10	9	**97.5**
Exp. 3, Ses. 2	9	9	9	9	**90.0**	9	10	9	9	**92.5**

sculpture's actions" are the best way to teach an interactive system engaging behavior, which is consistently reflected in his/her over 90 SUS score. These observations indicate that the system usability rating may be dependent on the expert's strong opinion on the best way to teach.

Users' Perception of The Preference Teaching System

Table 9.5 shows the experts' responses to the custom preference teaching questions. Because the Q4-Q6 in Emotion are measuring negative emotions, we use 10 minus the corresponding scales to convert them into positive emotions. The score of each category is calculated by taking the average scale over the questions within each category, i.e., emotion, utility, and social connection, then multiplying it with 10 to have a score in $[0, 100]$ where the higher the score, the better it is for each category. Overall, expert 3 has the highest score on all categories, which indicates expert 3 likes using the preference teaching method, and thinks this method could help in incorporating human knowledge into the

behavior of an interactive system, at the same time could feel social connection to the potential visitors who are going to interact with the interactive system. Expert 2 has the lowest score on utility and social connection, which indicates expert 2 does not think the preference teaching system is helpful to teach an interactive how to engage and he/she does not fell much social connection to the potential visitors of an interactive system by using the preference teaching system. Expert 1 has the lowest score on emotion, which indicates that even though expert 1 does not like the preference teaching method emotionally, he/she still thinks it is useful to teach the interactive system engaging behavior. According to emotion-Q4, expert 1 and 2 are both unhappy with the "Cannot Tell" queries where they cannot tell which one of the two segments is better. Expert 1 is particularly worried about providing unreliable preferences and ruining the sculpture's behavior, according to emotion-Q5 and emotion-Q6.

Table 9.6: What was your reasoning for your choice between two video clips?

Expert	**Response**
Expert 1	"mostly, the 'business' of the **patterns**, but also when there appeared to be **coordinated waves** of behaviour, I like that."
Expert 2	"Usually it had to do with **more happening**, and **more variety**, but sometimes it was about the relationship (**smoothness** and **fluidity)** of the expression I was seeing." "**activity**, and **diversity** of behaviour"
Expert 3	"I liked clips where it looked like there was some kind of recognizable **pattern** or effect that was intentional. Some clips were very **inactive**, which is not engaging. Some clips were '**hyper active**', which often looked random and unengaging." "Some behaviours were too **static**, some were too **empty**, and others were too **chaotic**. These better ones had some **gentle/subtle movement**, and the best ones had **occasional large (but still slow) movements**."

Users' Responses On Open Questions

It is hard to infer what is the expert's criteria on engaging behavior from the learned PB-Reward estimator, so we add the question "What was your reasoning for your choice between two video clips?" in the survey to directly ask them to see if experts share something in common in terms of engaging behavior. Table 9.6 shows experts responses to this question, where we highlighted some keywords of the experts' criteria. From the table, we can see that experts indeed share common preferences. For example, they all care about

patterns in the behavior of the sculpture in contrast to random behavior. In addition, variety and diversity are preferred over static and simple behavior. Activity is also considered when making a preference choice. However, as indicated by expert 3, neither inactive nor hyper active is good, but something in between with patterns is most favorable. Broadly speaking, experts share some common preferences at the high level, but we can still imagine at a more detailed level they may have distinct preference, especially given a much complicated scenario where human visitors are interacting with the sculpture.

Table 9.7: Do you think your preference has been shifted compared to that at the 1st session?

Expert	**Response**
Exp. 2, Ses. 2:	"Not really. I still find the clips quite limited in what they express and it's hard (even for me) to imagine how the activity depicted in the little thumbnails would translate to the real sculpture in all cases."
Exp. 3, Ses. 2:	"No I think my preferences are still the same as last session."

It is possible that human preference teachers change their preference after seeing more and more behavior in the segments. Therefore, in this study experts are asked if their preferences have shifted between the 1st session and the 2nd session. Table 9.7 shows the experts' responses on this question. It can be seen that neither expert 2 nor 3 reported changing their preference, which is consistent with our hypothesis in proposed in section D.6.2.

9.6 Limitations

One of the limitations of the work in this chapter is that no visitors, i.e., people who interact with the sculpture, were involved in the experiment. Therefore, we cannot examine if expert preferences are successfully passed to the behavior of the sculpture and lead to engaging behavior that can be perceived by visitors. This should be investigated in the future. In addition, in this work we only conducted 2 preference teaching sessions for each expert, which may be too limited; training performance with more sessions should be further investigated.

In this work, we only employed the reward model induced from preference data, which may be limiting. For example, the preference-based reward model may be unreliable,

especially at the beginning, due to the limited number of preference labels and the limited diversity of the segment pairs, which is detailed in section D.7.1. Moreover, there exists a world representation gap that the video segments used for collecting preference labels may capture information that cannot be represented by the experience segments. In addition, learning a reward model and the corresponding policy from scratch may cause very low learning speed. Therefore, it is interesting to investigate if the combination of a manually designed reward function, e.g., the engagement estimate proposed in Chapter 5, with a reward function learned from the human preference could induce more engaging behavior and at the same time enable faster policy learning.

Applying Preference Learning to Living Architecture System (LAS) via labeling of video segments is also challenging because moving from 3D world to 2D image results in information loss, especially for a LAS that is architectural-scale and intended for immersive crowd interaction. Video recordings captured by one camera may not be able to capture all necessary information required for preference learning. Moreover, it assumes that experts are able to accurately predict how simulations viewed in 2D videos can transfer to the user experience in the 3D world, which cannot be answered by the simulation experiment. All of these should motivate a future field study to validate the proposed approach.

There are many design choices involved in this work, such as the design of the various components in LAS-Intl-Env, i.e., the simulator, the design choices related to preference learning of a reward function, and the hyper-parameter choices related to DRL algorithms. Even though hyper-parameter search was conducted to choose the hyper-parameters empirically, it is not exhaustive and should be explored further in order to make the transfer from simulator to real world application successful.

9.7 Summary

In this chapter, we propose to generate engaging behavior from human preferences by replacing the hand-crafted reward function with a reward function induced from human preference. Specifically, we firstly introduce the overall framework and the learning algorithms at the high-level. Then, the control task is described. For the experiment, we take a step-by-step approach by starting from a hand-crafted reward function then moving towards preference-based reward function with different types of preference model, i.e., simulated preference, constrained human preference, and unconstrained expert preference. With these various experiment settings, we examine four DRL algorithms namely PPO, TD3, SAC and LSTM-TD3. Our results show PPO outperforms the other three algorithms on the hand-crafted reward, while SAC performs the worst. With simulated preferences,

TD3 and LSTM-TD3 achieve similar performance to that of PPO, but at faster learning speed. Our results on constrained human preferences show that the algorithms are more sensitive to segment length, where a longer segment length seems better than a short one. Finally, for experiment on unconstrained expert preference, we recruit three experts and each of them participates in two sessions. Our results show that the policy derived from the human preference does become better in term of generating videos preferred by the experts for the first few sessions compared to the initial policy. However, once the policy gets stuck to a local optical, experts tend to treat the similar video segments "equally unengaging" rather than "equally engaging", potentially because of the lack of diversity. In addition, we find that the preference based reward model becomes significantly better at predicting the preferences of the experts than the randomly initialized reward model and its old version, and the more the reward model is trained on a set of preference labels, the higher the prediction accuracy of the reward model will be on that set of preference labels. Our survey data of unconstrained experts shows experts have different attitudes towards the preference-based teaching system, where two of them have similar and higher ratings of the system and one of them is less favourable towards the preference-based teaching system and inclined to other methods, e.g., correction and demonstration based teaching.

Chapter 10

Conclusions and Future Work

As more and more robots move from lab/factory to people's houses and from safety cages to human-centered spaces, it is necessary to enable the robots to engage with their partners for productive, enjoyable, entertaining and long-lasting human-centered HRI. To help further this goal, this thesis applies state-of-art artificial intelligence techniques, especially Reinforcement Learning (RL) and Preference Learning (PL), to living architecture systems (LAS) to examine the various ideas towards learning to engage. Distinct from other HRI applications focus on one-to-one HRI, LAS particularly inclines to one-to-many (namely crowd) HRI given its architectural scale with hundreds of actuators and dozens of sensors. Considering the novelty of the robot, i.e., LAS, investigated in this thesis, we conduct both field study and simulation experiments on LAS to test the proposed approaches based on RL and PL. Along the journey, we recognize that it is still challenging to directly apply the state-of-art algorithms to our case, because of the limitations of these algorithms when facing POMDP and suffering from over-estimation problem, etc. Therefore, this thesis also studies these problems and proposes methods to solve them.

10.1 Conclusions

In Chapter 4, we introduced two physical testbeds, namely *Aegis Canopy* and *Meander*, and the LAS simulation toolkit. For the two testbeds, we first described the composition of sensors and actuators, then illustrated their pre-scripted behavior. Different from *Aegis Canopy*, which is installed in a museum and accessible to visitors who paid a ticket, *Meander* is installed in a building where part of the sculpture is publicly accessible. However, the physical testbeds are not always accessible to researchers and visitors. Therefore, we

developed a simulation toolkit to allow researchers to more easily develop and evaluate machine learning techniques to such challenging interactive systems.

In Chapter 5, we developed and evaluated algorithms for generating interactive behaviors in group environments. Specifically, we provide a way to estimate engagement during group interaction based on multiple IR sensors, where both individual engagement, passive and active interaction, and group engagement, i.e. occupancy, are taken into account. PB and PLA were examined to evaluate how the use of human knowledge influences interaction. By analyzing interaction and human survey data, we found that learned interactive behaviors, i.e. PLA, result in higher engagement and perceived likeability than pre-scripted behavior, i.e., PB. This study revealed that comparing to the fixed pre-scripted behavior, the adaptive behavior generated by a parameterized learning agent, that acts on the expert-designed action space and is driven by an engagement estimate, is more capable of engaging visitors and more promising for enabling a long-term interaction.

In Chapter 6, we empirically revealed multi-step methods' effect on alleviating overestimation in DRL, by proposing MDDPG and MMDDPG which are a combination of DDPG and multi-step methods, and discussed the underlying underestimation and overestimation tradeoff. Results show that employing multi-step methods in DRL helps to alleviate the overestimation problem by exploiting bootstrapping and improve the data efficiency of DDPG. This paper also discussed the advantages and disadvantages of three ways to implement multi-step methods from the point of view of extra computation cost and modeling error. This work helps to improve the data efficiency of DRL algorithms in order to make these algorithms more applicable to the LAS.

In Chapter 7, we proposed a memory-based DRL algorithm called LSTM-TD3 by combining a recurrent actor-critic framework with TD3. The proposed LSTM-TD3 was compared to standard DRL algorithms on both the MDP- and POMDP-versions of continuous control tasks. Our results show that LSTM-TD3 not only achieves significantly better performance on POMDPs than the baselines, but also retains the state-of-art performance on MDP. The ablation study shows that all components are essential to the success of the LSTM-TD3 where DC and TPS help in stabilizing learning, CFE is especially important to retain the good performance in MDP, and PA is beneficial for tasks where past actions provide information about the current state of the agent. This work is motivated by our concern that our LAS control task may be not a MDP but a POMDP, so an algorithm that is able to deal with POMDPs is required. For successful application to systems with an unknown state space such as LAS, the proposed LSTM-TD3 should not only be capable of solving POMDPs but also maintain good performance on MDPs.

In Chapter 8, we first highlight the counter-intuitive observation found when applying

DRLs to a novel complex robot, that PPO outperforms TD3 and SAC. We hypothesize that this degradation in performance is caused by partial observability. Then, we provide a potential explanation about why the multi-step bootstrapping employed in PPO makes it more robust to partial observability compared TD3 and SAC, which only rely on 1-step bootstrapping. Based on that, we proposed MTD3 and MSAC to verify our hypotheses on MDP- and POMDP-version of benchmark tasks. The same counter-intuitive observation can be reproduced on the POMDP-versions of the benchmark tasks, which confirms the problem is caused by partial observability. The results of MTD3 and MSAC with multi-step size $n = 5$ show that simply increasing the step size from $n = 1$ to $n = 5$ can significantly increase the performance of vanilla TD3 and SAC on POMDPs. The work in this chapter illustrates that when the observability of the state-space of a system is unknown, researchers could apply both DRL algorithms capable of POMDPs and algorithms that are designed for MDPs to the task at hand and compare their performance to help to diagnose if the task is a POMDP or MDP.

In Chapter 9, we propose to generate engaging behavior from human preferences by replacing the hand-crafted reward function with a reward function induced from human preference. Specifically, we firstly introduce the overall framework and the learning algorithms at the high-level. Then, the control task is described. For the experiment, we take a step-by-step approach by starting from a hand-crafted reward function then moving towards preference-based reward function with different types of preference model, i.e., simulated preference, constrained human preference, and unconstrained expert preference. With these various experiment settings, we examine four DRL algorithms namely PPO, TD3, SAC and LSTM-TD3. Our results show PPO outperforms the other three algorithms on the hand-crafted reward, while SAC performs the worst. With simulated preferences, TD3 and LSTM-TD3 achieve similar performance to that of PPO, but at faster learning speed. Our results on constrained human preferences show that the algorithms are more sensitive to segment length, where a longer segment length seems better than a short one. Finally, for experiment on unconstrained expert preference, we recruit three experts and each of them participates in two sessions. Our results show that the policy derived from the human preference does become better in term of generating videos preferred by the experts for the first few sessions compared to the initial policy. However, once the policy gets stuck to a local optical, experts tend to treat the similar video segments "equally unengaging" rather than "equally engaging", potentially because of the lack of diversity. In addition, we find that the preference based reward model becomes significantly better at predicting the preferences of the experts than the randomly initialized reward model and its old version, and the more the reward model is trained on a set of preference labels, the higher the prediction accuracy of the reward model will be on that set of preference labels.

Our survey data of unconstrained experts shows experts have different attitudes towards the preference-based teaching system, where two of them have similar and higher ratings of the system and one of them is less favourable towards the preference-based teaching system and inclined to other methods, e.g., correction and demonstration based teaching. The work in this chapter shows that the recruited experts do share common preferences, but the preferences are not exactly the same. Therefore, the proposed method can help derive engagement estimates that measure common preferences. As a complement to those common preferences, the derived engagement estimate can be combined with personalized preferences to further improve the engagement of the robot's behavior.

10.2 Future Work

Even though PLA proposed in Chapter 5 received higher average engagement and perceived likeability than PB, we cannot be certain about the cause of this difference. Therefore, a baseline with random policy can be tested to see if there is a difference between this baseline and PLA to confirm that the learning agent is indeed learning from and adapting to its interaction experience. Other advanced continuous control DRL algorithms such as SAC, TD3 and MDDPG are also worth investigating. Upcoming installations of even larger LAS are planned and this decentralization will become necessary as the number of actuated elements increases beyond one thousand in a single installation. In addition, hierarchical RL with PB bootstrapping could be a promising extension, where we could design a pool of PBs and various levels of reward functions, and see how complicated action patterns could emerge. It is also promising to introduce intrinsic motivation and a learning algorithm driven both intrinsically and extrinsically for LAS. To tackle the low pace of interaction in LAS and high sample requirement of RL, it is interesting to investigate how to transfer learned models from simulation to physical LAS. We realize conducting field study to examine these ideas is impossible, not only because a physical installation is not always available but also the time needed for such study is prohibitive. Therefore, in the future we could introduce visitor models in the simulation facility proposed in Chapter 4 to promote the accessibility of LAS and run multiple simulation experiments in parallel on a High Performance Computing system.

Chapter 6 shows the effect of multi-step method on alleviating overestimation problem in DRL. However, the step size is empirically chosen. Therefore, a principled way for choosing step size n is still needed. Perhaps dynamically tuning n during the course of learning is more suitable as at different stages of learning the trade-off between overestimation and underestimation needs to be balanced differently. The most important future

direction arising from this work is to find a more effective way to overcome overestimation since this is key to improving DRL algorithms' sample efficiency, while still retaining a simple exploration method in order to limit computational needs.

The proposed approach in Chapter 7 is particularly useful when engineers do not have enough knowledge about the environment model and the appropriate design of the observation space to capture the underlying state. Memory can be useful in such a scenario to help infer the underlying state. However, the interpretation of the extracted memory is a challenge. If there is a way to properly interpret the extracted memory, such information, e.g. if the current task is a POMDP or a MDP, can be exploited to improve the observation space design and advance the understanding of the task. Unfortunately, without adding specific constraint terms in the cost functions Eq. 7.4 and 7.6 to facilitate the interpretation, there is no way to properly interpret the extracted memory. Future research should give attention to this direction. In addition, for each run of LSTM-TD3 in Chapter 7 we treat the history length l as a hyper-parameter and fixed it for each run. While LSTM-TD3 with a history length $l = 5$ achieves good performance on the devised POMDPs, this may not be achieved for other tasks where the underlying state depends on less recent memory. However, a long history length increases computation resources and time, during both the training and the inferring, i.e. decision making, phases. In the future, an approach for dynamic adaptation of history length l that achieves the best performance while minimising training and decision making time should be investigated. Besides, more sophisticated POMDP tasks relying on long past history should be examined. SAC with LSTM is also worth to investigate in the future. With the insight of the importance of CFE of LSTM-TD3, TD3-OW with a separate CFE should also be studied and compared to LSTM-TD3.

In terms of the work in Chapter 8, a deeper understanding about why n-step bootstrapping can make TD3 and SAC better on POMDP is an interesting direction for the future. Besides, it is also worth to investigate if LSTM-TD3 and MTD3 can be combined to allow TD3 to solve POMDP better by learning temporal information both from the past experiences and from the future rewards.

Inspired by the results reported in Chapter 9, we identify the potential bias in preference based reward model, the reward scaling of PL in reward saturation, the world representation gap between video and experience segment, the interesting common and personalized preference problem, the potential difference between expert and novice teacher, and the limitations of the work in Appendix D. These challenges are not unique to our application, but are common to most real world applications where PL is employed. Therefore, they are all worth to be investigated deeply. Specifically, more sophisticated query generation and sampling methods should be investigated in order to reduce the bias by increasing the

diversity of the queries shown to the users. More attention should also be given to reward scaling of PL, because the non-stationary reward resulting from reward scaling is problematic and may cause sever fluctuation in the policy induced from that. One possible solution to that is to add an additional term in the cost function of reward model optimization to penalize huge change in reward estimation. Humans are very good at image processing and understanding, and it is reasonable to make the most of this capacity to transfer human preference. However, when the world representation gap exists between video and experience segment, it is questionable how much information can be effectively transferred to a reward model. Therefore, understanding the effect of the world representation on the effectiveness and efficiency of human preference transferring is also an interesting research direction. In this thesis, we were trying to extract the common preference from multiple experts into a reward function, whereas personalized preference should also be maintained in order to engage different people. Moreover, it is also interesting to understand if there is difference between expert and novice teacher. The last but not the least, applying PL to real world applications, rather than only experimenting in simulation, is challenging and needs more investigations, and will be left for the future.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Douglas Aberdeen, Olivier Buffet, and Owen Thomas. Policy-gradients for psrs and pomdps. In *Artificial Intelligence and Statistics*, pages 3–10. PMLR, 2007.

[3] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[4] Igor Adamski, Robert Adamski, Tomasz Grel, Adam Jedrych, Kamil Kaczmarek, and Henryk Michalewski. Distributed deep reinforcement learning: Learn how to play atari games in 21 minutes. In *International Conference on High Performance Computing*, pages 370–388. Springer, 2018.

[5] Nir Ailon. An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity. *Journal of Machine Learning Research*, 13(1), 2012.

[6] Riad Akrour, Marc Schoenauer, and Michele Sebag. Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer, 2011.

[7] Riad Akrour, Marc Schoenauer, and Michèle Sebag. April: Active preference learning-based reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 116–131. Springer, 2012.

[8] Arthur Allshire, Roberto Martín-Martín, Charles Lin, Shawn Manuel, Silvio Savarese, and Animesh Garg. Laser: Learning a latent action space for efficient

reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6650–6656. IEEE, 2021.

[9] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[10] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 176–185. JMLR. org, 2017.

[11] Salvatore M Anzalone, Sofiane Boucenna, Serena Ivaldi, and Mohamed Chetouani. Evaluating the engagement with social robots. *International Journal of Social Robotics*, 7(4):465–478, Aug 2015.

[12] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[13] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[14] John Asmuth, Lihong Li, Michael L Littman, Ali Nouri, and David Wingate. A bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 19–26. AUAI Press, 2009.

[15] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[16] Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965.

[17] Alexandre Attia and Sharone Dayan. Global overview of imitation learning. *arXiv preprint arXiv:1801.06503*, 2018.

[18] Federico Augugliaro, Angela P. Schoellig, and Raffaello D'Andrea. Dance of the flying machines: Methods for designing and executing an aerial dance choreography. *IEEE Robotics Automation Magazine*, 20(4):96–104, Dec 2013.

[19] Benedicte M Babayan, Naoshige Uchida, and Samuel J Gershman. Belief state representation in the dopamine system. *Nature communications*, 9(1):1–10, 2018.

[20] Yuqin Bai. An empirical study on bias reduction: Clipped double q vs. multi-step methods. In *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, pages 1063–1068. IEEE, 2021.

[21] Albert Bandura, William H Freeman, and Richard Lightsey. Self-efficacy: The exercise of control, 1999.

[22] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[23] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[24] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics*, 1(1):71–81, Jan 2009.

[25] Andrew G. Barto. *Intrinsic Motivation and Reinforcement Learning*, pages 17–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[26] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial intelligence*, 72(1-2):81–138, 1995.

[27] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[28] Philip Beesley. *Near-living Architecture: Work in Progress from the Hylozoic Ground Collaboration, 2011-2013*. Riverside Architectural Press Toronto, Ontario, Canada, 2014.

[29] Philip Beesley, Matthew Chan, Rob Gorbet, Dana Kulić, and Mo Memarian. Evolving systems within immersive architectural environments: New research by the living architecture systems group. *Next Generation Building*, 2:31–56, 2015.

[30] Philip Beesley, Pernilla Ohrstedt, and Rob Gorbet. *Hylozoic Ground: Liminal Responsive Architecture: Philip Beesley*. Riverside Architectural Press, 2010.

[31] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[32] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

[33] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[34] Erdem Bıyık, Nicolas Huynh, Mykel J Kochenderfer, and Dorsa Sadigh. Active preference-based gaussian process regression for reward learning. *arXiv preprint arXiv:2005.02575*, 2020.

[35] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

[36] Dan Bohus and Eric Horvitz. Models for multiparty engagement in open-world dialog. In *Proceedings of the SIGDIAL 2009 Conference*, pages 225–234. Association for Computational Linguistics, 2009.

[37] Dan Bohus and Eric Horvitz. Open-world dialog: Challenges, directions, and a prototype. Technical report, April 2009.

[38] Dan Bohus and Eric Horvitz. Managing human-robot engagement with forecasts and... um... hesitations. In *Proceedings of the 16th International Conference on Multimodal Interaction*, ICMI '14, pages 2–9, New York, NY, USA, 2014. ACM.

[39] Blai Bonet and Hector Geffner. Solving pomdps: Rtdp-bel vs. point-based algorithms. In *IJCAI*, pages 1641–1646. Pasadena CA, 2009.

[40] Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher Pal, and Jonathan Binas. Reinforcement learning with random delays. In *International conference on learning representations*, 2021.

[41] Darius Braziunas. Pomdp solution methods. *University of Toronto*, 2003.

[42] Cynthia Breazeal, Kerstin Dautenhahn, and Takayuki Kanda. *Social Robotics*, pages 1935–1972. Springer International Publishing, Cham, 2016.

[43] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[44] John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3), 1996.

[45] John Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[46] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pages 330–359. PMLR, 2020.

[47] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234, 2018.

[48] Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for markov decision processes. *Mathematics of Operations Research*, 22(1):222–255, 1997.

[49] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[50] Salvatore Candido and Seth Hutchinson. Minimum uncertainty robot navigation using information-guided pomdp planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 6102–6108. IEEE, 2011.

[51] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[52] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

[53] Anthony R Cassandra. A survey of pomdp applications. In *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, volume 1724, 1998.

[54] Ginevra Castellano, André Pereira, Iolanda Leite, Ana Paiva, and Peter W. McOwan. Detecting user engagement with a robot companion using task and social interaction-based features. In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, ICMI-MLMI '09, pages 119–126, New York, NY, USA, 2009. ACM.

[55] Carlos Celemin and Javier Ruiz-del Solar. Interactive learning of continuous actions from corrective advice communicated by humans. In *Robot soccer world cup*, pages 16–27. Springer, 2015.

[56] Carlos Celemin and Javier Ruiz-del Solar. An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Systems*, 95(1):77–97, 2019.

[57] Matthew TK Chan, Rob Gorbet, Philip Beesley, and Dana Kulič. Curiosity-based learning algorithm for distributed interactive sculptural systems. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3435–3441. IEEE, Sep. 2015.

[58] Matthew TK Chan, Rob Gorbet, Philip Beesley, and Dana Kulić. Interacting with curious agents: User experience with interactive sculptural systems. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 151–158. IEEE, Aug 2016.

[59] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression. In *Conference on robot learning*, pages 1262–1277. PMLR, 2021.

[60] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[61] Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *Conference on Robot Learning*, pages 422–431. PMLR, 2018.

[62] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

[63] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[64] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[65] Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144, 2005.

[66] Michael Jae-Yoon Chung and Maya Cakmak. "how was your stay?": Exploring the use of robots for gathering customer feedback in the hospitality industry. In *27th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2018, Nanjing, China, August 27-31, 2018*, pages 947–954, 2018.

[67] Pawel Cichosz. Truncating temporal differences: On the efficient implementation of td (lambda) for reinforcement learning. *Journal of Artificial Intelligence Research*, 2:287–318, 1994.

[68] Andrew L Comrey and Howard B Lee. *A first course in factor analysis*. Psychology press, 2013.

[69] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[70] Francisco Cruz, Sven Magg, Yukie Nagai, and Stefan Wermter. Improving interactive reinforcement learning: What makes a good teacher? *Connection Science*, 30(3):306–325, 2018.

[71] Catie Cuan, Ishaan Pakrasi, and Amy LaViers. Time to compile. In *Proceedings of the 5th International Conference on Movement and Computing*, MOCO '18, pages 53:1–53:4, New York, NY, USA, 2018. ACM.

[72] Mary L Cummings and Sylvain Bruni. Collaborative human–automation decision making. In *Springer handbook of automation*, pages 437–447. Springer, 2009.

[73] Raffaello D'Andrea and Max Dean. The table. https://raffaello.name/projects/table/, 2001.

[74] Kerstin Dautenhahn. Socially intelligent robots: dimensions of human–robot interaction. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 362(1480):679–704, 2007.

[75] Kristopher De Asis, J Fernando Hernandez-Garcia, G Zacharias Holland, and Richard S Sutton. Multi-step reinforcement learning: A unifying algorithm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[76] Robert F DeVellis. *Scale development: Theory and applications*, volume 26. Sage publications, 2016.

[77] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[78] Kevin Doherty and Gavin Doherty. Engagement in hci: Conception, theory and measurement. *ACM Comput. Surv.*, 51(5):99:1–99:39, November 2018.

[79] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *CoRR*, abs/1604.06778, 2016.

[80] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[81] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[82] Tom Everitt, Victoria Krakovna, Laurent Orseau, Marcus Hutter, and Shane Legg. Reinforcement learning with a corrupted reward channel. *arXiv preprint arXiv:1705.08417*, 2017.

[83] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[84] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[85] Amalia Foka and Panos Trahanias. Real-time hierarchical pomdps for autonomous robot navigation. *Robotics and Autonomous Systems*, 55(7):561–571, 2007.

[86] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[87] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[88] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[89] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[90] Johannes Fürnkranz and Eyke Hüllermeier. Pairwise preference learning and ranking. In *European conference on machine learning*, pages 145–156. Springer, 2003.

[91] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[92] Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.

[93] Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Marcin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of the SIGDIAL 2013 Conference*, pages 214–222, 2013.

[94] Milica Gašić and Steve Young. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40, 2013.

[95] Samuel J Gershman and Naoshige Uchida. Believing in dopamine. *Nature Reviews Neuroscience*, 20(11):703–714, 2019.

[96] Nadine Glas and Catherine Pelachaud. Definitions of engagement in human-agent interaction. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 944–949, Sep. 2015.

[97] Alyssa Glass. Explaining preference learning, 2006.

[98] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

[99] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[100] Michael A. Goodrich and Alan C. Schultz. Human-robot interaction: A survey. *Found. Trends Hum.-Comput. Interact.*, 1(3):203–275, January 2007.

[101] Goren Gordon, Samuel Spaulding, Jacqueline Kory Westlund, Jin Joo Lee, Luke Plummer, Marayna Martinez, Madhurima Das, and Cynthia Breazeal. Affective personalization of a social robot tutor for children's second language skills. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[102] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L. Isbell, and Andrea Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2625–2633, USA, 2013. Curran Associates Inc.

[103] Mikell P Groover. *Fundamentals of modern manufacturing: materials, processes, and systems*. John Wiley & Sons, 2020.

[104] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[105] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[106] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[107] Sami Haddadin and Elizabeth Croft. *Physical Human–Robot Interaction*, pages 1835–1874. Springer International Publishing, Cham, 2016.

[108] Martin T Hagan, Howard B Demuth, and Mark Beale. *Neural network design*. PWS Publishing Co., 1997.

[109] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[110] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

[111] Kotaro Hayashi, Daisuke Sakamoto, Takayuki Kanda, Masahiro Shiomi, Satoshi Koizumi, Hiroshi Ishiguro, Tsukasa Ogasawara, and Norihiro Hagita. Humanoid robots as a passive-social medium-a field experiment at a train station. In *2007*

2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 137–144. IEEE, March 2007.

[112] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[113] Marcel Heerink, Ben Krose, Vanessa Evers, and Bob Wielinga. Measuring acceptance of an assistive social robot: a suggested toolkit. In *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 528–533. IEEE, 2009.

[114] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

[115] Joey Hejna and Dorsa Sadigh. Few-shot preference learning for human-in-the-loop rl. *arXiv preprint arXiv:2212.03363*, 2022.

[116] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

[117] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[118] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[119] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[120] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[121] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[122] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[123] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Curiosity-driven exploration in deep reinforcement learning via bayesian neural networks. *arXiv preprint arXiv:1605.09674*, 2016.

[124] Ronald A Howard. Dynamic programming and markov processes. 1960.

[125] Jennifer L Hughes, Abigail A Camden, and Tenzin Yangchen. Rethinking and updating demographic questions: Guidance to improve descriptions of research samples. *Psi Chi Journal of Psychological Research*, 21(3):138–151, 2016.

[126] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21, 2017.

[127] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

[128] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

[129] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR, 2018.

[130] Alex Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[131] Charles Isbell, Christian R. Shelton, Michael Kearns, Satinder Singh, and Peter Stone. A social reinforcement learning agent. In *Proceedings of the Fifth International Conference on Autonomous Agents*, AGENTS '01, pages 377–384, New York, NY, USA, 2001. ACM.

[132] Carolina Islas Sedano, Verona Leendertz, Mikko Vinni, Erkki Sutinen, and Suria Ellis. Hypercontextualized learning games: Fantasy, motivation, and engagement in reality. *Simulation & Gaming*, 44(6):821–845, 2013.

[133] Serena Ivaldi, Sébastien Lefort, Jan Peters, Mohamed Chetouani, Joelle Provasi, and Elisabetta Zibetti. Towards engagement models that consider individual factors in HRI: on the relation of extroversion and negative attitude towards robots to gaze and speech during a human-robot assembly task. *CoRR*, abs/1508.04603, 2015.

[134] Michiel Joosse and Vanessa Evers. A guide robot at the airport: First impressions. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 149–150, New York, NY, USA, 2017. ACM.

[135] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018.

[136] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[137] Kirsikka Kaipainen, Aino Ahtinen, and Aleksi Hiltunen. Nice surprise, more present than a machine: Experiences evoked by a social robot for guidance and edutainment at a city service point. In *Proceedings of the 22Nd International Academic Mindtrek Conference*, Mindtrek '18, pages 163–171, New York, NY, USA, 2018. ACM.

[138] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[139] Takayuki Kanda and Hiroshi Ishiguro. *Human-Robot Interaction in Social Robotics*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2012.

[140] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning. In *2020 IEEE Conference on Games (CoG)*, pages 479–486. IEEE, 2020.

[141] Simon Keizer, Mary Ellen Foster, Zhuoran Wang, and Oliver Lemon. Machine learning for social multiparty human–robot interaction. *ACM transactions on interactive intelligent systems (TIIS)*, 4(3):14, 2014.

[142] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[143] Mehdi Khamassi, George Velentzas, Theodore Tsitsimis, and Costas Tzafestas. Robot fast adaptation to changes in human engagement during simulated dynamic social interaction with active exploration in parameterized reinforcement learning. *IEEE Transactions on Cognitive and Developmental Systems*, 10(4):881–893, 2018.

[144] Rajiv Khosla, Khanh Nguyen, and Mei-Tai Chu. Human robot engagement and acceptability in residential aged care. *International Journal of Human–Computer Interaction*, 33(6):510–522, 2017.

[145] Cory D Kidd and Cynthia Breazeal. Robots at home: Understanding long-term human-robot interaction. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3230–3235. IEEE, 2008.

[146] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[147] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[148] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *AAMAS*, 2010.

[149] W. Bradley Knox and Peter Stone. Reinforcement learning from simultaneous human and mdp reward. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 475–482, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[150] Samantha Krening and Karen M. Feigh. Interaction algorithm effect on human experience with reinforcement learning. *ACM Trans. Hum.-Robot Interact.*, 7(2):16:1–16:22, October 2018.

[151] Dana Kulic and Elizabeth A Croft. Affective state estimation for human–robot interaction. *IEEE Transactions on Robotics*, 23(5):991–1000, 2007.

[152] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[153] Kazumi Kumagai, Daiwei Lin, Lingheng Meng, Alexandru Blidaru, Philip Beesley, Dana Kulić, and Ikuo Mizuuchi. Towards individualized affective human-machine interaction. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 678–685. IEEE, IEEE, 2018.

[154] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[155] Edith Law, Vicky Cai, Qi Feng Liu, Sajin Sasy, Joslin Goh, Alex Blidaru, and Dana Kulić. A wizard-of-oz study of curiosity in human-robot interaction. In *Robot and Human Interactive Communication (RO-MAN), 2017 26th IEEE International Symposium on*, pages 607–614. IEEE, 2017.

[156] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[157] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

[158] Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021.

[159] Iolanda Leite, André Pereira, Ginevra Castellano, Samuel Mascarenhas, Carlos Martinho, and Ana Paiva. Modelling empathy in social robotic companions. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 135–147. Springer, 2011.

[160] Florent Levillain, Elisabetta Zibetti, and Sébastien Lefort. Interacting with non-anthropomorphic robotic artworks and interpreting their behaviour. *International Journal of Social Robotics*, 9(1):141–161, Jan 2017.

[161] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

[162] Kejun Li, Maegan Tucker, Erdem Bıyık, Ellen Novoseller, Joel W Burdick, Yanan Sui, Dorsa Sadigh, Yisong Yue, and Aaron D Ames. Roial: Region of interest active learning for characterizing exoskeleton gait preference landscapes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3218. IEEE, 2021.

[163] Xiang Li, Yongping Pan, Gong Chen, and Haoyong Yu. Adaptive human–robot interaction control for robots driven by series elastic actuators. *IEEE Transactions on Robotics*, 33(1):169–182, 2016.

[164] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[165] Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *NIPS*, volume 14, page 30, 2001.

[166] Phoebe Liu, Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. Learning interactive behavior for service robots the challenge of mixed-initiative interaction. In *Proceedings of the Workshop on Behavior Adaptation, Interaction and Learning for Assistive Robotics*, 2016.

[167] Phoebe Liu, Dylan F Glas, Takayuki Kanda, and Hiroshi Ishiguro. Learning proactive behavior for interactive social robots. *Autonomous Robots*, pages 1–19, 2018.

[168] Wei Liu, Seong-Woo Kim, Scott Pendleton, and Marcelo H Ang. Situation-aware decision making for autonomous driving on urban road using online pomdp. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1126–1133. IEEE, 2015.

[169] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017.

[170] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[171] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection science*, 15(4):151–190, 2003.

[172] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[173] Douglas G Macharet and Dinei A Florencio. Learning how to increase the chance of human-robot engagement. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2173–2179, Nov 2013.

[174] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning*, pages 561–591. PMLR, 2018.

[175] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[176] Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[177] Marcos Maroto-Gómez, Sara Marqués Villarroya, María Malfaz, Álvaro Castro-González, Josè Carlos Castillo, and Miguel Ángel Salichs. A preference learning system for the autonomous selection and personalization of entertainment activities during human-robot interaction. In *2022 IEEE International Conference on Development and Learning (ICDL)*, pages 343–348. IEEE, 2022.

[178] David C May, Kristie J Holler, Cindy L Bethel, Lesley Strawderman, Daniel W Carruth, and John M Usher. Survey of factors for the prediction of human comfort with a non-anthropomorphic robot in public spaces. *International Journal of Social Robotics*, 9(2):165–180, 2017.

[179] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[180] Lingheng Meng, Rob Gorbet, and Dana Kulić. The effect of multi-step methods on overestimation in deep reinforcement learning. *arXiv preprint arXiv:2006.12692*, 2020.

[181] Lingheng Meng, Rob Gorbet, and Dana Kulić. The effect of multi-step methods on overestimation in deep reinforcement learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 347–353. IEEE, 2021.

[182] Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626. IEEE, 2021.

[183] Lingheng Meng, Rob Gorbet, and Dana Kulić. Partial observability during drl for robot control. *arXiv preprint arXiv:2209.04999*, 2022.

[184] Lingheng Meng, Rob Gorbet, and Dana Kulić. The effect of multi-step methods on overestimation in deep reinforcement learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 347–353, 2021.

[185] Lingheng Meng, Rob Gorbet, and Dana Kulić. Memory-based deep reinforcement learning for pomdps. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626, 2021.

[186] Lingheng Meng, Daiwei Lin, Adam Francey, Rob Gorbet, Philip Beesley, and Dana Kulić. Learning to engage with interactive systems: A field study. *arXiv preprint arXiv:1904.06764*, 2019.

[187] Lingheng Meng, Daiwei Lin, Adam Francey, Rob Gorbet, Philip Beesley, and Dana Kulić. Learning to engage with interactive systems: A field study on deep reinforcement learning in a public museum. *J. Hum.-Robot Interact.*, 10(1), oct 2020.

[188] Lingheng Meng, Daiwei Lin, Adam Francey, Rob Gorbet, Philip Beesley, and Dana Kulić. Learning to engage with interactive systems: A field study on deep reinforcement learning in a public museum. *ACM Transactions on Human-Robot Interaction (THRI)*, 10(1):1–29, 2020.

[189] Marek P Michalowski, Selma Sabanovic, and Reid Simmons. A spatial model of engagement for a social robot. In *9th IEEE International Workshop on Advanced Motion Control, 2006.*, pages 762–767, March 2006.

[190] Marco Mirolli and Gianluca Baldassarre. Functions and mechanisms of intrinsic motivations. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, pages 49–72. Springer, 2013.

[191] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[192] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[193] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[194] Clément Moulin-Frier and Pierre-Yves Oudeyer. Exploration strategies in developmental robotics: a unified probabilistic framework. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pages 1–6. IEEE, 2013.

[195] Thibaut Munzer, Marc Toussaint, and Manuel Lopes. Preference learning on the execution of collaborative human-robot tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 879–885. IEEE, 2017.

[196] Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2:X3, 2000.

[197] Robin R Murphy. *Introduction to AI robotics*. MIT press, 2019.

[198] Bilge Mutlu and Jodi Forlizzi. Robots in organizations: The role of workflow, social, and environmental factors in human-robot interaction. In *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 287–294, March 2008.

[199] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[200] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[201] Kazushi Nakazawa, Keita Takahashi, and Masahide Kaneko. Unified environment-adaptive control of accompanying robots using artificial potential field. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 199–200. IEEE, 2013.

[202] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[203] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[204] Heather L O'Brien and Elaine G Toms. What is user engagement? a conceptual framework for defining user engagement with technology, 2008.

[205] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Pomdps for robotic tasks with mixed observability. In *Robotics: Science and systems*, volume 5, page 4, 2009.

[206] OpenAI. Learning dexterous in-hand manipulation. *arXiv preprint http-sarXiv:1808.00177v2*, 2018.

[207] Randall C O'reilly and Yuko Munakata. *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. MIT press, 2000.

[208] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[209] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[210] Jason W Osborne and Anna B Costello. Sample size and subject to item ratio in principal components analysis. *Practical Assessment, Research, and Evaluation*, 9(1):11, 2004.

[211] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[212] Vikas N O'Reilly-Shah. Factors influencing healthcare provider respondent fatigue answering a globally administered in-app survey. *PeerJ*, 5:e3785, 2017.

[213] Joni Pajarinen and Ville Kyrki. Robotic manipulation of multiple objects as a pomdp. *Artificial Intelligence*, 247:213–228, 2017.

[214] Joni Pajarinen, Jens Lundell, and Ville Kyrki. Pomdp planning under object composition uncertainty: Application to robotic manipulation. *IEEE Transactions on Robotics*, 2022.

[215] Egon S Pearson, Ralph B D "'AGOSTINO, and Kimiko O Bowman. Tests for departure from normality: Comparison of powers. *Biometrika*, 64(2):231–246, 1977.

[216] Paola Pennisi, Alessandro Tonacci, Gennaro Tartarisco, Lucia Billeci, Liliana Ruta, Sebastiano Gangemi, and Giovanni Pioggia. Autism and social robotics: A systematic review. *Autism Research*, 9(2):165–183, 2016.

[217] Rodrigo Pérez-Dattari, Carlos Celemin, Javier Ruiz-del Solar, and Jens Kober. Interactive learning with corrective feedback for policies based on deep neural networks. In *International Symposium on Experimental Robotics*, pages 353–363. Springer, 2020.

[218] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032. Citeseer, 2003.

[219] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *CoRR*, abs/1706.01905, 2017.

[220] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

[221] Stephen R Porter, Michael E Whitcomb, and William H Weitzer. Multiple surveys of students and survey fatigue. *New directions for institutional research*, 2004(121):63–73, 2004.

[222] Ahmed Hussain Qureshi, Yutaka Nakamura, Yuichiro Yoshikawa, and Hiroshi Ishiguro. Robot gains social intelligence through multimodal deep reinforcement learning. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 745–751. IEEE, 2016.

[223] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems*, pages 5690–5701, 2017.

[224] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

[225] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[226] Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*, pages 1–10. 2020.

[227] Charles Rich, Brett Ponsler, Aaron Holroyd, and Candace L Sidner. Recognizing engagement in human-robot interaction. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 375–382. IEEE, March 2010.

[228] Nicole L Robinson, Jennifer Connolly, Genevieve M Johnson, Yejee Kim, Leanne Hides, and David J Kavanagh. Measures of incentives and confidence in using a social robot. *Science Robotics*, 3(21):eaat6963, 2018.

[229] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[230] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[231] Stuart J. (Stuart Jonathan) Russell. *Artificial intelligence : a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, N.J, 3rd ed. edition.

[232] Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[233] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[234] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. *Active preference-based learning of reward functions*. 2017.

[235] Jyotirmay Sanghvi, Ginevra Castellano, Iolanda Leite, André Pereira, Peter W McOwan, and Ana Paiva. Automatic analysis of affective postures and body motion to detect engagement with a game companion. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 305–312. ACM, 2011.

[236] Sreeshankar Satheeshbabu, Naveen K Uppalapati, Tianshi Fu, and Girish Krishnan. Continuous control of a soft continuum arm using deep reinforcement learning. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 497–503. IEEE, 2020.

[237] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[238] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[239] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[240] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[241] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[242] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013.

[243] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[244] Thomas B. Sheridan. Human–robot interaction: Status and challenges. *Human Factors*, 58(4):525–532, 2016. PMID: 27098262.

[245] Chao Shi, Satoru Satake, Takayuki Kanda, and Hiroshi Ishiguro. A robot that distributes flyers to pedestrians in a shopping mall. *International Journal of Social Robotics*, 10(4):421–437, Sep 2018.

[246] Masahiro Shiomi, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. Interactive humanoid robots for a science museum. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, HRI '06, pages 305–312, New York, NY, USA, 2006. ACM.

[247] Candace L. Sidner, Cory D. Kidd, Christopher Lee, and Neal Lesh. Where to look: A study of human-robot engagement. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pages 78–84, New York, NY, USA, 2004. ACM.

[248] Candace L Sidner and Christopher Lee. Engagement rules for human-robot collaborative interactions. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, volume 4, pages 3957–3962 vol.4, Oct 2003.

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

[250] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR.

[251] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[252] Herbert A Simon. The new science of management decision. 1960.

[253] Herbert A Simon. Theories of decision-making in economics and behavioural science. In *Surveys of economic theory*, pages 1–28. Springer, 1966.

[254] Gautam Singh, Skand Peri, Junghyun Kim, Hyunseok Kim, and Sungjin Ahn. Structured world belief for reinforcement learning in pomdp. In *International Conference on Machine Learning*, pages 9744–9755. PMLR, 2021.

[255] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[256] Angela Sinickas. Finding a cure for survey fatigue. *Strategic Communication Management*, 11(2):11, 2007.

[257] William D Smart and L Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3404–3410. IEEE, 2002.

[258] Doo Re Song, Chuanyu Yang, Christopher McGreavy, and Zhibin Li. Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 311–318. IEEE, 2018.

[259] David St-Onge, Pierre-Yves Brches, Inna Sharf, Nicolas Reeves, Ioannis Rekleitis, Patrick Abouzakhm, Yogesh Girdhar, Adam Harmat, Gregory Dudek, and Philippe Gigure. Control, localization and human interaction with an autonomous lighter-than-air performer. *Robot. Auton. Syst.*, 88(C):165–186, February 2017.

[260] David St-Onge and Nicolas Reeves. Human interaction with flying cubic automata. In *Proceedings of 2010 IEEE/ACM Internations Conference on Human Robots Interaction*, 2010.

[261] Clara Kwon Starkweather, Benedicte M Babayan, Naoshige Uchida, and Samuel J Gershman. Dopamine reward prediction errors reflect hidden-state inference across time. *Nature neuroscience*, 20(4):581–589, 2017.

[262] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich. Common metrics for human-robot interaction. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, HRI '06, pages 33–40, New York, NY, USA, 2006. ACM.

[263] Halit Bener Suay and Sonia Chernova. Effect of human guidance and state space size on interactive reinforcement learning. In *RO-MAN, 2011 IEEE*, pages 1–6. IEEE, July 2011.

[264] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[265] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[266] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[267] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

[268] Victor Talpaert, Ibrahim Sobh, B Ravi Kiran, Patrick Mannion, Senthil Yogamani, Ahmad El-Sallab, and Patrick Perez. Exploring applications of deep reinforcement learning for real-world autonomous driving systems. *arXiv preprint arXiv:1901.01536*, 2019.

[269] Yunhao Tang. Self-imitation learning via generalized lower bound q-learning. *Advances in neural information processing systems*, 33:13964–13975, 2020.

[270] Andrea Thomaz, Guy Hoffman, and Maya Cakmak. Computational human-robot interaction. *Found. Trends Robot*, 4(2-3):105–223, December 2016.

[271] Andrea L. Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 1000–1005. AAAI Press, 2006.

[272] Andrea Lockerd Thomaz, Guy Hoffman, and Cynthia Breazeal. Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*, 2005.

[273] Robert L Thorndike. Who belongs in the family. In *Psychometrika*. Citeseer, 1953.

[274] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.

[275] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[276] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

[277] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[278] Panagiota Tsarouchi, Sotiris Makris, and George Chryssolouris. Human–robot interaction review and challenges on task planning and programming. *International Journal of Computer Integrated Manufacturing*, 29(8):916–931, 2016.

[279] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[280] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[281] Harm van Seijen. Effective multi-step temporal-difference learning for non-linear function approximation. *arXiv preprint arXiv:1608.05151*, 2016.

[282] George Velentzas, Theodore Tsitsimis, Iñaki Rañó, Costas Tzafestas, and Mehdi Khamassi. Adaptive reinforcement learning with active state-specific exploration for engagement maximization during simulated child-robot interaction. *Paladyn, Journal of Behavioral Robotics*, 9(1):235–253, 2018.

[283] Autilia Vitiello, Giovanni Acampora, Mariacarla Staffa, Bruno Siciliano, and Silvia Rossi. A neuro-fuzzy-bayesian approach for the adaptive control of robot proxemics behavior. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2017.

[284] Graham Wakefield, Tobias Hollerer, JoAnn Kuchera-Morin, Charles Roberts, and Matthew Wright. Spatial interaction in a multiuser immersive instrument. *IEEE computer graphics and applications*, 33(6):14–20, Nov 2013.

[285] Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019.

[286] Chen Wang, Yanan Li, Shuzhi Sam Ge, and Tong Heng Lee. Adaptive control for robot navigation in human environments based on social force model. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5690–5695. IEEE, 2016.

[287] Tixian Wang, Amirhossein Taghvaei, and Prashant G Mehta. Q-learning for pomdp: An application to learning locomotion gaits. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 2758–2763. IEEE, 2019.

[288] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[289] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[290] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[291] Nils Wilde, Erdem Bıyık, Dorsa Sadigh, and Stephen L Smith. Learning reward functions from scale feedback. *arXiv preprint arXiv:2110.00284*, 2021.

[292] Nils Wilde, Dana Kulic, and Stephen L Smith. Active preference learning using maximum regret. *arXiv preprint arXiv:2005.04067*, 2020.

[293] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25, 2012.

[294] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

[295] Christian Wirth and Johannes Fürnkranz. Preference-based reinforcement learning: A preliminary survey. In *Proceedings of the ECML/PKDD-13 Workshop on Reinforcement Learning from Generalized Feedback: Beyond Numeric Rewards*. Citeseer, 2013.

[296] Leo Woiceshyn, Yuchi Wang, Goldie Nejat, and Beno Benhabib. Personalized clothing recommendation by a social robot. In *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 179–185. IEEE, 2017.

[297] James E Young, JaYoung Sung, Amy Voida, Ehud Sharlin, Takeo Igarashi, Henrik I Christensen, and Rebecca E Grinter. Evaluating human-robot interaction. *International Journal of Social Robotics*, 3(1):53–67, Jan 2011.

[298] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[299] Fangyi Zhang, Jürgen Leitner, Michael Milford, Ben Upcroft, and Peter Corke. Towards vision-based deep reinforcement learning for robotic motion control. *arXiv preprint arXiv:1511.03791*, 2015.

[300] Juanjuan Zhang and Chien Chern Cheah. Passivity and stability of human–robot interaction control for upper-limb rehabilitation robots. *IEEE Transactions on Robotics*, 31(2):233–245, 2015.

[301] Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning deep neural network policies with continuous memory states. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 520–527. IEEE, 2016.

[302] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.

[303] Jakub Złotowski, Diane Proudfoot, Kumar Yogeeswaran, and Christoph Bartneck. Anthropomorphism: opportunities and challenges in human–robot interaction. *International Journal of Social Robotics*, 7(3):347–360, Jan 2015.

APPENDICES

Appendix A

Appendix for Chapter 5

A.1 Pseudo-code for DDPG-based PLA

ALGORITHM 4: DDPG-based PLA

Input: $first\_day$

Initialize: $train\_interval$, $train\_times$, $episode\_length$, replay buffer $\mathcal{D}$, mini-batch size N
if $first\_day$ **then**
    Initialize Critic $Q(obs, a|\theta^Q)$ and Actor $\mu(obs|\theta^\mu)$ with random parameters θ^Q, θ^μ
else
    Load pretrained Critic $Q(obs, a|\theta^Q)$ and Actor $\mu(obs|\theta^\mu)$
end
Target networks $\theta^{Q-} \leftarrow \theta^Q$, $\theta^{\mu-} \leftarrow \theta^\mu$
$start\_of\_day \leftarrow True$
while *True* **do**
    if $start\_of\_day$ **then**
        Receive initial observation $obs^{(1)}$
        Initialize parameter noise σ
        $start\_of\_day \leftarrow False$
    else
        $obs^{(1)} \leftarrow obs^{(t+1)}$, $\sigma \leftarrow \sigma_{j+1}$
    end
    for $t \leftarrow 1$ **to** $episode\_length$ **do**
        Select action $a^{(t)} = \mu(obs^{(t)}|\theta^\mu + \mathcal{N}(0, \sigma))$ according to the current policy and exploration noise.
        Execute action $a^{(t)}$, observe reward $r^{(t)}$ and observe new observation $obs^{(t+1)}$
        $\mathcal{D} \leftarrow \mathcal{D} \cup (obs^{(t)}, a^{(t)}, r^{(t)}, obs^{(t+1)})$
        if $t\%train\_interval == 0$ *and* $|\mathcal{D}| \geq N$ **then**
            for $j \leftarrow 1$ **to** $train\_times$ **do**
                Sample a random minibatch of N trainsitions $(obs^{(i)}, a^{(i)}, r^{(i)}, obs^{(i+1)})$ from $\mathcal{D}$
                `/* Update adaptive parameter noise scale */`
                `/* Update actor-critic */`
            end
        end
        $obs^{(t)} \leftarrow obs^{(t+1)}$, $\sigma_1 \leftarrow \sigma_{j+1}$
    end
end

Appendix B

Appendix for Chapter 7

B.1 Algorithms Implementation

The implementation of the algorithms is based on OpenAI Spinningup[1]. The code used for this work can be found in https://github.com/LinghengMeng/LSTM-TD3. Table C.2 details the hyperparameters used in this work, where − indicates the parameter does not apply to the corresponding algorithm. For the actor and critic neural network structure of LSTM-TD3, the first row corresponds to the structure of the memory component, the second row corresponds to the structure of the current feature extraction, and the third row corresponds to the structure of perception integration after combining the extracted memory and the extracted current feature.

B.2 Supplementary Results

B.2.1 Performance Comparison

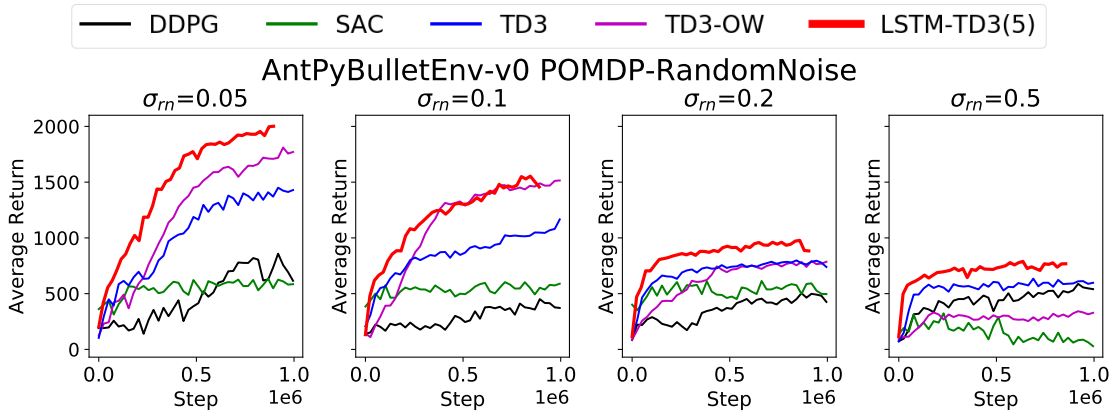
Table B.2 summarizes the maximum average return of each algorithm on different tasks, where the POMDP-FLK, POMDP-RN, and POMDP-RSM are examined with $p_{flk} = 0.2$, $\sigma_{rn} = 0.1$, and $p_{rsm} = 0.1$. From Table B.2, we can see that LSTM-TD3 either outperforms or achieves comparative performance to other baselines on MDP, and significantly outperforms other baselines on POMDP-versions of each task. These results provide evidence

[1]https://spinningup.openai.com

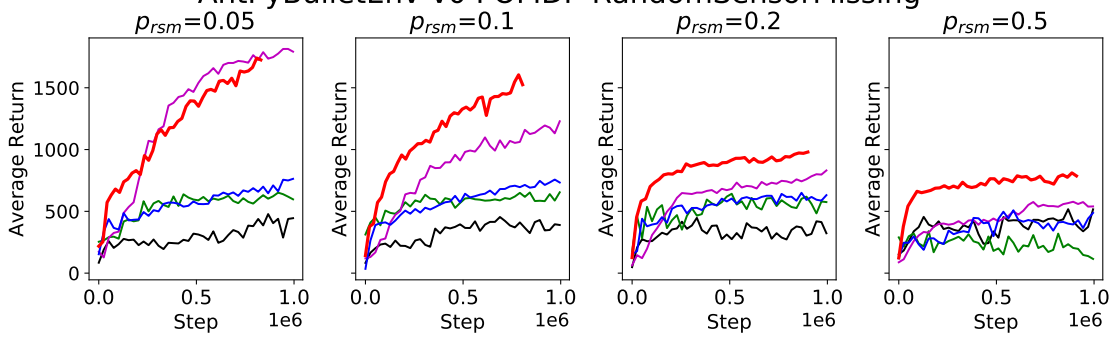
Table B.1: Hyperparameters for Algorithms

Hyperparameter	**Algorithms**			
	DDPG	**TD3**	**SAC**	**LSTM-TD3**
discount factor: γ	0.99			
batch size: N_{batch}	100			
replay buffer size: $\|D\|$	10^6			
random start step: N_{start_step}	10000			
update after N_{update_after}	1000			
target NN update rate τ	0.005			
optimizer	Adam [146]			
actor learning rate lr_{actor}	10^{-3}			
critic learning rate lr_{critic}	10^{-3}			
actor NN structure:	[256, 256]			[128] + [128] [128] [128, 128]
critic NN structure:	[256, 256]			[128] + [128] [128] [128, 128]
actor exploration noise σ_{act}	0.1		-	0.1
target actor noise σ_{targ_act}	-	0.2	-	0.2
target actor noise clip boundary c_{targ_act}	-	0.5	-	0.5
policy update delay	-	2	-	2
entropy regulation coefficient α	-	-	0.2	-
history length l	-	-	-	{0, 1, 3, 5}

of the promising advantages of memory based DRL on both MDP and POMDP. On one hand, LSTM-TD3 can be used out-of-box without caring too much of the design of the



(a) POMDP-RN



(b) POMDP-RSM

Figure B.1: Performance Comparison on POMDP-version of AntPyBulletEnv-v0 with different observabilities.

observation as memory component can partially compensate the lost information. On the other hand, by comparing the performance difference of adding and removing the memory component, LSTM-TD3 provides a way to detect if the current design of the observation space is improvable or not in terms of capturing the underlying state.

B.2.2 Robustness to Partial Observability

Fig. B.1 compares the proposed LSTM-TD3 with the baselines in terms of the robustness to different partial observabilities, where the higher the σ_{rn} and the p_{rsm}, the lower the observability. In general, LSTM-TD3 has better robustness than TD3-OW which has

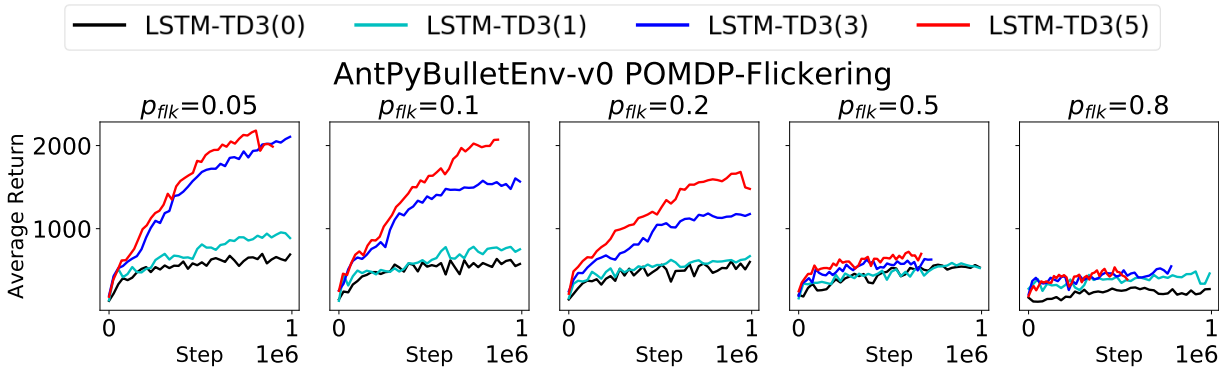


Figure B.2: Relationship Between Partial-Observability and History Length.

better robustness than TD3. A small reduction of observability can be handled by LSTM-TD3, but for POMDP with severe reduction of observability LSTM-TD3 performance also degrades.

B.2.3 Effect of History Length

The history length l is a hyperparameter of LSTM-TD3, and it determines the maximum history length in the observation window. As illustrated in Fig. B.4 and Table B.2, LSTM-TD3 with a relatively short history length $l = 5$ produces significantly better performance than other baselines without a memory component. Fig. B.2 shows the performance of LSTM-TD3 with different history lengths on POMDP-FLK AntPyBulletEnt-v0 with various flickering probabilities $p_{flk} = \{0.05, 0.1, 0.2, 0.5, 0.8\}$ where the higher the p_{flk} the lower the obervability. For $p_{flk} = 0.05$, LSTM-TD3(3) and LSTM-TD3(5) show similar performance and both significantly outperform LSTM-TD3(0) and LSTM-TD3(1). When p_{flk} increases from 0.05 to 0.1, LSTM-TD3(5) still maintains similar performance, but LSTM-TD3(3) experiences a dramatic decrease. This means the decreased observability can still be compensated with history of length 5, but cannot be compensated with history of length 3. When further reducing the observability to $p_{flk} = 0.2$, the performance of LSTM-TD3(5) is also degraded, as shown in the 3rd panel of Fig. B.2. When p_{flk} is increased to 0.5 and 0.8, all examined history lengths fail the task (the last two panels in Fig. B.2).

Fig. B.3 illustrates the performance of LSTM-TD3 with different history lengths. From this figure, we can see that when the history length increased, the final performance improves too. However, the long history length causes much extra computation consumption.

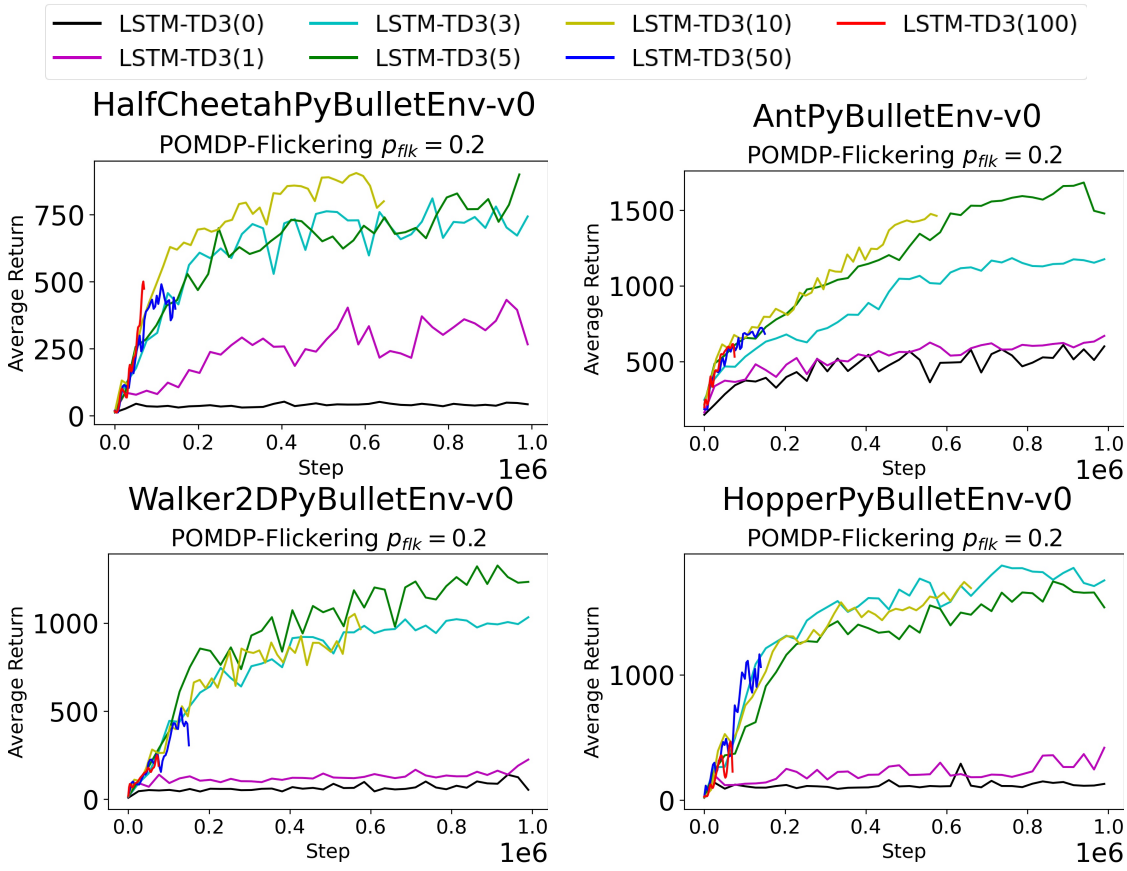


Figure B.3: Performance of LSTM-TD3 with Different History Lengths, where LSTM-TD3 with history length 10, 50 and 100 are not fully run up to 1 million steps due to the extra computation cost caused by long history.

B.2.4 Policy Generalization

The second extension is to evaluate the learned policy with different history length from that used during training. If we use l_{train} and l_{eval} to represent the history length used during training and evaluation respectively, the generalization capability for both $l_{train} > l_{eval}$ and $l_{train} < l_{eval}$ can be useful in different scenarios. On one hand, for $l_{train} > l_{eval}$, if l_{train} and l_{eval} can achieve the same performance, using a shorter history length during evaluation can reduce the inferring time of an action, and this is valuable for tasks where real-time decision making is important and training can be run in parallel and is not time-sensitive. On the other hand, for $l_{train} < l_{eval}$, if l_{eval} can achieve better

performance than l_{train} by increasing the history length, which means how to extract useful memory information can be learned with a shorter history length and longer history length during evaluation is only to extract more useful information, then using a shorter history length during training can speed up the training and reduce resource consumption, at the same time without compromising the performance. As for other DRL algorithms, the training is on a mini-batch while the evaluating is only on a single data, so if the l_{train} is large, the training will take more time and computation resources than evaluation. Therefore, it a good choice to achieve the same performance by reducing the cost during training and increasing the cost during evaluation. Normally, training cost is unvalued by researchers, especially some research [4, 138] proposed to use large distributed computer cluster. However, there are cases, where the robot cannot communicate fleetly with the remote computation center and onboard computation resources are limited, that the effort in saving computation is still required. This practical consideration inspires the second evaluation extension.

Evaluation with Different History Lengths

Fig. B.5 illustrates the evaluation results of LSTM-TD3 with various history lengths that may be different from the history length used for training. From this figure, it can be seen that when trained with a specific history length l_{train} but evaluated with a different history length $l_{eval} > l_{train}$, the performance remains at the same level. However, when evaluated with a history $l_{eval} < l_{train}$, the performance cannot be guaranteed, as for some cases a shorter evaluation history length will cause dramatic decrease in performance, e.g. LSTM-TD3(3) on POMDP-RN (the blue line in the 4th panel of Fig. B.5), while for others a shorter evaluation history length can still achieve similar performance, e.g. LSTM-TD(5) on POMDP-RN and on POMDP-RSM (the red line in the 4th and 5th panels of Fig. B.5). Based on this observation, it seems the performance can be generalized to a longer evaluation history length rather than a shorter one. It is worth to note that the longest training history length investigate here is only 5, so more valuable insights may be found with more results with longer training history length.

The results that when evaluating with history length longer than 0, the performance does not change (the cyan lines in Fig. B.5) of LSTM-TD3(0), are very interesting, because in the training phase LSTM-TD3(0) takes zero-valued dummy observation and action as history as defined in Eq. 7.1 and this dummy history cannot provide any useful information about how to extracting memory that is useful to current task. Therefore, when replacing this dummy history with real history in the replay buffer, the extracted memory can be anything and will disturb the decision making, which makes us to expect that when

evaluating the learned policy with history length longer than 0, the performance will be decreased. However, the results is surprisingly remained at the same level as that for LSTM-TD3(0). This reminds us that LSTM-TD3(0) may have learned a policy that intentionally ignores the history. To validate this conjecture, we plotted the average extracted memory of the actor in Fig. B.6c. As shown in Fig. B.6c, the average extracted memory of the actor of LSTM-TD3(0) initially starts with a non-zero value, but after a few thousands steps its value remains at a value around 0.0034 ± 0.0024, which is very close to 0, whereas for other LSTM-TD3s with longer history length the average extracted memories are relatively far away from 0 and have relatively large standard deviation. And this observation is consistent for both MDP and POMDPS. Even though we cannot claim the 0 in the extracted memory can be interpreted as neglect of past history, but at least we can say the history is uniformly mapped to a roughly fixed value rather than a random value for each history. In this way, even replacing the zero-value dummy history with a real history of experiences, the performance will not bad than that for a dummy history.

B.2.5 A Glance of The Relationship Among the Return, the Predicted Q-value, and the Extracted Memory of the Actor-Critic

The proposed LSTM-TD3 has been experimentally proved to be useful according to the results presented in Section B.2.1, but the understanding of the LSTM-TD3, especially the interpretation of the extracted memory, is still a big challenge. In Fig. B.6, we presented the average test return, the average predicted Q-value, the average extracted memory of actor and critic to have a glance of the relationship among them.

By comparing Fig. B.6a and B.6b, we found that the average test return and the average predicted Q-value match approximately perfect, which means there is no overestimation problem as studied in [274, 89, 180] and is desirable.

Our special interest is in the relationship between the test return and the extracted memory of the actor-critic. It is worth to note that both the actor and the critic have a memory component and there is no sharing of the memory component, which means they may learn different coding of the memory that is helpful for learning a Q-value function and a policy, respectively. As shown in Fig. B.6c and B.6d, no matter if we comparing these plots horizontally or vertically, there is no consistent trend can be found. Horizontally, for both the actor and the critic, the average extracted memories for different versions, i.e. the MDP and the various POMDPs, of the task are showing different trends, where an exception is the extracted memory of the actor of LSTM-TD3(0) as discussed in the

previous section B.2.4. Vertically, for different versions of the task, the extracted memories of the actor and the critic show different trends too. Especially, the average extracted memory of the critic of the LSTM-TD3(0) is not similar with that of the actor whose value is consistently close to 0. Contrarily, the average extracted memory of the critic of LSTM-TD3(0) is even further from 0 than that of LSTM-TD3(5). This may indicate the different roles of the memory component playing in the actor and the critic. Again, the interpretation of the extracted memory is not so straight-forward, and special constraints may be forced to improve the interpretability of the memory component, which is out of the scope of this paper and will be left for the future study.

B.2.6 Supplementary results for the Ablation Study

Fig. B.7 shows the learning curves of various ablated LSTM-TD3 by removing different components, namely (1) using double critics (DC), (2) using target policy smoothing (TPS), (3) having current feature extraction (CFE) component, and (4) including past actions (PA) in the history. Table B.3 reports the maximum average return of the investigated ablated algorithms.

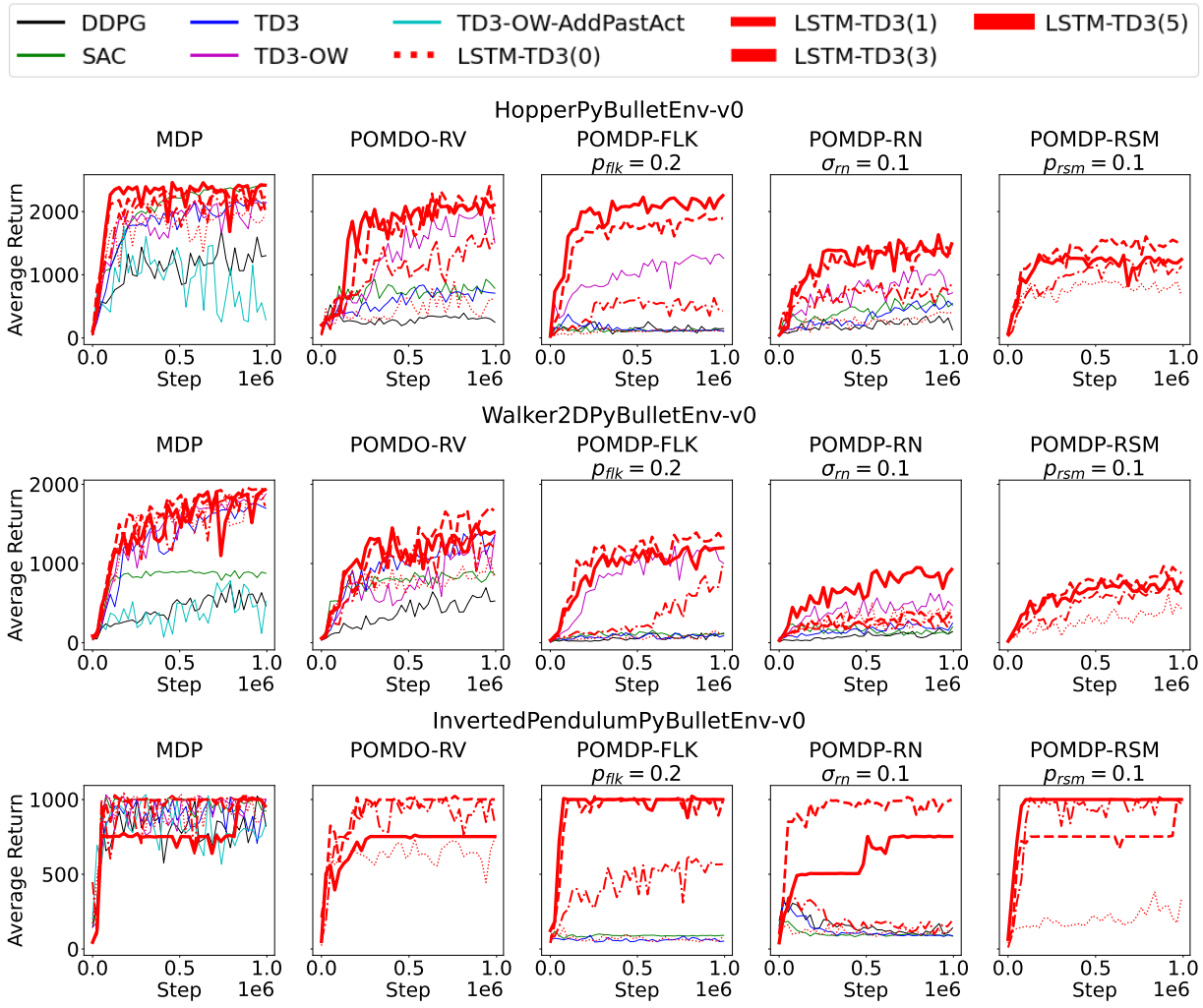


Figure B.4: Learning curves for PyBulletGym tasks, where to ease the comparison only average values are plotted. In the legend, the value in the bracket of LSTM-TD3 indicate the length of the history, e.g. LSTM-TD3(5) uses the history length 5.

Table B.2: Maximum Average Return over 10 evaluation episodes based on 4 different random seeds. Maximum value of evaluated algorithms for each task is bolded, and ± indicates a single standard deviation. LSTM-TD3 (5) corresponds to the LSTM-TD3 with the history length $l = 5$.

Task		**Algorithms**				
Name	Version	**DDPG**	**TD3**	**SAC**	**TD3-OW**	**LSTM-TD3 (5)**
HalfChee	MDP	487.6 ± 6.1	**1311.9 ± 49.7**	663.5 ± 30.8	1265.5 ± 8.9	1223.0 ± 582.3
	POMDP-RV	508.4 ± 23.9	1151.7 ± 74.9	631.5 ± 57.1	**1161.3 ± 17.2**	918.4 ± 44.0
	POMDP-FLK	84.8 ± 20.4	82.4 ± 45.7	117.0 ± 42.2	**1559.61 ± 559.9**	848.1 ± 60.2
	POMDP-RN	268.7 ± 70.2	501.9 ± 47.5	328.4 ± 62.1	703.9 ± 21.7	**771.6 ± 18.2**
	POMDP-RSM	283.7 ± 27.0	538.9 ± 32.2	587.4 ± 44.3	606.9 ± 13.4	**954.0 ± 362.9**
Ant	MDP	1210.8 ± 226.1	2433.5 ± 288.5	980.8 ± 96.3	2289.5 ± 154.8	**2574.9 ± 79.0**
	POMDP-RV	683.5 ± 101.4	1765.6 ± 2.2	800.4 ± 4.8	1265.3 ± 65.2	**1932.7 ± 199.6**
	POMDP-FLK	449.0 ± 93.3	654.4 ± 1.6	529.7 ± 23.7	1390.5 ± 736.6	**2036.7 ± 73.5**
	POMDP-RN	449.6 ± 18.5	1165.8 ± 59.0	620.8 ± 10.0	1520.1 ± 8.4	**1966.1 ± 171.4**
	POMDP-RSM	465.2 ± 51.0	763.7 ± 103.3	659.1 ± 3.1	1230.4 ± 124.0	**1324.9 ± 313.6**
Walker2D	MDP	835.0 ± 102.2	1783.1 ± 111.6	930.1 ± 53.2	1941.3 ± 128.5	**1970.5 ± 38.5**
	POMDP-RV	716.6 ± 224.5	1477.9 ± 164.3	921.9 ± 20.8	1220.9 ± 53.8	**1479.9 ± 283.2**
	POMDP-FLK	142.4 ± 29.6	181.9 ± 98.7	217.2 ± 90.6	1238.6 ± 385.4	**1264.9 ± 338.5**
	POMDP-RN	197.2 ± 96.2	295.8 ± 44.7	278.9 ± 44.5	648.3 ± 129.5	**984.7 ± 267.7**
	POMDP-RSM	283.6 ± 31.0	519.4 ± 17.5	630.5 ± 20.8	633.2 ± 23.2	**841.2 ± 91.6**
Hopper	MDP	1699.6 ± 80.2	2201.3 ± 180.4	2424.5 ± 85.4	2210.1 ± 286.4	**2465.0 ± 158.9**
	POMDP-RV	520.6 ± 105.3	926.0 ± 219.6	1145.8 ± 162.1	2212.1 ± 5.5	**2233.6 ± 176.6**
	POMDP-FLK	259.5 ± 63.9	401.1 ± 39.8	243.2 ± 161.3	1353.0 ± 467.8	**2264.6 ± 72.3**
	POMDP-RN	400.8 ± 62.8	644.2 ± 46.5	782.0 ± 65.2	962.0 ± 10.5	**1635.8 ± 180.7**
	POMDP-RSM	596.1 ± 57.1	873.2 ± 7.9	892.3 ± 2.5	1193.7 ± 193.3	**1349.1 ± 405.7**
InvPen	MDP	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**
	POMDP-RV	912.5 ± 13.5	**944.1 ± 41.6**	891.3 ± 108.7	876.3 ± 123.7	752.6 ± 428.5
	POMDP-FLK	147.1 ± 120.1	158.1 ± 115.5	121.1 ± 23.0	**1000.0 ± 0.0**	**1000.0 ± 0.0**
	POMDP-RN	422.5 ± 18.8	342.8 ± 46.8	289.8 ± 19.3	**1000.0 ± 0.0**	752.6 ± 428.5
	POMDP-RSM	624.9 ± 4.0	381.5 ± 180.1	445.0 ± 101.5	**1000.0 ± 0.0**	**1000.0 ± 0.0**
InvDouPen	MDP	4746.3 ± 4607.1	7054.3 ± 2304.5	9357.3 ± 0.3	9358.6 ± 0.3	**9359.77 ± 0.1**
	POMDP-RV	750.6 ± 130.4	953.6 ± 92.0	2027.6 ± 508.6	7998.1 ± 653.7	**9358.9 ± 0.3**
	POMDP-FLK	274.2 ± 62.8	382.6 ± 105.4	404.9 ± 11.8	**9358.6 ± 0.5**	9358.4 ± 1.1
	POMDP-RN	506.5 ± 159.3	470.4 ± 330.3	662.7 ± 34.7	**2005.3 ± 13.4**	1952.7 ± 503.4
	POMDP-RSM	881.7 ± 364.4	829.5 ± 96.1	1084.9 ± 103.6	**9304.4 ± 54.6**	9156.1 ± 348.6

For POMDP-FLK, $p_{flk} = 0.2$. For POMDP-RN, $\sigma_{rn=0.1}$. For POMDP-RSM, $p_{rsm} = 0.1$.

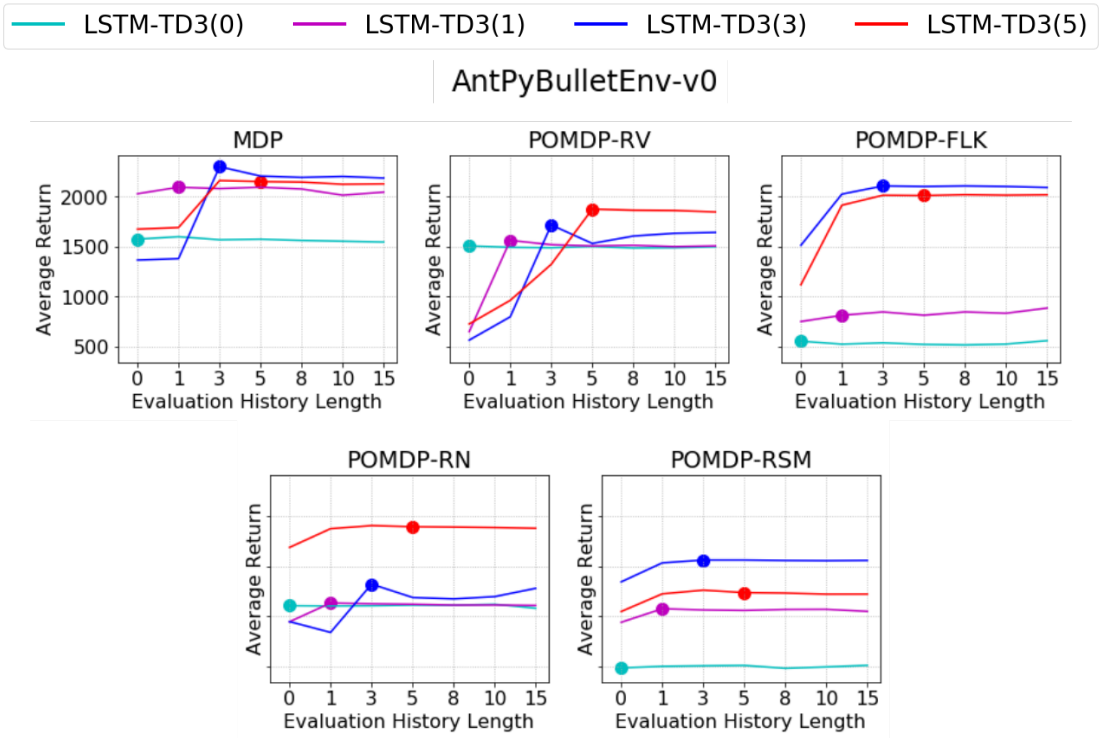
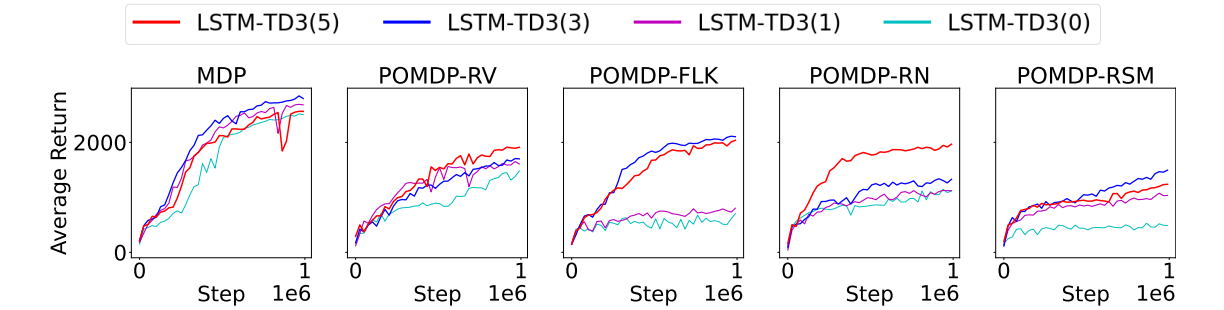
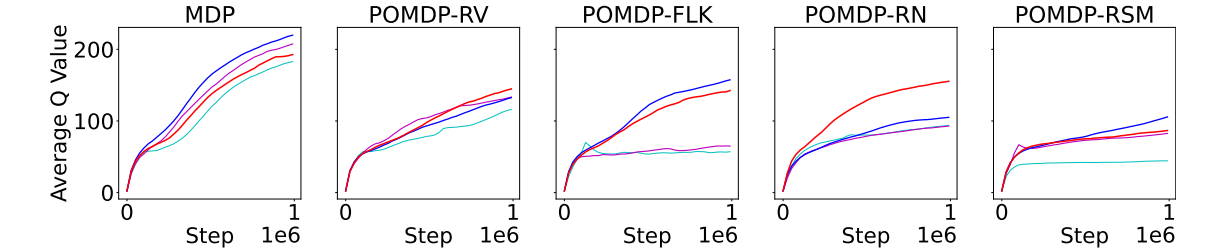


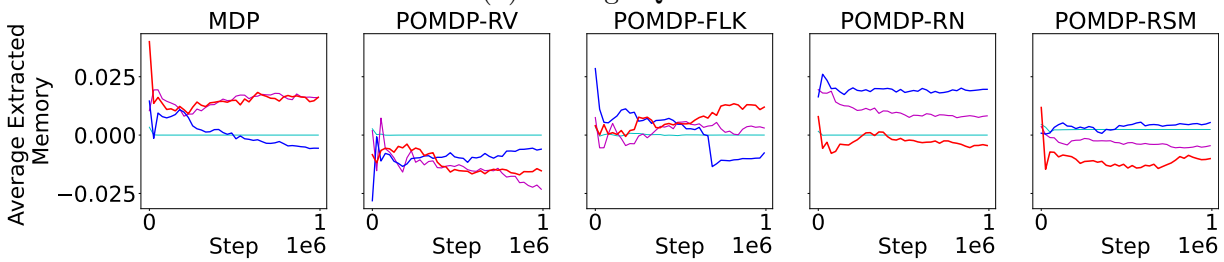
Figure B.5: Evaluation with History Length Different From that Used When Training, where in each panel the title indicates the version of the task, the x-axis corresponds to the history length used during evaluation, each line corresponds to the policy trained with a specific history length, and the marker indicates the point where the training and evaluation history length are the same.



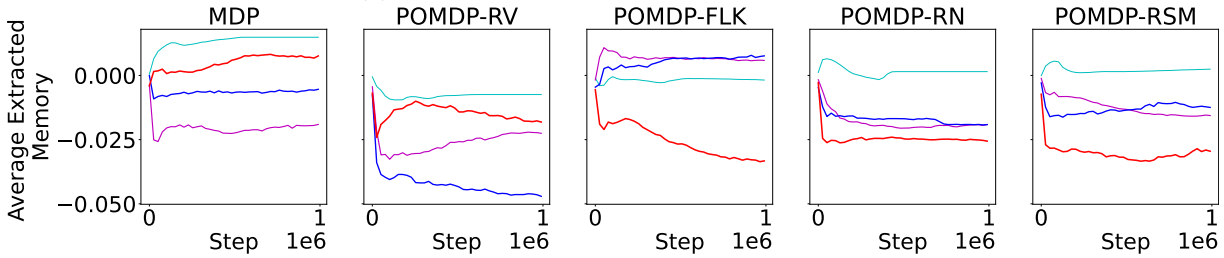
(a) Average Test Return



(b) Average Q Value



(c) Average Extracted Memory of Actor



(d) Average Extracted Memory of Critic

Figure B.6: Relationship Among the Return, Predicted Q-value, Extracted Memory of Actor-Critic, where (a), (b), (c), and (d) shows the average test return, the average predicted Q-value, the average extracted memory of actor, and the average extracted memory of critic. The average extracted memory of actor and critic is an average over all output neurons of the memory component.

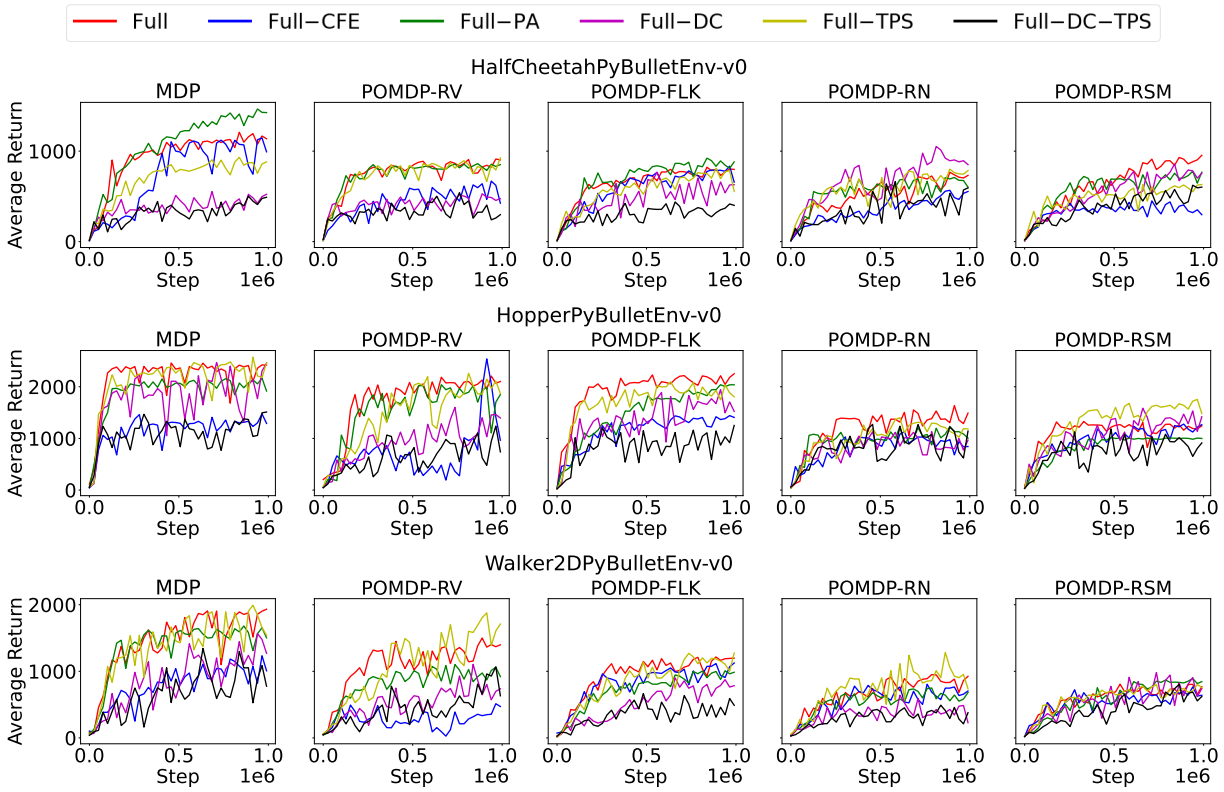


Figure B.7: Learning curves of ablation study, where to ease the comparison only average values are plotted. In the legend, Full, Full−CFE, Full−PA, Full−DC, Full−TPS, and Full−DC−TPS correspond to LSTM-TD3 with full components, removing current feature extraction, excluding past action, not using double critics, not using target policy smoothing, and simultaneously not using double critics and target policy smoothing.

Table B.3: Maximum Average Return for Ablation Study. Maximum value of evaluated algorithms for each task is bolded, and ± indicates a single standard deviation.

Task		Algorithms					
		LSTM-TD3	**Full−CFE**	**Full−PA**	**Full−DC**	**Full−TPS**	**Full−DC−TPS**
HalfChee	M	1223.0	1182.2 ± 522.0	**1469.6 ± 637.5**	569.1 ± 78.5	920.6 ± 48.8	517.4 ± 102.0
	P1	918.4	680.6 ± 129.9	871.0 ± 43.6	589.5 ± 73.8	**945.6 ± 187.5**	552.0 ± 1.4
	P2	848.1	844.7 ± 211.3	**946.9 ± 242.1**	836.3 ± 346.7	809.9 ± 42.7	690.8 ± 0.0
	P3	771.6	641.5 ± 40.2	742.9 ± 79.2	**1222.2 ± 313.4**	808.3 ± 468.1	731.1 ± 330.9
	P4	954.0	457.0 ± 99.1	784.0 ± 39.7	**958.1 ± 229.1**	693.9 ± 79.0	606.3 ± 63.0
Ant	M	2574.9	1510.8 ± 421.4	2421.0 ± 305.4	2121.1 ± 334.6	**2658.6 ± 1537.1**	1855.8 ± 494.2
	P1	**1932.7**	1133.2 ± 421.6	1535.0 ± 358.1	1711.9 ± 331.5	1814.3 ± 90.3	1068.6 ± 363.0
	P2	2036.7	1817.2 ± 323.0	1578.1 ± 466.2	2023.9 ± 348.0	2083.8 ± 159.9	**2145.1 ± 107.2**
	P3	**1966.1**	1705.7 ± 105.8	1287.4 ± 366.0	1588.0 ± 548.0	1885.8 ± 92.7	879.3 ± 446.9
	P4	1324.9	1737.4 ± 310.3	1817.6 ± 66.4	1728.8 ± 494.0	1730.0 ± 474.4	**1831.7 ± 33.9**
Walker2D	M	1970.5	1230.9 ± 329.2	1719.6 ± 431.1	1526.3 ± 96.5	**2011.7 ± 61.6**	1381.9 ± 801.0
	P1	1479.9	520.2 ± 85.4	1100.9 ± 243.6	1214.2 ± 313.7	**1871.7 ± 138.3**	1349.3 ± 101.1
	P2	1264.9	1132.6 ± 256.2	1027.3 ± 214.5	1017.1 ± 122.6	**1320.2 ± 275.9**	731.5 ± 160.0
	P3	984.7	844.9 ± 282.2	803.5 ± 22.9	662.7 ± 155.2	**1240.7 ± 228.2**	576.6 ± 403.8
	P4	841.2	762.9 ± 53.0	896.1 ± 46.8	**1070.8 ± 226.4**	820.8 ± 45.4	1010.2 ± 324.9
Hopper	M	2465.0	1517.0 ± 673.8	2178.2 ± 200.7	2504.7 ± 180.5	**2613.1 ± 81.5**	2015.0 ± 477.6
	P1	**2233.6**	1187.0 ± 749.1	2188.9 ± 240.2	1775.5 ± 471.6	2303.9 ± 156.6	1852.9 ± 327.4
	P2	**2264.6**	1459.0 ± 513.2	2055.9 ± 48.1	2087.9 ± 194.0	2117.9 ± 103.1	2083.3 ± 168.7
	P3	**1635.8**	1143.9 ± 298.7	1279.7 ± 444.7	1337.7 ± 262.1	1450.7 ± 357.3	1524.0 ± 516.0
	P4	1349.1	1343.1 ± 411.3	1008.6 ± 3.4	1634.3 ± 338.6	1769.5 ± 210.6	**1938.2 ± 0.0**
InvPen	M	**1000.0**	874.2 ± 217.9	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**
	P1	752.6	804.4 ± 246.2	752.6 ± 428.6	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**
	P2	**1000.0**	1000.0 ± 0.0	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 577.4**	**1000.0 ± 0.0**
	P3	752.6	752.7 ± 428.3	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	986.5 ± 19.0
	P4	**1000.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**	**1000.0 ± 0.0**
InvDouPen	M	**9359.73**	8677.9 ± 749.7	9359.0 ± 5403.4	9343.9 ± 5394.7	9358.8 ± 9358.8	9352.5 ± 5399.7
	P1	**9358.9**	9358.0 ± 1.0	5568.0 ± 4941.9	4748.1 ± 6616.0	9357.7 ± 5402.7	8974.0 ± 377.3
	P2	9358.4	9358.0 ± 1.1	9357.4 ± 1.9	9356.2 ± 1.2	9357.1 ± 5402.3	**9359.4 ± 0.0**
	P3	1952.7	1534.0 ± 462.1	1308.5 ± 1220.5	2453.9 ± 1576.2	2360.5 ± 1403.6	**2544.9 ± 3599.0**
	P4	9156.1	9139.5 ± 377.1	**9358.7 ± 0.5**	9156.9 ± 333.3	9357.7 ± 0.8	9355.0 ± 3.8

M: MDP, P1: POMDP-RV, P2: POMDP-FLK, P3: POMDP-RN, P4: POMDP-RSM.
The variance of the performance of LSTM-TD3 can be found in Table B.2.
For POMDP-FLK, $p_{flk} = 0.2$. For POMDP-RN, $\sigma_{rn=0.1}$. For POMDP-RSM, $p_{rsm} = 0.1$.
CFE: Current Feature Extraction; PA: Past Action; DC: Double Critics; TPS: Target Policy Smooth

Appendix C

Appendix for Chapter 8

C.1 The Potential Effect of Multi-step Bootstrapping on Passing Temporal Information

Table C.1 summarises the meaning of the key components of RL under the original and new interpretation of the reward function.

Table C.1: Interpretations of Reward Function

	Original Interpretation	**New Interpretation**
$R(o, a, o')$	the reward of taking action a in observation o which leading to a new observation o'	the 1 dimensional state-transition abstraction of taking action a in observation o leading to a new observationo' (a state-transition abstraction feature)
$Q^\pi(o, a)$	the expectation of the accumulated discounted reward when taking action a in observation o and following policy π thereafter	the expectation of the accumulated discounted state-transition abstraction when taking action a in observation o and following policy π thereafter
$V^\pi(o)$	the expectation of the accumulated discounted reward when in observation o and following policy π thereafter	the expectation of the accumulated discounted state-transition abstraction when in observation o and following policy π thereafter
$\pi(a\|o)$	policy π that is optimized to maximize the accumulated discounted reward	policy π that is optimized to maximize the accumulated discounted data representation

C.2 Algorithms Implementation

Table C.2 details the hyperparameters used in this work, where − indicates the parameter does not apply to the corresponding algorithm. For the actor and critic neural network structure of LSTM-TD3, the first row corresponds to the structure of the memory component, the second row corresponds to the structure of the current feature extraction, and the third row corresponds to the structure of perception integration after combining the extracted memory and the extracted current feature.

Table C.2: Hyperparameters for Algorithms

Hyperparameter	**Algorithms**					
	PPO	**TD3**	**MTD3**	**SAC**	**MSAC**	**LSTM-TD3**
discount factor: γ	0.99					
λ-return: λ	0.97	-				
clip ratio: ϵ	0.2	-				
batch size: N_{batch}	-	100				
replay buffer size: $\|D\|$	4000	10^6				
random start step: N_{start_step}	-	10000				
update after N_{update_after}	-	1000				
target NN update rate τ	-	0.005				
optimizer	Adam [146]					
actor learning rate lr_{actor}	$3 * 10^{-4}$	10^{-3}				
critic learning rate lr_{critic}	10^{-3}	10^{-3}				
actor NN structure:	[256, 256]					[128] + [128] [128] [128, 128]
critic NN structure:	[256, 256]					[128] + [128] [128] [128, 128]
actor exploration noise σ_{act}	-	0.1		-	-	0.1
target actor noise σ_{targ_act}	-	0.2	0.2	-	-	0.2
target actor noise clip boundary c_{targ_act}	-	0.5	0.5	-	-	0.5
policy update delay	-	2	2	-	-	2
entropy regulation coefficient α	-	-	-	0.2	0.2	-
history length l	-	-	-	-	-	{0, 1, 3, 5}
bootstrapping step size n	-	-	{2, 3, 4, 5}	-	{2, 3, 4, 5}	-

Appendix D

Appendix for Chapter 9

D.1 Pseudo-code For Training Preference-based Reward Function

Alg. 5 shows the pseudo-code for training the preference-based reward function.

D.2 Experiment Settings for Preference Learning

D.2.1 Trajectory Generation

A trajectory is a record of the interaction between an agent and its environment either from the start to the end, i.e., terminal, or to the maximum trajectory length. A trajectory is represented by a sequence of the interaction experiences of an agent and a sequence of images within a video recording that will be used for preference labeling.

Formally, a trajectory $l = (l_e, l_v)$ is a tuple of experience trajectory l_e and video trajectory l_v collected during the interaction of an agent with its environment. The experience trajectory $l_e = \langle (o_{t-1}, a_{t-1}, r_{t-1}, o_t, d_{t-1}) \rangle_{t=1}^{T}$ is represented by a sequence of experiences in discrete time t starting from $t = 1$ to $t = T$, while the video trajectory $l_v = \langle i_{\hat{t}} \rangle_{\hat{t}=\hat{t}_{o_0}}^{\hat{t}_{o_T}}$ is a sequence of images within a video recorded in real-world time $\hat{t}$ starting from $\hat{t}_{o_0}$ to $\hat{t}_{o_T}$, where the real-world time of an experience $(o_{t-1}, a_{t-1}, r_{t-1}, o_t, d_{t-1})$ is $\big(\hat{t}_{o_{t-1}}, \hat{t}_{a_{t-1}}, \hat{t}_{r_{t-1}}, \hat{t}_{o_t}, \hat{t}_{d_{t-1}}\big)$.

ALGORITHM 5: Training Preference-based Reward Function

Input: PL-Rew parameters θ_{pbr}, training dataset D_{pl}^{train}, test dataset D_{pl}^{test}, training epoch $N_{epoch} = 5$, training batch size $N_{batch} = 64$

Output: θ_{pbr}

Initialize early stopping patience $es_{patience} \leftarrow 0$, early stopping score $es_{score} \leftarrow -\infty$, maximum early stopping patience $max\_patience \leftarrow 2$
for $epoch\_i \leftarrow 1$ **to** N_{epoch} **do**
    `/* Training */`
    for $batch\_i \leftarrow 1$ **to** N_{batch} **do**
        calculate loss L_{train} on the batch according to Eq. 9.3
        update R^{pb} using gradient descent $\theta_{pbr} \leftarrow \theta_{pbr} - \alpha \nabla_{\theta_{pbr}} L_{train}$
    end
    `/* Validating and Early Stopping */`
    calculate loss L_{val} on test dataset
    if $epoch\_i == 1$ **then**
        $es_{score} \leftarrow -L_{val}$
    else
        if $es_{score} < -L_{val}$ **then**
            $es_{score} \leftarrow -L_{val}$
        else
            $es_{patience} \leftarrow es_{patience} + 1$
            if $es_{patience} \geq max\_patience$ **then**
                early stop training
            end
        end
    end
end

A **pre-training trajectory** is generated by a policy before an agent starts interacting with the environment during the pre-training stage. The pre-training trajectory is used to collect human preferences and initialize the reward function and avoid the agent learning on a random reward function. Once the agent starts interacting with the environment, the trajectories collected thereafter are called **online trajectories**.

D.2.2 Segment Generation

Formally, given a trajectory l, a segment $\sigma = (\sigma_e, \sigma_v)$ is a tuple of experience segment σ_e and video segment σ_v from l. Inside the experience segment $\sigma_e = \langle (o_{t+k-1}, a_{t+k-1}, r_{t+k-1}, o_{t+k}, d_{t+k-1}) \rangle_{k=1}^{K}$ are the K consecutive experiences starting from time step t to $t+K-1$, where $t \in [1, T-K+1]$ and $K \in [1, T]$. Correspondingly, assume the real-world time corresponding to the experience $(o_{t+k-1}, a_{t+k-1}, r_{t+k-1}, o_{t+k}, d_{t+k-1})$ is $\left(\hat{t}_{o_{t+k-1}}, \hat{t}_{a_{t+k-1}}, \hat{t}_{r_{t+k-1}}, \hat{t}_{o_{t+k}}, \hat{t}_{d_{t+k-1}}\right)$, then the video segment is defined as $\sigma_v = \langle i_{\hat{t}} \rangle_{\hat{t}=\hat{t}_{o_t}}^{\hat{t}_{o_{t+K}}}$, which is essentially a video clip starting from $\hat{t}_{o_t}$ to $\hat{t}_{o_{t+K}}$ in real-world time.

D.2.3 Segment Pair Sampling

A segment pair $c = (\sigma^0, \sigma^1)$ is a tuple of two segments drawn from the segment pool D_σ, which will be shown to a preference teacher to ask for a preference label. The simplest way to generate a segment pair is to randomly sample two segments and pair them up, we adopt this approach herein.

D.2.4 Preference Query Schedule

Periodically, a set of N_c segment pairs $\{c_n\}_{n=1}^{N_c}$ will be shown to a user to ask for his/her preference. We schedule N_c preference queries every F episodes of agent interaction.

D.2.5 Preference Labeling

Preference labeling is done by showing the video segments (σ_v^0, σ_v^1) of a pair of segments $(\sigma^0 = (\sigma_e^0, \sigma_v^0), \sigma^1 = (\sigma_e^1, \sigma_v^1))$ to a preference teacher and asking them to rate "left is better", "right is better", "equally engaging", and "equally un-engaging", through a web-based interface. The preference teacher's response will be interpreted into a preference label as

$$y \doteq \begin{cases} 0, & \text{if left is better,} \\ 1, & \text{if right is better,} \\ -1, & \text{if equally engaging,} \\ -1, & \text{if equally un-engaging,} \end{cases} \tag{D.1}$$

where $y = -1$ indicates two segments are equally engaging or un-engaging, which will not be used when training preference-based reward function. Once a preference data ($\sigma^0 =$

$(\sigma_e^0, \sigma_v^0), \sigma^1 = (\sigma_e^1, \sigma_v^1), y)$ is collected, it will be saved into either the training preference dataset D_{pl}^{train} with the probability $p_{pl} = 0.9$ or the test preference dataset D_{pl}^{test} with the probability $1 - p_{pl}$.

D.2.6 Web-based Interface for Human Preference Teaching

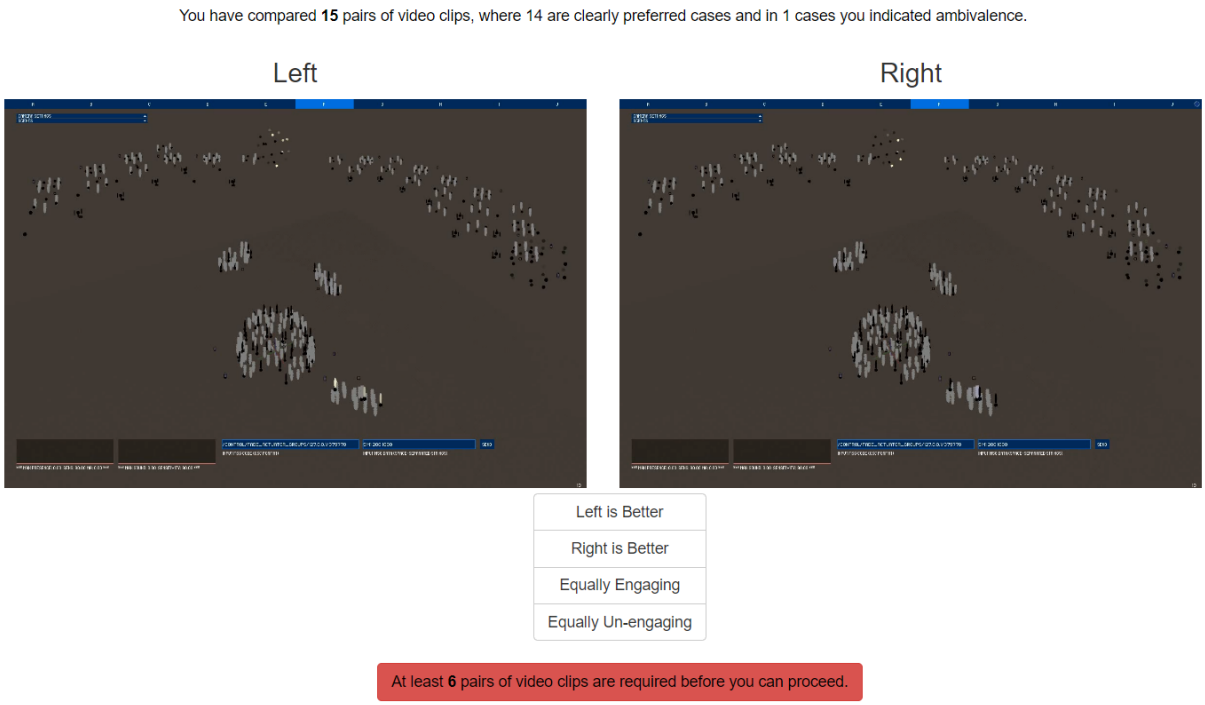


Figure D.1: Preference Query Page in Web-based Interface

We developed a web-based interface hosted on Google Cloud to collect preference labels. Fig. D.2 depicts the deployment of the web-based interface. In the local computer where the simulator is running, all data generated during the interaction between the agent and the environment is stored in a local database including both the experience and video trajectories. Every time the segments are generated, they will be uploaded to the cloud, where the video segments are stored in cloud storage and experience segments are stored in cloud database. The web-based interface will retrieve data from the cloud storage and database and save user data to the cloud database, each time a user participated in a session of the study. The users could access the interface remotely through any network accessible device.

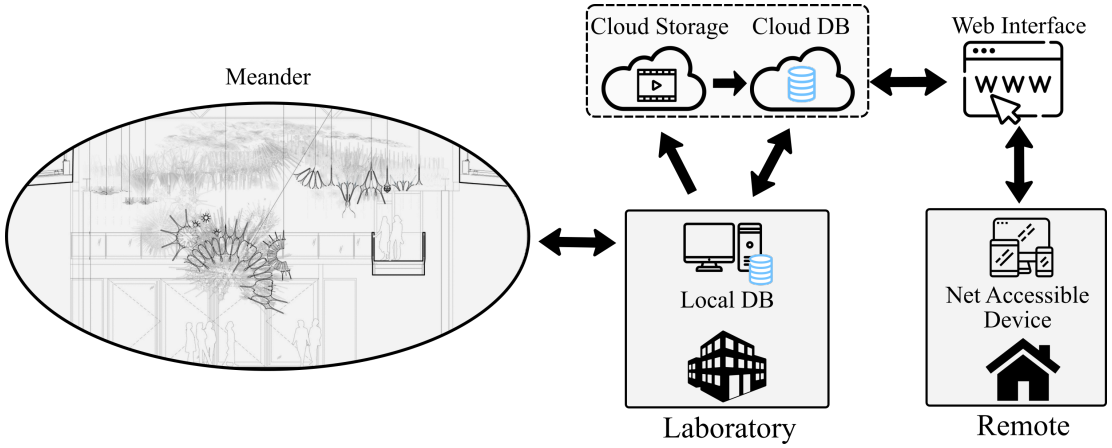


Figure D.2: Web-based Interface Deployment Diagram

D.3 Experiment Procedures for Different Conditions

To get a better understand of it and steadily advance towards generating engaging behavior from human preference, we designed various experiment conditions with gradual difficulties. This section will summarize these different experiment conditions to make it easy to differentiate the components and challenges involved in these experiment conditions, understand the experiment results, and get insights from the experiment. In summary, we will first experiment with hand-crafted reward, followed by experiment with preference-based reward that is induced from simulated preference label, constrained human preference label, unconstrained expert preference label, respectively.

D.3.1 Experiment With Hand-crafted Reward

When experiment with hand-crafted reward (HC-Rew), the reward function defined in Eq. 9.7 is used to generate reward signal. In this work, to ease the validation of different RL algorithms and the integration of the basic RL components with the preference learning related components, we unified the interaction interfaces, called *rl_agent.interact()* and *LAS-Intl-Env.step()*, between the internal environment *LAS-Intl-Env* and RL agent *rl_agent* as illustrated in Alg. 6. Specifically, *rl_agent.interact()* receives observation and reward signal from *LAS-Intl-Env* and generates action based on that, and *LAS-Intl-Env.step()* receives the action from the agent and executes it within the simulator then returns the next observation and reward signal. Alg. 6 shows the pseudo-code of the basic interaction paradigm

ALGORITHM 6: Interact with LAS-Intl-Env with Hand-crafted Reward Function

Input: LAS internal environment configuration, agent configuration, total experiment epochs $N_{rl\_episode}$, maximum episode length $max\_ep\_len = 100$

Initialize LAS-Intl-Env, Learning Agent rl_agent, total experiment steps $T = N_{rl\_episode} * max\_ep\_len$
new_obs, info ← LAS-Intl-Env.reset()
rew, done, terminal ← None, None, None
`/* Interact with environment */`
for $t \leftarrow 1$ **to** T **do**
    act ← rl_agent.interact(new_obs, rew, done, info, terminal)
    obs, rew, done, info = LAS-Intl-Env.step(act)
    terminal ← done or reach max_ep_len
    if *terminal* **then**
        `/* Save the last experience */`
        _ ← rl_agent.interact(new_obs, rew, done, info, terminal)
    end
end

in RL and will be expanded later to incorporate Preference Learning (PL). The *done* is always false in this work because there is no terminal state defined in LAS-Intl-Env, and *info* saves all necessary debugging information, e.g. when we test different reward scale, info can be used to save the original reward signal before scaling.

D.3.2 Experiment With Preference-based Reward

When experiment with preference-based reward (PB-Rew), the reward function is induced from preference labels which are either generated from simulated preference teachers or provided by real human users. Alg. 7 provides the pseudo-code of a RL agent interacting with the environment with PB-Rew, where the key differences from Alg. 6 are highlighted in light blue. Specifically, when experiment with PB-Rew, there additionally involves (1) collecting pretraining preference labels (Alg. 8/Alg. 11), (2) pretraining PB-Rew before a RL agent interacting with the environment, (3) online preference labels collection (Alg. 9/Alg. 12) and (4) online PB-Rew training which happen periodically during the interaction between RL agent and the environment.

Alg. 8 and Alg. 11 are the pseudo-codes on collecting pretraining preference labels, while Alg. 9 and Alg. 12 are the pseudo-codes on collecting online preference labels for simulated and human preference teachers, respectively. These algorithms all need Alg. 10

ALGORITHM 7: Interact with LAS-Intl-Env with Preference-based Reward Function

Input: LAS Internal environment configuration, preference collector configuration, PB-reward component configuration, learning agent configuration, PB-reward train frequency rew_train_freq

Initialize LAS-Intl-Env, RL Agent rl_agent, PB-Reward R_{pb}, Pref Collector $pref$, total interaction steps $T = N_{rl\_episode} * max\_ep\_len$
`/* Pre-train PB-Reward */`
$pref$.collect_pretraining_preferences($\cdots$) (Alg. 8/Alg. 11)
$R_{pb} \leftarrow$ train PB-Reward according to Alg. 5
update reward in environment LAS-Intl-Env.set_reward_component(R_{pb})
`/* Interact with environment */`
new_obs, info = LAS-Intl-Env.reset()
rew, done, terminal $\leftarrow$ None
for $t \leftarrow 1$ **to** T **do**
    act $\leftarrow$ rl_agent.interact(new_obs, rew, done, info, terminal)
    obs, rew, done, info $\leftarrow$ LAS-Intl-Env.step(act)
    terminal $\leftarrow$ done or reach max_ep_len
    if *terminal* **then**
        _ $\leftarrow$ rl_agent.interact(new_obs, rew, done, info, terminal)
        `/* Collect online preference labels */`
        $steps\_since\_last\_pref\_request \leftarrow pref$.collect_online_preferences($\cdots$) (Alg. 9/Alg. 12)
        `/* Train reward component */`
        if $steps\_since\_last\_rew\_train \geq rew\_train\_freq$ **then**
            $R_{pb} \leftarrow$ train PB-Reward according to Alg. 5
            update reward in environment LAS-Intl-Env.set_reward_component(R_{pb})
            $steps\_since\_last\_rew\_train \leftarrow 0$
        end
    end
    $steps\_since\_last\_pref\_request \leftarrow steps\_since\_last\_pref\_request + 1$
    $steps\_since\_last\_rew\_train \leftarrow steps\_since\_last\_rew\_train + 1$
end

to generate segments from a trajectory. Pretraining preference labels are collected before the agent interacting with its environment based on segments sampled from trajectories generated from some policies, where in this work we use a random policy to roll out trajectories for pretraining preference labels. After the PB-Rew is pre-trained, the agent starts interacting with its environment whose reward signal is given by PB-Rew. Online prefer-

ALGORITHM 8: Collect Simulated Pretraining Preference Labels

Input: Pretrain preference label number N_{pp}, pretrain agent policy π_{pp}, irrational probability p_{ip}, segment number per trajectory N_σ^l

pretrain segment number $N_{ps} \leftarrow N_{pp} \times 2$
`/* Generate pretrain segments */`
for $i \leftarrow 1$ **to** $\left\lceil \frac{N_{ps}}{N_\sigma^l} \right\rceil$ **do**
    roll out trajectory l by running π_{pp}
    $\{\sigma\} \leftarrow$ generate segments from trajectory l according to Alg. 10
    add to segment buffer $D_\sigma \leftarrow D_\sigma \cup \{\sigma\}$
end
`/* Generate pretrain preference labels */`
for $i \leftarrow 1$ **to** N_{pp} **do**
    sample a segment pair (σ^0, σ^1)
    generate preference label y' according to Eq. 9.10 with irrational probability p_{ip}
    `/* Add pretrain preference labels to traning and test dataset */`
    if $p \sim U_{[0,1]} <= p_{pl}$ **then**
        $D_{pl}^{train} \leftarrow D_{pl}^{train} \cup (\sigma^0, \sigma^1, y')$
    else
        $D_{pl}^{test} \leftarrow D_{pl}^{test} \cup (\sigma^0, \sigma^1, y')$
    end
end

ence labels are collected periodically during this interaction based on segments generated both in pretraining phase and online interaction phase.

Except the difference between pretraining and online preference collection, the procedures for collecting preference labels from simulated and human preference teachers also have slight difference which is mainly introduced by the web-based interface usage. The difference is highlighted in light blue in Alg. 11 and Alg. 12. As will be detailed in section D.4.4, the web-based interface is hosted on a cloud platform, so the data collected on local computer and cloud needs to be synchronized. In addition, different from simulated preference teacher who is always available, human preference teachers will only provide their preferences when they are available and this cannot be predicted. Therefore, when a preference request is scheduled, the whole learning system needs to wait until the requested preference labels are provided by human preference teachers.

Table D.1 summaries all hyper-parameters related to preference learning that are mentioned in Alg. 7 - 12. For values in {}, we will test all of them in this work.

ALGORITHM 9: Collect Simulated Online Preference Labels

Input: trajectory l, segment pool D_σ, *steps_since_last_pref_request*, *online_pref_request_freq*, preference number per request N_c

Output: *steps_since_last_pref_request*

`/* Generate online segments */`

$\{\sigma\} \leftarrow$ generate segments from trajectory l according to Alg. 10

add to segment buffer $D_\sigma \leftarrow D_\sigma \cup \{\sigma\}$

`/* Generate online preference labels */`

if *steps_since_last_pref_request* $\geq$ *online_pref_request_freq* **then**

  for $i \leftarrow 1$ **to** N_c **do**

    randomly sample a segment pair (σ^0, σ^1) from segment buffer D_σ

    generate preference label y' according to Eq. 9.10 with irrational probability p_{ip}

    `/* Add pretrain preference labels to traning and test dataset */`

    if $p \sim U_{[0,1]} <= p_{pl}$ **then**

      $D_{pl}^{train} \leftarrow D_{pl}^{train} \cup (\sigma^0, \sigma^1, y')$

    else

      $D_{pl}^{test} \leftarrow D_{pl}^{test} \cup (\sigma^0, \sigma^1, y')$

    end

  end

  steps_since_last_pref_request $\leftarrow 0$ `// Reset step count`

end

ALGORITHM 10: Segment Generation

Input: Trajectory $l = (l_e, l_v)$, segment length K, segment number N_σ^l for the current trajectory

Output: Segment set *seg_set*

Initialize segment set *seg_set* $\leftarrow \emptyset$ of the given trajectory

for $j \leftarrow 1$ **to** N_σ^l **do**

  randomly select a start time step $t \in [1, T-K+1]$

  extract experience segment $\sigma_e = \langle (o_{t+k-1}, a_{t+k-1}, r_{t+k-1}, o_{t+k}, d_{t+k-1}) \rangle_{k=1}^{K}$

  extract video segment $\sigma_v = \langle i_{\hat{t}} \rangle_{\hat{t}=\hat{t}_{o_t}}^{\hat{t}_{o_{t+K}}}$

  add segment to segment set *seg_set* $\leftarrow$ *seg_set* $\cup \{\sigma = (\sigma_e, \sigma_v)\}$

end

ALGORITHM 11: Collect Human Pretraining Preference Labels

Input: Pretrain preference label number N_{pp}, pretrain agent policy π_{pp}, irrational probability p_{ip}, segment number per trajectory N_σ^l

pretrain segment number $N_{ps} \leftarrow N_{pp} \times 2$
`/* Generate pretrain segments */`
for $i \leftarrow 1$ **to** $\left\lceil \frac{N_{ps}}{N_\sigma^l} \right\rceil$ **do**
  roll out trajectory l by running π_{pp}
  $\{\sigma\} \leftarrow$ generate segments from trajectory l according to Alg. 10
  add to segment buffer $D_\sigma \leftarrow D_\sigma \cup \{\sigma\}$
  one_way_sync_segment_table_local2cloud()
end
`/* Collect pretrain preference labels */`
while *not collected_pretraining_preferences* **do**
  one_way_sync_preference_table_cloud2local()
  if $N_{pp} \leq \left|\{(\sigma^0, \sigma^1, y) | (\sigma^0, \sigma^1, y) \in D_{pl} \text{ and } y \neq -1\}\right|$ **then**
    collected_pretraining_preferences $\leftarrow True$
  else
    sleep(5 minutes)
  end
end

D.4 Experiment Implementation

In this section, we will introduce some implementation details which are not mentioned in the proposed approach.

D.4.1 Simulation of LAS with *las_sim_tkt*

The simulation toolkit *LAS_Sim_Tkt* introduced in Section 4.2 is used in the simulation experiments conducted in this chapter. However, in Section 4.2 only the high-level concepts of *LAS_Sim_Tkt* is introduced. For the real experiments where many hyper-parameters are investigated, it is necessary to run multiple simulation experiments simultaneously. To achieve that and promote the reproducibility of the results reported in this chapter, we implemented the concept of *LAS_Sim_Tkt* to support running on High Performance Computing (HPC), which is available in https://github.com/LinghengMeng/las_sim_tkt.

ALGORITHM 12: Collect Human Online Preference Labels

Input: trajectory l, segment pool D_σ, $steps\_since\_last\_pref\_request$, $online\_pref\_request\_freq$, preference number per request N_c

```
/* Generate online segments */
```

$\{\sigma\} \leftarrow$ generate segments from trajectory l according to Alg. 10

add to segment buffer $D_\sigma \leftarrow D_\sigma \cup \{\sigma\}$

$one\_way\_sync\_segment\_table\_local2cloud()$

```
/* Collect online preference labels */
```

if $steps\_since\_last\_pref\_request \geq online\_pref\_request\_freq$ **then**

  $valid\_pref\_num\_before \leftarrow \left|\left\{(\sigma^0, \sigma^1, y) | (\sigma^0, \sigma^1, y) \in D_{pl} \text{ and } y \neq -1\right\}\right|$

  while *True* **do**

    sleep(5 minutes)

    one_way_sync_preference_table_cloud2local()

    $valid\_pref\_num\_after \leftarrow \left|\left\{(\sigma^0, \sigma^1, y) | (\sigma^0, \sigma^1, y) \in D_{pl} \text{ and } y \neq -1\right\}\right|$

    if $valid\_pref\_num\_after - valid\_pref\_num\_before \geq N_c$ **then**

      $steps\_since\_last\_pref\_request \leftarrow 0$ `// Reset step count`

      break

    end

  end

end

D.4.2 Reward Function Choosing

In this thesis, the Preference-based Reward (PB-Rew) function R^{pb} is approximated by a fully connected neural network with parameters θ_{pbr} and the hyper-parameters listed in Table D.2.

D.4.3 RL Algorithms Implementation

The implementation of RL algorithms in this chapter are very similar to those investigated in the previous chapters. Specifically, the neural network structures and hyper-parameters of PPO, TD3, SAC are the same as the implementation in OpenAI Spinning Up (https://spinningup.openai.com). The LSTM-TD3(5) follows the implementation as that in Chapter 7, where n in the bracket indicates the step size in multi-step bootstrapping.

Table D.1: Preference-Learning Related Hyper-parameters

Hyper-parameters	**Description**	**Value**
max_ep_len	Maximum episode length	100
N_{pp}	Pretrain preference label number	20
N_{σ}^{l}	Segment number per trajectory	2
rew_train_freq	Reward train frequency	1000
p_{ip}	Irrational probability of simulated teacher	$\{0, 0.1, 0.3\}$
p_{pl}	Probability a preference data is added to tranining dataset	0.9
$online\_pref\_request\_freq$	Online preference request frequency	1000
N_c	Preference number per request	20
K	Segment length	$\{1.5s, 3s, 4.5s\}$

PB-Reward related hyper-parameters can be found in Table D.2.
p_{ip} is only for simulated preference teacher.

Table D.2: Hyper-parameters of Preference-based Reward Function

Hyper-parameter	Value
Input	(o_t, a_t, o_{t+1})
Hidden Layers	$[64, 64]$
Hidden Layer Activation	ReLu
Output Layer Activation	Tanh
Dropout Rate	0.5

D.4.4 Data Management

For this experiment, we need to save experience data, video data, and user data, which makes simple text log file prohibitive. To enable data management, we employed relational database to save experience and user data. In addition, we kept video segments separately in Google Bucket Storage and only stored the link to the video segments in the database tables. Another practical thinking is there is no need to store experience in cloud database, since the cloud database is just for collecting preference based on video segment and all learning related works are happened on local computer. Therefor, we maintained two

databases where one on local computer and another one on the cloud. The local and cloud database are implemented by SQLite https://www.sqlite.org and PostgreSQL https://www.postgresql.org, respectively. The tables in each database are shown in Table D.3. Specifically, the *experience_table* saves all experience data generated during the interaction between an agent and its environment. *segment_table* saves video segment data where the experience segment is represented by the start and end experience id and the video segment is represented by a URL link to the video segment stored on cloud bucket storage. *exp_and_seg_match_table matches* a video segment and the experiences within the start and end experience id corresponding to the video segment to ease data retrieving. *pref_user_table* saves the user, i.e., the preference teacher, data. *pref_label_table* saves all preference labels provided by the users. *pref_survey_table* saves all questionnaire related data. Fig. D.3 depicts the relationship between these tables and the synchronization of the tables between the local and cloud database. In particularly, *segment_table* is synchronized from local to cloud database, while *pref_user_table*, *pref_label_table*, and *pref_survey_table* are synchronized from cloud to local database.

Table D.3: Tables in Local and Cloud Database

DB Tables	**Description**	**Local DB**	**Cloud DB**
experience_table	all experiences	✓	✗
segment_table	all generated segments	✓	✓
exp_and_seg_match_table	the correspondence between segments and experiences	✓	✗
pref_user_table	preference user data	✓	✓
pref_label_table	preference data	✓	✓
pref_survey_table	questionnaire data of preference user	✓	✓

Note: ✓ or ✗ indicates a table is included or not included in a database.

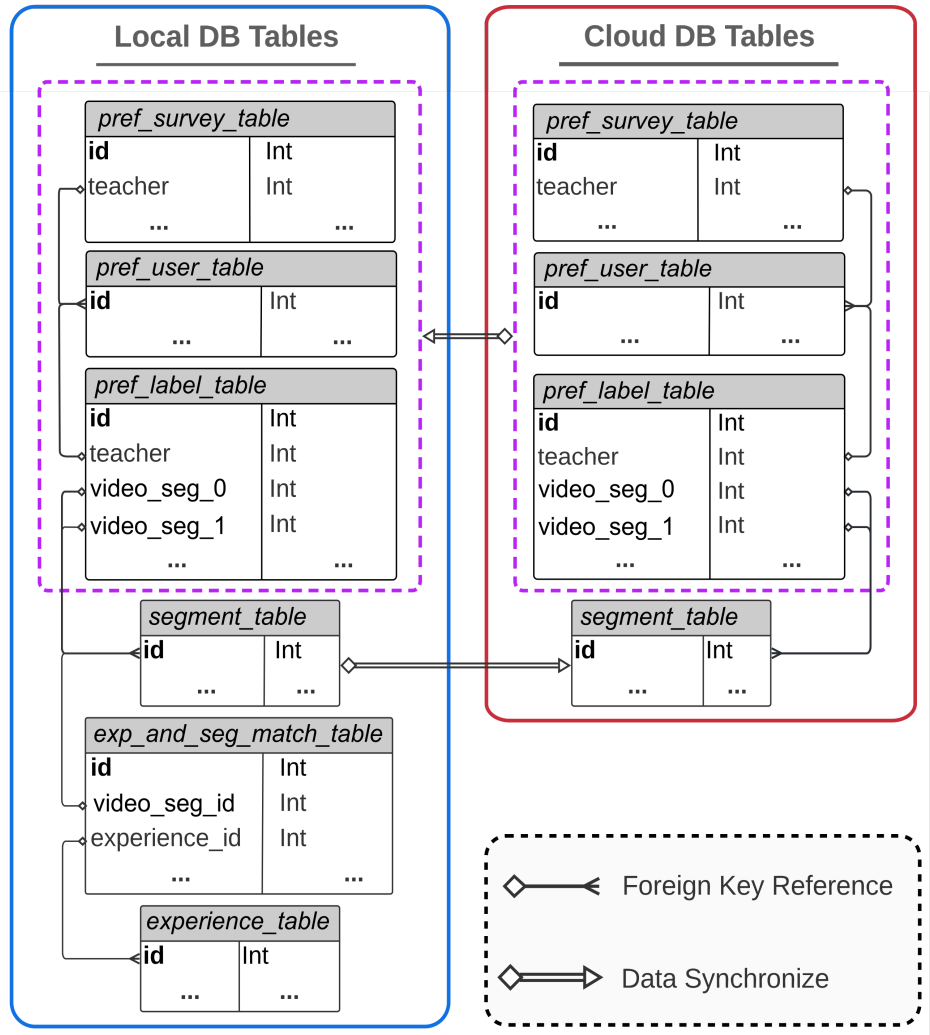


Figure D.3: Database Schema and Synchronization Between Local and Cloud Database

D.5 Preliminary Experiment Results

The learning curves reported in this chapter are averaged over three random seeds and smoothed by 1-D Gaussian filter (Gaussian kernel standard deviation $\sigma = 5$) for better visual effect. The shaded area corresponds to the standard deviation over the three random seeds.

D.5.1 Results on Hand-crafted Reward

The experiment with hand-crafted reward is to validate different design choices, such as observation window size T_{ow} and different DRL algorithms.

Effect of Observation Window Size

Fig. D.4 shows the learning curves on hand-crafted reward defined in Eq. 9.7 over 3 random seeds (curves on hand-crafted reward with different random seeds are shown in Fig. D.5), Table D.5 summaries the best performance on hand-crafted reward for each algorithm within 20000 steps, and Table D.4 shows the observation space size corresponding to different observation window sizes, where T_{ow} indicates the size of observation window used to construct observation as introduced in section 9.3.1. We tested different observation window sizes to confirm there is no sever partial observability problem based on two intuitions that: (1) if there is sever partial observability problem in $T_{ow} = 1$, then increasing the observation window size will get better results because larger observation window size is able to incorporate temporal information within the observation space; and (2) if there is sever partial observability problem, we would be able to observe significant performance difference between TD3 and LSTM-TD3, as LSTM-TD3 is proposed to deal with POMDP. Comparing the three panels in Fig. D.4, there is not much performance difference among different observation window sizes, neither significant difference between TD3 and LSTM-TD3 for $T_{ow} = 1$s and $T_{ow} = 2$s. Even though LSTM-TD3 experiences performance decrease, this is more likely related to the large observation space size shown in Table D.4. Therefore, we empirically conclude that for $T_{ow} = 1$ there is no sever partial observability problem.

Table D.4: Observation Space Size for Different T_{ow}

Observation Window Size	$T_{ow} = 1$s	$T_{ow} = 2$s	$T_{ow} = 5$s
Observation Space Size	724	1448	3620

Observation reading frequency $f_o = 1$.

By focusing on the first panel in Fig. D.4, we can see that TD3 and LSTM-TD3 learn faster than PPO at the beginning, but achieve worse final performance. The first part of this observation is not surprising, but the second part is very uncommon compared to the results reported on MuJoCo tasks [89] where TD3 outperforms PPO significantly

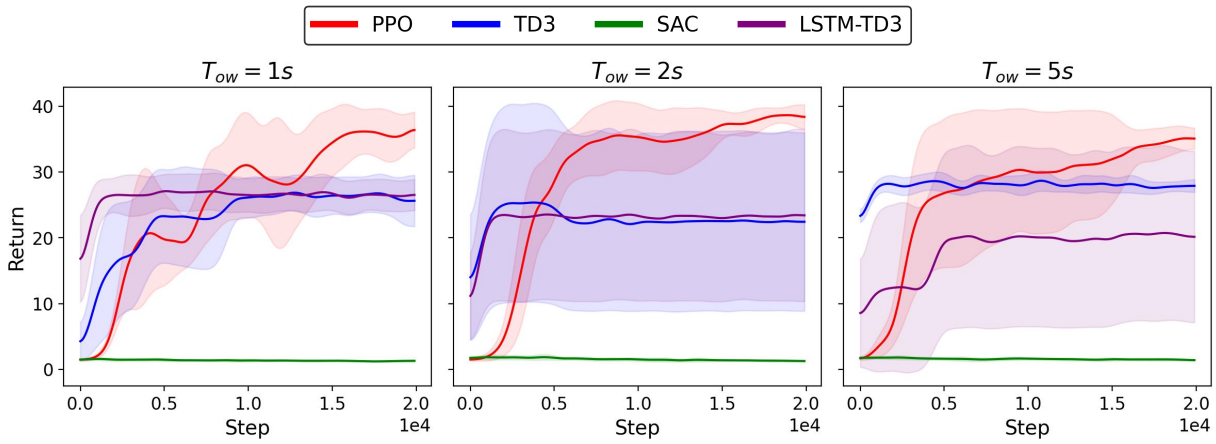


Figure D.4: Learning Curves on Hand-crafted Reward

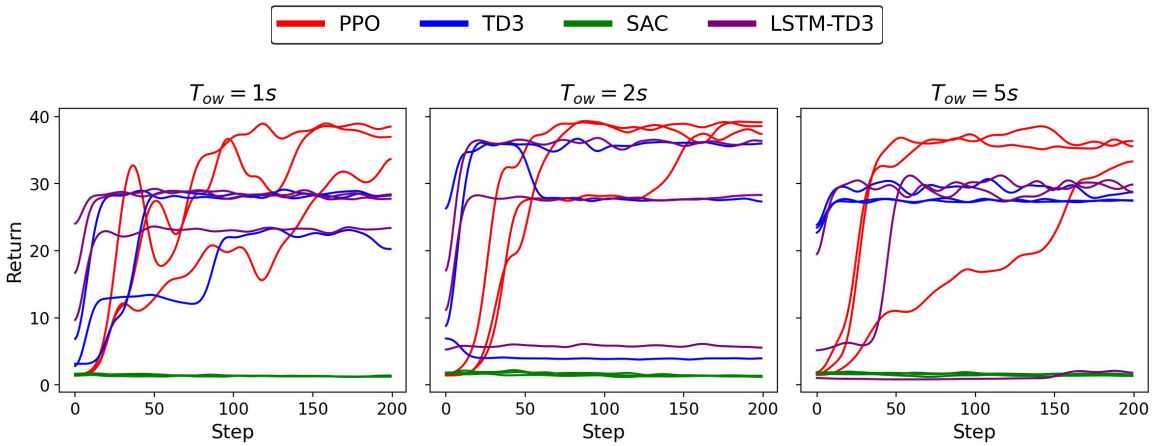


Figure D.5: Learning Curves on Hand-crafted Reward with Different Random Seeds

Table D.5: Best Learning Performance on Hand-crafted Reward

Obs Window Size	PPO	TD3	LSTM-TD3	SAC
$T_{ow} = 1$s	**44.77**	34.07	34.83	1.54
$T_{ow} = 2$s	**43.44**	43.43	32.70	3.22
$T_{ow} = 5$s	**40.99**	32.33	32.33	8.40

in terms of final performance. Counter-intuitively, SAC performs worst among the four

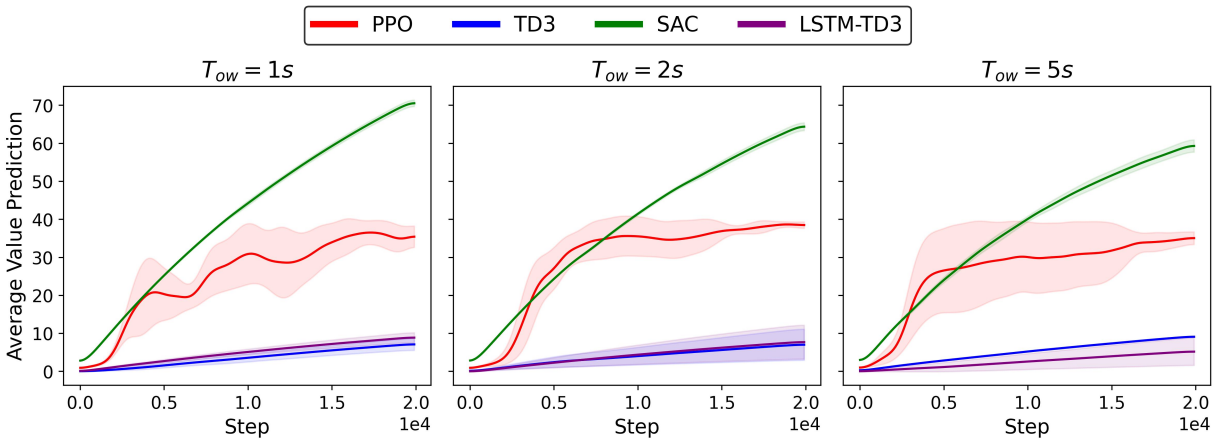


Figure D.6: Average Value Prediction

algorithms. This is very unexpected because on MuJoCo takes SAC is always the best or comparable to TD3 as reported in [106]. To exclude the implementation bugs, we tested our implementation of the algorithms on MuJoCo tasks, and the results are similar to the results reported in [89, 106]. To provide more insights into this, Fig. D.6 shows the average value predictions correspond to the learning curves in Fig. D.4, where the PPO's the state-value and others' Q-value functions are plotted. It can be seen from Fig. D.4 and Fig. D.6 that SAC experienced sever overestimation problem which is indicated by the huge difference between the learning performance and Q-value prediction of SAC. Even though SAC employs the same method (Eq. 3.18) with TD3 (Eq. 3.15) to overcome overestimation by taking the minimum of two Q-value estimators, SAC still suffers from the problem while TD3 does not. Therefore, we suspect these counter-intuitive observations are task specific.

Effect of Reward Scale

Based on the counter-intuitive observation that TD3 and SAC are worse than PPO, we investigated if reward signal scale is the reason of that. Particularly, in the standard MuJoCo benchmark, most tasks have maximum immediate reward that is greater than 1, while in the LAS the hand-crafted reward value range is in [0,1]. Formally, assume the original reward value range is $r \in [min, max]$ and we want to rescale the reward to range $r' \in [min', max']$, then the new reward r' can be rescaled by

$$r'_{[min',max']} = \frac{r_{[min,max]} - min}{max - min}(max' - min') + min' \tag{D.2}$$

where in this work $r \in [0, 1]$ which is rescaled to $r'_{[0,2]}$, $r'_{[-1,1]}$, $r'_{[-2,2]}$, $r'_{[0,10]}$, $r'_{[0,100]}$.

Fig. D.7 shows the learning curves averaged over 3 random seeds[1] on hand-crafted reward with different reward scales and fixed observation window size $T_{ow} = 1$, where $r'_{[0,1]}$ corresponds to the original reward, and the agent is trained with the rescaled reward but measured in the original reward scale. Table D.6 summaries the best performance within 20000 steps among different random seeds for different reward scales and observation window sizes, where the best among each row is highlight in bold. From the figure and the table, the performance of PPO on different reward scales is very similar and consistent. Even though TD3 and LSTM-TD3 have slight changes on most reward scales, the changes are not significant. The most interesting observation is on SAC which completely fails the task with $r'_{[0,1]}$, $r'_{[0,2]}$, $r'_{[-1,1]}$, $r'_{[-2,2]}$ but has significant improvement with $r'_{[0,10]}$ and $r'_{[0,100]}$, even though the learning is unstable.

Table D.6: Best Learning Performance on Hand-crafted Reward with Different Reward Scale and Observation Window Sizes

Alg.	Obs Window	$r'_{[0,1]}$	$r'_{[0,2]}$	$r'_{[-1,1]}$	$r'_{[-2,2]}$	$r'_{[0,10]}$	$r'_{[0,100]}$
PPO	$T_{ow} = 1$s	44.77	42.59	43.00	42.29	**45.15**	43.70
	$T_{ow} = 2$s	43.44	41.57	44.07	43.92	**44.69**	41.79
	$T_{ow} = 5$s	**40.99**	40.30	40.71	40.63	39.91	40.62
TD3	$T_{ow} = 1$s	31.62	31.50	42.58	34.71	40.82	**43.32**
	$T_{ow} = 2$s	**40.41**	30.20	39.76	39.05	37.65	38.85
	$T_{ow} = 5$s	33.88	36.19	35.00	34.70	**37.12**	14.65
LSTM-TD3	$T_{ow} = 1$s	31.50	38.79	**43.35**	41.75	40.05	41.44
	$T_{ow} = 2$s	40.80	39.45	38.56	38.76	**41.03**	30.95
	$T_{ow} = 5$s	35.21	22.56	4.53	17.53	**37.65**	35.95
SAC	$T_{ow} = 1$s	2.45	2.35	2.33	2.88	32.35	**42.93**
	$T_{ow} = 2$s	3.97	3.39	2.62	2.91	34.32	**39.72**
	$T_{ow} = 5$s	2.94	6.49	2.97	5.19	33.50	**41.64**

D.5.2 Results on Simulated Preference

The simulated human preference is generated according to Eq. 9.8 - 9.10 with different segment lengths and irrational probabilities. Specifically, we investigated irrational probability $p_{ip} = 0$,$p_{ip} = 0.1$,$p_{ip} = 0.3$, and segment length $l = 1.5$s, $l = 3$s, and $l = 4.5$s.

[1] the learning curves on hand-crafted reward with different reward scales and random seeds are shown in Fig. D.8.

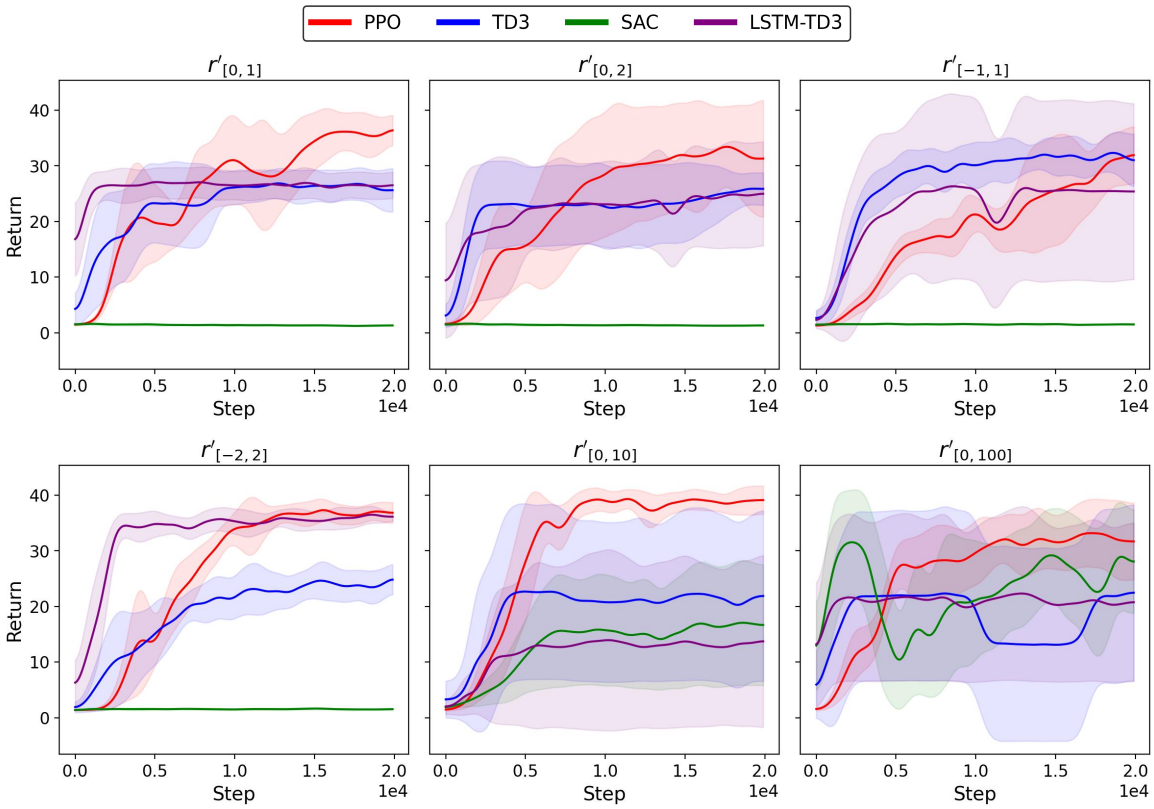


Figure D.7: Learning Curves on Hand-crafted Reward with Different Reward Scales

Fig. D.9 shows the learning curves of various RL algorithms on preference-based reward (PB-Rew) with different irrational probabilities and segment lengths, where the performance is measured in hand-crafted reward (HC-Rew) and the fist column is the baseline that RL algorithms both learn and measured in hand-crafted reward. Because the preference-based reward signal that is used by RL agent is induced from simulated preference and is different among different runs, it is more fair to compare the performance measured in hand-crafted reward which is fixed and direct measurement of how good the learned PB-Rew is in terms of maximizing the HC-Rew. From Fig. D.9, we can see that overall the performance of each algorithm trained on PB-Rew with different irrational probabilities and segment lengths is comparative to their performance on HC-Rew, and for some cases the performance of PB-Rew is even better than that of HC-Rew, e.g., TD3 on $p_{ip} = 0.1$ and segment length 1.5s gets better performance than it on HC-Rew. These observations indicate a descent PB-Rew is derived from the simulated preference labels. It

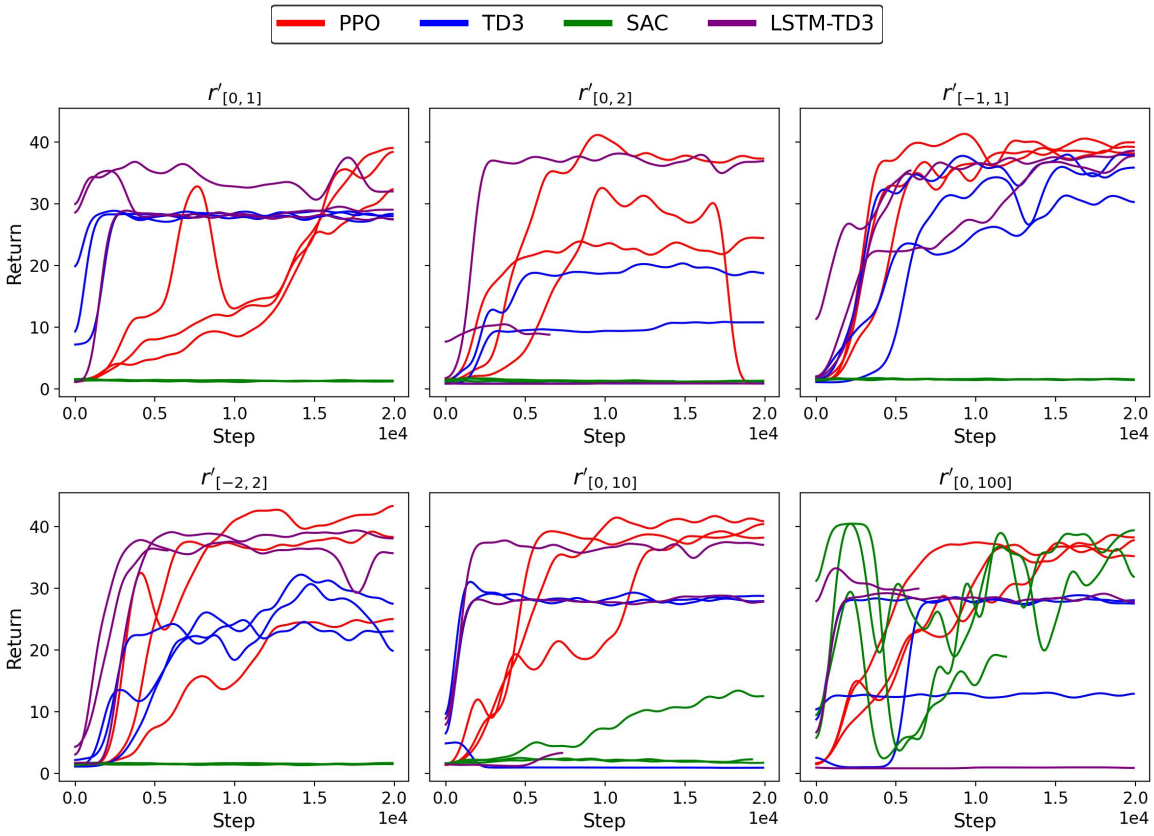


Figure D.8: Learning Curves on Hand-crafted Reward with Different Reward Scales and Random Seeds

is particularly interesting that SAC on PB-Rew gets better performance on all cases that it on HC-Rew, even though SAC is still worse than other algorithms. Note that the irrational probability indicates the probability that a simulated preference teacher will make irrational, i.e., random, choice rather than the indicating the error rate of preference labeling. Fig. D.10 illustrates the true error rate of the collected preference labels of different irrational probabilities, where the box extends from the lower to upper quartile values of the data, with a line at the median, and the whiskers extend from the box to show the range of the data. From Fig. D.10, we can see that the true error rate of preference label is lower than the corresponding irrational probability.

Table D.7 summaries best performance of different algorithms with different irrational probabilities and segment lengths measured in HC-Rew among different random seeds. In

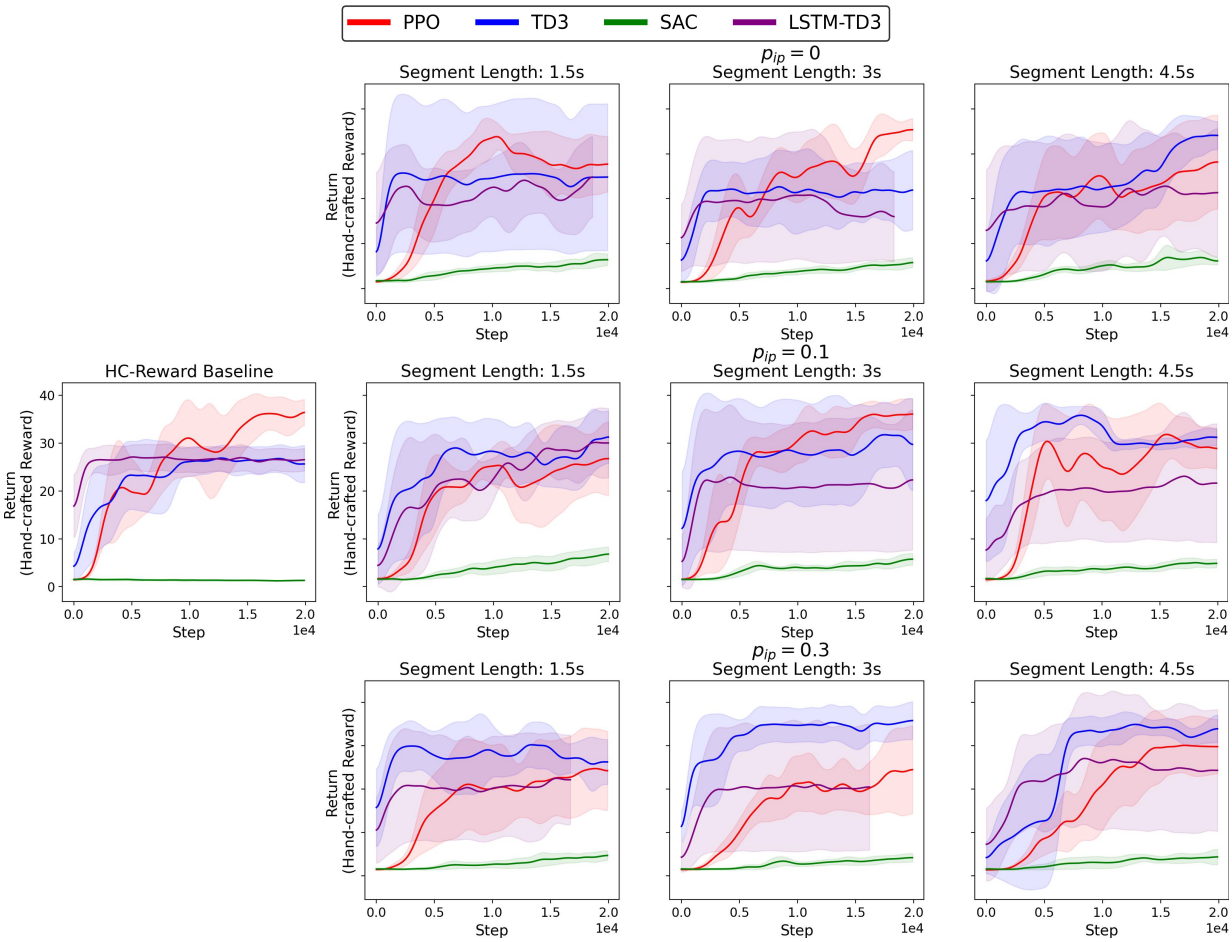


Figure D.9: Learning Curves Measured in HC-Rew on Preference-based Reward

Table D.7, if the best performance of a RL agent on PB-Rew is worse than the its baseline, i.e., its performance on HC-Rew, it will be in gray. The bold corresponds to the best performance of a RL agent among different segment lengths for each irrational probability. From the table, we can see that except PPO, other algorithms all get better best performance on PB-Rew than their baseline on HC-Rew, which indicates the effectiveness of PB-Rew induced from preference labels. The results in Table D.7 and Fig. D.9 also show the performance is not sensitive to irrational probability, and it seems that for higher irrational probability longer segment length is better than shorter segment length.

Fig. D.11 shows the learning curves of RL agents trained on PB-Rew that are also measured in PB-Rew. From the perspective of PB-Rew, SAC seems making pretty good

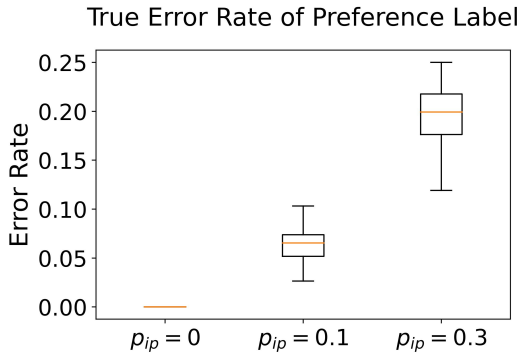


Figure D.10: True Error Rate of Preference Label

Table D.7: Best Performance Measured in Hand-crafted Reward for Simulated Preference Teacher with Different Irrationality and Segment Lengths

Alg.	BL	$p_{ip} = 0$			$p_{ip} = 0.1$			$p_{ip} = 0.3$		
		1.5s	3s	4.5s	1.5s	3s	4.5s	1.5s	3s	4.5s
PPO	44.77	**45.13**	43.33	41.42	40.72	43.03	**45.79**	41.82	38.73	**42.54**
TD3	31.62	**42.16**	36.24	41.37	40.94	**43.44**	41.52	38.76	**42.31**	41.85
LSTM-TD3	31.50	38.58	42.21	**43.50**	**42.65**	38.83	40.74	40.24	41.95	**42.23**
SAC	2.45	10.24	11.83	**15.00**	11.16	10.96	**11.37**	8.41	7.45	**9.26**

BL: baseline with hand-crafted reward. For all results, $T_{ow} = 1$s.

and steady progress and having less performance gap with other algorithms than that from the perspective of HC-Rew. This contrary observation on the performance of SAC measured in PB-Rew and HC-Rew seems to indicate the learned PB-Rew is not an exact match of HC-Rew, even though they may share some in common. Another interesting observation in Fig. D.11 is that for many cases TD3 and LSTM-TD3 experience sever performance decrease when measured in PB-Rew, which is not observed in Fig. D.9 when measured in HC-Rew. We suspect the performance decrease measured in PB-Rew observed in TD3 might be caused by the reward function shift.

To investigate the aforementioned collapse in TD3 and validate our suspicion, Fig. D.12 shows the return, reward and action collected during the interaction of an agent with its environment, where the irrational probability p_{ip} is 0.1, segment length is 1.5s, and the

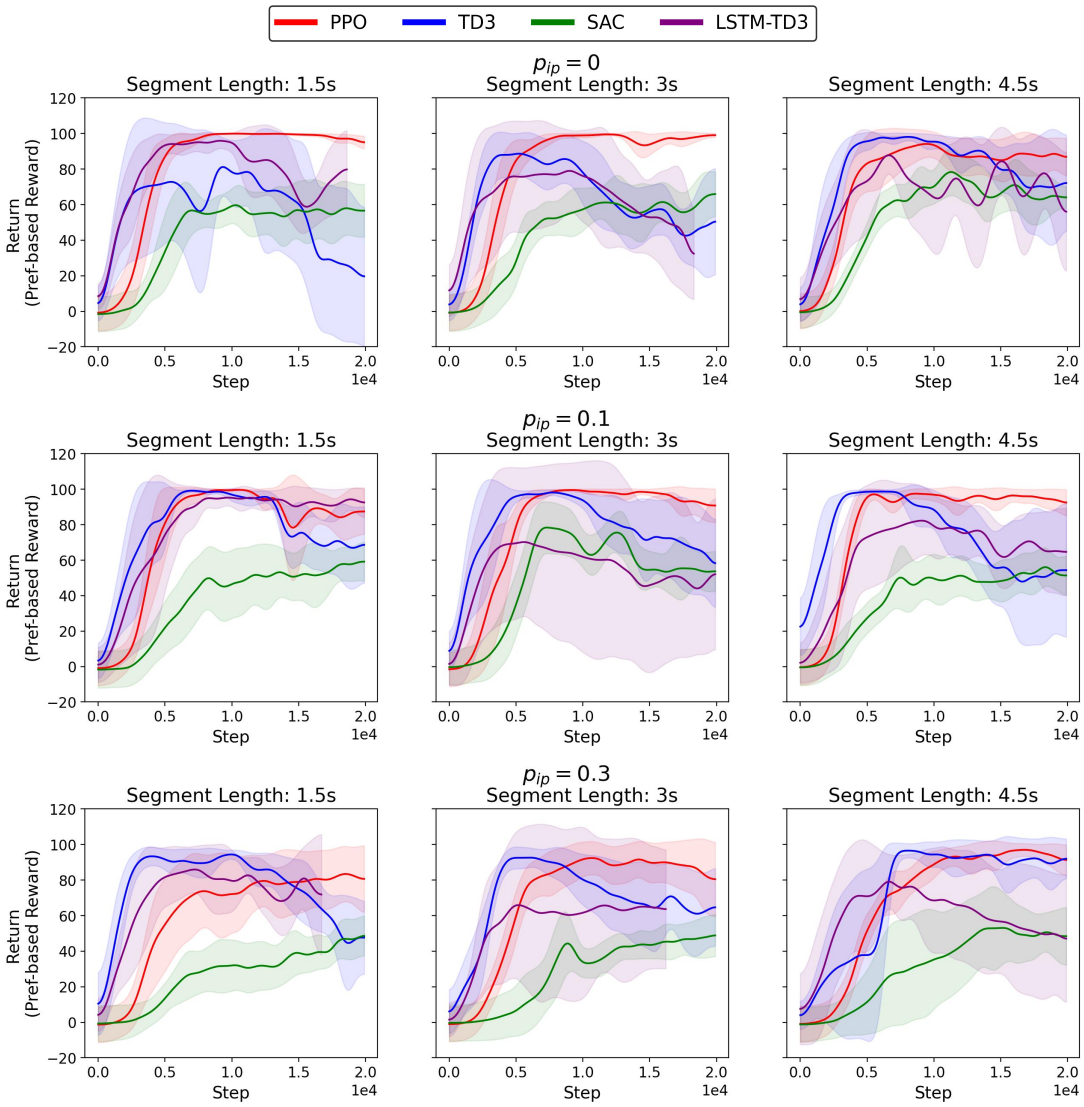


Figure D.11: Learning Curves Measured in PB-Rew on Preference-based Reward

rewards and actions before step 0 are generated by a random policy during collecting pre-training preference label (more results can be found in Appendix D). The data shown in Fig. D.12 is collected during a RL agent is interacting with its environment where only the PB-Rew is accessible to the agent and HC-Rew is only saved for investigation and not accessible to the agent. Each row of Fig. D.12 corresponds to the results from a specific RL algorithm. In each row, the first column shows the return measured in PB-Rew and

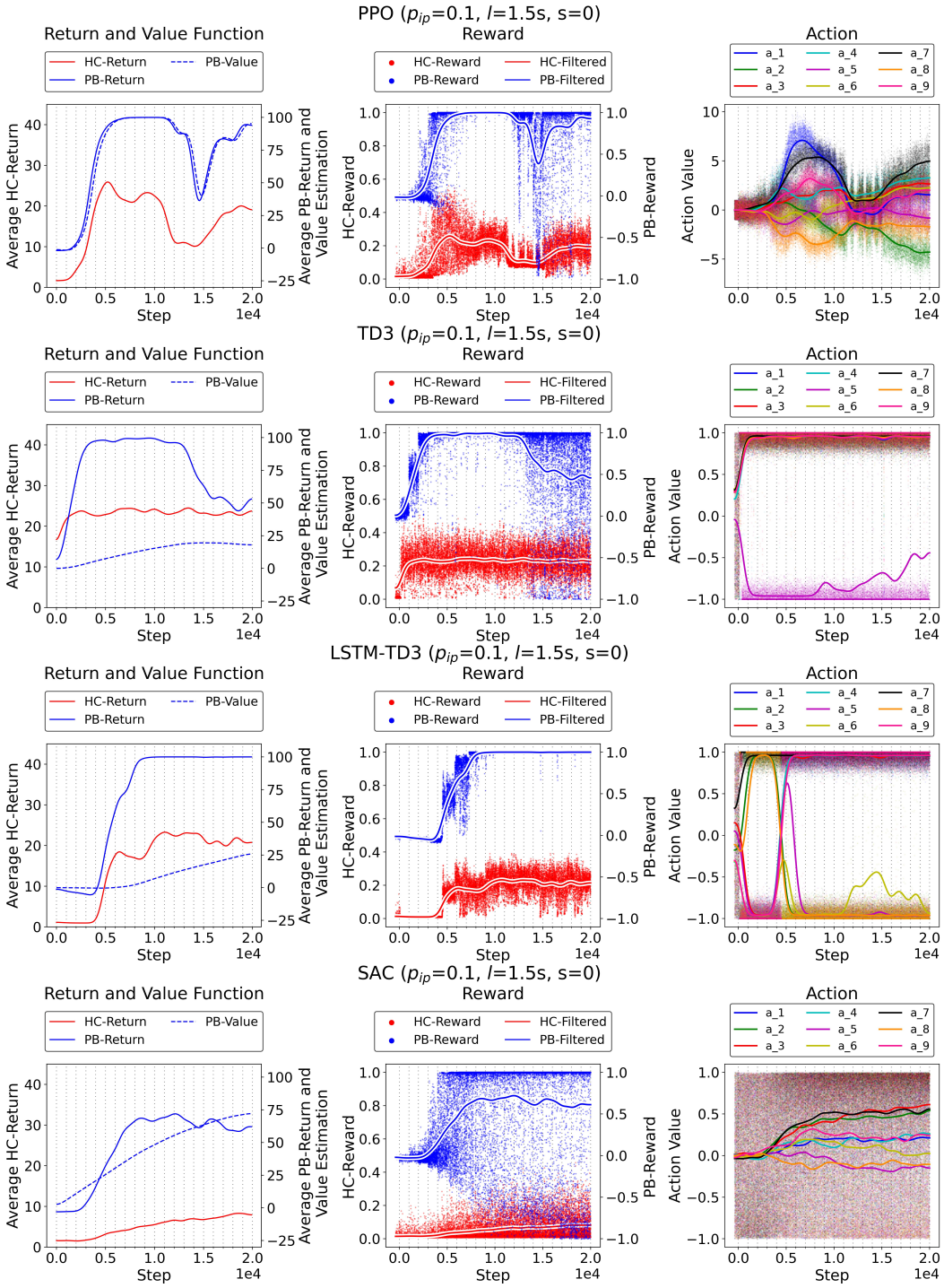


Figure D.12: Investigate Into PB-Rew Based Return Collapse

HC-Rew, indicated as PB-Return and HC-Return respectively, and the estimated value function indicated as PB-Value. The second column shows the PB-Rew and HC-Rew for each experiences collected over the interaction, and for better visualization the corresponding filtered lines are plotted as well. The third column shows the actions generated by the agent over the interaction, and similarly for better visualization the corresponding filtered lines are plotted. From the 1st column of the 2nd row in the figure, we can see that there is a collapse in the PB-Return of TD3, but the HC-Return is not affected. This indicates even though the PB-Rew is changed, the policy derived from PB-Rew does not changed much. This can be supported by the collected rewards in the 2nd column of the 2nd row of the figure, where start from around 14000 step there are more experiences have PB-Rew value covering whole value range from -1 to 1, while it seems there is no obvious change in HC-Rew. In addition, it can also be supported by the generated action where in the 3rd column of the 2nd row only one action dimension has obvious change while others do not. Similar observations can be found for LSTM-TD3 and SAC as well. It is worth to mentioned that the PB-Rew plotted in Fig. D.12 is calculated for each step, while during the policy training of TD3 the PB-Rew is recalculated for the experiences sampled from the experience replay buffer based on the updated PB-Rew function. This further emphasizes the importance of recalculating PB-Rew during policy training for off-policy RL algorithms that relying on experience replay. Different from other three off-policy algorithms, PPO is more sensitive to the PB-Rew change. As shown in the the 1st row of Fig. D.12, when the PB-Rew changed around 15000 step (blue dots in the 2nd column), the performance of the policy on HC-Rew is also affected (red dots in the 2nd column), because of the change of policy (3rd column). At the first glance, PPO is an on-policy algorithm which means it always has access to the latest PB-Rew, and the other three also have access to the latest PB-Rew during training by recalculating PB-Rew. However, PPO only uses the experiences from the last episode to train its policy, while the other three sample a mini-batch of experiences from the experience buffer with much more experiences from many episodes. Because the PPO updates policy only on a small set of experiences, it may not generalize well and cause performance collapse.

Another interesting observation in Fig. D.12 is the sudden PB-Rew change always happens after a long time of reward plateau, e.g. the 1st and 2nd row. An explanation to this is when the reward gets saturated around the maximum it is impossible to accommodate more preference labels because the reward predictions are so close to each other. Therefore, the neural network output needs to be pushed to cover the whole output range to accurately predict preference.

D.5.3 Results on Constrained Human Preference

The whole experiment with unconstrained expert preference follows the same procedure for human preference teacher as demonstrated in Alg. 7 in Appendix D.3.1. The preference learning related hyper-parameters in the experiment with unconstrained experts is the same as that shown in Table D.1, where in this experiment only segment length $K = 4.5$s is tested. The RL agent used is TD3, and the preference-based reward estimator uses the hyper-parameters shown in Table D.2.

Fig. D.13 shows the error rate of constrained human preference corresponding to different video segment length, where the error rate is the ratio of preference queries that are wrongly labels and the true label is calculated based on the handcrafted reward as defined in Eq. 9.8 adn 9.9. For TD3 and LSTM-TD3, the shorter the segment length is, the higher the error rate is. PPO is less sensitive to segment length, compared to other three algorithms. SAC has very high error rate no matter what the segment length is.

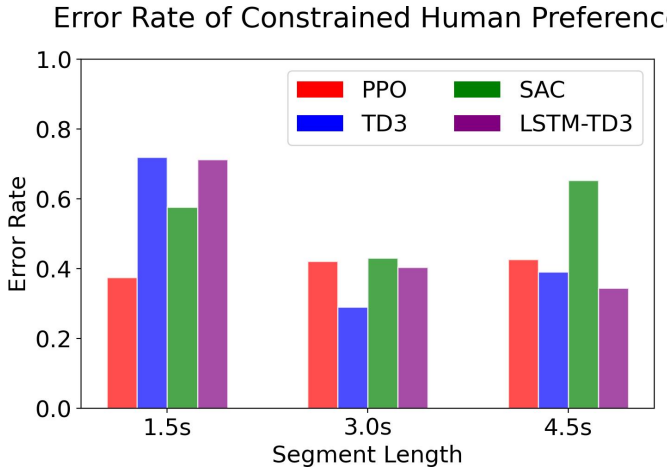
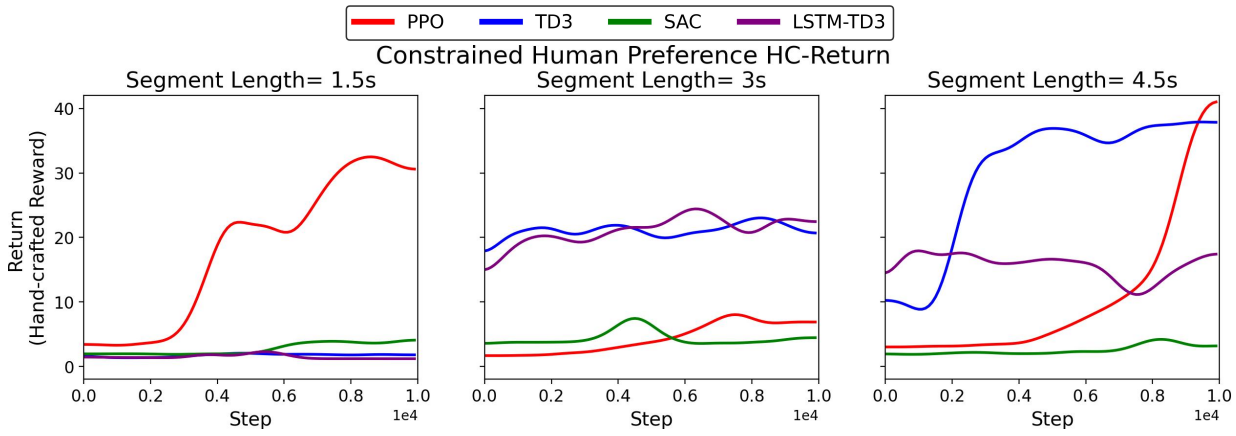


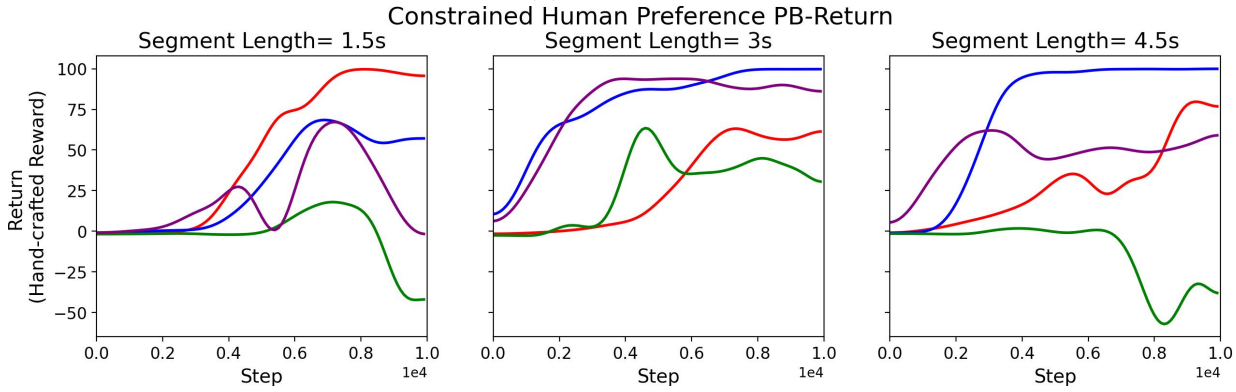
Figure D.13: Error Rate of Constrained Human Preference

Fig. D.14 shows the learning results of various RL algorithms with constrained human preference, where the 1st, 2nd, and 3rd row show the HC-Return, PB-Return and average value prediction, respectively. By comparing the HC-Return, PB-Return and average value prediction, we can roughly conclude the bad performance of SAC is closely related to its bad value estimation, For instance, SAC's HC-Return is the lowest, but it gets the highest average value prediction, compared to TD3 and LSTM-TD3.

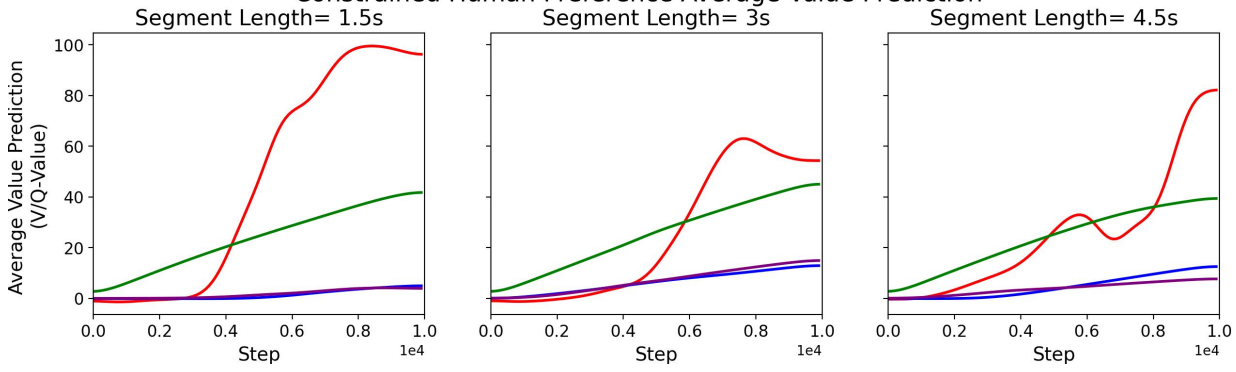
More data from these four algorithms on different segment length are shwon in Fig. D.15, D.16, D.17, D.18, where the rewards and actions collected along the learning are



(a) HC-Return



(b) PB-Return



(c) Average Value Prediction

Figure D.14: RL Learning Results on Constrained Human Preference

presented as well.

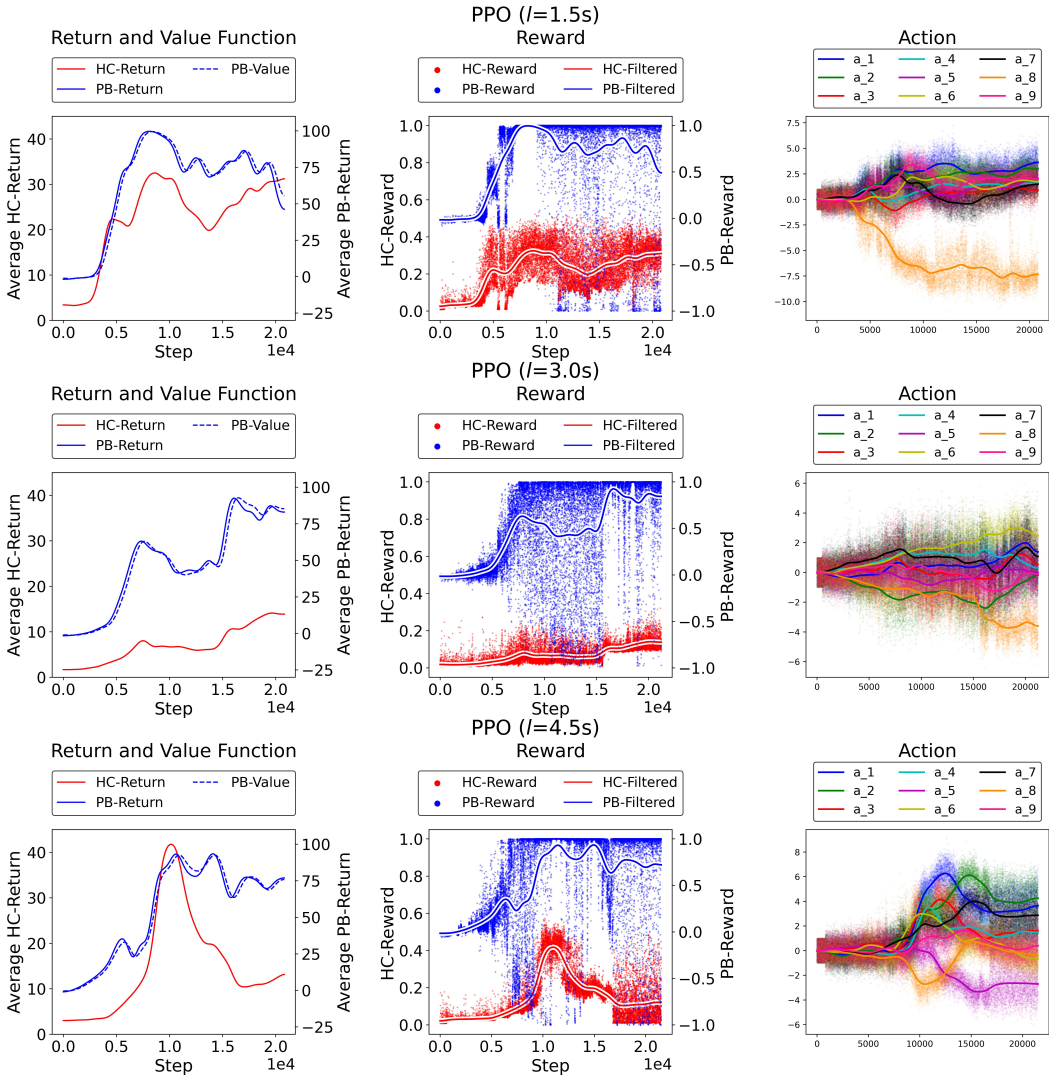


Figure D.15: PPO with Constrained Human Preference on Various Segment Lengths

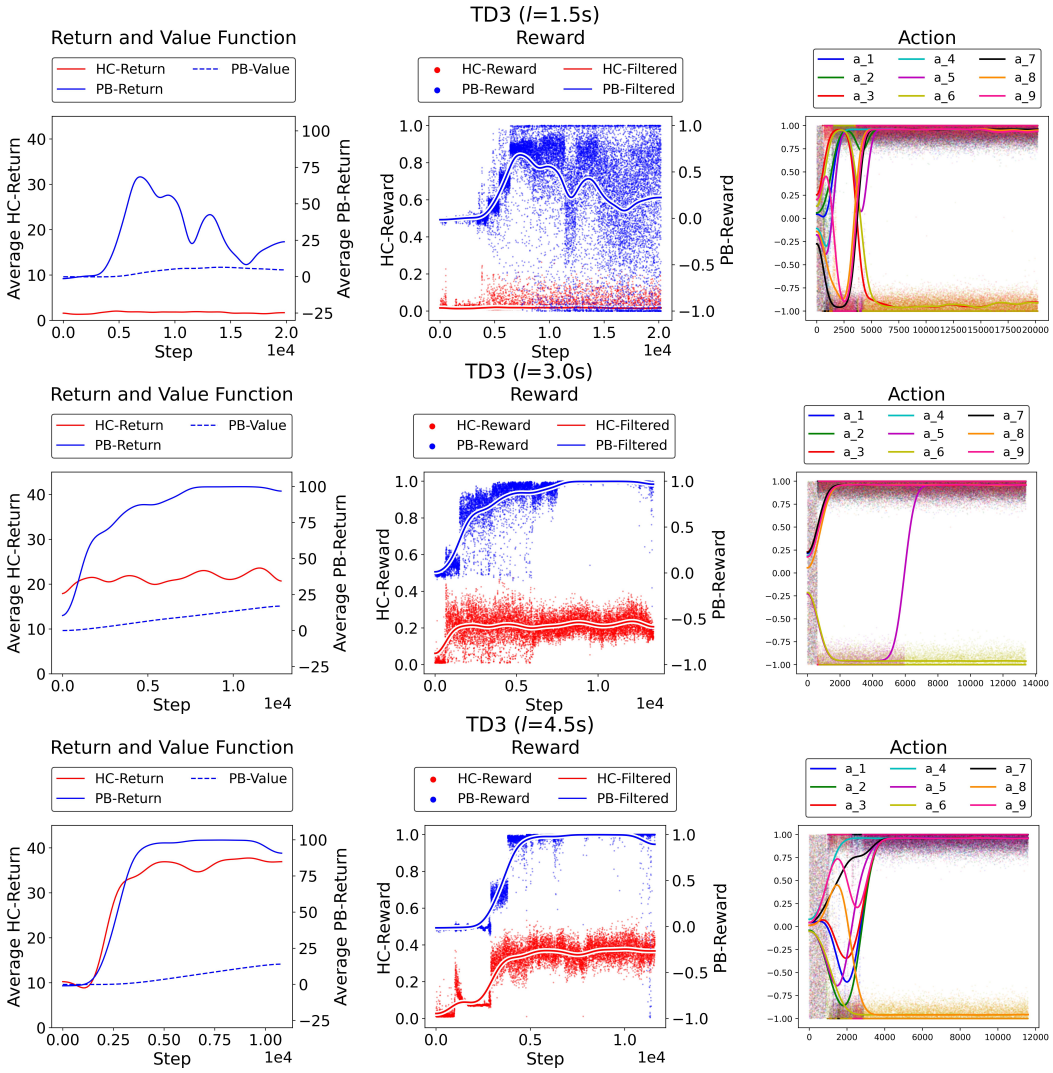


Figure D.16: TD3 with Constrained Human Preference on Various Segment Lengths

D.6 Results on Unconstrained Expert Preference

D.6.1 Expert Data Summary

Table D.8 summarizes the participation of the experts, where the Request Sequence corresponds to the sequence of participation, and the numbers in the 3rd to 7th columns indicate how many preference labels are provided in total and for each preference choice. As men-

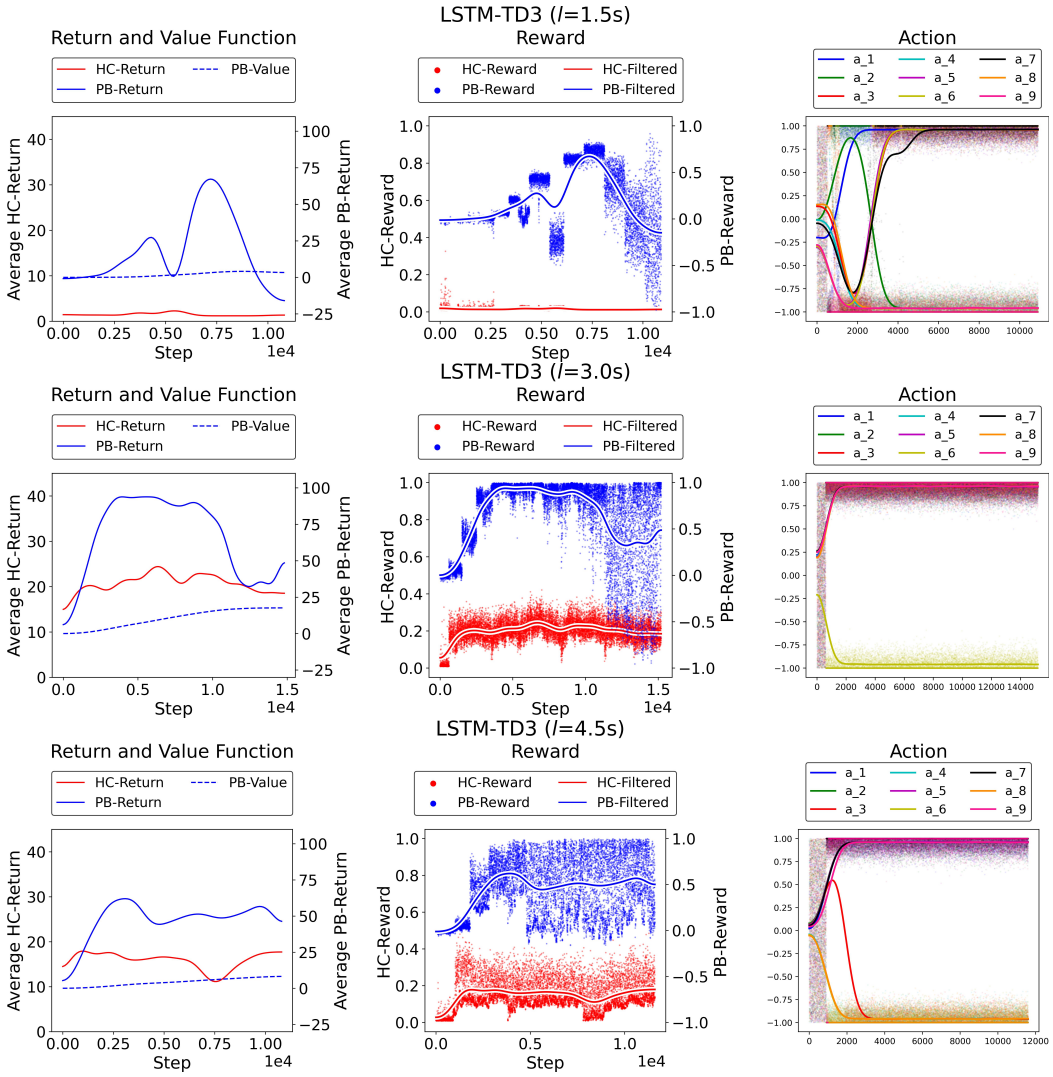


Figure D.17: LSTM-TD3 with Constrained Human Preference on Various Segment Lengths

tioned in section D.2.6, there may be cases where the two segments of a segment pair are equally engaging or unengaging and these preference labels cannot be used in preference learning as we formulate it as a two-class classification problem, so in this experiment, we ask expert participants to provide at least 20 distinct preferences, i.e., either "Right is Better" or "Left is Better", in each session to make sure we have sufficient training data. As seen in Table D.8, each session has at least 20 distinct preferences. The preference labels are collected based on the segment pool where two segments are sampled from to

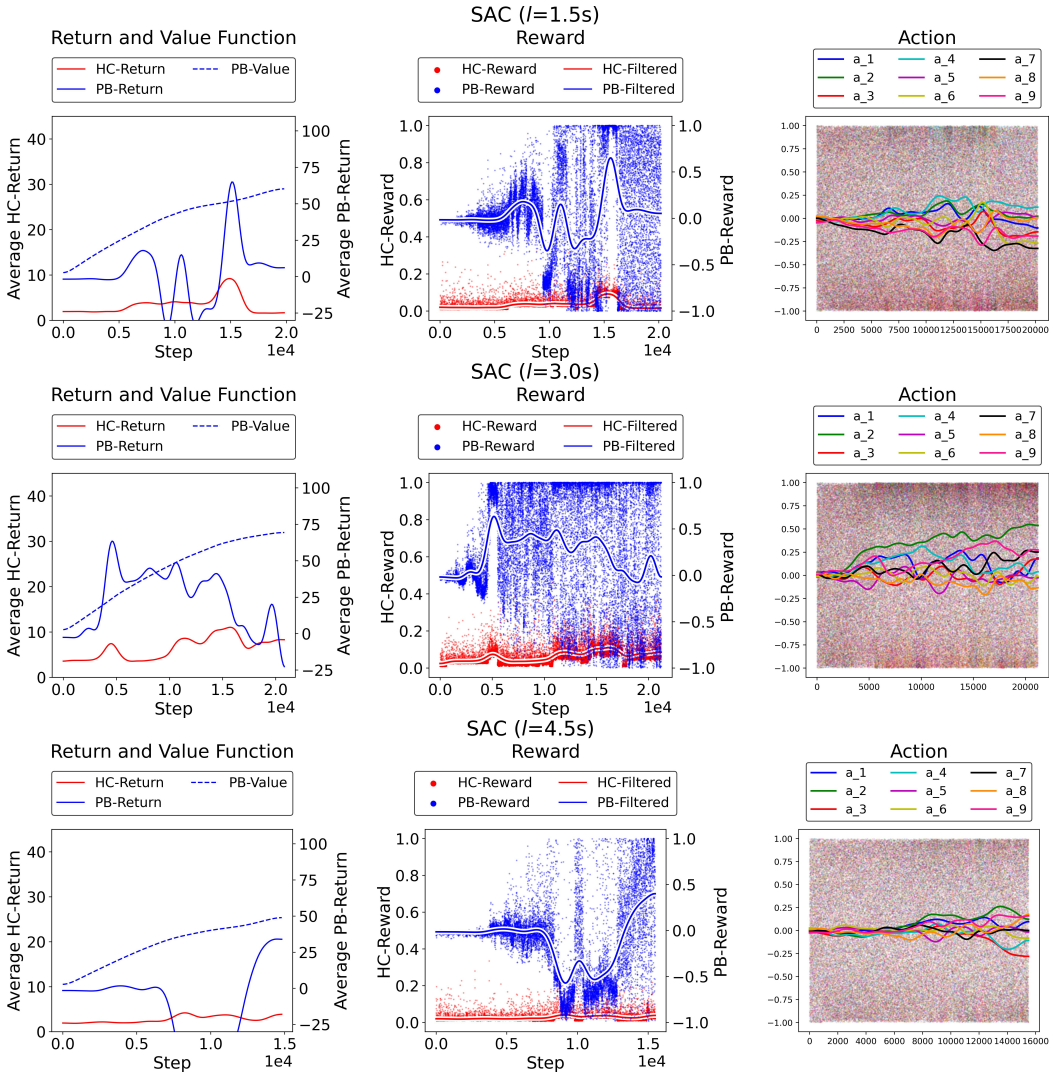


Figure D.18: SAC with Constrained Human Preference on Various Segment Lengths

ask for preference. Fig. D.19 illustrates the segments and segment pairs. Specifically, the top left shows the segments indicated as dots and the segment pairs connected by a solid line, where x-axis is the time step of RL agent[2], y-axis is the times a segment is sampled for preference, different colors correspond to the segment pairs shown to different preference requests, and the segments after the gray-dotted vertical line are added after the last

[2]Segments before time step 0 are collected for pretraining preference-based reward.

preference request. The bottom left of Fig. D.19 is to show how the the segments are distributed over the time, where each short vertical line corresponds to a segment. The top right summarizes the number of segments that are sampled for different times, where the inset pie chart shows the proportion of segments with different sampled numbers. The bottom right shows the proportion of experiences that are at least seen once by preference teachers. From Fig. D.19 we can see that (1) segments are sparsely distributed over the whole trajectory, (2) segments added earlier have higher opportunity to be seen by a preference teacher, and (3) only about 6.1% of experiences[3] is seen by preference teachers.

Table D.8: Summary of Expert Participation

Request Sequence	Pref Request	Total	Distinct Preference			Cannot Tell		
			Right is Better	Left is Better	Sum	Equally Eng.	Equally Uneng.	Sum
PT1	Exp. 1, Ses. 1	24	6	14	20	1	3	4
PT2	Exp. 2, Ses. 1	30	9	15	24	4	2	6
PT3	Exp. 3, Ses. 1	28	11	12	23	5	0	5
PT4	Exp. 1, Ses. 2	28	9	12	21	7	0	7
PT5	Exp. 3, Ses. 2	28	11	10	21	3	4	7
PT6	Exp. 2, Ses. 2	30	11	9	20	2	8	10

Exp.: Expert, Ses.: Session, Eng.: Engaging, Uneng.:Unengaging

Fig. D.20 shows the time spent on each preference query in a session by an expert, where the "equally engaging" and "equally unengaging" labels are highlighted in green and red respectively, the dotted vertical lines separate the different sessions, and the dashed line corresponds to a 3-degree polynomial fit to the time spent on preference queries within a session. Specifically, the time spent on each preference query is measured as the time between when a segment pair is shown to a preference teacher and a preference choice is provided. This is a rough estimation of time spent on a preference query as a preference teacher may be distracted from the current task. For example, there is one preference query of Expert 3, Session 1 cost over 400s which is extremely abnormal. Fig. D.21 shows the distribution of the time spent over each preference choice in each session, where "eng." and "uneng." stands for engaging and unengaging respectively, and rating times over 100s are ignored. From these two figures, we can see that within a session the time spent normally decreases for most sessions, which indicates the experts adapt quickly on how to make a preference choice. An especially outstanding example is the 1st session of exert 2, where

[3]To be more precisely, preference teachers only see the video segments correspond to the experience segments.

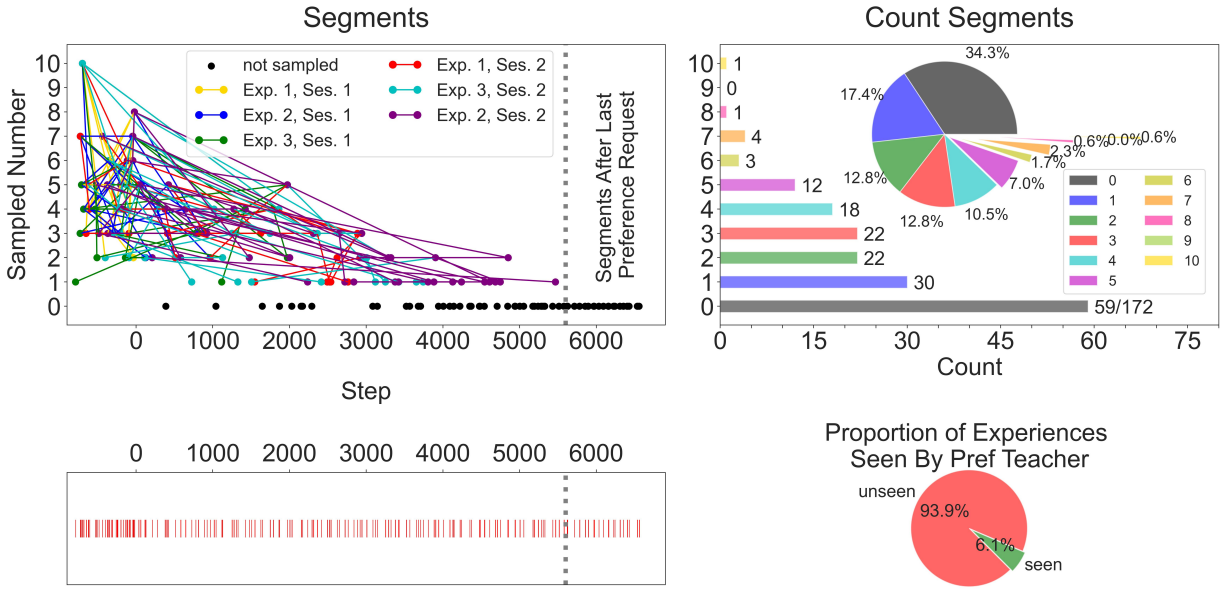


Figure D.19: Segments and Segment Pairs

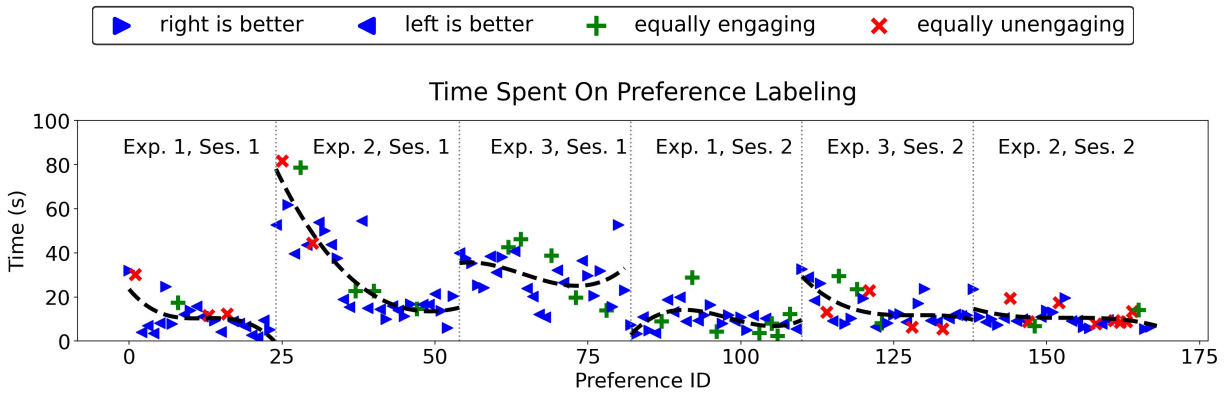


Figure D.20: Expert Preference Choice Time Spending

the first few queries cost over 50 seconds on average while the last few queries cost less than 20 seconds. In addition, for the same expert it is shown that less average time is spent on the 2nd session compared to the 1st session. For instance, expert 2 spent much less time on preference labeling in session 2 than that in session 1, as shown in the green markers in Fig. D.20 and the 2nd and 6th panel in Fig. D.21. Besides, equally engaging and unengaging usually cost more time than distinct preference where either left or right

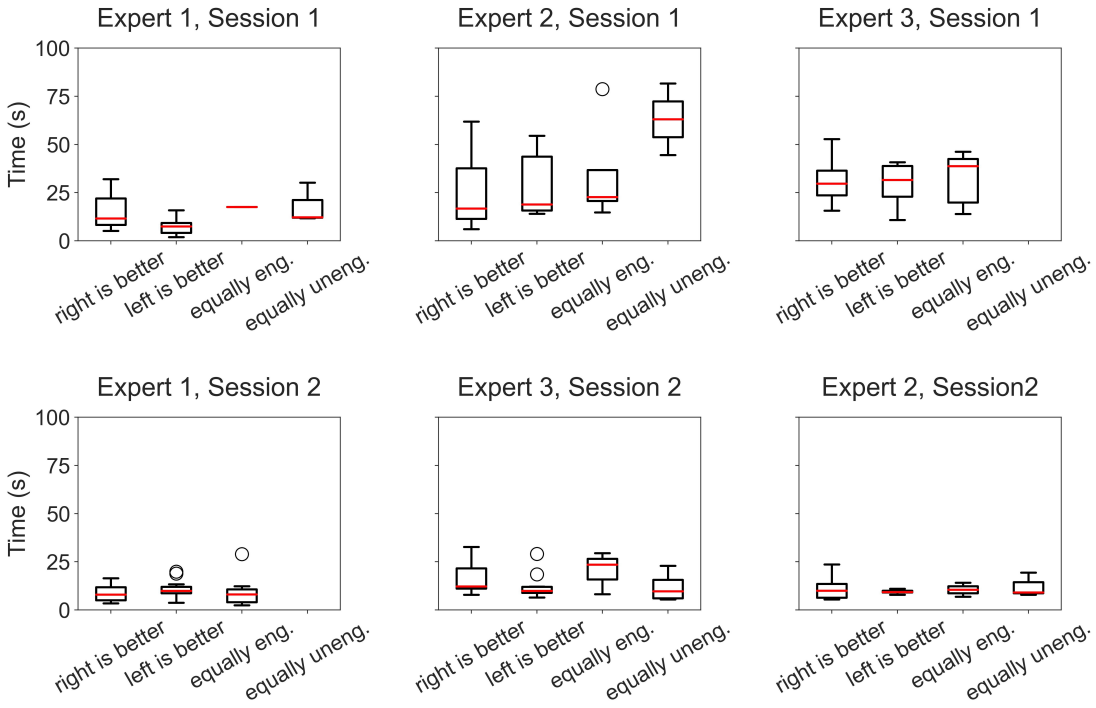


Figure D.21: Distribution of Expert Preference Choice Time Spending

is preferred as shown in Fig. D.21 that the median of "equally engaging" or "equally unengaging" is much higher than that of "right is better" or "left is better".

Fig. D.22 summarizes the agreement between two consecutive PB-Rew checkpoints (where CP0 is the randomly initialized PB-Rew, and CP1-CP6 are the checkpoints of PB-Rew trained after each of the 6 preference teaching sessions) on 500 segment pairs randomly sampled from the segment pool. It can be seen that after trained on the preference labels from the 1st session CP1 is significantly different from the randomly initialized CPO. Then, after continuously trained with new preference labels from the 2nd and 3rd session, the disagreement between two consecutive PB-Rew checkpoints decreases. However, the disagreement between CP3 and CP4 is significantly higher than other cases followed by the disagreement between CP4 and CP5. Overall, the PB-Rew does change much, in terms of preference prediction, after the 1st session, with the maximum of 30% disagreement for CP3 and CP4 and the minimum of 1.8% disagreement for CP2 and CP3.

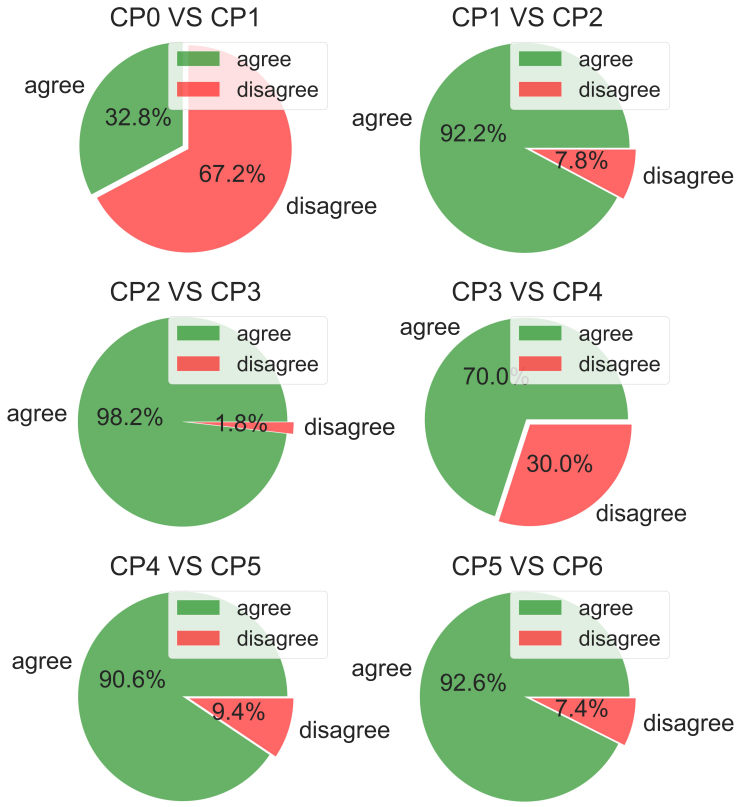
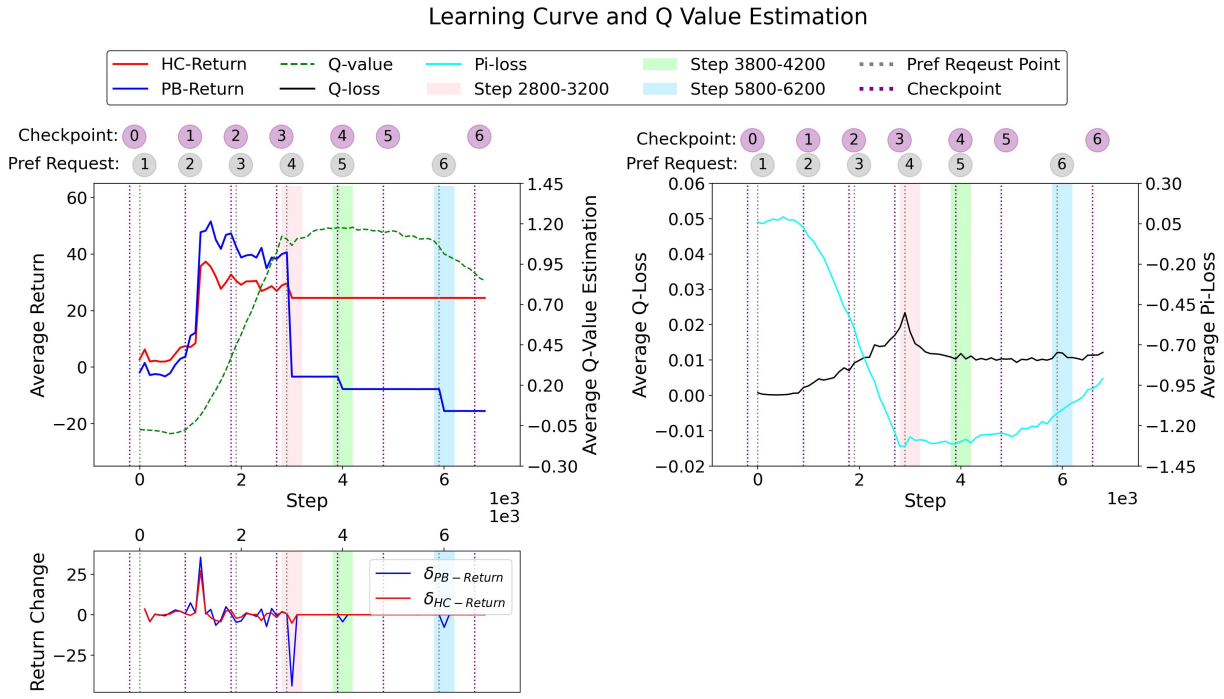


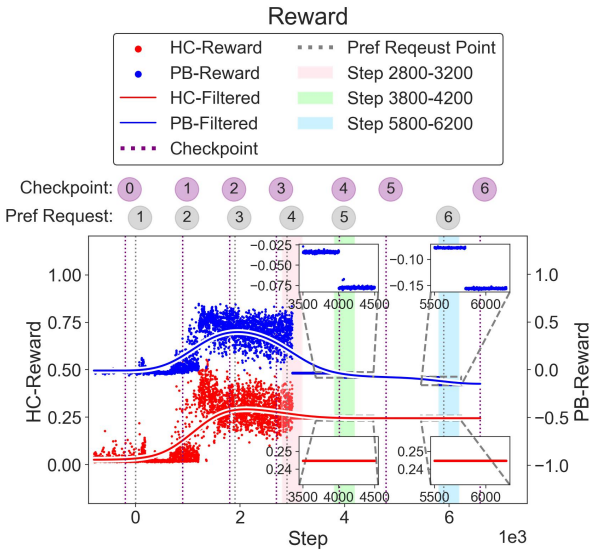
Figure D.22: Preference Prediction-Agreement Between Two Consecutive PB-Rew Checkpoints

D.6.2 Policy Evolution Accompanying Preference-based Reward

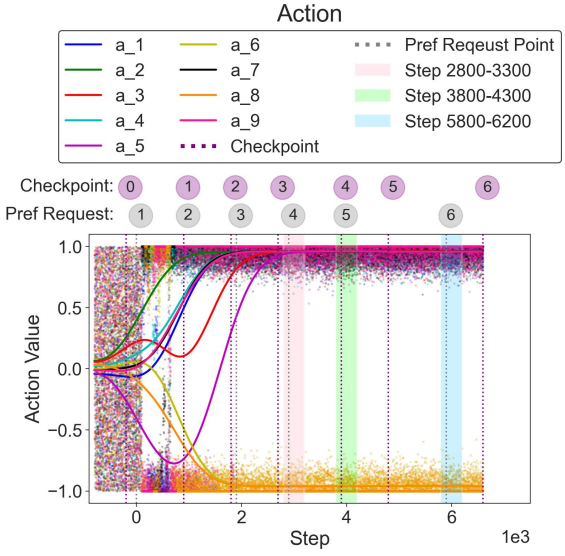
To better understand the observations in the last section, in this section we will take a close look at the evolution of the policy derived from the preference based reward function. Fig. D.23 plots the data collected during the PL+RL. Specifically, the first panel of Fig. D.23a shows the PB-Return (solid blue line) and HC-Return (solid red line) measured in PB-Rew and HC-Rew respectively, and the average Q-value (dashed green line), and the second panel shows the Q-loss (solid black line) and Pi-loss (solid cyan line) of the TD3 agent. The third panel of Fig. D.23a shows the change of PB-Return and HC-Return in the first panel. Fig. D.23b shows the HC-Rew in red and PB-Rew in blue collected during the learning, where HC-Rew is in $[0, 1]$ and PB-Rew is in $[-1, 1]$, the lines are Gaussian filtered rewards for better visualization of reward change, and rewards



(a) Learning Curve on Unconstrained Expert Preference



(b) Rewards Collected During Learning



(c) Actions Collected During Learning

Figure D.23: RL and PL on Unconstrained Expert Preference

before time step 0 are those collected from a random policy for generating segments for pretraining preference labels. Fig. D.23c shows the actions collected during the learning, where different colors corresponds to different dimensions of an action represented by dots, the lines are Gaussian filtered actions for better visualization of action change, and actions before time step 0 are generated by the random policy for preference pretraining. In each sub-figure, from left to right, the gray vertical lines correspond to 6 preference requests (labeled in gray circles above the vertical lines) where the first preference request is for pretraining PB-Rew which happens before RL, while the purple vertical lines correspond to the 7 checkpoints (labeled in purple circles above the vertical lines) of RL agent and PB-Rew, namely CP0(-200), CP1(899), CP2(1799), CP3(2699), CP4(3899), CP5(4799), and CP6(6599), where the number within the brackets is the time step that the checkpoint is created and CP0(-200) is the checkpoint saved before any learning. In addition, the vertical color bars are to highlight the time when the RL agent experienced outstanding PB-Return decreases, where the light red, light green and light blue correspond to the 1st, 2nd and 3rd PB-Return drops respectively. Note that in the PL+RL setting, only the PB-Rew derived from human preference is used to learn a policy, and PB-Rew is only used for comparison purpose.

When comparing the shape of HC-Return (solid red line) and PB-Return (solid blue line) in the first panel and their changes in the thrid panel of Fig. D.23a, they look very similar. When PB-Return increases the HC-Return increases too, and when the PB-Return decreases at about 3000 steps the HC-Return also decreases. This means overall PB-Rew is correlated with HC-Rew. However, there are still distinctions. For example, the increase and decrease magnitudes are different between PB-Return and HC-Return. In addition, the 2nd (light green area) and 3rd (light blue area) PB-Return drops do not cause HC-Return drop. This can be better observed in Fig. D.23b, where the PB-Rew decreases in the 2nd and 3rd drops of PB-Return while the HC-Rew does not. These observations indicate the PB-Rew is to some extent distinct from HC-Rew.

When comparing the average Q-value estimation (dashed green line) and the PB-Return (solid blue line) in the first panel of Fig. D.23a, the average Q-value is underestimated even with discount factor $\gamma = 0.99$ as it is far smaller than PB-Return. We can see that the Q-value estimation increases as the increase of PB-Return and experiences a small fluctuation when PB-Return drops (light red area), then stays relatively stable until the 3rd drop (light blue area) in PB-Return. The average Q-value loss (solid black line) first increases along with the increase of Q-value, then decreases dramatically at the first drop of PB-Return, followed by staying stable from 4000 to 6000 step and increasing gradually after the 3rd PB-Return drop (light blue area). Because the policy loss is defined as the negative of Q-value in order to maximize the Q-value of the action generated by the policy,

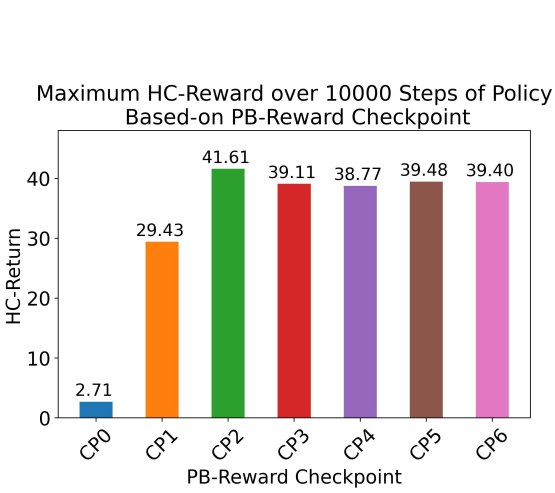
the opposite trend of average Q-value estimation can be observed in the Pi-loss (solid cyan color).

It is particularly interesting to investigate **(1) what causes the decreases in PB-Return**, and to understand **(2) why the HC-Return does not change for the 2nd and the 3rd decrease in PB-Return**. It is worth to mention that in the PL+RL setting, it is very difficult to analyze the causality of performance change, because both the PL and RL can cause significant change in the performance measured in PB-Rew and they also interfere with each other. On one hand, if the policy of RL does not change, significant change in PB-Rew from PL can cause significant change in PB-Return. On the other hand, if the PB-Rew does not change, significant change in the policy can lead to significant change in PB-Return as well. What makes it more complicated is both the PB-Rew and policy are non-stationary.

Based on our observation in Fig. D.23c that the policy does not show significant change, we hypothesize: *the performance drops in PB-Return are mainly caused by the scaling change of PB-Reward, rather than the change of underlying human preference*. To validate our hypothesis, we show results in the rest of this sub-section based on the following ideas: (1) if the PB-Reward corresponding to the drops changed significantly, the performance (measured in HC-Reward) of the policy learned from the fixed PB-Reward checkpoints should be significantly different too; and (2) if the performance drops are caused by the change of human preference, similar observation should be able to be reproduced in simulation.

Fig. D.24 compares the performance (measured in HC-Reward) of TD3 with different PB-Reward checkpoints saved during the study with unconstrained experts as indicated in Fig. D.23, where the TD3 agents are trained from scratch with the reward calculated from the fixed PB-Reward checkpoints, and only the best performance for each checkpoint is compared. We can see that the PB-Reward increases from CP0 to CP2, and experiences a slight decrease from CP2 to CP3, then stays relatively stable thereafter. This result is contrary to the idea (1) and supports our hypothesis.

Fig. D.25 shows the results of a simulated preference model switch, where a predefined preference model is switched to its opposite after 4000 steps as indicated by the vertical dotted line. Here, the simulated preferences switch from active to calm, i.e., the more active actuators the better for the first 4000 steps, then afterwards the less active actuators the better. The solid bold lines correspond to the success cases where the agents successfully adapt from one preference to another. Nevertheless, it is interesting to see that the adaption takes a long time rather than happens immediately, because we use all preference labels collected until now to train a PB-Reward model. This means only when the number



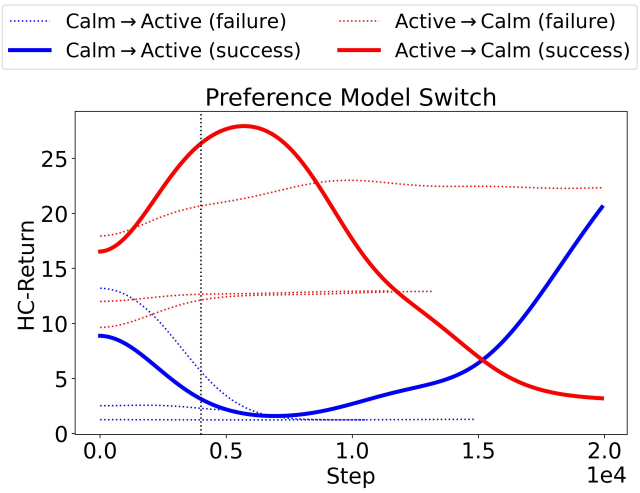


Figure D.24: Compare Performance of TD3 with Different PB-Reward Checkpoints

Figure D.25: Effect of Preference Model Switch

of the preference labels based on the new preference model exceeds that based on the old preference model, the agent is able to dominantly be driven by the new preference. Surprisingly, we also observe many cases where the agents fail to adapt to the switched preference model, as indicated by the dotted lines which are stopped early when no change is observed in the learning curves for a long time. On the one hand, this indicates that the proposed method is not suitable for the cases where the preference model may change over time and a faster adaption to the change is critical. However, on the other hand, the slow adaption means the method is robust to preference noise which may be either introduced by mistake or wrong preference labeling caused by very similar segment comparison. Overall, the results rejects the idea (2) and supports our hypothesis.

Fig. D.26 shows a small set of video segments from the segment pool collected along the whole process, where the dots in the learning curves indicate the time steps corresponding to the video segments, and the segment before time step 0 is collected during PL pre-training phase. Due to the difficulty of showing a video segment with hundreds of frames, we take the maximum value over the frames for each pixel to form an image to represent the maximum intensity of the actuators within a video segment. Fig. D.26 shows the images of the video segments and the histograms of these images. We can see that as the HC-Return increases when more actuators are activated and the last three video segments in Fig. D.26 look very similar which indicates the policy does not change much

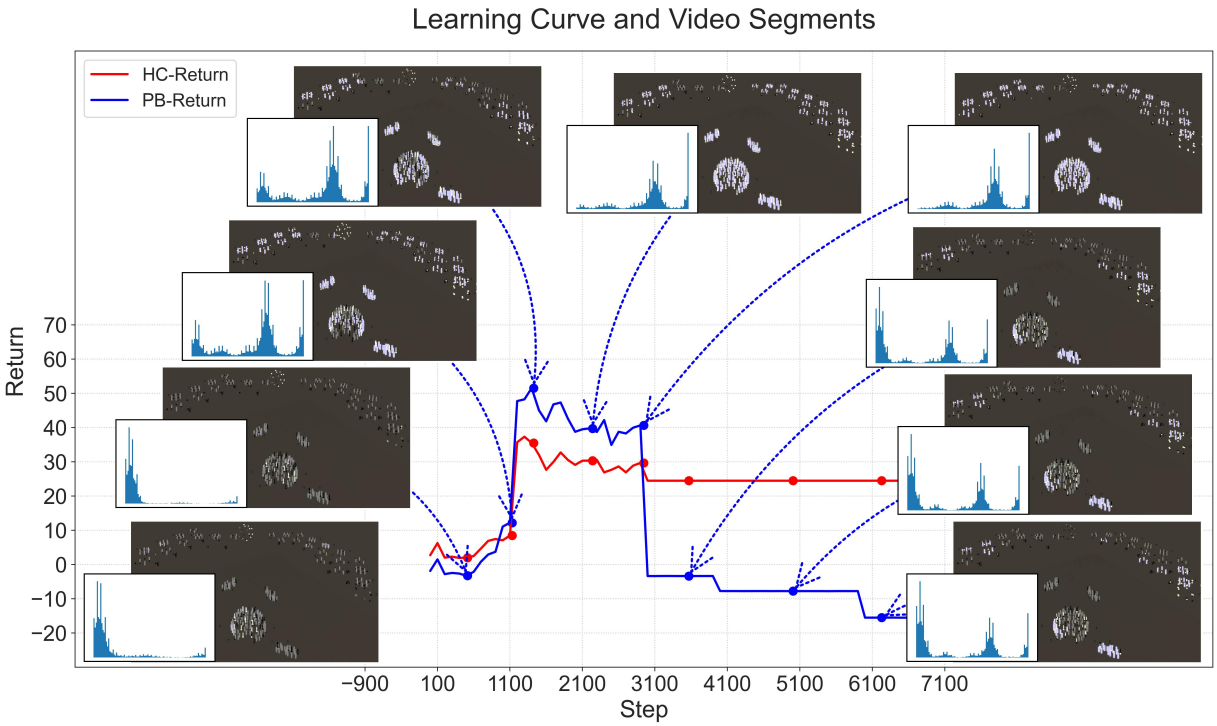


Figure D.26: RL Agent Learning Curve and Video Segments

Further Analysis on PB-Return Drops

From Fig. D.23a and Fig. D.23b, it can be seen that the three drops happened immediately after the 4th (Exp. 1, Ses. 2), 5th (Exp. 3, Ses. 2), and 6th (Exp. 2, Ses. 2) preference request (indicated in gray circles in Fig. D.23a), where these preference requests are followed by PB-Rew training on all collected preference labels. Then, a natural assumption is that the drops in PB-Return are mainly caused by the change of PB-Rew. To validate the assumption, we sample all experiences from the replay buffer of RL agent and predict PB-Rew on these experiences using the PB-Rew model checkpoints, and plot the distribution of the predictions in Fig. D.27 where the x-axis corresponds to the checkpoint version, e.g., CP3 (2699) is the 3rd checkpoint saved immediately after time step 2699, and the red and blue dotted lines connect the medians and means of the predictions respectively. Besides PB-Rew prediction, we also use the PB-Rew model checkpoints to predict the preference to see to what extent the two consecutive checkpoints of PB-Rew model agree with each other, where we randomly sample 500 segment pairs in total from the segment pool to predict the preference label with irrational probability $p_{ip} = 0$. Fig. D.28 is the

preference prediction-"accuracy" between two consecutive PB-Rew checkpoints, where the prediction of the previous checkpoint is treated as the target and the diagonal values of each panel indicate the proportion of samples that the two consecutive checkpoints make the same prediction of preference. Fig. D.22 summarizes the results in Fig. D.28 into agree and disagree categories to show how the agreement between two consecutive PB-Rew checkpoints changes. According to Fig. D.23a, the 1st drop in PB-Return happened between the 3rd and 4th checkpoint, the 2nd drop in PB-Return happened between the 4th and 5th checkpoint, and the 3rd drop in PB-Return happened between the 5th and 6th checkpoint. This observation aligns well with the results in Fig. D.27 where we can see that there is a dramatic decrease in the median and mean of the PB-Rew predictions between CP3 and CP4, while the change from CP4 to CP5 and from CP5 and CP6 is minor which corresponds to the last two minor drops in Fig. D.23a. In addition, these changes are also reflected in the preference prediction illustrated in Fig. D.28 and Fig. D.22, where the disagreement between CP3 and CP4 in terms of preference prediction is significantly higher than other cases followed by the disagreement between CP4 and CP5. To summarize, all of the aforementioned results indicate the cause of the decreases in PB-Return is the change of the PB-Rew induced from human preference, which answers the question **(1)** in the previous paragraph.

Because HC-Rew is fixed, it can be roughly used as an indicator of the policy change, i.e., if the policy is changed the HC-Return will change, otherwise HC-Return will stay unchanged. When comparing the HC-Return and PB-Return in Fig. D.23a, we found that HC-Return does not change for the 2nd and 3rd drop in PB-Return but only experiences a mild decrease for the 1st drop in PB-Return. Therefore, we suspect that the policy only has a mild change at the 1st drop in PB-Return and keeps unchanged for the other two. To validate that, we visualize the action generated from the RL policy in Fig. D.23c and Fig. D.29. Fig. D.23c shows the actions collected during learning where the actions are those leading to the rewards shown in Fig. D.23b and return shown in Fig. D.23a, from which no obvious difference can be found near the PB-Return drops. To catch more subtle changes in the policy, Fig. D.29 depicts the actions[4] generated by the policy checkpoints in the observations from the experience replay buffer of the RL agent, where each panel corresponds to an action dimension of the action space defined in Table 9.2 with value range $[-1, 1]$ and the inset figure of each panel with much smaller y-axis range is to visualize the action differences on different observations. According to Fig. D.29, overall except exitorSpeedLimit the difference among policy checkpoints is very small. However, as shown in the inset figures, the policy change is much larger from CP3 to CP4 than that from CP4 to CP5 and from CP5 to CP6. Therefore, we conclude the answer to the question **(2)** is

[4]Note that exploration action noise is not included.

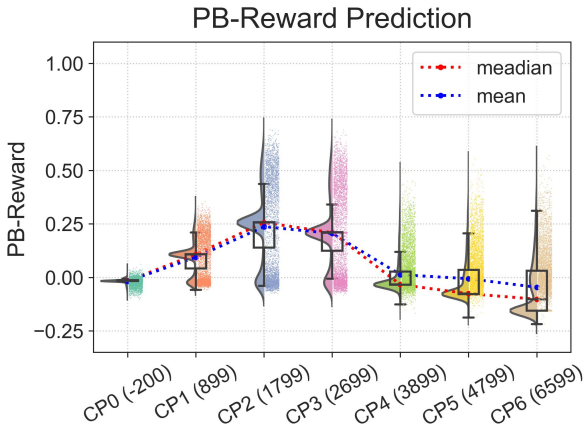


Figure D.27: PB-Reward Prediction of PB-Rew Checkpoints

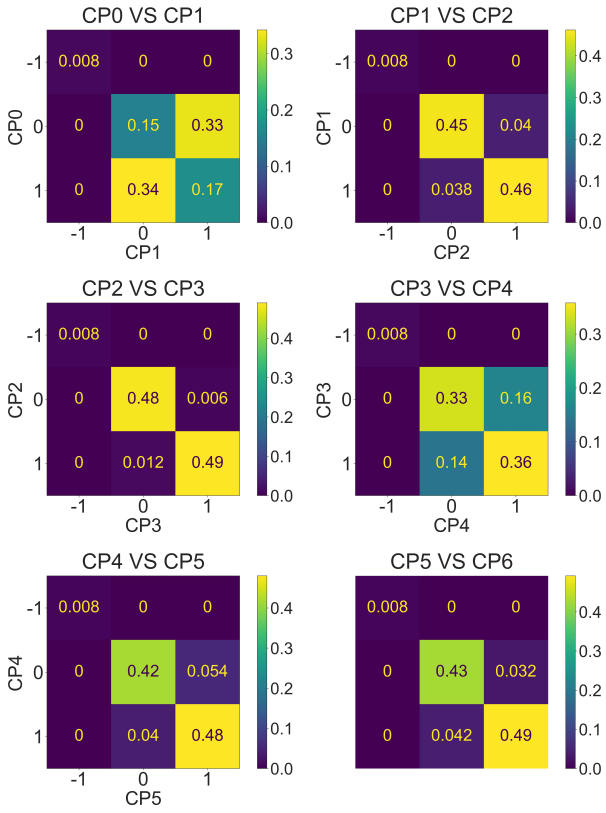


Figure D.28: Preference Prediction-"Accuracy" Between Two Consecutive PB-Rew Checkpoints

HC-Return does not change for the 2nd and 3rd drop in PB-Return is because the policy does not change much. Another question can be raised is that if PB-Rew changed why the policy does not change for the 2nd and 3rd drop in PB-Return, since the policy is derived from PB-Rew. One possible explanation is under some limitation even if the PB-Rew has changed along the preference learning, as long as the relative relationship, i.e., which experience is better than another among the PB-Rew of the experiences does not change, the policy will keep stationary as well. Note that the prerequisite of this explanation is the PB-Rew change is within some limitation rather than a dramatic change as that from CP3 to CP4 in Fig. D.27.

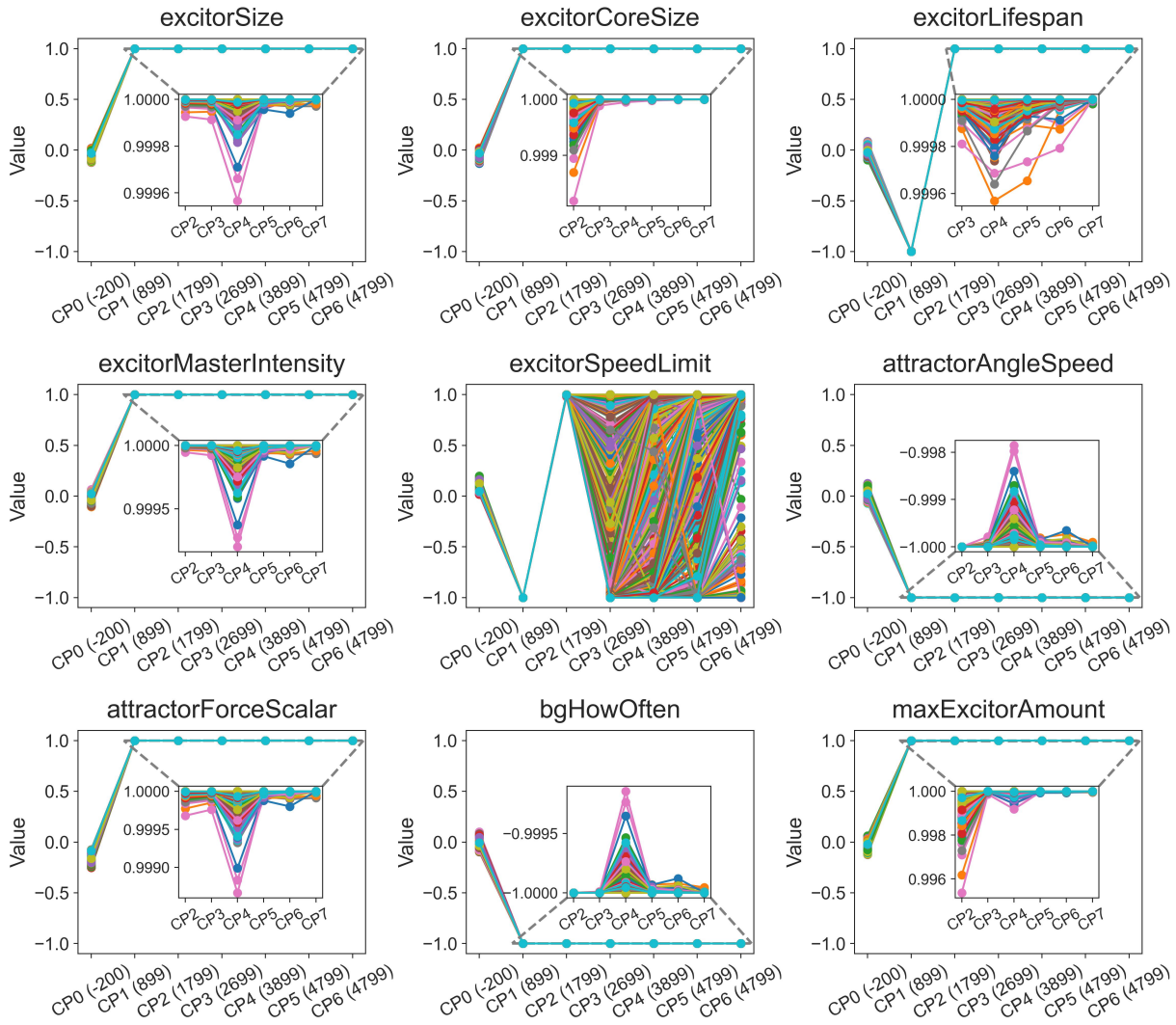


Figure D.29: Action Prediction of Policy Checkpoints

D.6.3 Preference Prediction of Various PB-Reward Models

In Table 9.7, the PB-Reward checkpoints are trained with all preference labels collected after each preference request session in an incremental online learning method, which means these checkpoints incorporate all preference teachers' preference and may not able to reflect the difference among the preference labels from various sessions. Therefore, we separately trained a set of PB-Reward estimators (called Exp. x, Ses. y where x is the expert

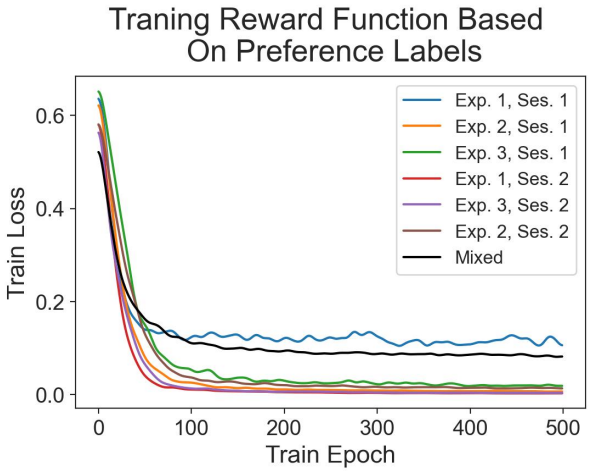


Figure D.30: Offline PB-Reward Estimator Training

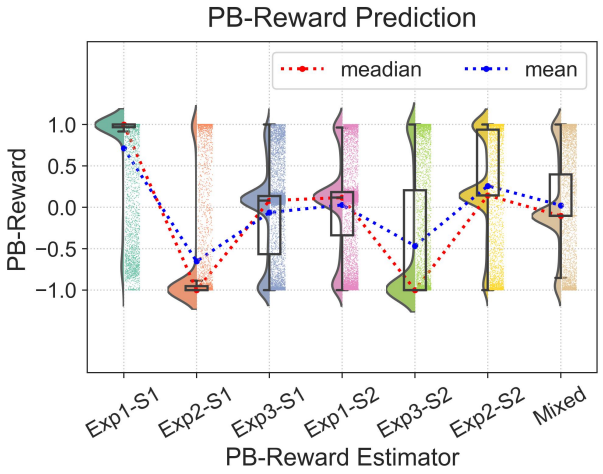


Figure D.31: Reward Prediction of Separate PB-Reward Estimator

id and y is the session id of that expert) for the preference labels collected within each preference request session and an estimator (called mixed) with all preference labels, where all estimators use the same neural network structure listed in Table D.2 and the training of these estimators is offline batch learning. Fig. D.30 shows the training loss of these estimators within 500 epochs, where the all preference labels within each session are used for training and there is no validation and test dataset because of the limited number of labels. We can see the training loss decreases quickly at the first dozens of epochs and remains relatively stable. Fig. D.31 shows the distribution of the reward prediction of these estimators on experiences in the replay buffer of the RL agent. From this figure, it can been seen that on the same experience set the distribution of the predicted reward looks very different for different estimators, except the estimator from Exp3-S1 looks similar to that of Exp1-S2. Particularly, even for the the same expert, the estimators from different sessions look quite different from each other, e.g., Exp1-S1 and Exp1-S2. It is naturally to conclude that (1) experts have different preference, and (2) the same expert changed his/her preference from one session to another. However, these conclusions are inconsistent with the experts' the survey data where none of the experts indicating their preference changes over the time, which will be shown in session D.6.4 and discussed in session D.7.

Even the reward prediction distributions of different estimators have different shapes, it is still possible to make the same preference prediction from different estimators as long as the relative relationship, i.e., which one is better than another, is similar. Therefore,

Table D.9: Preference Prediction Accuracy of Separately Trained PB-Reward Estimators

Estimator \ Pref Request	PT1	PT2	PT3	PT4	PT5	PT6
Exp. 1, Ses. 1	**0.950**	**0.667**	**0.522**	**0.810**	0.476	**0.600**
Exp. 2, Ses. 1	0.450	**1.000**	**0.522**	**0.667**	**0.714**	**0.550**
Exp. 3, Ses. 1	**0.600**	**0.583**	**1.000**	**0.571**	0.381	**0.600**
Exp. 1, Ses. 2	**0.900**	**0.750**	**0.652**	**1.000**	**0.571**	**0.550**
Exp. 3, Ses. 2	**0.600**	**0.750**	**0.609**	**0.619**	**1.000**	0.450
Exp. 2, Ses. 2	**0.600**	**0.625**	**0.652**	**0.619**	**0.524**	**1.000**
Mixed	**0.950**	**1.000**	**0.957**	**1.000**	**1.000**	**0.950**

Table D.9 shows the preference prediction accuracy of the separately trained PB-Reward estimators on the preference labels provided by the experts, where the results in bold are those greater than 0.5, the results in red indicate the horizontal estimator is trained with the vertical preference labels, the results in blue highlight the prediction accuracy of an estimator trained on labels from one session but examined on another session, and the curves connect the sessions from the same expert. It is not surprising that all estimators' preference prediction accuracy (results in red) are above 0.95 on the preference dataset that they are trained. In terms of generalization, only 4 of 30 cases the estimators have prediction accuracy that are lower than 0.5 (results with smaller font-size and not in red) on preference data that they did not see during training, which indicates the estimator from one session can generalize better than a random guess in terms of preference prediction. This observation reveals that even though the reward prediction distributions of the estimators look significantly different from each other (Fig. D.31), the preference predictions of these estimators may be very similar. For instance, the reward prediction distributions of Exp1-S1 and Exp1-S2 look very different in Fig. D.31, but their preference predictions are not so different, as estimator trained on PT1 with training accuracy 0.950 has prediction accuracy 0.810 on PT4 and estimator trained on PT4 with training accuracy 1.00 has prediction accuracy 0.900 on PT1.

Except the separately trained PB-Reward estimator, it is particularly interesting to see how the estimator trained offline, i.e., mixed in Table D.9, with all preference labels performs compared to that of trained online, i.e., CP6 in Table 9.7, in terms of preference prediction accuracy. Table D.10 compares the preference prediction accuracy of CP6 and that of mixed, where the maximum in each column is in bold and the total number of training epochs of the preference dataset from each session is shown in the brackets. It

Table D.10: Preference Prediction Accuracy Comparison Between Online and Offline Mix-trained Estimator

Estimator \ Pref Request	PT1	PT2	PT3	PT4	PT5	PT6
CP6 (Table 9.7)	0.900 (30)	0.708 (25)	0.565 (20)	0.952 (15)	0.619 (10)	0.500 (5)
Mixed (500) (Table D.9)	**0.950**	**1.000**	**0.957**	**1.000**	**1.000**	**0.950**

is clear that the most recent preference labels are less trained than old preference labels in online training. For example, PT6 is trained 5 epochs while PT1 is trained 30 epochs for CP6. This means the estimator is biased to old preference labels. This is reflected by the prediction accuracy, where CP6 has preference prediction accuracy 0.900 on PT1 but only has 0.500 on PT6. For the offline training, all preference labels are equally went through the training, which causes very similar preference prediction accuracy on data from different sessions. Due to the large training epochs, mixed has higher preference prediction accuracy than CP6 on all sessions.

Table D.11: Preference Prediction Agreement on Random Segment Pairs

Estimator \ Estimator	Exp. 1 Ses. 1	Exp. 2 Ses. 1	Exp. 3 Ses. 1	Exp. 1 Ses. 2	Exp. 3 Ses. 2	Exp. 2 Ses. 2	Mixed
Exp. 1, Ses. 1	**1.0**	0.304	0.496	**0.62**	0.278	**0.658**	0.482
Exp. 2, Ses. 1	0.304	**1.0**	0.372	**0.526**	**0.848**	0.418	**0.668**
Exp. 3, Ses. 1	0.496	0.372	**1.0**	0.476	0.368	**0.514**	0.408
Exp. 1, Ses. 2	**0.62**	**0.526**	0.476	**1.0**	0.47	**0.598**	**0.578**
Exp. 3, Ses. 2	0.278	**0.848**	0.368	0.47	**1.0**	0.418	**0.672**
Exp. 2, Ses. 2	**0.658**	0.418	**0.514**	**0.598**	0.418	**1.0**	**0.522**
Mixed	0.482	**0.668**	0.408	**0.578**	**0.672**	**0.522**	**1.0**

Until now we investigated the preference prediction accuracy on CP1-CP6 where the human preference labels are available, it is also worth to see how the PB-Reward estimators induced from the human preference labels agree with each other on the segment pairs that are not seen by a user. Since the newly sample segment pairs are not labeled, there will not be prediction accuracy but only preference prediction agreement between estimators. Table D.11 shows the preference prediction agreement, i.e., the proportion of segment pairs over 500 segment pairs randomly sampled from the segment pool that two estimators make

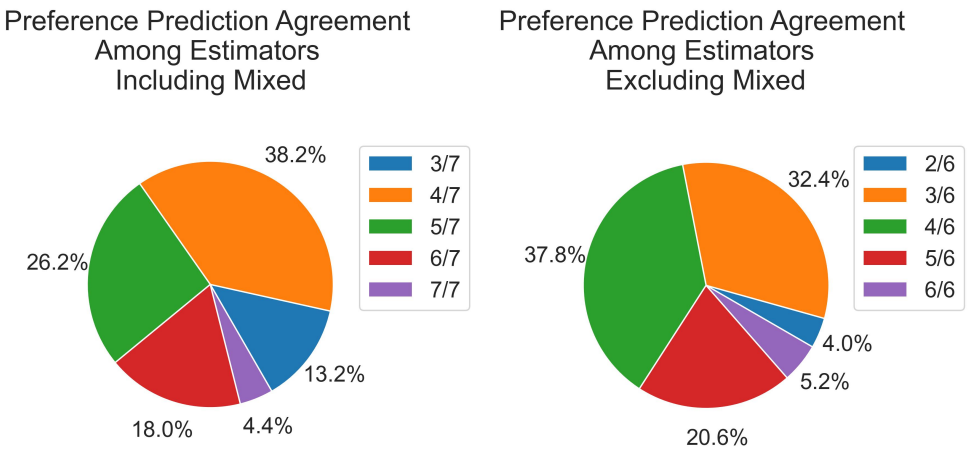


Figure D.32: Preference Prediction Agreement Among Separately Trained Estimators

the same preference prediction, where the agreements greater than 0.5 are in bold and the agreement between the estimators induced from the different sessions of the same expert is highlighted in red rectangle. Overall the separately trained estimators (not including mixed) do not always agree with each other, where for some cases, e.g. Exp.1, Ses.1 vs Exp. 3, Ses. 3, estimators have very low agreement, while for others, e.g., Exp. 2, Ses. 1 vs Exp. 3, Ses. 2, they have high agreement. Compared to separately trained estimators, mixed, which is trained with labels from all sessions, has higher agreement with the separately trained estimators. The agreement between Exp. 2, Ses. 1 and Exp. 2, Ses. 2 and that between Exp. 3, Ses. 1 and Exp. 3, Ses. 2 are lower than 0.5, which indicates Expert 2 and Expert 3 disagree with themselves in different sessions. This observation is in line with the observation from Fig. D.31, but is inconsistent with the experts' the survey data where none of the experts indicating their preference changes over the time. Again, we will show the survey data in session D.6.4 and have a discuss on this in session D.7.

Fig. D.32 summarizes the proportion of 500 randomly sampled segment pairs that the estimators make the same preference prediction, where the left panel includes the mixed estimator while the right panel excludes the mixed estimator. It can be seen that on 38.2% and 32.4% of the 500 segment pairs that 4/7 and 3/6 estimators are agree with each other, and on 4.4% and 5.2% of the 500 segment pairs that all estimators are agree with each other. This also indicates these estimators are not completely agree with each other, but they still share something in common.

D.6.4 Expert Teacher Survey Data

Table D.12 shows the questions adapted from System Usability Scale (SUS) [44, 23, 45] and used to measure the usability of our web-based interface. The SUS score [22] is calculated as follows

$$S_{sus} = 2.5 \times \left(\sum_{i=1,3,5,7,9} (Q_i - 1) + \sum_{j=2,4,6,8,10} (5 - Q_j) \right), \tag{D.3}$$

where the final score is in range $[0, 100]$.

Table D.13 shows questions adapted from Robot Incentives Scale (RIS) [228, 113] and used to measure user's perception of his/her emotion, utility, and social connection of using the preference teaching system.

Table D.12: Questions Adapted From System Usability Scale

Question Label	**Question**
SUS-Q1	I think that I would like to use this interface frequently.
SUS-Q2	I found the interface unnecessarily complex.
SUS-Q3	I thought the interface was easy to use.
SUS-Q4	I think that I would need the support of a technical person to be able to use his interface.
SUS-Q5	I found the various functions in this interface were well integrated.
SUS-Q6	I thought there was too much inconsistency in this interface.
SUS-Q7	I would imagine that most people would learn to use this interface very quickly.
SUS-Q8	I found the interface very cumbersome to use.
SUS-Q9	I felt very confident using the interface.
SUS-Q10	I needed to learn a lot of things before I could get going with this interface.

In the survey, we also welcome expert participants to leave general comments on our study in order to catch oversights that are not well considered before. Among the experts, only expert 2 left general comments in the two sessions as follows:

- Exp. 2, Ses. 1:

Table D.13: Questions Adapted From Robot Incentives Scale

Question Label	**Question**
Emotion-Q1	I like this teaching process.
Emotion-Q2	I enjoy providing my preference of video clip to teach the sculpture how to engage visitors.
Emotion-Q3	I find it is easy to choose the video clip I prefer.
Emotion-Q4	I am unhappy with the "Cannot Tell" queries.
Emotion-Q5	If I should use the teaching interface, I would be afraid to provide unreliable feedback.
Emotion-Q6	If I should use the teaching interface, I would be afraid to ruin the sculpture's behavior by providing incorrect feedback.
Utility-Q1	This teaching process would be able to make the sculpture's behavior more engaging.
Utility-Q2	This teaching process would be useful for me to teach the sculpture how to generate more engaging behavior.
Utility-Q3	This teaching process would be able to help any user to adapt the sculpture's behavior to his/her preferences.
Utility-Q4	This teaching process would provide reliable assistance to adapt the sculpture's behavior.
Social-Connection-Q1	I feel socially connected to the visitors of the sculpture, even though they are not aware I am one of the teachers who taught this sculpture.
Social-Connection-Q2	I am aware that the sculpture I am teaching is intended to engage its visitors.
Social-Connection-Q3	I am proud to tell the visitors of the sculpture that I contributed to teaching the sculpture.
Social-Connection-Q4	I would recommend to my friends that they use the interface to help teach the sculpture.

"It can be hard to judge what is 'best' for 'engagement' at this scale and without context of the physical environment – some things that look very interesting in these clips might be annoying or impossible to really see in the physical context of the sculpture. Also, for the interface, I found I was constantly maximizing and minimizing the videos and that got annoying. And would be nice to directly select the image rather than needing to say 'left' or 'right' - i kept being afraid I was picking the wrong one because I

might mix up left and right."

- Exp. 2, Ses. 2:

  " I think 'engaging activity' relies on changes to the behaviour from one such behaviour to another over time and in response to activity of the space, and sometimes it is the contrast between two behavioiurs that will be 'engaging' – not sure how to represent this in the study."

It is very interesting to see that expert 2 has very high expectation by extending the scenario to very complicated one where human visitors are within the space of the sculpture. Indeed, in real application engaging behavior depends on the context rather than simply determined by the sculpture's behavior. For example, if a visitor is very excited and intended to expect more reaction from the sculpture, highly active behavior may be more engaging, while if the visitor is very cautious and much enjoyed smooth reaction, less active behavior may be more engaging than highly active one.

D.7 Discussion

Even though our results indicate learn how to engage from preference is promising, there are still questions worth further investigation. Based on our observations, we will devote this section to discuss the bias in preference learning based reward model, world representation gap between video and experience segment, common preference VS personalized preference, expert VS novice preference teacher, and some limitations of the work shown in this chapter.

D.7.1 Bias in Preference Based Reward Model

In the previous section, we mentioned that according to the results on reward and preference prediction of PB-Reward estimators, it seems the same expert disagrees with him/herself between different sessions, while the experts clearly expressed in the survey data that their preference do not change. These contrary results remind us there must be something is overlooked. If the experts' preference do not change, then the reward estimator must be biased so that it cannot preciously capture the experts' preference. To further investigate the source of the potential bias, we employ t-distributed stochastic neighbor embedding (t-SNE) [279] to embedding the observations from whole experience buffer into 2 dimensions for better visualization. Fig. D.33 shows the observation embedding. Inside, all observations are shown in the top left panel, where the observations within the segments for

different preference request are shown in different colors, the size of the markers indicates the magnitude of HC-Reward of the experience that the original observation is from, and the purple ones are those not included within any segment sampled for human preference. We visually group all embedded observations into three groups, i.e., G1, G2 and G3, according to the 2D distance, and the top right panel shows the distribution of the embedded observations based on their HC-Reward which indicates the distinct difference among the three groups with respect to HC-Reward. The rest sub-panels show the embedded observations from the segments for different sessions. Combining and comparing these panels, we can see that the for the first session only very limited observations (only observations in G2 and G3 are included) are shown to experts for preference labeling, whereas for the second session broader observations (all three groups are included for expert 2 and 3) are seen by experts.

These biased observations included in the segments used for collecting human preference along with the limited human knowledge that can be carried by preference labels are believed to be the cause of biased reward estimators. Let us consider an expert who prefers active behavior but not hyper active behavior, as mentioned by expert 3 in Table 9.6 and depicted in Fig. D.34. At the beginning because the policy is random, it is more likely to have segments that are less active. Then, when two of them are shown to the expert, the one that is more active will be preferred. However, this does not mean the more active, the better. Then, along with the policy evolving, more segments that are much more active are added, and when two very active segments are shown to an expert, the expert may prefer the one less active. Similarly, this does not mean the less active, the better. As depicted in Fig. D.34, when only partial preference is included within the preference dataset, this will cause the preference based reward model cannot capture the whole picture of human preference and generalize badly to unseen data. Therefore, only when diverse segment pairs, that are able to cover sufficient diverse observations, are preference labeled, the experts' preference can be more preciously captured.

Since we can not change the information that can be carried by a preference label, the best way to avoid biased reward model is to reduce the bias of the observations in the segments sampled for human preference. This can be realized by having a large pretraining preference labels with diverse segment pool. However, this is impractical for applications where collecting the pretraining preference labels are expensive and the policy that can be used to collect diverse segments is unavailable. Therefore, another approach, that does not rely on a very large pretraining preference dataset but using continuously growing preference dataset, is to periodically reinitialize the reward estimator and train with the whole preference dataset from scratch. However, unless the policy of RL agent is reinitialized and retrained, the reward bias will be maintained within the policy for a very long time. In

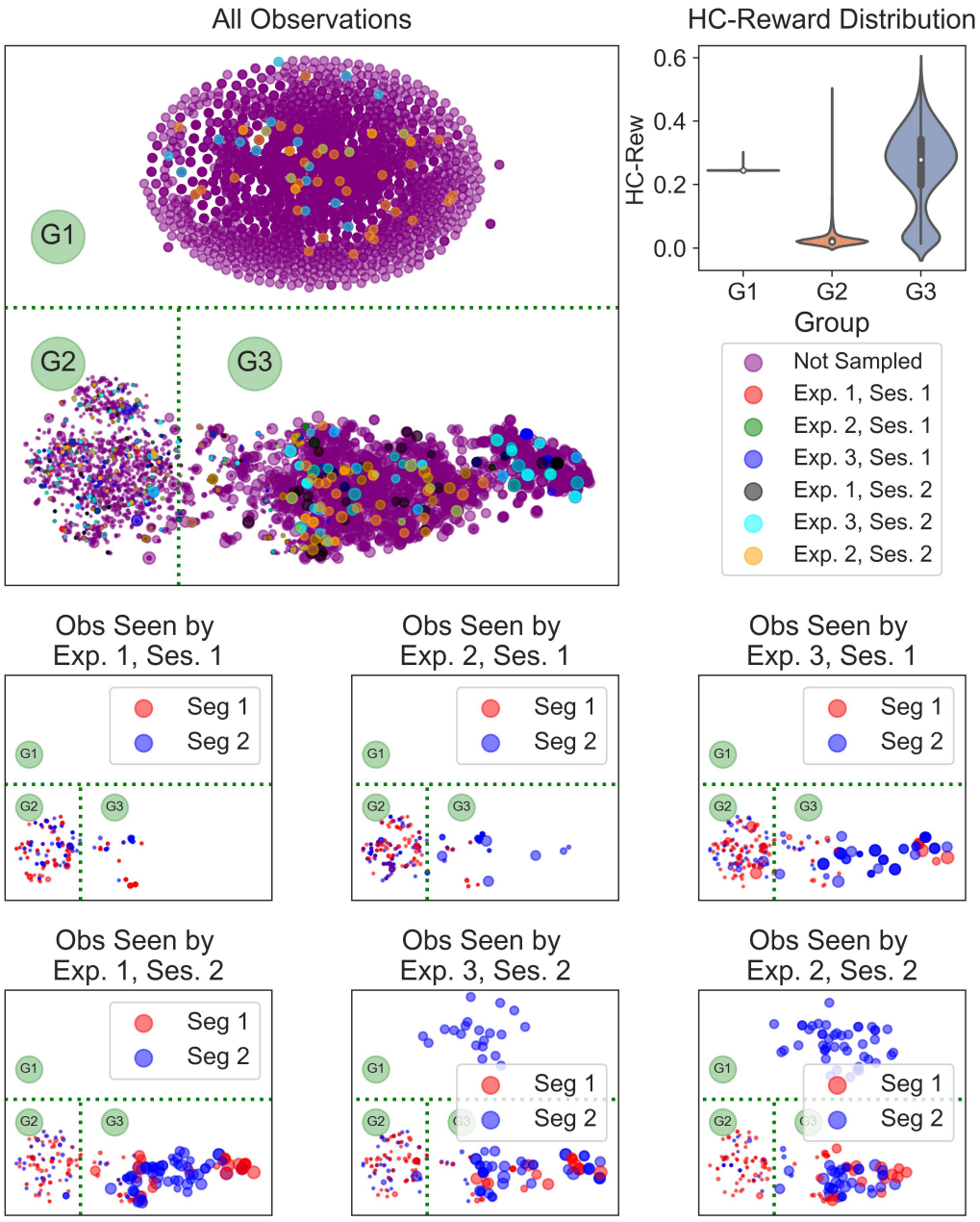


Figure D.33: Observation Embedding

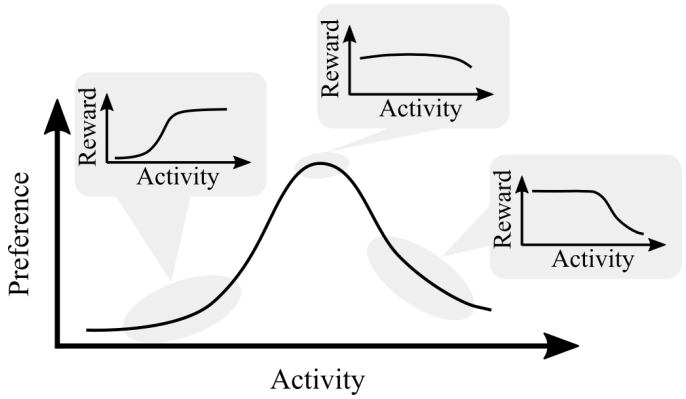


Figure D.34: Example Expert Preference Model and Biased Reward Models

general, these potential approaches can only alleviate the bias, but it is basically inevitable because of the co-evolution of RL policy and preference-based reward model.

D.7.2 Reward Scaling In Reward Saturation

Not only the bias may cause the shift of preference based reward model, but also the reward scaling after reward saturation will lead to the change of the preference based reward model. By definition, *reward saturation* refer to the situation where the current reward model makes reward predictions around the boundaries, either upper or lower bound of the output range, so that no more reward prediction can be squeezed into to keep the accurate preference prediction without interference. Therefore, in order to accommodate more preference labels, the *reward scaling* will happen, which will loosely spread the reward predictions.

To understand the concept of reward saturation and reward scaling, we need to first understand how the reward model is updated given segment pairs and the corresponding preference labels. Without loss of generality, let us consider an extreme case where only one experience is included within a segment $\sigma = (o_t, a_t, r_t, o_{t+1}, d_t)$, which is essentially $K = 1$ according to the representation of segment introduced in Section D.2.2. Given a segment pair (σ^0, σ^1) and a PB-Reward reward estimator R^{pb} approximated by a neural network, whose output activation function is tanh, then the preference prediction of the estimator defined in Eq. 9.2 can be represented as

$$\hat{P}\left[\sigma^0 \succ \sigma^1\right] = \frac{e^{R^{pb}(\sigma^0)}}{e^{R^{pb}(\sigma^0)} + e^{R^{pb}(\sigma^1)}}. \tag{D.4}$$

Now let us only consider the loss on this segment pair, which has the following form

$$L\left(R^{pb}\right) = -\left((1-y)\hat{P}\left[\sigma_e^0 \succ \sigma_e^1\right] + y\hat{P}\left[\sigma_e^0 \prec \sigma_e^1\right]\right), \tag{D.5}$$

If the preference label $y = 0$, then $L\left(R^{pb}\right) = -\frac{e^{R^{pb}(\sigma^0)}}{e^{R^{pb}(\sigma^0)}+e^{R^{pb}(\sigma^1)}}$. Then, to minimize the loss, we could either increase $R^{pb}(\sigma^0)$ or decrease $R^{pb}(\sigma^1)$, as shown in the top right shaded area of Fig. D.35. Similarly, if $y = 1$, then $L\left(R^{pb}\right) = -\frac{e^{R^{pb}(\sigma^1)}}{e^{R^{pb}(\sigma^0)}+e^{R^{pb}(\sigma^1)}}$. To minimize the loss, we could either increase $R^{pb}(\sigma^1)$ or decrease $R^{pb}(\sigma^0)$, as shown in the bottom left shaded area of Fig. D.35. The increase and/or decrease of the reward prediction is realized by tuning the weights of the neural network used to approximate the reward model.

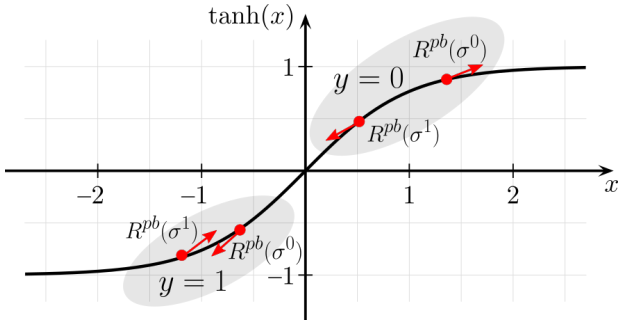


Figure D.35: Reward Learning Example

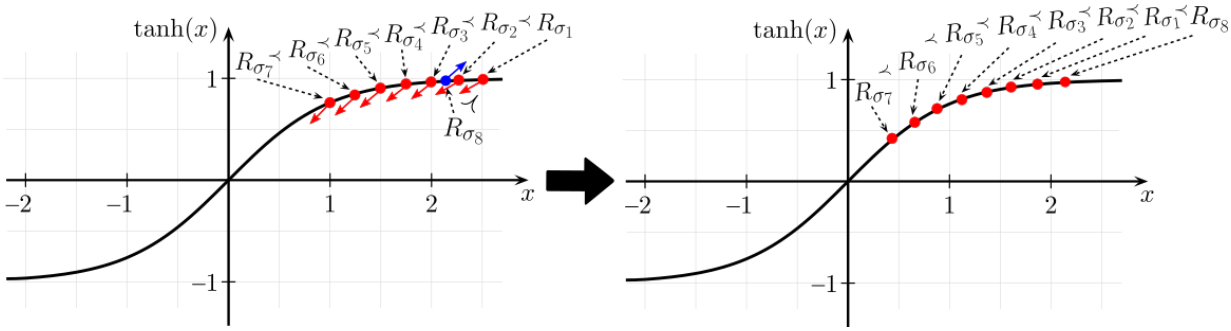


Figure D.36: Reward Scaling In Reward Saturation

Now, let see how reward scaling happens in reward saturation. As shown in the left panel of Fig. D.36, the preference relationship, indicated by $\prec$, among σ_1 to σ_7 is correctly incorporated into the reward model R, and the predictions, indicated in red dots, are concentrated around the upper bound of the tanh activation function. Than, when a new

preference data point ($\sigma^0 = \sigma_1, \sigma^1 = \sigma_8, y = 1$) comes, the reward prediction R_{σ_8} on σ_8 (in blue dot) is actually lower than that on σ_1. Therefore, to minimize the loss, we need to either increase R_{σ_8} or decrease R_{σ_1}, or do both. However, since R_{σ_8} is already very close to the upper bound of the output, which is what we called reward saturation, it is easier to decrease R_{σ_1} by tuning the model weights. At the same time, we still need to keep the preference relationship among σ_1 to σ_7, which consequently causes the decreases in σ_2 to σ_7 as well, which is what we called reward scaling, and finally leads to the results in the right panel in Fig. D.36. Similar phenomenon will happen when reward is saturated around the lower bound.

Either reward saturation nor reward scaling in Preference Learning is undesirable for RL agent. To be more concrete, if reward saturation happened, this means many experiences have very similar reward predictions, which will make it hard to differentiate which action in an observation is better than others. This will further cause extremely slow learning speed of a good policy that maximizes the preference based reward. Reward scaling would also to harmful, because the change in the scale of the reward signal will cause the change in value function approximation that is commonly adopted in RL, and the unstable value function approximation will further lead to unstable policy update. Ideally, the reward prediction could nicely spread over the whole output range of the reward estimator, at the same time similar experiences could have similar reward predictions while distinct experiences could have differentiable reward predictions. In order to realize this, new techniques must be introduced and need further study.

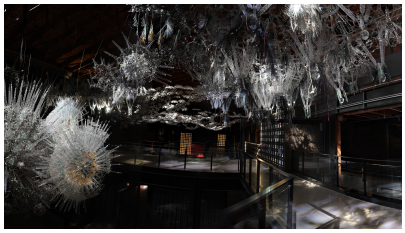
Different from the bias introduced in Section D.7.1 where the biased reward model may generalize to completely wrong reward as shown in Fig. D.34 and lead to a policy that inconsistent with the human preference, reward scaling in PL does not necessarily cause different policy as long as the preference relationship is unchanged, which we call policy invariance under genuine reward scaling. It is not hard to imagine that if only the reward scale is changed but which action is better than another does not change, the final optimal policy should be the same. This is essentially the a reward shaping [202], where the shaped reward R', original reward R, and shaping reward function F satisfies $R' = R + x \times F$, $F = R$ and $x \in (-1, \infty)$.

D.7.3 World Representation Gap Between Video and Experience Segment

The fundamental idea of this work is to learn a preference based reward function from human preference by showing preference teacher two video segments. However, we should

notice that not the video segments but the experience segments are used in training the reward function. Therefore, we cannot ignore the fact that there is a representation gap between video and experience segments, which further results in a perception gap and influences the effectiveness of preference transferring from human to the preference-based reward function. Take our application as an example, on one hand, the video segment rendered in the simulator cannot visually reflect subtle activation intensity difference of actuators, while this can be captured by the experience segment where actuator intensity is represented by real value. On the other hand, the action pattern is easier to be perceived by preference teacher from the video segment than that from experience segment, because neither the representation of observation space nor the reward estimator are designed to deal with temporal information that is essential for learning preferred behavior pattern. Therefore, it is not hard to imagine that transferring human preference from human to reward model is easier for tasks where the representation gap is small than the tasks where the gap is large.

If applying the proposed method to real world application, the gap will be more prominent because of limited sensing capacity, complex lighting condition, and complicated surrounding environment, etc. Fig. D.37 shows some distinct scenes that could exacerbate the representation gap. Fig. D.37a and Fig. D.37b show different camera positions will have different views with different obstacles and be affected different by light coming through the windows. Fig. D.37c shows a partially public accessible gate which may be open or closed depending on the business owner, where when it is closed only the part close to elevator publicly accessible. Fig. D.37d shows the light condition on a special event at night, which is significantly distinct from that at daytime. Fig. D.37e shows a group of three visitors during daytime, where one (highlighted in red arrow) of them knows more about the sculpture and leads the group interaction with the sculpture by giving demonstration. Fig. D.37f shows two group interactions, where one group of two is close to the camera and another group of four beyond the gate as highlighted with red arrows, and they are all guests from the event held onsite. Fig. D.37g shows a worker (highlighted in red arrow) is introduce the sculpture to her potential customers. Fig. D.37h shows a one-to-one interaction where a visitor self-explored how to interact with the sculpture. Fig. D.37i shows a worker who is just passing by the sculpture without any intention to interact. The various scenes shown in the figure are good examples to demonstrate how complex the context can be perceived by a human when a human see a sequence of such images from a video segment. As mentioned by expert 2 in section D.6.4, the engagement of the sculpture's behavior also depends on the context, whereas this context cannot be represented by the observation of an RL agent. Without a good understanding of world representation gap, it will be very difficult to successfully apply the proposed method to

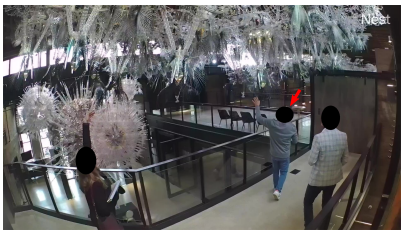


(a) Different Camera Position

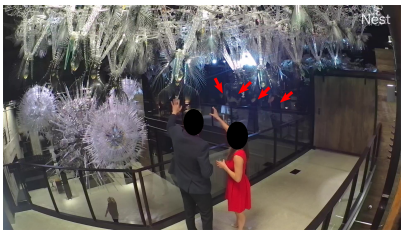
(b) Different Camera Position

(c) Partially Public Accessible Gate

(d) Light Condition on Special Event

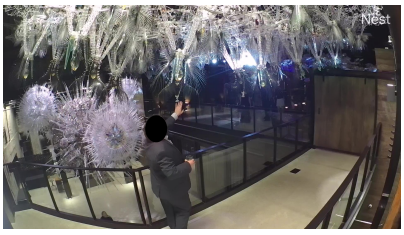


(e) Group Visitors at Daytime

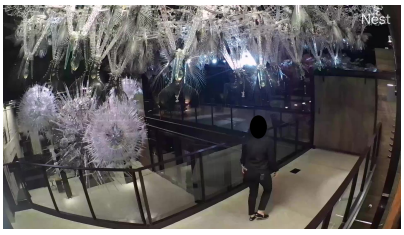


(f) Group Visitors at Night

(g) Worker Guided Visitors



(h) Self-explorer



(i) Worker Passing By

Figure D.37: Example Scenes Challenging World Representation Gap

real world and may possible to result in wrong reward and cause safety concern.

D.7.4 Common Preference VS Personalized Preference

In this work, we continuously trained the reward model with the preference labels from different expert teachers, with the assumption that experts share common preference with each other and with potential visitors of the sculpture. One may argue that Preference Learning should target to individual rather than a group of people, as the group of people may have very different preferences. It does make sense if learning from different underlying preference models, the reward model induced from the preference labels provided

by these different preference models may fluctuate continuously resulting in highly non-stationary reward model, especially for opposite preference models. However, experts' preference cannot be dramatically different and must share something in common, because otherwise, there is no way for experts to design a pre-scripted behavior that everyone is happy. Actually, for pre-scripted behavior there is more strict assumption that not only the experts but also the visitors share the same preference on engaging behavior. The common preference assumption is, to some extent, validated by our survey data shown in Table 9.6 that experts do share something in common in term of engaging behavior. Another reason, that we use the whole preference dataset to train a reward model rather than personalizing the reward model for each expert, is the number of preference labels is too small for each expert to avoid highly biased reward model as discussed in section D.7.1, unless we can introduce techniques to estimate uncertainty in the reward model which is definitely an interesting direction for the future. In addition, if there are hundreds of preference teachers, it is impractical to maintain a set of reward model and policy for each of them.

Nevertheless, it does not mean learning personalized preference is not a good idea or impossible. If the preference teacher population is not too large, we could separately learn different reward functions and policies for different teachers by tracking each of them. In this way, personalized engaging behavior would be more appealing for the teacher. However, if the preference teacher population is too large to separately maintain the personalized reward model and policy, or if it is impossible to track each teacher, we can probably cluster preference teachers by some metrics to reduce the preference teacher types and separately train reward model and policy for each cluster. For this method, we also need to find a way to connect a specific preference type with the user that the robot is facing without much input from the user. For example, ideally we can infer the user's preference from the way he/she interact with the robot, or collect a small set of preference labels, e.g., only 1 or 2, from the user, then infer the user's preference from the provided preference labels, given the segment pairs for this small set of preference labels are representative to differentiate one preference type from the others.

D.7.5 Expert VS Novice Teacher

Expert and novice preference teachers may share common preference as discussed before, but it is very likely that they have different preference on the tools that can be used to teach a LAS how to engage. Because of the limited knowledge carrying capacity of the proposed preference teaching, expert 1 and 2 are prone to other ways to teach engaging behavior, such as action demonstration, action correction, or programming, as shown in Table 9.4.

However, we need to understand that these methods are way cognitively demanding and difficult to use for novices, especially considering on parameterized action space, novices do not know how to preciously interpret parameterized action into the raw actuator space. In addition, one practical challenge in our application is the robot, i.e., a LAS, cannot be paused for a user to correct its action and resumed later, which is quite unique to other action correction applications where the robot could be paused and wait for human input. What makes things worse is the there are two types of HRI involved in real application as illustrated in Fig. D.38 without considering the potential human-to-human interaction popular in group interaction, where one type of HRI happens between the teacher and the robot and the second type of HRI happens between the robot and visitors, which means even if we could pause the robot for human input, we could not pause the visitors who are engaging with the robot. For experts, they have a lot of experiences and are able to imagine various of scenario so that they could possibly design the behavior of a robot without the involvement of the second type of HRI. However, this is impossible for novice teacher. Therefore, we believe preference teaching is probably the most feasible way to allow novices to teach the LAS how to generate engaging behavior.

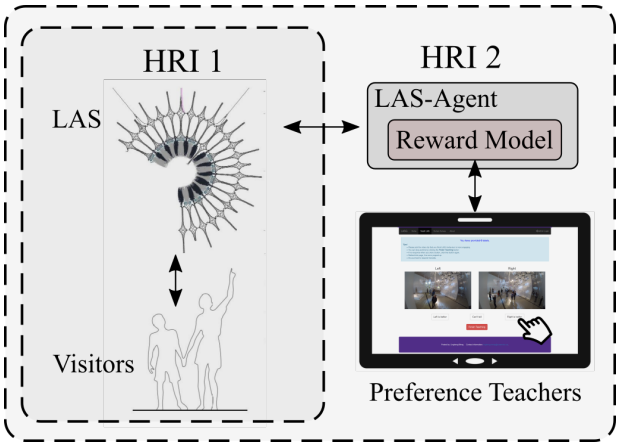


Figure D.38: Two Types of HRI Involvement