Practical Reinforcement Learning for Adaptive Robotic Manipulation: Sample Efficiency, Sim-to-Real Transfer, and Context Inference

by

Amir Mehdi Soufi Enayati
M.Sc., Sharif University of Technology, 2018
B.Sc., Sharif University of Technology, 2016

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Mechanical Engineering

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/ Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Practical Reinforcement Learning for Adaptive Robotic Manipulation: Sample Efficiency, Sim-to-Real Transfer, and Context Inference

by

Amir Mehdi Soufi Enayati
M.Sc., Sharif University of Technology, 2018
B.Sc., Sharif University of Technology, 2016

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor
(Department of Mechanical Engineering)

Dr. Yang Shi, Departmental Member
(Department of Mechanical Engineering)

Dr. Teseo Schneider, Outside Member
(Department of Computer Science)

ABSTRACT

Modern robotic systems seek the ability to adapt to novel tasks and environments in a sample-efficient and robust manner. A framework is proposed to enable such adaptability through three interrelated and complementary contributions in the field of reinforcement learning (RL) for robot manipulation. The central challenges compromising the practical use of RL are addressed, including data efficiency, sim-to-real transfer, and context-aware generalization to unseen tasks.

The first contribution addresses the sample-efficiency challenge. **Demo**nstration **E**xploitation by **A**bstract **S**ymmetry of **E**nvironments (**Demo-EASE**), introduces a sample-efficient RL framework with limited demonstrations that can be augmented by exploiting the symmetry. By identifying and leveraging symmetry in manipulation environments, abstract demonstrations are reused across multiple sub-regions of the task space. The implemented masked behavior cloning allows online adaptive balance between pure RL and imitation learning. Demo-EASE shows effective knowledge transfer, improved learning efficiency, and fewer interactions required while generalizing on the workspace of the expert policy.

The second contribution focuses on improving the reliability of sim-to-real transfer. A novel concept, **R**eal-**T**ime **I**ntrinsic **S**tochasticity (**RT-IS**), is introduced, demonstrating that inherent noise of real-time simulations can be beneficial when approximating real-world uncertainty. Experimental validation on simulated and physical robot tasks confirms that RT-IS improves deployability, requiring less explicit tuning than domain randomization and relaxing the threshold on modeling precision.

The third contribution addresses the challenge of inferring task representation in a meta-RL setting. A transformer-based belief model, **C**ontext **R**epresentation via **A**ction-**F**ree **T**ransformer encoder-decoder (**CRAFT**), is developed to infer variational latent task belief from sequences of states and rewards, without access to the agent's actions. This action-agnostic approach improves adaptability in partially observable environments and supports effective zero-shot learning. Tested on the MetaWorld benchmark, CRAFT outperforms existing baselines in generalization with meaningful task inference quality.

Together, these contributions complete the roadmap to create a framework for adaptive reinforcement learning in robotics. The results demonstrate how structured data, intentional noise, and agent-agnostic long-horizon attention can create efficient, robust, and adaptable learning systems. This work lays a theoretical and practical foundation for future developments in adaptive robotics, enabling robots to operate across a broad spectrum of environments and objectives while cutting down on retraining.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

After every milestone, it is worthwhile to sit back and reflect on the journey. This path was forged, for ideas are born in discourse, and progress is made through collective action. First and foremost, I extend my deepest gratitude to Dr. Homayoun Najjaran. His trust and welcoming nature opened the door for me to join the Advanced Control and Intelligent Systems (ACIS) Lab. His steadfast support motivated me, and his mentorship guided me through the journey and made this work possible.

I would like to thank Dr. Zengjie Zhang and Dr. Kashish Gupta, brilliant minds and researchers. I am grateful to Zengjie for his dedication, sharp mathematical thinking, and attention to detail. Insightful discussions with Kashish and his passion for machine learning and reinforcement learning were inspiring.

My sincere gratitude goes to my smart teammates, Homayoun Honari and Mehran Ghafarian Tamizi. Late-night brainstorming, pair coding, and debugging in the gym were among my most cherished experiences. I am also indebted to my co-authors, most of whom are colleagues at ACIS, who were instrumental in shaping my research. Thanks to Ram Dershan, Dr. Dean Richert, Jayden Hong, Navid Mahdian - my energetic friend and roommate, Mohammad Jani, and Joel Sol. I also thank the Natural Sciences and Engineering Research Council (NSERC) Canada for financial support through the Alliance Grant CRDPJ 543881-19, Discovery Grant RGPIN-202305408, and RTI-2023-00418.

I was fortunate to have age-old great friends and comrades who supported me and shared so many good memories with me. I thank Amin Mohammadi NasrAbadi, Dr. YussufReza Esmaeili, Dr. Mahshid Mahbod, and Dr. Aliakbar Izadkhah. I also appreciate my colleagues at ACIS for creating an enjoyable and productive environment, and the friends I made in the beautiful cities of Kelowna and Victoria for providing a sense of community far from home.

Finally, and most importantly, my deepest thanks go to my family and loved ones. To my father, who taught me the value of curiosity and dependability; to my mother, her love and belief in me have been a constant source of strength; to my sister, her care and compassion have never wavered; to my uncle, his sincerity and willingness to challenge me when I was unreasonable shaped my growth; and to my significant other, her kind spirit and devoted companionship have been my anchor.

"The impediment to action advances action. What stands in the way becomes the way."

- Marcus Aurelius

DEDICATION

To my father, my mother, my sister, my uncle, and my sweet companion.

In loving memory of my late grandfather, grandmother, and grandaunt, whom I lost throughout the course of my Ph.D., without being able to say a proper goodbye.

PREFACE

Several collaborative publications have contributed to the research presented in this dissertation. The author was primarily responsible for ideation, theoretical development, method design, implementation, and manuscript preparation. All research was conducted under the supervision of Dr. Homayoun Najjaran in the Advanced Control and Intelligent Systems (ACIS) Lab, affiliated with the University of British Columbia, Okanagan, and the University of Victoria.

– The concept of the *Adaptive Robotics Framework*, along with its philosophical, technological, and algorithmic background, is introduced in a survey and formulation paper, which also proposes a unified mathematical framework. Dr. Zengjie Zhang played a key role in the theoretical formalism and co-authorship of this publication:

  - [5] Amir M. Soufi Enayati, Zengjie Zhang, and Homayoun Najjaran. A methodical interpretation of adaptive robotics: Study and reformulation. *Neurocomputing*, 512:381–397, 2022.

  This publication constitutes Chapters 2 and 3.

– The author's initial inspiration for leveraging the notion of abstraction in boosting reinforcement learning stemmed from discussions with Dr. Kashish Gupta on the EASE framework [6, 7]. Building on this idea, the Demo-EASE project introduced the use of symmetry in robotic manipulation, investigated minimal demonstration requirements, and proposed an on-policy extension. A preprint is available [8], and the final version is currently under revision:

  - Amir M. Soufi Enayati, Zengjie Zhang, Kashish Gupta, and Homayoun Najjaran. Sample-Efficient Reinforcement Learning with Symmetry-Guided Demonstrations for Robotic Manipulation. Submitted to *Springer Neural Computing and Applications*, 2025.

  Parts of Chapter 4 are borrowed from this preprint. Also, the implementation developed in Demo-EASE was subsequently used in:

  - [9] Zengjie Zhang, Jayden Hong, Amir M. Soufi Enayati, and Homayoun Najjaran. Using implicit behavior cloning and dynamic movement primitive to facilitate reinforcement learning for robot motion planning. *IEEE Transactions on Robotics*, 2024.

– The idea of exploiting the stochastic behavior of real-time simulators was conceptualized by the author, with substantial support from Mr. Ram Dershan in implementation and real-world experiments.

  - [10] Amir M. Soufi Enayati, Ram Dershan, Zengjie Zhang, Dean Richert, and Homayoun Najjaran. Facilitating sim-to-real by intrinsic stochasticity of real-time simulation in reinforcement learning for robot manipulation. *IEEE Transactions on Artificial Intelligence*, 5(4):1791–1804, 2023.

  This work contributed to Chapter 5.

– The CRAFT model emerged from a shared interest in the generalization bottleneck in RL and the meta-RL paradigm with Mr. Homayoun Honari. His discussions helped shape the notion of action-free task inference, and he significantly contributed to the theoretical development and coding. The material presented in Chapter 6 is currently being prepared as a manuscript:

  - Amir M. Soufi Enayati, Homayoun Honari, and Homayoun Najjaran. Action-Free Transformer Encoder-Decoder for Task Inference in Robotic Manipulation with Context-Adaptive Meta Reinforcement Learning. *In Preparation*, 2025.

Dr. Homayoun Najjaran has guided the research throughout all stages of development and publication, consistently supporting the author and providing constructive feedback on the research, experimental work, and publications. Partial equipment support was provided by Kinova®Inc., and the financial support was received from the Natural Sciences and Engineering Research Council (NSERC) Canada through the Alliance Grant CRDPJ 543881-19, Discovery Grant RGPIN-202305408, and RTI-2023-00418.

Chapter 1

Introduction

1.1 Problem Statement

Modern robotics is undergoing a transformative phase marked by rapid advancements in artificial intelligence (AI) and machine learning (ML), and especially the intelligent decision-making sub-field of them, reinforcement learning (RL). However, a significant bottleneck remains in the lack of adaptability in robotic systems when faced with variations in tasks or environment dynamics. Intelligent robot control agents are regularly trained in simulation for safety and financial concerns. These models trained in simulation often perform suboptimally or fail when deployed in real-world settings, where subtle discrepancies in dynamics, perception, or hardware characteristics can drastically affect performance.

The central issue is that these systems are generally rigid, i.e., they lack the mechanisms needed to infer understandings, adapt internally, and optimize behavior in new or changing circumstances without retraining. This limitation hinders the operational reliability of autonomous systems and limits the scalability of RL-based solutions in industrial, medical, and assistive robotics. Adaptive robotics, therefore, is a necessary paradigm. Adaptive robotics encompasses algorithmic, architectural, and operational innovations enabling robotic agents to generalize, transfer knowledge, and react robustly to uncertainties and new objectives in the real world. Amongst all the paradigms under control theory and machine learning, RL holds the highest promise as a mathematical and computational common ground for addressing the concerns mentioned.

1.2 Questions and Objectives

This dissertation explores and aims to propose solutions to three critical challenges in developing adaptive robotic systems using RL and meta-RL approaches. The primary research questions guiding this investigation are:

1. How can robots learn manipulation tasks better from limited supervised experience?
2. What mechanisms can enable robust sim-to-real transfer of learned policies, especially when dealing with unmodeled noise?
3. How can a robotic task identity be inferred effectively, particularly in an agent-agnostic manner with minimal dependence on action observations?

These questions frame the objectives of the work as follows:

- Develop a sample-efficient learning pipeline that reduces reliance on extensive training data by bootstrapping from limited expert guidance.
- Propose a novel approach to sim-to-real transfer that employs the inherent stochasticity in real-time simulators rather than suppressing it for increased robustness.
- Introduce a belief model capable of inferring latent task identity from action-free trajectories, eliminating dependence on action traces for flexible inter-task adaptation.

The dissertation investigates each of these objectives built upon a unified theoretical and empirical reinforcement learning framework, demonstrating their efficacy on both simulated benchmark tasks and physical robots.

1.3 Contributions and Scope

This work contributes to the field of reinforcement learning for robotics by introducing three novel components as the proposed solutions to the three questions in Section 1.2, called Demo-EASE (Chapter 4), RT-IS (Chapter 5), and CRAFT (Chapter 6), respectively. These three components are all concerned with training an RL-based policy for improved adaptability in robotic environments. This adaptability can be either

in the form of rapid knowledge transfer from an expert, sim-to-real domain shift, or facing a completely unseen objective in the same domain.

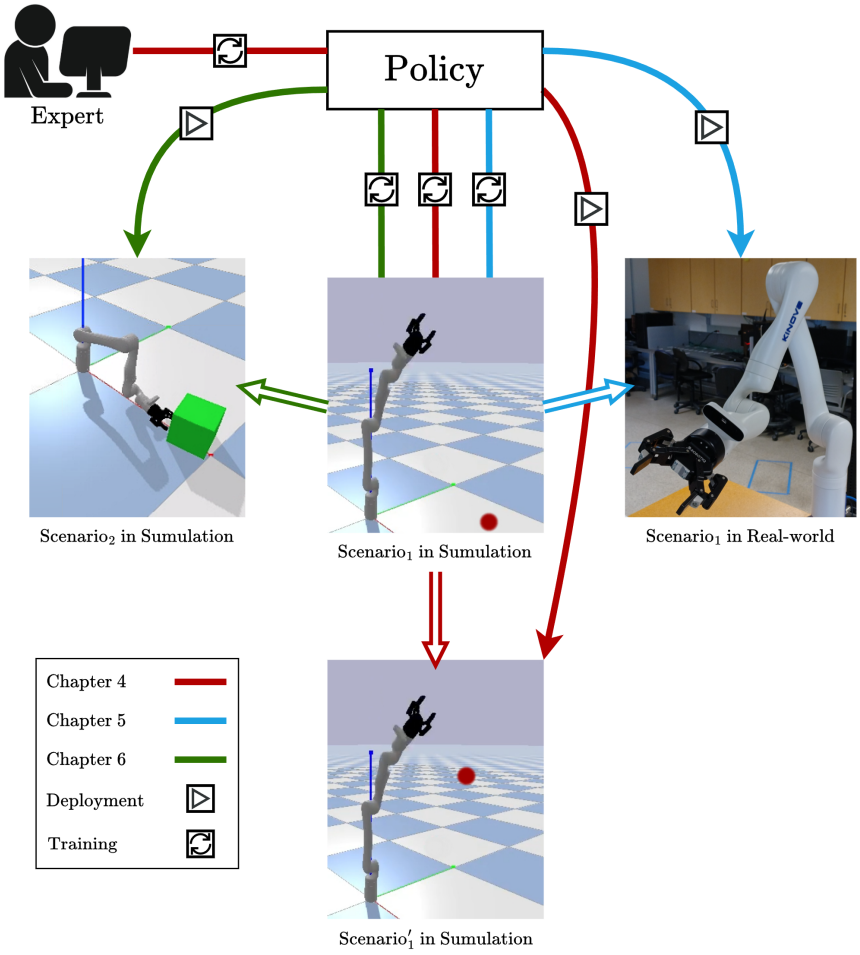


Figure 1.1: Visualization of the conceptual similarities and differences of the components introduced in different chapters of this dissertation: Demo-EASE (Chapter 4), RT-IS (Chapter 5), and CRAFT (Chapter 6)

Demo-EASE: A symmetry-aware learning algorithm that leverages few structured, though short, demonstrations to enhance sample efficiency in robotic manipulation tasks. Abstracting demonstrations based on environment symmetry allows replication and reuse across tasks with shared underlying structure.

RT-IS: A sim-to-real transfer technique that uses the intrinsic stochasticity natural to real-time simulation. This inherent variability in real-time simulation environments serves as a proxy for real-world noise. RT-IS reduces the need for

extensive domain randomization, a conventional approach to diversifying the data used in RL, and improves the robustness of trained policies.

CRAFT: A transformer-based encoder-decoder model is developed that infers task identity from contextual trajectories of states and rewards without accessing action information. Although this information loss might sound harmful, it enables effective task adaptation on the grounds of introducing less structure and motivating flexibility and exploration. This supports context inference in partially observable meta-RL settings and could present a future step towards cross-modal generalization to new observation spaces.

These contributions are validated through various simulation environments and real-world robot deployment of RT-IS using the Kinova® Gen3 robotic arm.

1.4 Thesis Outline

This dissertation is organized into seven chapters:

- **Chapter 2:** reviews the conceptual and theoretical foundations of adaptive robotics and introduces a taxonomy connecting structure, function, and adaptability in robotic systems.
- **Chapter 3:** presents a mathematical formulation of decision-making in adaptive robotics, outlining how uncertainty, variability, and task differences can be modeled within an adaptive MDP framework, the building block of reinforcement learning.
- **Chapter 4:** explains the Demo-EASE algorithm, focusing on how symmetry-guided mini demonstrations can be used to accelerate and generalize policy learning in large state spaces.
- **Chapter 5:** introduces the RT-IS concept and evaluates its effectiveness for sim-to-real transfer through both analytical models and real experiments.
- **Chapter 6:** presents CRAFT, the transformer-based action-free belief model for task inference in a meta-reinforcement learning robot manipulation setting.
- **Chapter 7:** finally summarizes the key findings, evaluates practical significance, discusses limitations, and suggests future research directions.

Chapter 2

An Insight Into Adaptive Robotics General Framework

The rise of automation and intelligentization in industrial manufacturing and social services, driven by robots and AI technology, has brought numerous benefits, including freeing humans from hazardous and monotonous tasks. However, the lack of flexibility and adaptability of robots and AI in broader manufacturing and social settings remains a limitation. To address this, the concept of adaptive robotics has been proposed, aiming to enable AI-powered robots to reprogram themselves rapidly in response to changes. Here, first, a systematic review of the field of adaptive robotics is presented, drawing on previous research, to establish objectives for a proposed framework.

2.1 The Foundations of Adaptive Robotics

Since the inception of artificial intelligence, there has been an expectation for machines to possess high levels of autonomy and learning ability. However, robotic manipulation, especially in scenarios involving new tasks [11], or transitioning from simulated to physical environments [12], remains a significant challenge [13]. This section aims to provide a taxonomy of the state-of-the-art in adaptive robotics, addressing the need for improved adaptability and flexibility in robots for complex tasks, to establish a unified framework for future research.

Adaptability originates from the innate drive of living beings to survive environmental changes. While animals and plants possess instinctive knowledge for imme-

diate survival, human infants require learning and training, usually more than other beings [14], before mastering basic skills such as walking. Improved learning ability is crucial for surviving harsh conditions. Learning can lead to mastering developed behaviors or adapting to new settings within an individual's lifetime. An analogy in robotic systems can be drawn as shown in Figure 2.1. Adaptability in nature is intertwined with learning as accidents, disabilities, and environmental changes necessitate revising ways of life. Adaptability is essential for survival in uncertain conditions, allowing individuals to adjust and thrive in new environments.

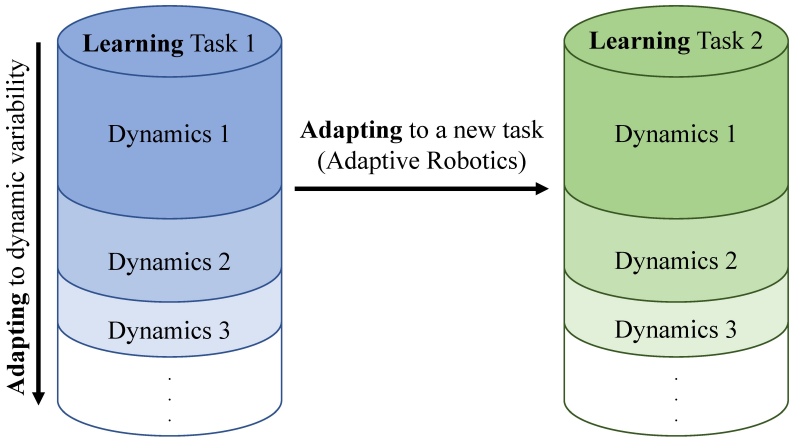


Figure 2.1: Interpretations of adaptability regarding learning

The development of numerically controlled mechatronic systems has been instrumental in industrial manufacturing, but the rise of Industry 4.0 demands more intelligent and adaptive systems [15]. This shift requires flexibility, human interaction, and multi-objective optimization, prompting the implementation of artificial intelligence. Future factories will prioritize flexibility and customization [16], with robots sharing intelligence through an internet-based framework for collaboration. The growing applications of robots in human-involved scenarios, such as robotic surgery [17], necessitate adaptability and guaranteed safety. To meet these evolving needs, mechatronic systems must become more intelligent, flexible, and capable of operating in a changing work environment.

2.2 Definition of Adaptive Robotics Concept

The integration of generalization and adaptability is crucial for artificially intelligent systems inspired by biological systems [14]. Industry 4.0 presents an ideal environment

for flexible robotics that can efficiently handle changing production requirements [18], regardless of variations in raw materials, hardware, schedules, or product assembly. The concept of Adaptive Robotics is a key focus of research, with the need to address multiple objectives such as agility, safety, precision, and speed in applications like human-robot interaction. Several metrics have been proposed for measuring the effectiveness of transfer, but none apply to inter-task transference and heterogeneous adaptation. A broader set of metrics specific to adaptive robotics is identified to distinguish various levels of adaptability and is presented in detail in the following. A graphic summary of these metrics, which are synthesized from a review of the literature surveyed in this chapter, is illustrated in Figure 2.2. It is noted that the definition of metrics is usually practice-oriented.

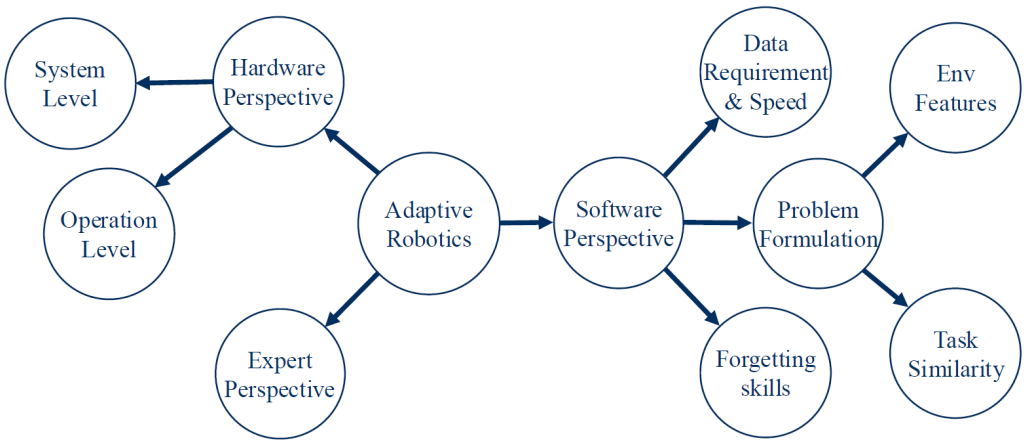


Figure 2.2: Interpretations of adaptability regarding learning

Hardware perspective

I. *System level: hardware and software*
In designing an adaptive system, the constraints imposed by generic hardware should be compensated by software, as the industry shifts towards relying more on control programs for adaptation while using multi-purpose hardware tools.

II. *Operation level: perception, planning, and control*
Autonomous systems, consisting of perception, planning, and control modules [19], face unpredictable situations that require adaptability across these three dimensions for real-world operations.

Software Perspective

I. *Adaptability data requirement and speed*
Naturally, an agent's exposure to different amounts and frequencies of anomalies or changes in a dynamic system returns inconsistent results in terms of self-adjustment. The requirements of abrupt permanent changes on a learning agent, such as hardware failure or the sim-to-real transfer effect, are significantly different from gradual wear and tear.

II. *Remembering or forgetting previous skills*
A robot must maintain proficiency in certain skills, especially when new tasks are variations or extensions of old ones. However, in some cases, permanent change or multi-tasking ability may not be relevant, and the decision to remember or forget old capabilities depends on the learning algorithm's depth and scope of adjustments.

III. *Formulation of the problem: structural similarity of tasks*
Mathematical models are commonly used to solve robotic task execution problems, and deep learning methods have incorporated adaptive traits to handle slight parameter changes. However, existing algorithms struggle to adapt to structural or fundamental changes in the environment [3], such as differences in manipulation tasks that involve varying influential entities, goals, and decision logic. As a result, the adaptability of robots is still underdeveloped for handling these immense task variations.

IV. *Formulation of the problem: environment features representation*
Differences between tasks are not always coarse and high scale as mentioned in the previous definition. Adapting to changes in the governing dynamics or the representation of a mathematical model, such as the *sim-to-real* transfer [20], requires intricate attention despite the big magnitude of the impact.

Human Expert Perspective

The adaptability of intelligent robot software, defined by the influence of a human expert, relies on identifying changes, motivating learning, controlling the learning sequence, and leveraging the oracle's knowledge for task mapping.

2.3 Related Approaches and Methodologies

Robotic systems face the challenge of self-adaptivity and decision-making in the presence of changes, which can range from intricate environmental behaviors to fundamental shifts in objectives. Adaptability can be achieved through structural design considerations or by employing evolutionary, knowledge transfer, or other methods. An overview of the related paradigms is shown in Figure 2.3, and the following explanation of each is laid out in the following sections. This figure highlights the concepts and paradigms that are repeatedly referenced throughout the rest of this dissertation.

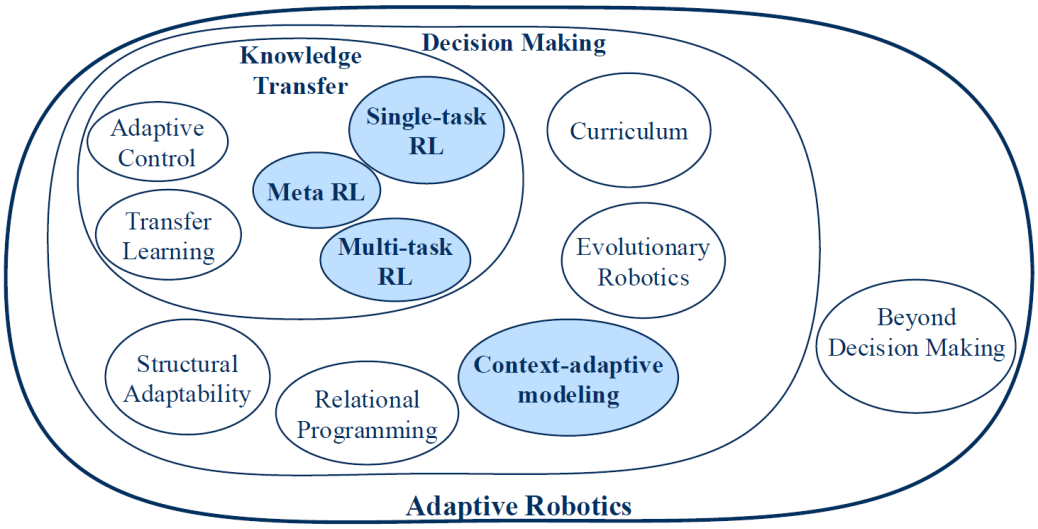


Figure 2.3: Illustrative taxonomy of related paradigms under the Adaptive Robotics concept

2.3.1 Structural Adaptability

Structural adaptability is defined on three levels: controller design, middleware architecture, and hardware. This adaptability is implemented with either adjustable architecture or modification of the morphology [18]. Fukuda and Shibata [21] proposes the coexistence of conventional control and AI to enhance the intelligence of controllers. In a practical approach, Edwards et al. [22] introduced the required middleware components in the architecture design such that the robot is able to self-adapt and self-manage. Although a valuable adaptive layered design guideline is suggested, their solution is limited to enumerating the required tools and software architecture for failure scenarios. An interesting recent work by Nygaard et al. [23] combines the

morphological hardware adaptability of a quadrupedal robot with online optimization of control strategy for minimum energy use in alternating terrains.

2.3.2 Evolutionary robotics

Almost all of the recent key breakthroughs in AI occur under the supervised learning umbrella, which has a weak natural explanation. It is believed that learning from huge datasets is not feasible for many species considering their urgent survival needs after birth [14]. This Darwinian principle of natural selection has been accepted and transformed into engineering design and formed the concept of the *Evolutionary Robotics* domain [24].

Some works modeled the evolution in the form of a tuning factor that determines the tendency of a robot to learn from its interactions with its changing environment [25]. Others changed the learning rules through generations with the help of evolution [26]. In the latter design, robots gradually become more adaptive to any general uncertain environments, rather than becoming sensitive to interactions with fast-changing environments as in the former.

However, evolution can hardly be implemented online [27] in industrial and social robotic applications. Essentially, evolution is effective during the design phase, either in the control system design or the structural design. It offers the advantage of an experimentalist and holistic approach toward robotic system design [28], but lacks the on-the-fly capability needed for working robots. An on-the-fly change refers to any change during deployment without reintegration downtime [29], and a working robot refers to a deployed system under production constraints as opposed to a lab prototype.

2.3.3 Knowledge Transfer

Various paradigms, here called *knowledge transfer* methods, have been investigated to develop adaptive robots by using feedback-based regression models and machine learning techniques. The following summary provides an overview of these fields with a particular emphasis on machine learning.

Adaptive control

Before the widespread application of machine learning in robotics, adaptive control played a central role in the online adaptation of manipulators in response to payload and changing actuator dynamics [30]. Adaptive control theory uses a regression model of the plant that can adjust governing dynamics by interpreting the latent perception or direct instrumentation feedback. Direct implementation of deep learning models in conventional control frameworks has been presented before. In Su et al. [31], a recurrent deep neural network is used for online estimation of surgical robots because of its flexibility and superior learning ability. Another successful example of simultaneous implementation of the conventional control scheme and machine learning in robotic manipulation is [32]. This work proposes an RNN-based constrained control command for redundant robotic manipulators with unknown structural parameters that can successfully adapt to the real values and perform inverse kinematic tasks. In [33], authors used a neural network as a noise estimator which improves the accuracy in simulation platforms. Yet, despite the large presence of learning-based algorithms, trajectory-based policies often generate waypoints and leave the command execution to conventional controllers [34].

Reinforcement learning

While deep reinforcement learning has been able to solve a number of complex problems in machine learning [35], it still fails to perform effectively, despite in a few well-constrained cases [36], when the problem domain or environment undergoes considerable changes or dwells in less-visited states during the training phase. Adaptive robotics fits this description perfectly. Hence, some extensions to baseline RL algorithms that might improve its capabilities are pointed out in the following.

Multi-task learning

Multi-task learning is one technique to improve this challenge in baseline machine learning algorithms. In multi-task learning, the agent faces a number of multiple scenarios during training and tries to leverage similarities. The goal is to create a way of learning that is more data-efficient than tackling several single-task problems. Per Kalashnikov et al. [11], mastery of different tasks with sharing exploration and experience among them simultaneously results in the robot's better performance while facing a new one. Instead of forcing data acquisition from multiple sources, in other

instances, *auxiliary tasks* [37] and continuous task change [38] are also examined. In the latter formulation, for better knowledge transfer during training, the authors suggest that the policy should be a function of both state and task to handle the complexity. From a different perspective, the agent can develop task-specific behavior after being exposed to multiple tasks. In any case, a destructive downside of multi-task learning is forgetting, which is a dead-end in sequential learning. To tackle this problem, Xiong et al. [39] proposed an algorithm with multiple local policies to guide the main policy searching. However, the tasks are only small variations of each other, and multi-task learning has yet to be put to harder tests.

Transfer learning

Knowledge transfer plays a key role in multi-task learning. Similarly, in transfer learning, the knowledge should be leveraged such that the agent learns the new task. Also, a major assumption in multi-task and transfer learning is that all the tasks or domains should share a similar data structure. The fundamental difference between the two is that in transfer learning, the highest priority is not maintaining an acceptable level of performance in the previous task(s) after adapting to the new one, as in multi-task learning. Nevertheless, in transfer learning, retaining acceptable performance on source tasks is often desirable and achieved through methods such as regularization, constrained optimization, etc., to mitigate forgetting.

Therefore, transfer learning has become a dominant paradigm when adaptiveness is needed to bridge the reality gap in *sim-to-real transfer*. In Golemo et al. [40], a recurrent neural network (RNN) is trained to learn the differences between virtual and physical manipulators. A mapping is then created to transfer a policy solely learned in simulation to the real robot. In many cases, rather than improving the model, policy randomization facilitates the transfer from simulation to real system [13].

Meta-learning

Meta-learning focuses on the learning mechanism itself. In many implementations, it involves the optimization of the learning parameters in a way that learning new tasks only requires a minimum number of examples. Also, similar to supervised learning, meta-learning process consists of two stages of meta-training and meta-testing, where during each the learning involves more than one task [41]. Essentially, meta-learning also shares an important characteristic with multi-task learning and transfer learning

during meta-training time: leveraging prior features acquired while facing a collection of previous tasks. However, two major differences can be pointed out: the emphasis on data efficiency and automated knowledge transfer. This approach is particularly well suited to online applications where a robot has to adapt to a new setting with a minimum number of trials.

Meta-learning has been proven to improve the adaptability of model-based reinforcement learning in mobile robots to terrain change, damage, and malfunctions [42, 43]. This impressive result was consistent using two different methods, and it can be considered an adaptation to dynamic variability. However, task adaptability was not addressed in these works. On the other hand, model-agnostic meta-learning (*MAML*) algorithm has been proposed for such an approach toward adaptive robotics [44]. Regardless of the model structure, this gradient-descent-based method updates the parameters of a model according to meta-training. The selection and interpretation of the said parameters are upon the user's decision. Certain extensions or variations have been proposed to handle noise using evolutionary search methods instead of gradient-descent [45], or to remedy overfitting [46]. In a broader sense, meta-RL faces some general constraints that have gained attention. Rakelly et al. [47] introduces an off-policy extension to meta-reinforcement learning that uses a probabilistic model to predict latent features for each new task. Gupta et al. [48] presents an automatic task generation to fulfill the task variability requirement. Notably, Yu et al. [3] develops a huge library of 50 robotic manipulation tasks as a benchmark for task adaptability. The lack of metrics and references for evaluating ideas based on each of the paradigms presented in this section is a serious setback that was also highlighted in Section 2.2. Lastly, a remarkable feature of meta-reinforcement learning is its flexibility of problem formulation such that multi-objective tasks can also be adapted to meta-learning terms and algorithms [49].

2.3.4 Curriculum

Curriculum learning is inspired by how humans learn or teach others to accomplish complex tasks. Reinforcement learning has demonstrated a lot of potential and significant achievements, and a curriculum is a viable option for extending its capability [50]. Hierarchical skill acquisition can be considered a special case of the curriculum. When a task is split into sub-tasks or skills, the agent can focus on learning them in a specific order [51, 52, 53]. This is as opposed to traditional reinforcement

learning methods, which normally label an entire scenario as a success or failure. Eppe et al. [54] considers a problem formulated as a multi-goal Markov decision process (MDP) instead of a multi-skill single-goal environment. They devised a masking vector system for evaluating complexity based on the combination of accomplishment criteria. This evaluation is then used in designing the curriculum for learning. Moreover, for single-goal problems, Curriculum Hindsight Experience Replay (CHER) is introduced such that the agent can interpret a general direction or judgment from its failures [55]. Automated generation of intermediate goals is also widely discussed as opposed to depending on user-defined functions or heuristics [56]. The objective of automatic goal generation is always to produce sufficiently hard and achievable new goals for the agent to maintain a learning improvement. Although most curriculum-based reinforcement learning methods are in simulated or video game environments, this approach might be of inspiration to adaptive robotics researchers. Specifically, curricula improve within-domain training; however, they might underperform under unmodeled dynamics or domain shifts as they do not explicitly incorporate distribution diversity.

2.3.5 Relational Programming

An approach to enhance a machine's intelligence in terms of adaptability is to structure the mathematical model to develop semantics or meanings in a similar way to humans. Relational Reinforcement Learning is inspired by inductive logic programming, which helps the agent understand relations between elements of an environment and then make a decision. Zambaldi et al. [57] implemented this functionality using specific structures such as graph neural networks or self-attention in a way that the agent becomes aware of spatial or logical relations among entities. They showed that when a robot can understand the connection between a lock and a corresponding key in a video game, scaling up the problem to a greater number of locks and keys is almost effortless. This idea has been implemented in applications other than robotics for expanding on the capabilities of model-based reinforcement learning [58, 59, 60], and also in robotic manipulation [61]. Li et al. [61] use graph-based relation for blocks in a robotic manipulation stacking task and shows that agents develop internal knowledge for modularizing the problem. This resulted in zero-shot generalization ability to build taller stacks.

2.3.6 Context-adaptive Models

Context denotes an extra parameter that cannot be directly identified or measured but will influence the model's behavior. A contextual MDP is used to describe a specific subset of MDPs where the transition dynamics, reward function, and observations distribution are affected by a hidden, but sometimes known, nuisance factor called *context*. For this type of problem, the objective is usually to find an optimal policy that maintains a certain level of performance for the whole range of context variables [62]. Human-centered applications are one of the most common instances of such problems, where the agent interacts with various users. As long as the number of context variables is small and the agent has any information, learning policies for a contextual MDP will be feasible. Therefore it is imperative to focus on learning the context variables or at least acknowledging the change in context from indirect information. Eghbal-zadeh et al. [63] used unsupervised learning for that purpose and proposed a protocol to examine the necessity of introducing context to regular MDP decision-making problems.

In conclusion, despite the mentioned accomplishments, a very big challenge hinders knowledge transfer methods: the tasks must share structural similarity [44]. Hence, for distinct robotic tasks, this similarity is meticulously hand-designed by the oracle such that the dynamic parameter spaces and objective functions are normalized and tuned to a single standard [3]. Unfortunately, deriving an explicit quantification of this feature is not easy, but Achille et al. [64] proposed a measure to estimate the complexity, however, this metric is hard to implement for reinforcement learning paradigms where the network parameters are less stable due to its stochastic nature and constant exploration.

2.3.7 Beyond Decision-Making

Robots are benefiting from adaptive learning not only for decision-making problems but also for a more general class of problems such as imitation and inverse learning. A robot can efficiently learn hierarchies and skills by learning the dynamic model of a system or imitating motions controlled by an expert [13]. Yet, the main focus of this research is on decision-making tasks. For completeness, other methods that tackle the variability of task execution in robotics are briefly mentioned here. The basic idea behind many works on policy search for robotic manipulation is using either knowledge priors or surrogate models for dynamics, structure, or the parameters of

the environment so that the robot controller matures with a handful of trials [65]. Accordingly, learning dynamical movement primitives [66], learning task priors [67], and transferring knowledge from the human demonstration in imitation learning [68, 69] are notable approaches.

This chapter lays the conceptual groundwork for adaptive robotics and motivates reinforcement learning as the common decision-making substrate. First, the scope of adaptive robotics is defined, then the chapter continues with a survey of key paradigms and concepts, organizing them into a taxonomy of how systems gain robustness and flexibility, schematically shown in Figure 2.3. The review highlights where existing approaches fall short in practice, for instance, by limited reuse of structured task regularities, sensitivity to deployment noise and latency, and agent-dependent context, setting up the need for the three contributions of this dissertation that follow later.

Chapter 3

Decision-Making Mathematical Formulation in Adaptive Robotics

The main goal of adaptive robotics is to give robot systems the capability to adapt to various tasks in a changing environment. As mentioned in the previous chapter, this dissertation is particularly focused on the decision-making tasks. Given the significant potential and bright future of adaptive robotics, one particular challenge is the ambiguous context of *adaptability* due to the lack of a uniform definition. Although the previous chapter has provided several specific perspectives on adaptive robotics, the connection between general concepts and practical solutions is still missing. It is still not clear how the technology of adaptive robotics is implemented to improve the performance of conventional robotic systems. One of the main reasons is that the application of robotics is concerned with a vast domain that can hardly be covered by a unified paradigm.

To overcome this challenge, this chapter proposes a mathematical formulation, mainly based on the definition of Markov Decision Processes and the reinforcement learning paradigm, for adaptive robotics. Firstly, the foundational mathematics of RL is laid out, with a focus on its flexibility in Robotic environment modeling. Then, a taxonomy of representative robotic tasks in the literature is conducted using a Markov Decision Process mathematical model to generalize the robotic tasks, where the achievement of the tasks is uniformly formulated as an optimization problem. Then, the *adaptability* of robotic systems is defined, associated with the specific task models, and the possible solutions to give the adaptive capability to robots in an uncertain environment based on adaptive MDP models are addressed. Finally, the

mathematical formulations of the related research topics introduced in the previous sections are summarized. It is noted that the presented interpretation here is far from comprehensive in terms of covering the entire domain of robotic engineering. However, it is hoped that this work will inspire further work on defining a broader scope of adaptive robotics in the future.

3.1 The Role of Reinforcement Learning

Reinforcement Learning (RL) is a computational framework in which an agent learns to make decisions by interacting with an environment to maximize cumulative rewards.

3.1.1 Fundamentals of RL

RL, at its core, is modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{P}_R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}_S(s'|s,a)$ the transition dynamics, $\mathcal{P}_R(s,a)$ the reward function, and γ the discount factor. The agent observes a state $s \in \mathcal{S}$, selects an action $a \in \mathcal{A}$ based on a policy $\pi(a|s)$, receives a scalar reward, and transitions to a new state. The objective is to find an optimal policy π^* that maximizes the expected return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{3.1}$$

3.1.2 RL with Function Approximation

In many real-world applications, the state or action spaces are large or continuous, making tabular methods infeasible. Function approximation techniques, such as deep neural networks, allow RL agents to estimate value functions or policies in high-dimensional spaces. These methods form the basis of Deep Reinforcement Learning (DRL), which has been instrumental in extending RL capabilities to complex robotic systems. The action-value function $Q^\pi(s,a)$ is often approximated by a neural network parameterized by θ:

$$Q^\pi(s,a;\theta) \approx \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right]. \tag{3.2}$$

While DRL has enabled breakthroughs in control and perception [70, 71], it remains data-intensive and sensitive to hyperparameter choices, necessitating the development of more adaptive solutions.

3.2 The Essential Mathematical Model for Robotic Decision-Making

The decision-making tasks for robotic systems are defined in various domains, ranging from physical motor control at the lowest level to task planning at a higher level. A taxonomy of these tasks in a structured paradigm helps provide specific solutions for these tasks in practice. This subsection categorizes the robotic decision-making tasks using an adopted hierarchical structure proposed for safe human-robot collaboration [1].

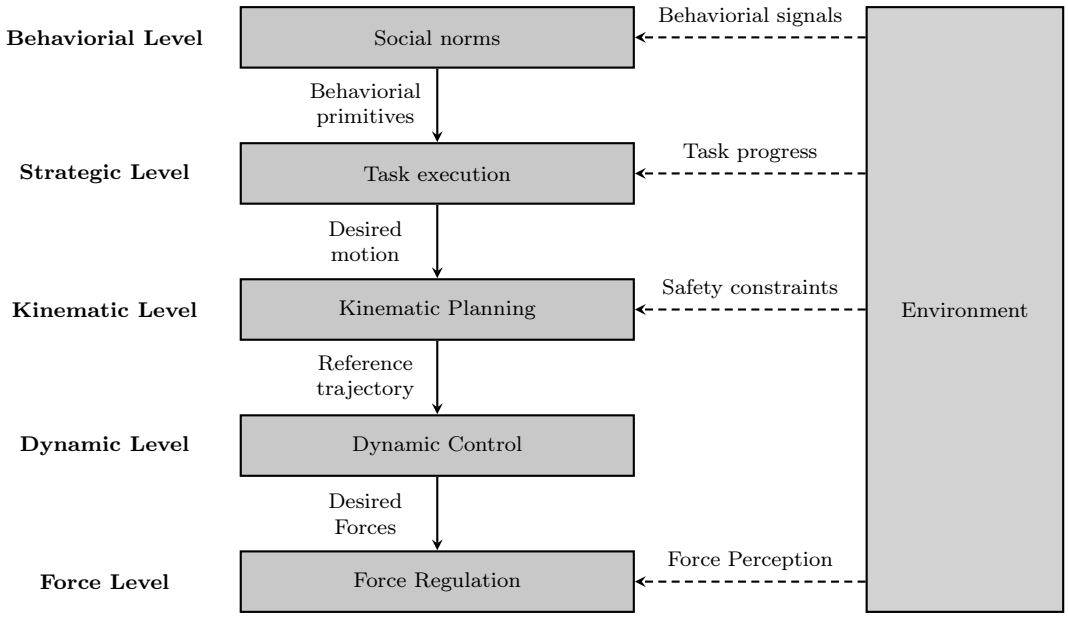


Figure 3.1: The proposed hierarchical paradigm adopted from [1].

3.2.1 Generic Robotic Decision-Making at Different Levels

Shown in Figure 3.1, the decision-making tasks are classified into five levels, namely the *Force Level*, the *Dynamic Level*, the *Kinematic Level*, the *Strategic Level*, and the *Behavioral Level*. From bottom to top, the robotic tasks vary from those with clear

physical definitions to those with abstract formulations. In the meantime, from top to bottom, the tasks tend to have a higher requirement on the measurement precision of the sensors and the execution bandwidth of the actuators. The aims of the tasks on different levels are briefly introduced as follows.

I. ***The Force Level.*** This level is concerned with the decision-making in the lowest level of a robotic system. A force-level task is dedicated to achieving a desired regulation of the joint forces given the desired forces from the upper levels and the force measurement from the environment. Typical force-level tasks include force regulation [72], force estimation [73], impedance control [74], admittance control [75], and contact transition control [76]. In this sense, decision-making means how much joint forces should be exerted to achieve the desired forces given the force measurement from the environment, as shown in Figure 3.1. It is worth mentioning that the performance of these tasks is usually very sensitive to the bandwidth and the stability of the closed-loop of the system. Therefore, the methods to solve the force-level tasks are usually based on precise force models of the robotic actuators. Also, high-precision and high-frequency force sensors are needed to provide reliable measurements.

II. ***The Dynamic Level.*** This level is defined to solve the dynamic motion control problems for robotic systems. The input of the dynamic control module is the reference trajectory generated by the kinematic level. In the meantime, it produces the desired forces to the actuators of the robotic systems, which is recognized as a decision-making process to precisely track the reference trajectory. The dynamic-level tasks are typically formulated as a trajectory tracking control problem [77] for which a stable tracking controller is designed by utilizing the dynamic model of the robotic system. It is noted that the precision of the robotic dynamic model is critical to guarantee the decent performance of the dynamic-level tasks. The deviation between the imprecise model and the true dynamics of the robotic system, usually referred to as *uncertainty* [78], is the main factor that affects the performance of the dynamic controller. Therefore, a robot motion controller is usually required to be *robust* to the modeling uncertainty attributed to the imperfect knowledge of the system [79].

III. ***The Kinematic Level.*** This level mainly focuses on generating feasible reference trajectories for the robot system subject to a certain set of kinematic constraints. The kinematic constraints are imposed due to the limitation of

the physical structure or the requirement of obstacle avoidance. These constraints are usually represented as equations, unilateral or bilateral inequalities of the system state that consists of the angular position and velocity of the robot. The kinematic-level tasks are formulated as motion planning problems where decision-making is reflected by generating feasible trajectories for the dynamic level. Due to the high bandwidth of the dynamic-level controller, the kinematic-level tasks are not keen on real-time implementation. Popular methods for kinematic-level motion planning include rapidly-exploring random tree (RRT) [80], artificial potential field [81], and stochastic policy gradient [82]. The results of these methods are usually in the form of complete trajectories which might become invalid when the environment changes due to the movement of the obstacles or the variation of the kinematic constraints. In this sense, the planned trajectories should be regenerated to adapt to the changes. Thus, the main challenge of the kinematic-level tasks is the tremendous computational load of the algorithms designated for modeling the changes in the variant environment.

IV. ***The Strategic Level.*** In this level, decision-making is concerned with the execution of strategic tasks, including grasping [83], pick-and-place [84], environment exploration [85], and assembly [86]. For these tasks, the robot is not concerned with its movement features. Instead, it focuses on the solution of certain properly defined temporal logic. Also, it formulates the tasks using event-triggering-based abstract representations. For example, probabilistic finite automata are frequently used to describe advanced robotic tasks such as cooking [87], dancing [88], and waste recycling [89]. The execution of the strategic-level tasks is usually required to be flexible such that it can automatically recover from an error state when the task process is disturbed or interrupted [90]. This indicates the ability of the robot to tackle unexpected changes in the environment. It is also challenging to extract the experience of the robot from one task and apply it to another task.

V. ***The Behavioral Level***, beyond all other four essential levels, focuses on defining behavioral or social regulations for the robots to make the robot more acceptable to humans [91]. In this sense, this level brings up particular interests for seamless human-robot interaction [92]. The tasks defined at this level are supposed to perceive the human behavioral signals, such as facial expressions [93],

emotions [94], gestures [95], or body movements [96], and generate proper behavioral primitives in the present context, such as distance [97], movement velocity [98], motion impedance [99], grip forces [100], or logic-based principles [101]. The tasks at this level are faced with a large extent of uncertainties since the behavioral signals may greatly vary among different subjects. Generating human-like behavioral primitives in various social contexts is still an open and challenging question.

3.2.2 MDP as the Essential Model for Robotic Decision-Making

The robotic decision-making tasks categorized above are formulated as problems with various domains and mathematical structures. Nevertheless, it is still possible to use a general model to provide a uniform representation for these tasks. This subsection provides a uniform formulation for these robotic tasks based on MDP. An MDP is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{P}_R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are respectively the state and action spaces of the robotic system, $\mathcal{P}_S$ denotes the state transition of the system, $\mathcal{P}_R$ is the reward or utility which is usually used to depict the degree of completion of the desired tasks, and γ is the discounting factor which balances between the rewards at the current moment and in the future. The MDP $\mathcal{M}$ can be deterministic or stochastic, depending on the type of uncertainty models incorporated in the formulation.

1) For a deterministic MDP, $\mathcal{P}_S$ is usually denoted by a joint mapping from the current state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ to the successive state $s' \in \mathcal{S}$, i.e., $s' = f(s, a)$, where $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Also, $\mathcal{P}_R$ is defined by a mapping $l : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that $r = l(s, a)$ indicates the instant reward obtained after executing the action a. The robot task is defined as solving the following optimization problem,

$$\max_{\pi} J(\pi) = \max_{\pi} \sum_{i=0}^{N} \gamma^i l(s_i, a_i), \tag{3.3}$$

where $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$ respectively represent the system state and action at time instant i, N denotes the horizon of an implemented path of the system which is also known as an *episode*, and $\pi : \mathcal{S} \to \mathcal{A}$, also known as *policy*, is a function that maps the current system state s to a desired action $a = \pi(s)$. The desired robotic task can be achieved under an optimal policy $\pi^* = \arg\max_{\pi} J(\pi)$ that maximizes the given reward. Note that the optimization problem defined in Equation (3.3) can be

maximizing an accumulated *reward* or minimizing the summary of instant costs. For a deterministic MDP, the uncertainty is usually reflected by the imperfect knowledge of the transition function $f(s,a)$ or the reward function $l(s,a)$. In most cases, the exact models of f and l are not available. Instead, they are usually only partially known. The uncertainties of the task and the environment are represented by the modeling deviations $\Delta f = f - \hat{f}$ and $\Delta l = l - \hat{l}$. Solving the problem defined in Equation (3.3) using the imperfect models may lead to a degradation of the actual performance of the system, i.e., a smaller actual reward than the optimal reward $J(\hat{\pi}^*) < J(\pi^*)$. It is often required to confine the policy such that the performance deviation is restricted in a tolerable range, i.e., with given $\delta \in \mathbb{R}^+$, $|J(\hat{\pi}^*) - J(\pi^*)| < \delta$. This is also referred to as the *robustness* of the system. However, when the modeling deviations Δf and Δl are overlarge, meaning that they are bigger than certain respective thresholds Δf_{thr} and Δl_{thr}, a robust policy solution is difficult to find since the problem becomes too conservative. In this sense, $\hat{f}$ and $\hat{l}$ will be recognized as a new environment and a new task. Adaptation means that the system should be able to automatically adjust the nominal policy $\hat{\pi}$ to the actual optimal policy π^* such that the performance deviation $|J(\hat{\pi}^*) - J(\pi^*)|$ remains in an acceptable range.

2) For a stochastic MDP, $\mathcal{P}_S$ and $\mathcal{P}_R$ are represented as conditional probabilistic distributions $p(s'|s,a)$ and $p(r|s,a)$ respectively defined on the state domain $\mathcal{S}$ and the reward domain $\mathcal{P}_R$. The successive state and the instant reward are recognized as the sampling from $s' \sim p(s'|s,a)$ and $r \sim p(r|s,a)$. An implementation of the sequential sampling of the MDP is referred to as a path $\tau = \{s_0, s_1, \cdots, s_N\}$, where N is the number of the steps in the path. Then, the robot task is defined as solving the following optimization problem

$$\max_{\pi} J(\pi) = \max_{\pi} \mathop{\mathbb{E}}_{\tau \sim p(\tau)} \left(\sum_{t=0}^{N} \gamma^t r_t \right), \tag{3.4}$$

where $p(\tau)$ is the distribution of the system path. Also, r_t is used to represent the instant reward to indicate that it changes over time. Note that the policy π is also in the form of a conditional probabilistic distribution $\pi(a|s)$ from which an action is sampled $a \sim \pi(a|s)$. In most cases, the policy is usually expressed in the form of a parameterized distribution function $\Pi_\theta(a|s)$, such as a neural network [102] or a Bayesian decoder [103]. Thus, the desired robotic task can be achieved by solving the optimal parameter θ^* such that the metric $J(\pi)$ in Equation (3.4) is optimized. For

a stochastic MDP, the uncertainty is usually reflected by the imperfect knowledge on the models for the transition $p(s'|s,a)$ and $p(r|s,a)$. In most cases, the knowledge of $p(s'|s,a)$ and $p(r|s,a)$ are usually partially or completely unknown. Solving the optimization problem using the imperfect model may also lead to degradation of the performance of the system. Nevertheless, for stochastic MDPs, there is need to introduce a measure $|p-\hat{p}|$ to represent the deviation between the two models p and $\hat{p}$. If the new environment or tasks $\hat{p}(s'|s,a)$ and $\hat{p}(r|s,a)$ are introduced, adaptation must be enabled to adjust the policy π, such that the performance deviation remains in the acceptable range. It is noticed that the stochastic MDP model with intrinsic uncertainty. The stochastic property is only used to depict the uncertainty reflected in the data. Supposing that $p(s'|s,a)$ is subject to Gaussian distribution of which the higher moments are zeros and the variance is close to zero. Under this condition, the corresponding stochastic MDP becomes very close to a deterministic MDP.

The MDP-based model can cover most of the decision-making tasks in the five-level structure in Figure 3.1. For the lower levels, namely the *force level* and the *dynamic level*, deterministic MDP with infinite continuous state and action spaces is mostly used with the form of differential or difference equations [104, 105]. At the *kinematic level*, both deterministic [106] and stochastic MDPs [107] are used for motion planning. The *strategic level* mainly uses stochastic MDP to formulate the event-based tasks [108]. Nevertheless, at the *behavioral level*, how to formulate behavioral-primitive generation as decision-making problems is still an open question, although many efforts are devoted based on user studies [109, 110]. As mentioned above, the conventional MDP models are not adaptive to environmental uncertainties. The next subsection addresses how to give the adaptive ability to MDP models to render adaptive robots in an uncertain environment.

3.3 Mathematical Formulation of Adaptive Robotics

Given the uniform formulation of decision-making problems, the robotic systems adaptability can be described from a generic mathematical perspective. This subsection depicts the formulation of robotic adaptability based on the deterministic and stochastic MDP models. An uncertainty model is used to measure the change in the environment or the deviation of the robotic tasks. Based on the uncertainty model, it is pointed out that adaptability can be refined as the robustness and the variability of the robot control policies.

3.3.1 The Uncertainty Model

In the formulation of MDP, the state transition $\mathcal{P}_S$ stands for the evolution of the states of the robotic system, which mainly reflects the intrinsic dynamic features of the robot, such as the inertia, the centrifugal forces, and the effects of the gravity and friction. These features are not associated with specific tasks but may vary due to change in the environment, such as temperature, lubrication, or mounting structures. Also, the identification process of the dynamic features is usually tedious and challenging since it has to ensure the sufficient exploration of the feature space of the robot systems. Even with the most careful identification, there still exists variation between the identified model and the true dynamic features of the robot system, which is referred to as *uncertainty*. As a result, the policy designed according to the identified model may lead to a suboptimal solution due to the existence of uncertainty. Meanwhile, the reward $\mathcal{P}_R$ measures the extent to which a robotic task is completed. The reward can be sparse, such as a binary digit that indicates whether a condition is satisfied [111], or dense, such as a continuous cost that is exerted on each instant during the task execution [107]. In most cases, different rewarding schemes $\mathcal{P}_R$ correspond to different robotic tasks. However, even for the same task, a slight change in the task configurations may lead to different reward distributions. In this sense, the policy designed for a nominal task may fail when the configuration changes.

Thus, any changes in the robot dynamics or task configurations may lead to the failure of a previously solved policy. This problem is especially critical for the models built based on the deterministic MDP. As discussed in Section 3.2.2, the uncertainty of deterministic MDP model is measured by the deviation between the nominal model $\hat{f}(s,a)$, $\hat{l}(s,a)$ and the actual dynamics $f(s,a)$, $l(s,a)$, i.e., $\Delta f = f(s,a) - \hat{f}(s,a)$ and $\Delta l = l(s,a) - \hat{l}(s,a)$ [112]. A policy $a = \pi(s)$ is referred to as *robust* if, for all dynamics $f = \hat{f} + \epsilon\Delta f$ and reward $l = \hat{l} + \epsilon\Delta l$, it ensures that the performance deviation $|J - \hat{J}| < \delta$ is within the predefined range, where $\delta \in \mathbb{R}^+$ is a predefined threshold. Thus, a robust policy should be able to guarantee the successful execution of a task under all possible uncertain conditions. Within the scope of this chapter, robustness is recognized as a conservative solution for adaptability since the robustness condition becomes very strict when the uncertainty is large. The stochastic MDP is intrinsically integrated with an uncertainty model which is represented in the form of probabilistic distributions, $p(s'|s,a)$ and $p(r|s,a)$, defined on the state and reward domains. However, when the deviation of the models is substantial such that

it is far from the nominal probabilistic distributions, the policy may not be able to ensure the successful execution of the robotic task.

When the uncertainty or the task deviation is so large that a robust policy is difficult to solve or even infeasible, a new policy for the changed task should be solved. In the conventional sense, this means that a completely different policy should be generated with the new models. At most, the old policy serves as an initial guess of the new policy to boost the solving process [113], during which very little knowledge from the old task is utilized. Thus, the essential motivation of adaptability is to provide a general solution to solve the new policy while fully exploiting the knowledge of the old task. In this sense, the adaptability of a robotic policy can be interpreted from two perspectives. First, in the situation where uncertainty is confined within a clear bound, adaptability means the robustness of the fixed policy to all possible variations of the system conditions. Second, in a situation where uncertainty is not confined but can be very large, adaptability means that the policy is able to change its structure such that performance degradation is mitigated.

3.3.2 The Formulation of Adaptive MDP

According to the connection between system uncertainty and adaptability, as discussed in the previous subsection, one important problem is to measure the extent of the uncertainty and evaluate whether regenerating a new policy is necessary. This naturally leads to the measure of the similarity between two MDPs. The similarity measure between two deterministic MDPs seems to be straightforward. Let us respectively represent the old and the new robotic tasks as two MDPs $\mathcal{M}_1 = \{\mathcal{S}_1, \mathcal{A}_1, \mathcal{P}_{S,1}, \mathcal{P}_{R,1}, \gamma_1\}$ and $\mathcal{M}_2 = \{\mathcal{S}_2, \mathcal{A}_2, \mathcal{P}_{S,2}, \mathcal{P}_{R,2}, \gamma_2\}$. Then, similar to the notation in Section 3.3.1, the function difference $\|\Delta f\| = \|f_1 - f_2\|$ and $\|\Delta l\| = \|l_1 - l_2\|$ can be used to depict the deviation between the two MDP models $\mathcal{M}_1$ and $\mathcal{M}_2$, where $\|\cdot\|$ stands for some measure in the functional space. For stochastic MDPs, however, there is not a mature definition of model deviation. Instead, previous work has attempted to propose quantitative metrics to evaluate the similarity between different tasks which is an important concept to avoid the negative transfer of knowledge. For example, in Carroll and Seppi [114], four different task similarity metrics are presented.

1. **Target advantage,** or the advantage that the new policy gained compared to the old policy when applied to the new task.

2. **Policy overlap,** or the number of the states that share the same values among the two tasks, which is recognized as an approximation of the target advantage.

3. **Task difference,** which is defined as the mean squared error between the values of the different tasks.

4. **Reward difference,** which is defined as the mean squared error between the instant rewards of the two tasks.

In Ammar et al. [115], an approximation model is used to represent the MDP. Then, the mean squared errors are calculated as a measure of task similarity. Also, a measure referred to as task compliance is proposed in [116] to assist the adaptation from the old policy to a new policy. For the purpose of generalization, the model deviation is represented as $\|\mathcal{M}_1 - \mathcal{M}_2\|$. Meanwhile, the optimal policies for the two MDPs are respectively $\pi_1 : \mathcal{S}_1 \rightarrow \mathcal{A}_1$ and $\pi_2 : \mathcal{S}_2 \rightarrow \mathcal{A}_2$. Then, the adaptability of a robotic system is reflected by the process of solving π_2 using the knowledge of π_1, such that the model deviation $\|\mathcal{M}_1 - \mathcal{M}_2\|$ is minimized. In the next subsection, this general formulation is further investigated in detail for the most relevant topics of adaptive robotics.

3.4 The Formulation of Related Topics

With the generic mathematical description of robotic adaptability, many of the existing robot-design paradigms, presented in Section 2.3, can be discussed in this common context. Therefore, this subsection presents the mathematical formulations of these paradigms using MDPs with the uncertainty models introduced in Section 3.3. Based on these formulations, it is possible to build up an essential framework for adaptive robotics with clearly defined research questions and methods.

3.4.1 RL Sim-to-Real by Domain Randomization

In RL, Sim-to-Real literally means applying a policy generated from simulation to a real robot platform. Without fully considering the variance between the two environments, a well-solved policy in simulation may lead to failure in reality due to the lack of robustness. Therefore, Sim-to-Real usually relaxes the optimality of the generated policy under the simulation conditions, but instead, requires the tolerance

of the uncertainties when transferring the policy from simulation to reality. As a result, the simulated policy can be explicitly applied to the real robot without changing its structure or parameters, although its performance is sacrificed. From this perspective, Sim-to-Real seeks a conservative but robust policy that survives in different environments. Robust decision-making or robust control, as a typical Sim-to-Real topic designated to deterministic MDPs, has been widely and deeply investigated by previous work [117]. However, robust Sim-to-Real transference is still an open question for stochastic MDPs. One prospective method for stochastic MDP is domain randomization which solves the policy in the simulation environment with a set of different configurations [118]. The stochastic MDP models in the simulation are represented using two parameterized distribution functions $p(s'|s,a,\theta)$ and $p(r|s,a,\mu)$. Then, domain randomization attempts to solve an optimal policy for various parameters $\theta \in \Theta$ and $\mu \in U$ in the feasible parameter spaces Θ and U. In this sense, the objective to be optimized becomes

$$J = \mathop{\mathbb{E}}_{\substack{\theta\sim p(\Theta)\\ \mu\sim p(U)}} \left[\mathop{\mathbb{E}}_{\tau\sim p(\tau|\pi,\mu)} \left(\sum_{t=0}^{T} \gamma^t r_t \right) \right], \tag{3.5}$$

where $p(\Theta)$ and $p(U)$ are the sampling distributions of the parameters θ and μ. Different from the conventional policy solving process described in Equation (3.4), Sim-to-Real seeks an expected optimal solution by sampling the parameters from heuristic distributions. The policy generated from such simulation conditions ensures smooth transference to reality as long as the distribution of the randomized parameters covers the maximum Sim-to-Real variation. A successful example of domain randomization can be referred to in [119], where a robust policy is solved for the pushing task on a 7-degree-of-freedom robot.

3.4.2 Modeling Adaptive Control

Domain randomization is usually only feasible if the variance between two environments is small. However, when the deviation is overlarge as described in Section 3.2.2, seeking a robust policy for both environments will be very challenging due to the conservativeness of the problem. In this situation, the policy of the old environment should be abandoned and another policy should be solved for the new environment. This process is usually tedious and inefficient. A more elegant manner is to actively

change the structure of the old policy to automatically adapt it to the new environment, such that task performance is continually ensured. Let us use $\mathcal{M}_1$ and $\mathcal{M}_2$ to denote the models of the old and the new environments, respectively, and use π_1 and π_2 to represent their corresponding optimal policies. Also, the old policy π_1 is assumed to be known and is represented in a parameterized form $a = \pi_1(s, \theta)$, where θ is the structural parameter of policy π_1 sampled in the parameter domain Θ. Then, the problem is formulated as seeking the searching direction for the new policy π_2, i.e., the parameter increment $\Delta\theta$, such that π_2 is solved within finite steps. Adaptive control, built upon ordinary differential equations, is a well-developed technology that ensures a feasible solution for this problem, where the policy parameter θ is automatically tuned using

$$\dot{\theta} = -\Gamma \nabla_\theta V(s), \tag{3.6}$$

a gradient searching law where Γ is a gain parameter denoting the step size of the searching process and $\nabla_\theta V(s)$ is the gradient of the value function $V(s)$. From the control theory perspective, $V(s)$ is usually selected as a state-dependent positive semi-definite function, also known as a *Lyapunov function*. If the state transition $f(s, a)$ has a linear-in-the-input form and the value function is convex for the state s, the gradient searching law in Equation (3.6) ensures the convergence of π_1 to the globally optimal policy π_2 with rigorously theoretical proofs [120]. Also, adaptive control ensures online implementation which significantly improves the computation efficiency of the policy solving process. A survey of adaptive control methods can be referred to in [121]. However, the convergence of the policy is only guaranteed when the *persistent excitation* condition is satisfied, which indicates sufficient exploration in the system state space. Additionally, policy adaptation for non-convex value functions or stochastic MDP models is still a great challenge.

3.4.3 Modeling Knowledge Transfer

Knowledge transference represents a large class of technologies that are dedicated to exploiting the knowledge in the old pattern to generate a model for the new pattern. Knowledge transference in the machine learning domain is usually referred to as *transfer learning*. A comprehensive review of transfer learning for supervised learning problems with a well-defined mathematical formulation can be referred to in [122]. Within the scope of this chapter, it is considered that transfer learning can also

be applied to strengthen the adaptability of robotic systems, although it has not attracted much attention. Specifically, in the same context as adaptive control in Section 3.4.2, transfer learning can be used to extract knowledge from the old policy π_1 for the old task $\mathcal{M}_1$, and use the knowledge to train the new policy π_2 for the new task $\mathcal{M}_2$. Let us define $\mathcal{S}_1 \times \mathcal{A}_1$ as the source domain, $\mathcal{S}_2 \times \mathcal{A}_2$ as the target domain, $\mathcal{T}_1$ as the source task, and $\mathcal{T}_2$ as the target task. Also, the value functions of the two MDP models $\mathcal{M}_1$ and $\mathcal{M}_2$ are defined as $Q_1 : \mathcal{S}_1 \times \mathcal{A}_1 \to Q_{\mathcal{T}_1}$ and $Q_2 : \mathcal{S}_2 \times \mathcal{A}_2 \to Q_{\mathcal{T}_2}$. Thus, the target of transfer learning is to explicitly solve Q_2 using the knowledge of Q_1. Note that transfer learning assumes that solving the value function Q_1 for the source domain $\mathcal{S}_1 \times \mathcal{A}_1$ and the source task R_1 is straightforward, but solving Q_2 is nontrivial. It is also assumed that the domains or tasks are not identical, i.e., $\mathcal{S}_1 \times \mathcal{A}_1 \neq \mathcal{S}_2 \times \mathcal{A}_2$ or $\mathcal{T}_1 \neq \mathcal{T}_2$. The critical technical point of transfer learning is to determine the domain mapping $\Psi : \mathcal{S}_2 \times \mathcal{A}_2 \to \mathcal{S}_1 \times \mathcal{A}_1$ and the task mapping $\Phi : \mathcal{T}_1 \to \mathcal{T}_2$. Then, the value Q_2 can be obtained through the combination of the complex mapping $\Phi \to Q_1 \to \Psi$. It is worth mentioning that most of the conventional work applies heuristic domain and task mappings which originate from the engineering experience [123]. Automatically solving these mappings is still an open question.

3.4.4 Multi-Target Optimization and Multi-Task Planning

Different from adaptive control and knowledge transference, multi-task learning considers solving a policy that achieves a compromised performance among several tasks. A variety of tasks are defined using a cluster of MDP models $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N$ which share the same state space $\mathcal{S}$, action space $\mathcal{A}$, and state transition $\mathcal{P}_S$, but different rewards $\mathcal{P}_R^i$ and discounting factors γ_i, $i = 1, 2, \cdots, N$, where N is the number of tasks. Then, a multi-task planning problem aims at solving the following multi-target optimization problem

$$J = \sum_{i}^{N} \omega_i \mathop{\mathbb{E}}_{\tau \sim p(\tau)} \left(\sum_{t}^{T} \gamma_i^t (r_t)_i \right), \tag{3.7}$$

where $(r_t)_i$ denotes the instant reward of MDP model $\mathcal{M}_i$, $i = 1, 2, \cdots, N$ and ω_i is a predefined weight of task i. Compared to the single-task problem defined in Equation (3.4), a multi-task problem is concerned with the weighted combination of the objectives of all tasks. Thus, the solved solution is not optimal for any individual task but should be considered as a compromise among these tasks. Multi-task plan-

ning is often applied to versatile robot systems that are required to execute several similar tasks using a common configuration [124]. This problem can be recognized as a special instance of domain randomization (Section 3.4.1) if the parameter spaces are finite sets and the randomization is conducted on a uniform distribution over the parameter spaces. Also, different from knowledge transference in Section 3.4.3, multi-task planning can be considered as an incomplete transference of knowledge from one task to the other tasks. Note that most of the conventional work on multi-task planning is discussed for MDP models with identical state and action spaces. Multi-task planning among heterogeneous robotic tasks with different state-action domains is still an open problem.

3.4.5 Meta-RL

Meta Learning for Decision-making

Meta-reinforcement learning (meta-RL) builds on the premise of learning to adapt: rather than solving each task independently, the agent learns a learning strategy across tasks. This is often framed as a two-loop process, with an outer loop for acquiring meta-knowledge and an inner loop for task-specific adaptation. Model-free methods like MAML [44] optimize the initial policy parameters θ such that one or few gradient steps improve performance on a new task:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta), \tag{3.8}$$

where $\mathcal{T}_i$ is a task sampled from the distribution of tasks. On the other hand, recurrent methods learn adaptation through memory.

Meta-RL with Task Representation

To enable fast generalization, meta-RL often incorporates task inference mechanisms that estimate latent variables representing the task. These representations can be learned via variational inference, memory networks, or sequence models. A common approach involves maximizing the evidence lower bound (ELBO) on task-conditional trajectories:

$$\log p(\tau) \geq \mathbb{E}_{q(z|\tau)} \left[\log p(\tau|z)\right] - D_{\mathrm{KL}}[q(z|\tau)||p(z)]. \tag{3.9}$$

By conditioning policies or value functions on these inferred variables, agents can tailor their behavior to the current task without explicit supervision.

Recent advances in task representation learning focus on context-based inference, where past trajectories are used to derive a task embedding. This embedding augments the observation space or modulates the policy directly. Techniques like context encoders, hypernetworks, and attention-based models are employed to construct robust task representations. A context-conditioned policy can be expressed as,

$$\pi(a_t|s_t, z_t); z_t = f(\text{context}). \tag{3.10}$$

These methods support rapid adaptation, sample efficiency, and generalization across a wide range of robotic manipulation and locomotion tasks.

3.4.6 Modeling Curriculum

Compared to the previous topics, the curriculum is a novel concept introduced in the most recent work of hierarchical reinforcement learning. Different from the multi-task planning problem (Section 3.4.4) which solves a common policy for several independent tasks, the curriculum is dedicated to strategically planning an appropriate sequence for a series of dependent tasks. Dependency means that some of the tasks should only be executed when other tasks are completely or partially executed, which increases the challenge of task planning problems. In [125], curriculum is formulated as a directed graph, where the dependency between the tasks is depicted by the edges in the graph. With a cluster of MDP models $\mathcal{M}_1$, $\mathcal{M}_2$, $\cdots$, $\mathcal{M}_N$ defined as in Section 3.4.4, $\mathcal{D}^{\mathcal{P}_S}$ can also be defined as the set of all possible transition samples from the transition $\mathcal{P}_S$. Then, curriculum is defined as a directed acyclic graph $C = (\mathcal{V}, \mathcal{E}, g, \mathcal{P}_S)$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E} = \{(x, y)|(x, y) \in \mathcal{V} \times \mathcal{V} \wedge x \neq y\}$ is the set of directed edges, and $g : \mathcal{V} \to \mathcal{P}(\mathcal{D})$ is a function that associates the vertices $\mathcal{V}$ to the subsets of the samples in $\mathcal{D}$, where $\mathcal{P}(\mathcal{D}^{\mathcal{P}_S})$ is the power set of $\mathcal{D}^{\mathcal{P}_S}$. A directed edge $\langle v_j, v_k \rangle$ in C indicates that the samples associated with $v_j \in \mathcal{V}$ should be trained before the samples associated with $v_k \in \mathcal{V}$ are trained. Also, all paths in the graph terminate on a single sink node $v_t \in \mathcal{V}$. Thus, the aim of a curriculum-based task is to generate a proper curriculum C for $\mathcal{M}_i$, $i = 1, 2, \cdots, N$, such that a certain utility $U(g)$ associated with the curriculum mapping g is optimized. Intuitively, the curriculum can be seen as *planning of the tasks*, instead of *planning for the tasks*. In this sense, the curriculum describes the adaptability of the robots at a higher strate-

gic level, compared to the problems in the previous subsections. Note that automatic planning of curriculum is still an unexplored domain, and most of the existing work on curriculum mainly relies on heuristics from experts.

This chapter formalizes robot decision-making and the sources of uncertainty that matter in deployment. The chapter starts with clarifying the role of RL in the decision-making formulation of robot manipulation, then it presents the MDP as the essential model and extends it to an adaptive formulation that captures task variability. This formalism is connected to all related mechanisms used throughout the rest of the dissertation: formalising robot manipulation in general, adopting domain randomization in the formulation, and the definition of meta-RL under diverse task distribution. The chapter serves as the mathematical bridge to Chapters 4 to 6, where all practical contributions to RL for improved adaptability follow the fundamental mathematical basis introduced in this chapter.

Chapter 4

Symmetry-Guided Demos for Sample-Efficient RL

Reinforcement learning (RL) can learn control policies in numerous domains but suffers from low sample efficiency in robotic manipulation. A major issue is the high-dimensional continuous state-space of such tasks. The performance of RL in learning a complex task can improve by leveraging prior knowledge, even if the knowledge is only suitable for abstract and simplified cases of the task. The natural symmetry of robotic environments is defined and used to create abstract environments for improved data aggregation in both on-policy and off-policy algorithms. A dual buffer design is proposed here, containing datasets of expert and original RL experiences, and regulating the RL update with behavior cloning. This architecture introduces a rich yet compact initiation and steady guidance. The method is verified in three simulation experiments with a Kinova® Gen3 robot. Experiments include off-policy learning of two point-to-point reaching tasks, with and without an obstacle, and on-policy learning for a pick-and-place task. A PID controller creates the expert demonstrations by tracking the linear joint-space trajectories with pre-calculated midpoints. The effect of different demonstration buffer sizes and weighted behavior cloning loss is analyzed. Results show a clear advantage in learning performance and potential value for faster implementations in real-world applications.

4.1 Sample Inefficiency of RL in Robotics

A major limitation of RL is low sampling efficiency [126, 127], which limits its real-world applications [5]. Different from the conventional models, which are fixed for specific tasks and scenes, such as control [79, 77] and fault diagnosis [128], an RL agent is designed to be automatically tuned using the interaction data. As a result, a large amount of data is often required for an RL policy to achieve theoretical optimality. However, only a small amount of this data is really useful for policy improvement. Thus, the training of an RL agent is expensive, especially for robot manipulation tasks, given the high-dimensional search spaces. In this sense, an end-to-end learning scheme is seldom practical for robotic engineering. To tackle this challenge, for example, compact state representation [129, 130] shrinks the search space, and hierarchical reinforcement learning [131, 132, 133] creates simpler sub-problems. Similarly, curricula for efficient sweeping over the state space [50], or heuristic dynamic modeling [134] have been proposed to tackle the mathematical complexity. In general, these schemes either reduce unnecessary exploration or remap the policy space to a simpler one.

Two other types of heuristics commonly used to improve the efficiency of RL are *symmetry* and *demonstrations*. *Symmetry* helps create a shared abstract model for different domains of the environment using a common feature set [135]. With a well-defined symmetry, the features, namely state, and actions for the case of RL, can be mapped among different symmetric domains without additional interaction. This minimizes the model complexity and decisively improves the sampling efficiency. *Demonstrations* are the reference trajectory clips of the system states performed by experts [136]. Unlike in programming by demonstration (PbD) [137], imitating sub-optimal human motion is not the ultimate goal for pure robotic manipulation. Fanatical imitation may lead to over-fitting and a lack of generalizability. In [6] and [7], the general idea of Exploitation of Abstract Symmetry of Environments (EASE) is proposed to utilize the symmetry of a complex environment to allow for sample efficient RL policy learning. EASE uses RL training for both abstract and global environments; thus, it suffers from RL sample inefficiency but to a lesser degree.

This chapter proposes a novel data aggregation method based on the inherent symmetry of the robotic systems for on-policy and off-policy RL. The concept of **EASE** is extended with heuristic **Demo**nstrations (Demo-EASE) by replacing the local RL training with a control-based expert. Although EASE was only proposed for an off-

policy RL algorithm, the current extension is adapted to on-policy RL methods as well. The presented work further studies Demo-EASE from [6] and presents a rigorous framework to allow for repeatability and generalizability by introducing different sub-processes and associated symmetry definitions, specific to robot manipulation tasks. Deep Deterministic Policy Gradient (DDPG) [138], as the base off-policy method and Proximal Policy Optimization (PPO) [139] as the base on-policy method are chosen. A PID controller is used to generate samples of robot trajectory, although it can be substituted with other planning methods or human motion. The demonstrations are duplicated by the symmetrical mapping among local environments. Demonstrations are stored in a demo buffer, which, together with the experience buffer, forms a dual-memory learning scheme. A behavior cloning (BC) loss function is added to update the actor-critic agent with the dual-memory structure [140]. The rest of the chapter is organized as follows. Section 4.2 formulates the problem to be investigated in this chapter. The technical details of the proposed method are presented in Section 4.3. The implementation results are discussed in Section 4.5. Finally, Section 4.6 concludes the chapter. Additional information about the algorithm and experiment setup can be found in Section 4.3.5 and Section 4.4.

Notation and units: $\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}^n$, $\mathbb{N}^+$ are used to represent the sets of real numbers, positive real numbers, n-dimensional real vectors, and positive integers, respectively. $|\cdot|$ is the absolute value of a scalar and $\|\cdot\|$ stands for the 2-norm of any real vector. All angles are in radians, and all lengths are in meters if otherwise specified.

4.2 Problem Formulation

Consider a general manipulation task as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{P}_R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces of the robotic system, $\mathcal{P}_S : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the state transition model, $\mathcal{P}_R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the task-specific reward function, and $\gamma \in (0, 1]$ is the discounting factor. Note that the above definition uses a *deterministic* MDP. The definition of *stochastic* MDP can be referred to in [141]. The target of the MDP problem is to solve the optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ such that the following accumulated reward is maximized

$$J(\pi) = \sum_{t=0}^{N_\tau} \gamma^t R(s_t, a_t)\,, \tag{4.1}$$

where N_τ is the length of the robot trajectory, and $s_i \in \mathcal{S}$ and $a_i = \pi(s_t) \in \mathcal{A}$ are respectively the observed state and action. A model designed to solve the optimal policy $\pi^* = \arg\max_\pi J(\pi)$ of the MDP $\mathcal{M}$ is also referred to as an *agent*. The optimal policy π^* can be solved by either off-policy methods or on-policy methods [142].

Three essential robot tasks are studied in this chapter as representatives of general robot manipulation tasks, namely a point-to-point reaching (P2P) task, a P2P task with obstacle avoidance (P2P-O), and a pick-and-place (P&P) scenario. These three tasks are the most essential primitives where workspace symmetries are well-defined, and they can be combined to form more complicated manipulation tasks. For application, a collaborative arm reflects the target use-case, for instance, assembly, and isolates symmetry in a bounded static environment; however, the framework itself is generally agnostic to embodiment. For an n-degree-of-freedom (n-DOF) robot manipulator, the action in all three tasks can be defined as the vector of reference joint velocities, i.e., $a_t = \dot{q}_t^{\mathrm{r}} \in \mathbb{R}^n$, where $q_t^{\mathrm{r}} \in \mathbb{R}^n$ is the joint angles of the robot with the subscript t indicating the time and the superscript r referring *reference trajectory*. The state of the P2P task is defined as

$$s_t = [\, q_t,\, \sin(q_t),\, \cos(q_t),\, \dot{q}_t,\, P_g,\, e_t \,], \tag{4.2}$$

where $q_t, \dot{q}_t \in \mathbb{R}^n$ are respectively the actual joint angles and joint velocities at time t, $P_g \in \mathbb{R}^3$ is the Cartesian position of the goal, and $e_t = P_g - P_t^e$ is the vector distance between the goal position P_g and the robot end-effector position $P_t^e \in \mathbb{R}^3$. The state vector of the P2P-O and P&P tasks are defined similarly to Equation (4.2) with an addition of the object position $P_o \in \mathbb{R}^3$. Here, the object is assumed to be a certain cube, and P_o is its geometrical center.

4.3 Demo-EASE

The overall structure of the proposed framework is illustrated in Figure 4.1 and Figure 4.2. Both of these agents seamlessly adapt to the continuous state and action spaces. The off-policy agent in here, different from the previous work, which uses a locally trained DDPG agent to generate the demonstration, focuses on the knowledge transfer from a developed but sub-optimal controller to incorporate the capability of leveraging the existing controllers. Also, the on-policy training process is a direct extension. Although the proposed frameworks are based on only two existing methods,

this method can be extended to other RL methods equipped with online or offline experience replay buffers.

Demo-EASE has a demonstration storing procedure in addition to the RL training process. In the demonstration storing procedure, two experience buffers, an original replay buffer, and a demo buffer, are used to record the interaction data. The original replay buffer collects the agent's interaction data with the global environment, and the demo buffer stores the demonstration samples recorded in the local symmetric environments and their duplicates. The concepts of global and local environments are defined in Section 4.3.1, and the recording of the demonstrations is explained in Section 4.3.2. The masked behavior cloning integration to the optimization problem is explained in Section 4.3.4, and the training algorithms are provided in Section 4.3.5.

4.3.1 Symmetric Local Environment

One technical point of this chapter is the exploitation of environment symmetry to duplicate the demonstration data and improve the training efficiency. As symmetry is conventionally task-specific and noted as expert-defined in [6], the following general definition for symmetry in robot manipulation tasks is proposed.

Definition 1. *For the MDP $\mathcal{M}$, $\mathcal{S}_0 \subset \mathcal{S}$ and $\mathcal{S}_1 \subset \mathcal{S}$ are symmetric, or $\mathcal{S}_0 \rightleftharpoons \mathcal{S}_1$, if there exist reversible mappings $\phi : \mathcal{S}_1 \to \mathcal{S}_0$ and $\psi : \mathcal{A} \to \mathcal{A}$, such that $\phi(\mathcal{P}_S(s,a)) = \mathcal{P}_S(\phi(s), \psi(a))$ holds for any $s \in \mathcal{S}_1$ and $a \in \mathcal{A}$ that satisfy $\mathcal{P}_S(s,a) \in \mathcal{S}_1$ and $\mathcal{P}_S(\phi(s), \psi(a)) \in \mathcal{S}_0$.*

Definition 1 gives a mathematical formulation of symmetry for deterministic MDP. This allows for accurate prediction on a domain given the historical observation on its symmetric counterpart even though the real observation is not available.

Note that the symmetry relation is reversible, i.e.,

$$\mathcal{S}_0 \rightleftharpoons \mathcal{S}_1 \Rightarrow \mathcal{S}_1 \rightleftharpoons \mathcal{S}_0, \tag{4.3}$$

since $\phi(\mathcal{P}_S(s,a)) = \mathcal{P}_S(\phi(s), \psi(a))$ leads to $(\phi^{-1}[\phi(s)], \psi^{-1}[\psi(a)]) = \phi^{-1}[\mathcal{P}_S(s,a)]$. Building on the definition of symmetry, the symmetrically partitionable environment can also be defined.

Definition 2. *For the MDP $\mathcal{M}$, the domain $\mathcal{S}$ is symmetrically partitionable, if $\exists\, \mathcal{S}_0$, $\mathcal{S}_1, \cdots, \mathcal{S}_{M-1} \subset \mathcal{S}$, where M is a positive integer, such that $\mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_{M-1} = \mathcal{S}$, and $S_i \rightleftharpoons \mathcal{S}_j$ and $S_i \cap \mathcal{S}_j = \varnothing$ for all $i, j = 0, 1, \cdots, M-1$ and $i \neq j$.*

Definition 2 introduces a special type of MDP, of which the state space can be split into several mutually exclusive and symmetric partitions. In these MDPs, for any sample $(s_0, a_0) \in \mathcal{S} \times \mathcal{A}$ and its observation $s'_0 \in \mathcal{S}$, another $M-1$ counterparts (s_1, a_1, s'_1), $\cdots$, $(s_{M-1}, a_{M-1}, s'_{M-1})$ can be accurately predicted. In this chapter, the *global environment* is considered a symmetrically partitionable space $\mathcal{S}$, and the *local environment* is any of its symmetric partitions $\mathcal{S}_k$, $k = 0, 1, \cdots, M-1$.

4.3.2 Definition of Demonstration

A demonstration episode is a trajectory of the MDP $\mathcal{M}$

$$\tau(\pi^{\mathrm{d}}) = \{s_0, a_0, s_1, a_1, \cdots, s_{N_\tau - 1}, a_{N_\tau - 1}, s_{N_\tau}\}, N_\tau \in \mathbb{N}^+,$$

where π^{d} is the policy used to generate the demonstration. The demonstration policy can be generated from previous data [140], or explicit teaching [143], or an imperfect controller [144]. Demonstration serves as prior knowledge to guide the agent through the initial learning stage. However, demonstration can cause a local-optima problem in the later stages of agent training [145]. Therefore, the agent should use the demonstration knowledge in the early learning stage and start to balance between the demonstration and its own experience when it becomes mature.

The expert agent trajectories are recorded subject to a PID controller. As a conventional and widely applied control method, PID benefits from simplicity and robustness. Therefore, it can be used as a conservative solution for general robot control and planning, although it is hard to be customized for certain tasks. For any time instant $t = 0, 1, \cdots, N_\tau$ and state s_t, a PID action $a_t^{\mathrm{PID}} \sim \pi^{\mathrm{d}}(s_t)$ is defined as

$$a_t^{\mathrm{PID}} = K_{\mathrm{P}} \tilde{q}_t + K_{\mathrm{I}} \int_0^t \tilde{q}_\tau \mathrm{d}\tau + K_{\mathrm{D}} \dot{\tilde{q}}_t, \tag{4.4}$$

where $\tilde{q}_t = q_t^{\mathrm{sp}} - q_t$, $q_t^{\mathrm{st}} \in \mathbb{R}^n$ is the desired joint angles obtained by calculating the inverse kinematics of the Cartesian set points, and q_t is the current joint angles of the robot. For the P2P and P&P tasks, q_t^{sp} is the inverse kinematics of the goal position P_g. For the P2P-O task, heuristic midway points are added to q_t^{sp} such that the robot moves away from the obstacle. K_{P}, K_{I}, and K_{D} are respectively the proportional gain, the integral gain, and the derivative gain of the PID controller. With the state s_t and action a_t^{PID}, the successive state s'_t is observed and the instant reward $r_t = R(s_t, a_t^{\mathrm{PID}})$ is calculated. Note that the whole demonstration is recorded

in one local environment $\mathcal{S}_k$, $k = 0, 1, \cdots, M-1$. Each demonstration episode is terminated by a binary ending flag $\zeta_t \in \{0, 1\}$ which is enabled by success, timeout, collision, or exiting the local environment. $\mathcal{D}_t^k = \left\{s_t, a_t^{\text{PID}}, s_t', r_t, \zeta_t\right\}$ represents the demonstration sample at time t, and k stands for the local environment from which the sample is obtained. Additional samples $\phi\left(\mathcal{D}_t^k\right) = \{\phi(s_t), a^{\text{PID}}, \phi(s_t'), r_t, \zeta_t\}$ can be duplicated on all the symmetrical local environments of $\mathcal{S}_k$.

4.3.3 Storing Demonstration Samples

As shown in Figure 4.1 and Figure 4.2, Demo-EASE is equipped with the original online or offline replay buffer $\mathcal{B}_{\text{o}}$ of RL baseline and a demonstration replay buffer $\mathcal{B}_{\text{d}}$. $\mathcal{B}_{\text{d}}$ only contains the demonstration samples $\mathcal{D}_t^k$ and their duplicates $\phi\left(\mathcal{D}_t^k\right)$ in all local environments, while $\mathcal{B}_{\text{o}}$ contains both the demonstration and training transitions. In a symmetrically partitionable space, the demonstration policy is executed uniformly in all local environments. Apart from a small noise, the initial position q_0 is fixed for each local environment, and the goal and object positions P_g and P_o are randomly sampled from uniform distributions.

In the off-policy framework, all demonstrations and their duplicates are stored before training. In the on-policy framework, a similar offline demo buffer is implemented. Also, demonstrations are copied in the online buffer in early iterations such that the policy update is not solely guided by behavior cloning. Hence, for the first or a considerably small number of experiences of the early epochs, i.e., until filling the demo buffer to its nominal capacity, demonstration transitions are stored in the online buffer as well.

4.3.4 Masked Behavior Cloning Loss

Not all the samples in the replay buffers $\mathcal{B}_{\text{o}}$ and $\mathcal{B}_{\text{d}}$ are used to update the actor and critic networks of the RL agents. Instead, two smaller batches $\overline{\mathcal{B}}_{\text{o}}$ and $\overline{\mathcal{B}}_{\text{d}}$ are sampled, which contain the shuffled and randomly chosen transitions of $\mathcal{B}_{\text{o}}$ and $\mathcal{B}_{\text{d}}$, respectively. The sizes of $\overline{\mathcal{B}}_{\text{o}}$ and $\overline{\mathcal{B}}_{\text{d}}$ are respectively $\overline{N}_{\text{o}}$ and $\overline{N}_{\text{d}}$ which satisfy $\overline{N}_{\text{d}} < N_{\text{d}}$ and $\overline{N}_{\text{o}} < N_{\text{o}}$. The conventional critic loss function is minimized to update the critic networks in both off-policy and on-policy versions of Demo-EASE. The actor networks, however, are updated using both batches $\overline{\mathcal{B}}_{\text{o}}$ and $\overline{\mathcal{B}}_{\text{d}}$ with a combined loss function

$$\mathcal{L}_A\left(\overline{\mathcal{B}}_{\text{o}}, \overline{\mathcal{B}}_{\text{d}}\right) = \mathcal{L}_O\left(\overline{\mathcal{B}}_{\text{o}}\right) + \lambda_{BC}\mathcal{L}_{BC}\left(\overline{\mathcal{B}}_{\text{d}}\right), \tag{4.5}$$

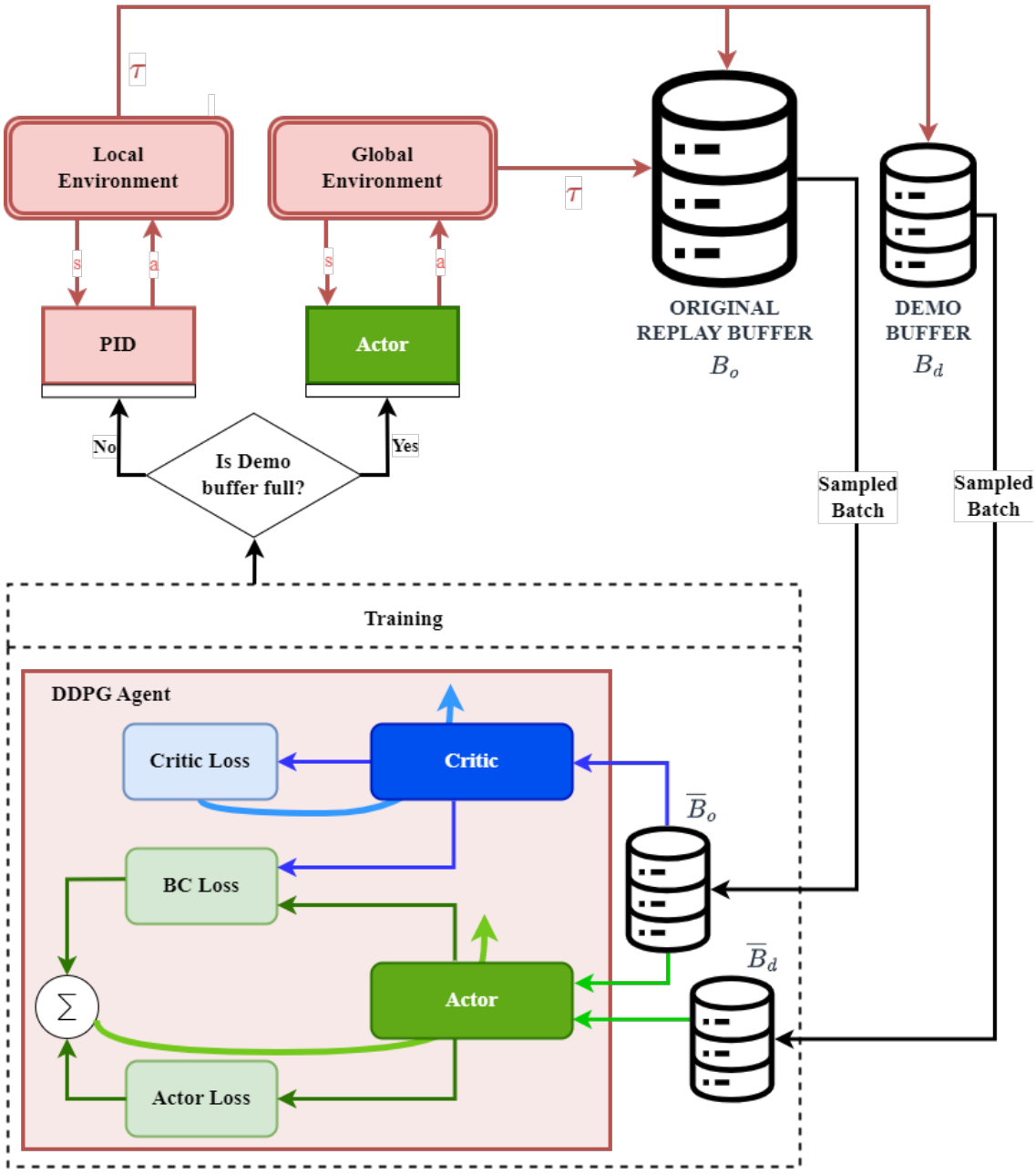


Figure 4.1: The flowchart of off-policy Demo-EASE based on DDPG

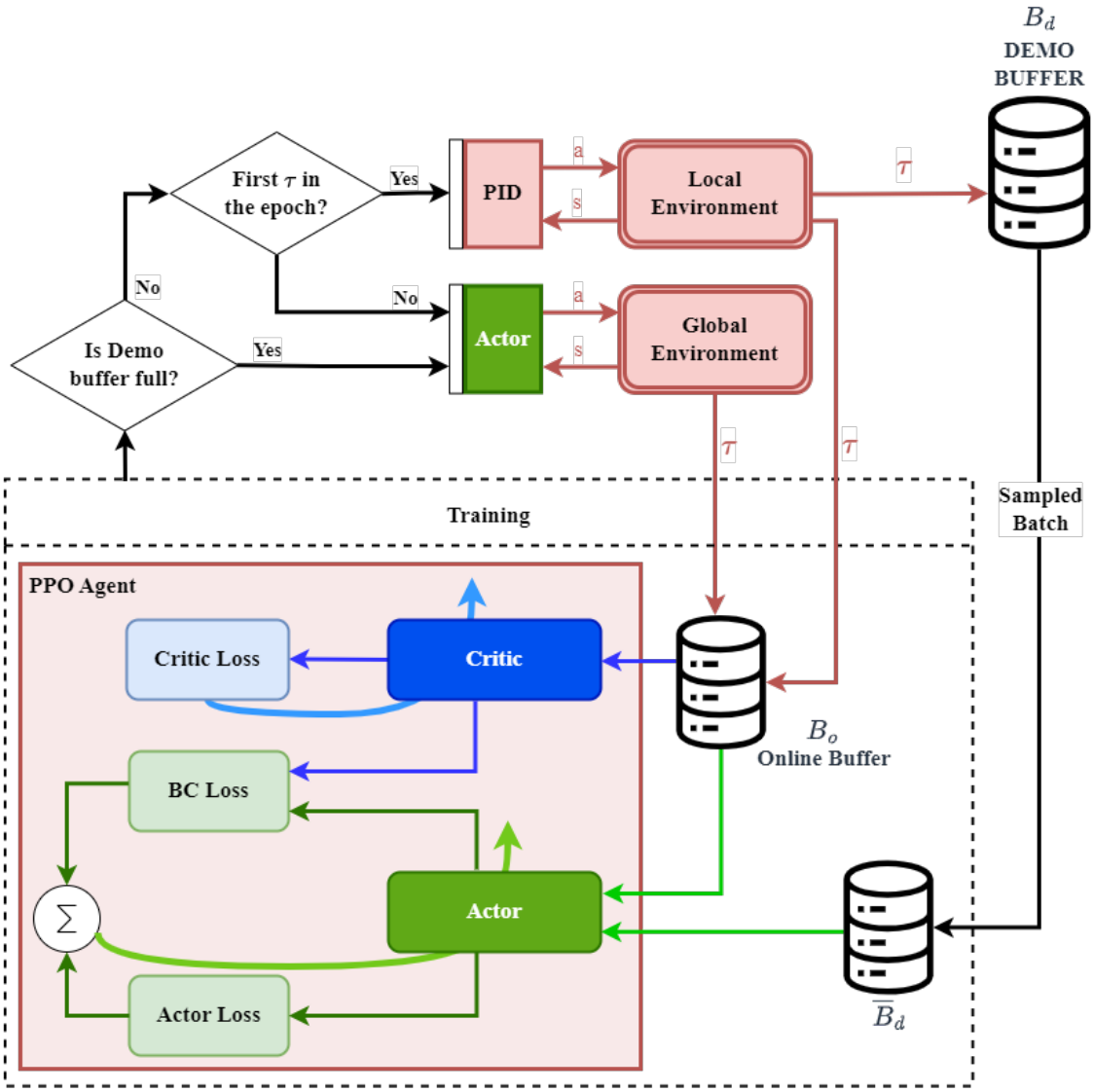


Figure 4.2: The flowchart of on-policy Demo-EASE based on PPO

where $\mathcal{L}_O$ is the original actor loss function either based on DDPG or PPO. $\mathcal{L}_{BC}$ is the behavior cloning loss function used to encourage the compatibility between the policies and λ_{BC} is a hyperparameter to adjust the regulation. The behavior cloning loss function $\mathcal{L}_{BC}$, inspired by the behavior cloning in imitation learning [140], is defined as

$$\mathcal{L}_{BC}\left(\overline{\mathcal{B}}_{\mathrm{d}}\right)=\frac{1}{\overline{N}_{\mathrm{d}}}\sum_{t=0}^{\overline{N}_{\mathrm{d}}-1}M_t\left(a_t^{\mathrm{d}}-\hat{a}_t\right)^2, \tag{4.6}$$

where $\hat{a}_t = \pi\big(s_t^{\mathrm{d}}\big)$ is the agent's action subject to the current policy π and the demonstration state s_t^{d}, $\big(s_t^{\mathrm{d}}, a_t^{\mathrm{d}}\big) \sim \overline{\mathcal{B}}_{\mathrm{d}}$. Note that in the on-policy method, a variational actor produces a mean μ and variance σ for actions, so $\hat{a}_t = \mu\big(s_t^{\mathrm{d}}\big)$. M_t is a binary mask that indicates whether the demonstration action a_t^{d} is superior to the current policy action a_t. The superiority in the case of the DDPG-based method is defined as a higher state-action pair value for a_t^{d}, i.e.,

$$Q(s_t^{\mathrm{d}}, \hat{a}_t) < Q(s_t^{\mathrm{d}}, a_t^{\mathrm{d}}).$$

In the on-policy version, this superiority is defined as when the immediate reward received by the action in the demonstration a_t^{d} is higher than the estimated state value, i.e.,

$$V(s_t^{\mathrm{d}}) < R(s_t^{\mathrm{d}}, a_t^{\mathrm{d}}).$$

The purpose of $\mathcal{L}_{BC}$ is to balance between the demonstration and the original experience of the agent. In the initial stages of the training, when the policy is not yet trained, π imitates the demonstration policy π^{d}. When the agent is well-trained such that its actor performs better than the demonstration, $\mathcal{L}_{BC}$ vanishes and the demonstration policy π^{d} is ignored. In general, when no sufficiently good demo subset is found, e.g., due to any shortcomings in the demo policy, such as avoiding dynamic obstacles, Demo-EASE defaults to the vanilla RL algorithm it is built on. In such rare cases, the performance gracefully degrades to the base learner rather than failing catastrophically. The behavior cloning gain hyperparameter, λ_{BC}, adjusts the extent of cloning. In other words, λ_{BC} balances between the training speed and the generalizability of the agent.

4.3.5 Training Algorithms

The algorithmic processes of off-policy and on-policy Demo-EASE are presented in Algorithm 1 and Algorithm 2, respectively. In Algorithm 1, $N_{\mathrm{eq}} \in \mathbb{N}^+$ is the number of episodes, $N_{\mathrm{up}} \in \mathbb{N}^+$ is the number of epochs after which the networks are updated, $N_\tau \in \mathbb{N}^+$ is the maximum length of each episode, $\sigma \in \mathbb{R}^+$ is the standard deviation of the action exploration, and $\omega \in \mathbb{R}^+$ is the coefficient of target network updates.

4.4 Experiment Setup

To validate the proposed method, a simulation experiment is set up in PyBullet using a 6-DOF Kinova® Gen3 robot model with a Robotiq® 2F-85 gripper as shown in Figure 4.3. Since the experiments do not involve complex dexterity, only four actuators (# 1, # 2, # 3, and # 4) are active and the other two are locked, as shown in Figure 4.3, which results in a 4-DOF active robot arm. The experimental simulation is performed on a computer workstation equipped with an AMD Ryzen 9 5900X CPU, an Nvidia RTX 3090 GPU, and 64GB (2x32) Corsair DDR4 memory, with Ubuntu 18.04 LTS operating system. The DDPG and PPO benchmarks are taken from *spinningup* [146] for implementation of the proposed method. The configurations of the three tasks are interpreted as follows. Note that cylinder coordinates (θ, ρ, z) represent the position in the workspace for the remaining part of this chapter.

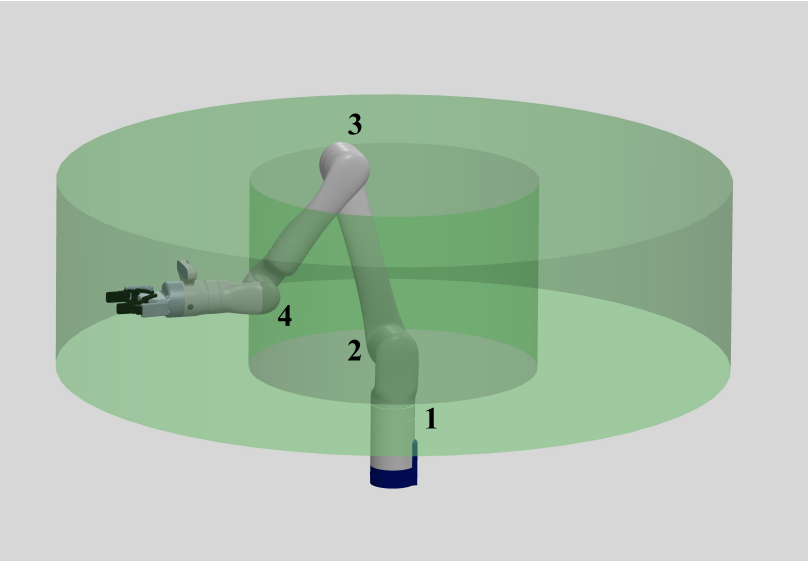


Figure 4.3: The 6-DOF Kinova® Gen3 robot model. The workspace is the green tube showing *min* and *max* extremities of polar radius and height. The four active joints are labeled as well.

4.4.1 Demonstrations

In a symmetrically partitionable space, the demonstration policy is executed uniformly in all local environments. Apart from a small noise, the initial position q_0 is fixed for each local environment. The goal and object positions P_g and P_o are randomly sampled from uniform distributions. For each episode of the chosen tasks, the

Algorithm 1: Off-policy Demo-EASE Training based on DDPG

Input: Parameters μ, ν for initial policy π_μ and value Q_ν;
Parameters $\mu' \leftarrow \mu$, $\nu' \leftarrow \nu$ for target networks $\pi_{\mu'}$ and $Q_{\nu'}$;
Full replay buffers $\mathcal{B}_\mathrm{o}$, $\mathcal{B}_\mathrm{d}$;
for $i \leftarrow 1$ **to** N_ep **do**
  for $j \leftarrow 0$ **to** N_up **do**
    $s_t \leftarrow q_0$, $t \leftarrow 0$;
    while $t < N_\tau$ *&* $\zeta_t = 0$ **do**
      Observe s_t and execute $a_t \sim \mathcal{N}(\pi_\mu(s_t), \sigma)$;
      Observe s'_t, r_t and determine ζ_t;
      Store sample $\{s_t, a_t, r_t, s'_t, \zeta_t\} \rightarrow \mathcal{B}_\mathrm{o}$;
      $s_t \leftarrow s'_t$, $t \leftarrow t + 1$;
  Sample batch buffers $\overline{\mathcal{B}}_\mathrm{o} \sim \mathcal{B}_\mathrm{o}$, $\overline{\mathcal{B}}_\mathrm{d} \sim \mathcal{B}_\mathrm{d}$;
  Calculate $\mathcal{L}_C$ and $\nabla_\nu \mathcal{L}_C$ using

$$\mathcal{L}_C\big(\overline{\mathcal{B}}_\mathrm{o}\big) = \frac{1}{\overline{N}_\mathrm{o}} \sum_{t=0}^{\overline{N}_\mathrm{o}-1} \left(r_t - Q(s_t, \pi(s_t)) + \gamma(1-\zeta_t) Q'(s'_t, \pi'(s'_t))\right)^2$$

  Calculate $\mathcal{L}_O$ using

$$\mathcal{L}_O\big(\overline{\mathcal{B}}_\mathrm{o}\big) = -\frac{1}{\overline{N}_\mathrm{o}} \sum_{t=0}^{\overline{N}_\mathrm{o}-1} Q(s_t, \pi(s_t))$$

  Calculate $\mathcal{L}_{BC}$ using Equation (4.6);
  Calculate $\mathcal{L}_A$ using Equation (4.5) and $\nabla_\mu \mathcal{L}_A$;
  Update networks π_μ, Q_ν by gradient descent:
  $\mu \leftarrow \mu + \alpha_\mu \nabla_\mu \mathcal{L}_A$, $\nu \leftarrow \nu + \alpha_\nu \nabla_\nu \mathcal{L}_C$;
  Update the target networks $\pi_{\mu'}, Q_{\nu'}$:
  $\mu' \leftarrow \omega\mu' + (1-\omega)\mu$, $\nu' \leftarrow \omega\nu' + (1-\omega)\nu$;

Algorithm 2: On-policy Demo-EASE Training based on PPO-Clip

Input: Parameters ω, ν for initial policy π_ω and value V_ν networks;
Empty replay buffers $\mathcal{B}_\text{o}, \mathcal{B}_\text{d}$;
for $i \leftarrow 1$ **to** N_ep **do**
  for $j \leftarrow 0$ **to** N_o **do**
    if $j \leq N_\text{o}$ **and** $\mathcal{B}_\text{d}$ *is not* **full then**
      Run a demonstration episode and store the trajectory $\rightarrow \mathcal{B}_\text{d}, \mathcal{B}_\text{o}$;
    else
      $s_t \leftarrow q_0$, $t \leftarrow 0$;
      while $t < N_\tau$ *&* $\zeta_t = 0$ **do**
        Observe s_t and execute $a_t \sim \mathcal{N}(\mu_{\omega_i}(s_t), \sigma_{\omega_i}(s_t))$;
        Observe s_{t+1}, r_t and determine ζ_t;
        $s_t \leftarrow s_{t+1}$, $t \leftarrow t+1$;
      Store the trajectory $\rightarrow \mathcal{B}_\text{o}$;
  Compute Rewards-to-go for the trajectories in $\mathcal{B}_\text{o}$, $\hat{R}_t = \sum_{l=t}^{N_\tau} r_l$;
  Sample batch buffers $\overline{\mathcal{B}}_\text{o} \sim \mathcal{B}_\text{o}$, $\overline{\mathcal{B}}_\text{d} \sim \mathcal{B}_\text{d}$;
  $k \leftarrow 0$;
  while $k \leq N_\text{up}$ *&* $D_{KL}(\pi_{\omega_{i,k}}, \pi_{\omega_{i,0}}) < D_{KL}^{target}$ **do**
    Compute Advantage estimates for the trajectories in $\mathcal{B}_\text{o}$ using the Generalized Advantage Estimator (GAE):

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=t}^{N_\tau - 1} (\gamma\lambda)^{l-t}(r_l + \gamma V_{\omega_i,k}(s_{l+1}) - V_{\omega_i,k}(s_l))$$

    where γ is MDP discount factor and λ is GAE discount factor;
    Calculate $\mathcal{L}_O$:

$$\mathcal{L}_O = -\frac{1}{N_\text{o} N_\tau} \sum_{\mathcal{B}_\text{o}} \sum_{t=0}^{N_\tau} \min\left(r_t^\omega \hat{A}_t, \text{clip}(r_t^\omega, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

    with $r_t^\omega = \pi_\omega(a_t|s_t)/\pi_{\omega_{i,k}}(a_t|s_t)$ and ϵ as the clipping threshold;
    Calculate $\mathcal{L}_{BC}, \mathcal{L}_A$ (Equations (4.5) and (4.6)), and $\nabla_\omega \mathcal{L}_A$;
    Calculate $\mathcal{L}_C$ and $\nabla_\nu \mathcal{L}_C$:

$$\mathcal{L}_C = \frac{1}{N_\text{o} N_\tau} \sum_{\mathcal{B}_\text{o}} \sum_{t=0}^{N_\tau} \left(V_\nu(s_t) - \hat{R}_t\right)^2$$

    Update networks π_ω, V_ν by gradient descent:
    $\omega_{i,k+1} \leftarrow \omega_{i,k} + \alpha_\omega \nabla_\omega \mathcal{L}_A$, $\nu_{i,k+1} \leftarrow \nu_{i,k} + \alpha_\nu \nabla_\nu \mathcal{L}_C$;
    $k \leftarrow k+1$;

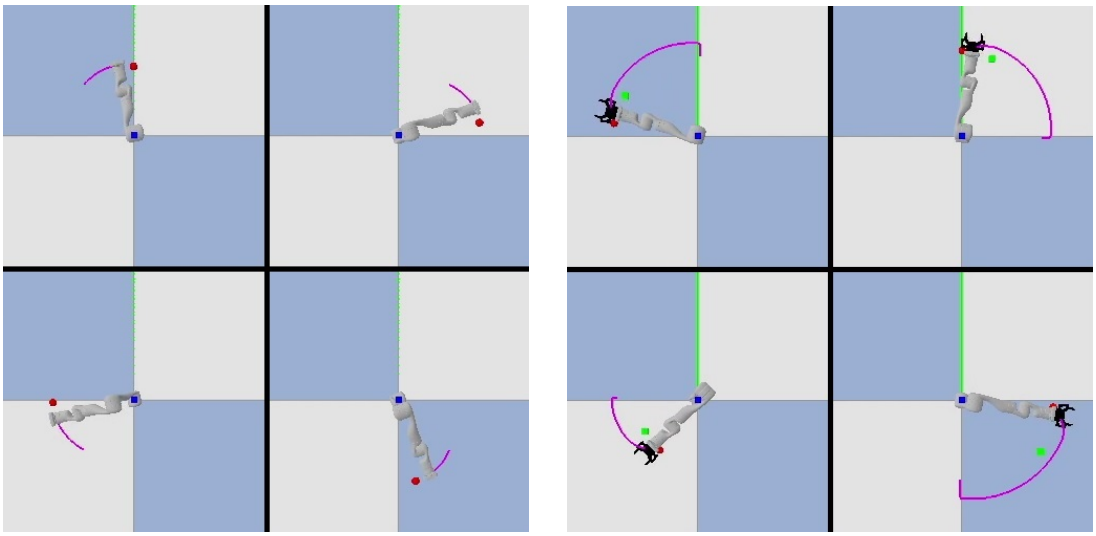
robot moves from an initial joint angle $q_0 \in \mathbb{R}^n$ to reach the goal position $P_g \in \mathbb{R}^3$. For different stages, demonstration recording, training, and testing, the initial joint angle q_0 and the goal position P_g are sampled differently to evaluate the generalizability of the agent. The P2P-O and P&P tasks also involve the sampling of the object position P_o, which is sampled according to the initial robot configuration. The demonstration is recorded in local environments $\mathcal{S}_k$, where $k \sim \{1, 2, 3, 4\}$ is uniformly sampled, while the training and test are performed in the global environment $\mathcal{S}$.

Configuration of the P2P task

For the P2P task, the Off-policy Demo-EASE was chosen to train the RL agent. The initial position of the end effector is sampled uniformly such that $\theta_0 \sim \left\{\frac{k\pi}{2} - \frac{\pi}{4}\right\}$, $\rho_0 = 0.52$, $z_0 = 0.42$, for all $k \sim \{0, 1, 2, 3\}$. Here, the polar coordinates (θ_0, ρ_0, z_0) represent the end-effector position in the workspace. Then, inverse kinematics is calculated to obtain the initial joint angles q_0. The goal position (θ_g, ρ_g, z_g) is uniformly sampled from $\theta_g \sim \left[\frac{k\pi}{2} - \frac{\pi}{2}, \frac{k\pi}{2}\right)$, $\rho_g \sim [\,0.4,\, 0.6\,)$, $z_g \sim [\,0.35,\, 0.55\,)$, which is then transformed to the Cartesian coordinate P_g. In such a manner, the sampled goal is restricted to the same local environment as the initial position of the robot end-effector. The number of the demonstration episodes N_{demos} is selected as 0, 80, and 160 for the comparison study. Four examples of P2P demonstrations are illustrated in Figure 4.4a.

Configuration of the P2P-O and P&P

Unlike P2P-O, trained with Off-policy Demo-EASE, P&P is trained with the On-policy version. For both tasks, the initial position of the robot end-effector is uniformly sampled from $\theta_0 \sim \left\{\frac{k\pi}{2}\right\}$, $\rho_0 = 0.52$, $z_0 = 0.42$, for all $k \sim \{0, 1, 2, 3\}$, which is then transformed to the initial joint angle q_0. The target position P_g is sampled from $\theta_g \sim \left[\frac{k\pi}{2} - \frac{\pi}{4}, \frac{k\pi}{2}\right)$, $\rho_g \sim [\,0.4,\, 0.6\,)$, $z_g \sim [\,0.35,\, 0.55\,)$. A cube sized 0.04 m is placed at position P_o which is determined based on the goal position so that it is located between the initial position of the gripper and the target, $\theta_o \sim \theta_g + \{-1, 1\} \times \left[\frac{\pi}{12}, \frac{\pi}{6}\right)$, $z_o \sim [\,z_g - 0.1, z_g + 0.1\,)$. The subscript o indicates the coordinate of the object. The number of demonstration episodes N_{demos} is set at 100, 200, and 400 for P2P-O and 150, 250, and 400 for P&P. They are larger than what is chosen in the P2P task due to their complexity. Four examples of P2P-O demonstrations are shown in Figure 4.4b.



(a) P2P (b) P2P with obstacle

Figure 4.4: Samples of the demonstrations used to prepare the local demo replay buffers

4.4.2 Agent Training

The training of the agent is performed in the global environment $\mathcal{S}$. For both the P2P and the P2P-O tasks, the initial position of the robot end-effector is fixed in $\theta_0 = \pi/4$, $\rho_0 = 0.52$, $z_0 = 0.42$, and the target positions P_g is sampled from

$$\theta_g \sim [\,-\pi,\, \pi\,),\ \rho_g \sim [\,0.3,\, 0.7\,),\ z_g \sim [\,0.25,\, 0.65\,). \tag{4.7}$$

For the P2P-O and P&P tasks, the object position is sampled from $\theta_b = \theta_g + \mathrm{sgn}(\theta_1 - 0.5)\theta_2$, $\rho_b \sim [\,0.4,\, 0.6\,)$, $z_b \sim [z_g - 0.1, z_g + 0.1)$, where $\mathrm{sgn}(\cdot)$ is the sign function that produces 1, -1, and 0 for positive, negative, and zero values, respectively. $\theta_1 \sim [\,0,\ 1\,)$ and $\theta_2 \sim [\,\pi/12, \pi/6\,)$ are random variables sampled from uniform distributions. Such a definition of θ_b is to ensure that the block is forced to be towards a clockwise or counterclockwise direction from the target in the global space.

The reward functions R are selected according to the following concerns.

1. **Approaching:** Penalize the reaching error at each time.
2. **Effort:** Encourage the agent to apply minimum effort.
3. **Reaching:** Encourage ultimately reaching the goal.
4. **Collision:** Penalize any collisions with the object.

5. **Grasping:** Encourage the agent to pick up the object when intended.

Thus, the following reward function is designed,

$$R(s_t, a_t) := r_t^{\text{dst}} + r_t^{\text{eft}} + r_t^{\text{rch}} + r_t^{\text{cls}} + r_t^{\text{grsp}}. \tag{4.8}$$

The first term $r_t^{\text{dst}} = -\alpha_1 ||e_t||$ is to penalize the reaching error e_t, where $\alpha_1 \in \mathbb{R}^+$ is design parameter to be determined. This term stimulates the reaching towards the goal. The second term $r_t^{\text{eft}} = -\alpha_2 \|\hat{\tau}_t\|$ is to penalize the actuation torques of the robot, where α_2 is a parameter and $\hat{\tau}_t = \begin{bmatrix} \hat{\tau}_t^{(1)} & \hat{\tau}_t^{(2)} & \hat{\tau}_t^{(3)} & \hat{\tau}_t^{(4)} \end{bmatrix}^\top$ are the normalized actuation torques of the four active joints of the robot. For each joint j, $\hat{\tau}_t^{(j)} = \|\tau_t^{(j)}\|/\tau_{\max}^{(j)}$, where $\tau_t^{(j)}$ is the actual actuation torque at time t and $\tau_{\max}^{(j)}$ is the maximum torque, where $\tau_{\max}^{(1,2,3)} = 39\,\text{N·m}$ and $\tau_{\max}^{(4)} = 9\,\text{N·m}$. The third term $r_t^{\text{rch}} = \begin{cases} 0, & \text{if } \|e_t\| \geq \varepsilon \\ R_1, & \text{else} \end{cases}$ gives a big positive reward for ultimately reaching the goal, where $\varepsilon = 0.05\,\text{m}$ is the tolerated reaching error. The term $r_t^{\text{cls}} = \begin{cases} -R_2, & \text{if collision detected} \\ 0, & \text{else} \end{cases}$ exerts a big negative reward for collision either with the object or the robot itself, and the last term $r_t^{\text{grsp}} = \begin{cases} R_3, & \text{if } \|P_t^e - P_t^o\| \leq \varepsilon \text{ and } d(\mathbf{q}_t^e - \mathbf{q}_t^o) \leq \varepsilon_q \\ 0, & \text{else} \end{cases}$ is a one-time reward in the P&P task when the robot reaches the object in a proper position with proper orientation, where $\varepsilon, R_1, R_2, R_3 \in \mathbb{R}^+$ are parameters. For the reward function, the parameters are selected as $\alpha_1 = 2 \times 10^{-3}$, $\alpha_2 = 10^{-3}$, $R_1 = 10$, $R_2 = R_3 = 2$ wherever needed for each task. In the definition of r_t^{grsp}, $d(\mathbf{q}_t^e - \mathbf{q}_t^o)$ is the distance between the end-effector and object quaternions. Also, if the robot satisfies the condition for receiving R_3, the grasping occurs automatically until the end of the episode.

The architecture of the actor and critic networks in P2P and P2P-O are multilayer perceptron networks with $[\,128,\ 512,\ 128\,]$ and $[\,256,\ 1024,\ 256\,]$ units, respectively. A similar architecture with $[\,512,\ 512,\ 256\,]$ units is used for both actor and critic networks in P&P. The remaining parameters of the agent training are indicated in Table 4.1 and Table 4.2.

4.5 Experimental results

This section analyzes the results of the implementations and evaluates the performance of the proposed method. Additional information about the environment con-

Table 4.1: Training parameters for off-policy Demo-EASE experiments

Notation	Parameter	P2P	P2P-O
N_τ	Max Episode Length	400	500
N_{ep}	Number of epochs	250	500
N_{up}	Update interval	20	25
N_{o}	Size of $\mathcal{B}_{\text{o}}$	10^6	
$\overline{N}_{\text{o}}$	Size of $\overline{\mathcal{B}}_{\text{o}}$	100	
$\overline{N}_{\text{d}}$	Size of $\overline{\mathcal{B}}_{\text{d}}$	100	
γ	Discount factor	0.99	
σ	Exploration Noise Std. Dev.	0.1	
ω	Target update coefficient	0.995	
α_μ	Actor's learning rate	10^{-3}	
α_ν	Critic's learning rate	10^{-3}	

figuration and platforms is presented in Section 4.4. The codebase used for the experiments is publicly available.[1]

4.5.1 The Training Results

Two important hyperparameters play the major role in the influence of the method on performance, namely the number of demonstrations N_{d}, and the behavior cloning gain λ_{BC}. For each combination of N_{demos} and λ_{BC}, multiple repeats are performed to reduce the effect of randomization. A conventional DDPG and PPO agent is also trained for comparison, corresponding to $N_{\text{demos}} = 0$ and $\lambda = 0$ for the corresponding experiments.

Along with the learning curve that shows the changes in accumulated reward over time, the following numerical metrics are also used to evaluate the performance:

- **Initial return** $\overline{R}_{10}$: the average return over the initial 10% of episode steps;
- **Ultimate return** $\overline{R}_{90}$: the average return over the final 10% of episode steps;
- **Reward Increment** $I_R = \overline{R}_{90} - \overline{R}_{10}$: the increase of the returns during training;
- **Half-trained time** T_{50}: the episode when the return first exceeds $\frac{1}{2}\overline{R}_{90}$.

[1] https://github.com/amsoufi/DemoEASE

Table 4.2: Training parameters for on-policy Demo-EASE experiment

Notation	Parameter	P&P
N_τ	Max Episode Length	800
N_{ep}	Number of epochs	50
N_{up}	Policy updates per epoch	20
N_{o}	Size of $\mathcal{B}_{\text{o}}$	4000
$\overline{N}_{\text{o}}$	Size of $\overline{\mathcal{B}}_{\text{o}}$	800
$\overline{N}_{\text{d}}$	Size of $\overline{\mathcal{B}}_{\text{d}}$	100
γ	Discount factor	0.99
γ^{GAE}	GAE factor	0.97
ϵ	Clip ratio	0.2
D_{KL}^{target}	KL-Divergence threshold	0.02

The P&P Task

Figure 4.5 shows the training curves for the P&P task. It can be noticed that agents trained using Demo-EASE show a significant advantage over the original baseline. As expected, the learning curves of all Demo-EASE agents, even those with minimal adjustments, start to grow early on. Also, more demonstrations or larger λ_{BC} in training leads to higher accumulated rewards during the training process. Additional training metrics for the P&P task are listed in Table 4.3.

Table 4.3: Numerical training metrics of the P&P agent

Agent	λ_{BC} ($N_{\text{demos}} = 400$)			N_{demos} ($\lambda_{BC} = 15$)			PPO
	5	10	15	150	250	400	
$\overline{R}_{10}$	0.36	0.37	0.36	0.35	0.19	0.36	-0.60
$\overline{R}_{90}$	1.67	2.29	3.03	1.73	2.26	3.03	-0.51
I_R	1.30	1.92	2.67	1.38	2.07	2.67	0.09
T_{50}	649	612	548	721	572	548	655

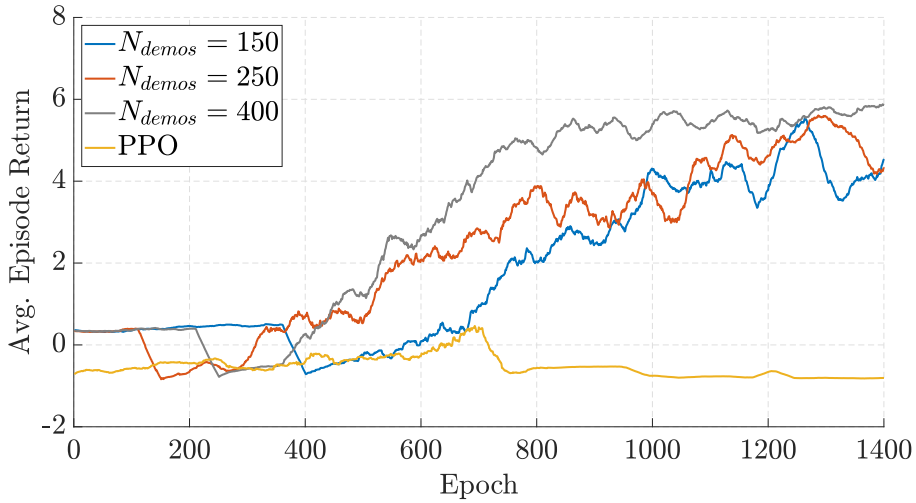
The P2P Task

The learning curves of the P2P task during the agent training stage are presented in Figure 4.6. Specifically, the results depicting the influence of the number of demon-

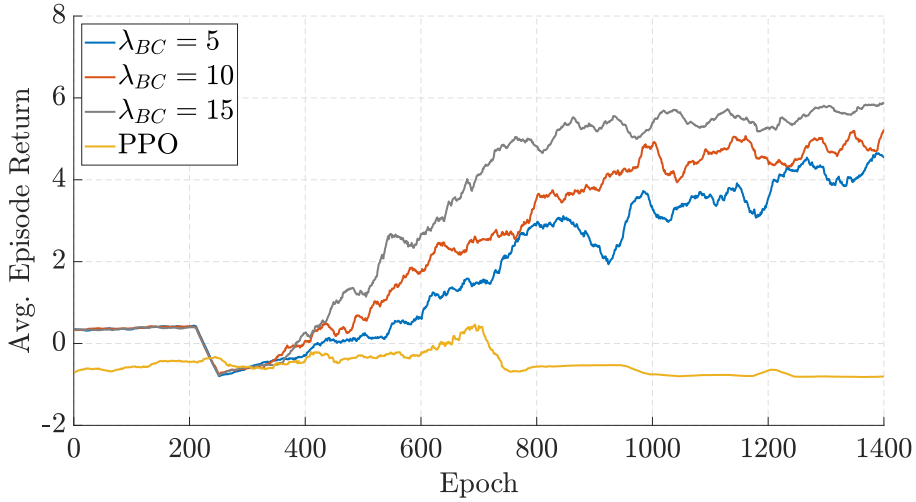
strations N_{demos} and the behavior cloning gain λ_{BC} are respectively illustrated in Figure 4.6a and Figure 4.6b. Additional training metrics are listed in Table 4.4.

The P2P-O Task

The learning curves of the P2P-O task during the agent training stage are shown in Figure 4.7. Specifically, the results showing the influence of the number of demonstrations N_{demos} and the behavior cloning gain λ_{BC} are respectively illustrated in Figure 4.7a and Figure 4.7b. Other numerical metrics are shown in Table 4.5.



(a) Different N_{demos} with $\lambda_{BC} = 15$



(b) Different λ_{BC} with $N_{\text{demos}} = 400$

Figure 4.5: The effect of hyperparameters on the P&P learning curves

Table 4.4: Numerical training metrics of the P2P agent

Agent	λ_{BC} ($N_{\text{demos}} = 100$)			N_{demos} ($\lambda_{BC} = 1$)			DDPG
	0.1	0.6	1.8	0	80	160	
$\overline{R}_{10}$	-0.44	-0.06	-0.12	-0.28	-0.13	0.06	-0.56
$\overline{R}_{90}$	3.72	6.83	8.94	5.55	8.58	8.67	0.90
I_R	4.16	6.89	9.06	5.83	8.71	8.61	1.49
T_{50}	657	292	398	281	457	379	486

Table 4.5: Numerical training metrics of the P2P-O agent

Agent	λ_{BC} ($N_{\text{demos}} = 250$)			N_{demos} ($\lambda_{BC} = 1$)			DDPG
	0.5	1	2	100	200	400	
$\overline{R}_{10}$	-1.61	-1.51	-1.62	-1.53	-1.51	-1.67	-1.51
$\overline{R}_{90}$	5.79	6.80	7.19	3.40	3.44	5.95	-1.52
I_R	7.40	8.31	8.81	4.93	4.95	7.63	-0.01
T_{50}	1503	1603	1441	1787	1780	1483	N/A

4.5.2 The Test Study

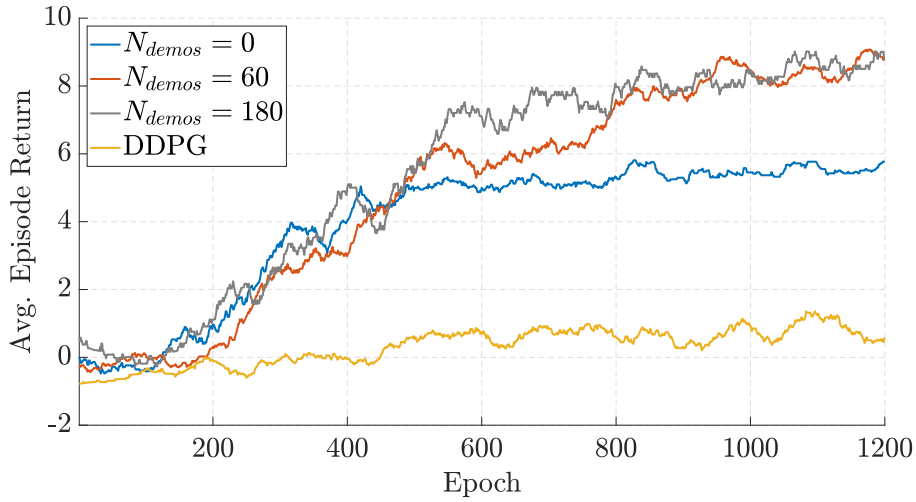
In this section, the trained agents are deployed onto the simulated robot model, and their test performance is evaluated. The test study is conducted in the *global environment*, with 500 trials per task. In the test study, the following metrics are used for evaluation.

1. **Success rate** P_{scs}: the ratio of the number of successful trials N_{scs} to all trials N_{ttl}.

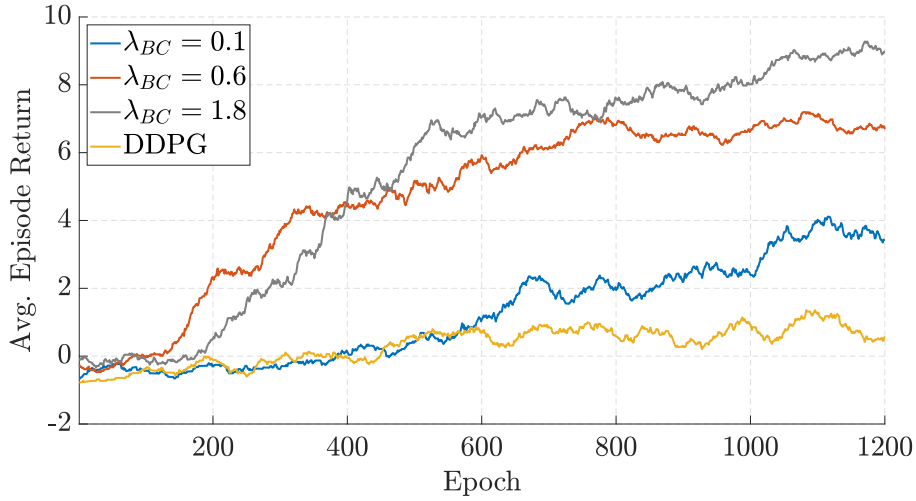
2. **Average effort** $\overline{T}_{\text{eff}}$: average torque utility rate of the joints in the successful trials, as in the following,

$$\overline{T}_{\text{eff}} = \frac{1}{N_{\text{scs}}} \frac{1}{N_\tau} \sum_{i=1}^{N_{\text{scs}}} \sum_{t=1}^{N_\tau} \left\| \hat{\tau}_t^i \right\|. \tag{4.9}$$

   where $\hat{\tau}_t$ is the norm of the vector of active joint utilities, and i denotes the i-th trial.



(a) Different numbers of demos N_{demos}



(b) Different behavior cloning gains λ_{BC}

Figure 4.6: The effect of hyperparameters on the P2P learning curves

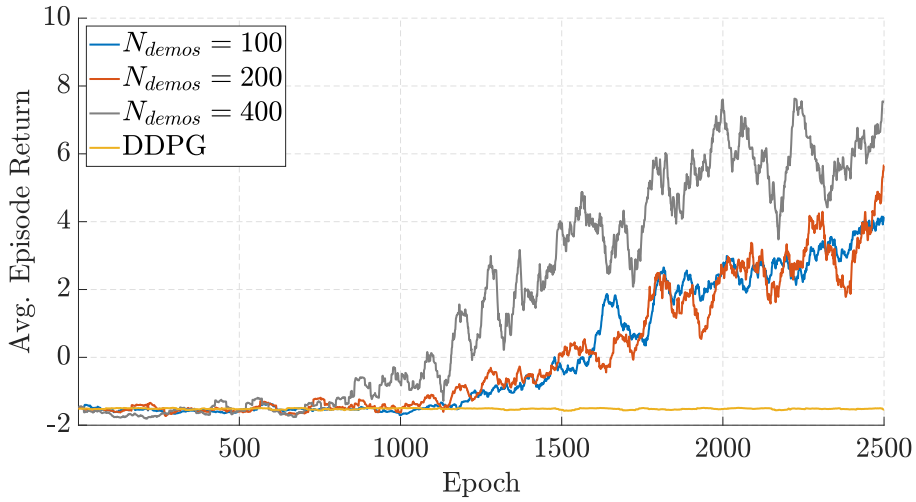
3. **Average test return** $\overline{R}_{\text{test}}$: the average return over successful trials.

4. **Ultimate error** $\overline{E}_{\overline{95}}$: the average ultimate reaching error over the final 5% of a successful trial,

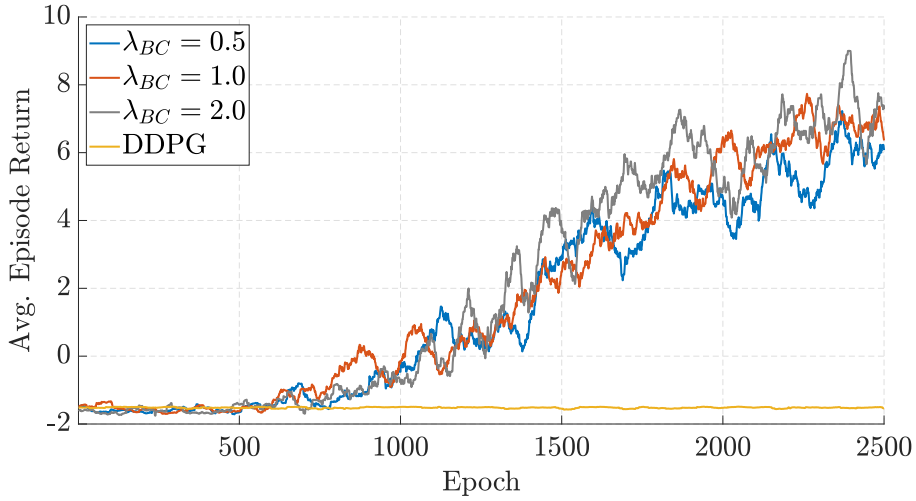
$$\overline{E}_{\overline{95}} = \frac{1}{N_{\text{scs}}} \frac{1}{0.05 N_\tau} \sum_{j=1}^{N_{\text{scs}}} \sum_{t=0.95N_\tau}^{N_\tau} \left\| e_t^{(j)} \right\| \tag{4.10}$$

where $e_t^{(j)}$ is the reaching error of trial j at time t.

The initial robot end-effector position is fixed as $\theta_0 = -\pi/4$, $\rho_0 = 0.5$, $z_0 = 0.45$



(a) Different numbers of demos N_{demos}



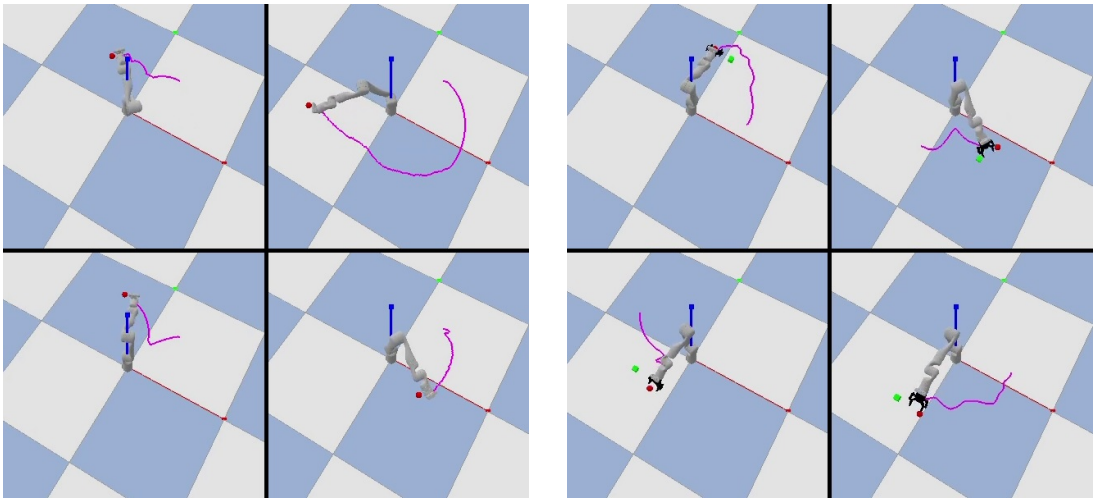
(b) Different behavior cloning gains λ_{BC}

Figure 4.7: The effect of hyperparameters on the P2P-O learning curves

for P2P and $\theta_0 = -\pi/2$, $\rho_0 = 0.5$, $z_0 = 0.45$ for P2P-O and P&P. The robot goal configuration is sampled from the same distribution as the training configuration in Section 4.4.2. Each test trial ends if the **Reaching**, **Timeout**, and **Collision** criteria are met. Besides, the trajectories that meet the **Reaching** criterion are considered *successful trials*. Examples of the successful trials in two of the simulation environments of the test study are illustrated in Figure 4.8.

The reaching error trajectories $\|e_t\|$ of the P2P, P2P-O, and P&P tasks in the test study, subject to different values of λ_{BC} and N_{demos}, are respectively illustrated in Figure 4.9, Figure 4.10, and Figure 4.11. In these figures, the dark curve represents

the averaged trajectory over successful trials, and the shaded areas represent the standard deviation of the trajectories. It can be seen that implementing the behavior cloning and demonstrations generates smooth and stable trajectories in this test.



(a) P2P (b) P2P with obstacle

Figure 4.8: Samples of successful trials of trained agents

The numerical metrics of the P2P agent in the test study, subject to different hyperparameters N_{demos} and λ_{BC}, are presented in Table 4.6. Additionally, the deployment evaluation results of the tests for P2P-O and P&P tasks are shown in Table 4.7 and Table 4.8. The PID controller designed as Equation (4.4) is also tested for comparison. The results indicate that the PID baseline shows a mediocre success rate and ultimate error compared to the Demo-EASE agents. Meanwhile, PID also uses the lowest effort and gets the smallest reward. This confirms that PID is a conservative control baseline that produces smooth and energy-saving trajectories but neglects task-specific optimality. Note that the reaching errors are not relatively big because episodes are truncated at a 5 cm threshold.

4.5.3 Result Discussion

It is noticed that even the Demo-EASE agent with the minimum number of demonstrations or behavior cloning gain performs significantly better than the standard RL agent in terms of not only the reward but also the sampling efficiency. It is also interesting to find that the behavior cloning gain shows more contribution to improving the training performance compared to the number of demonstrations. In the test

Table 4.6: Numerical evaluation results of the P2P agent in test

Agent	λ_{BC} ($N_{\text{demos}} = 100$)			N_{demos} ($\lambda_{BC} = 1$)			PID
	0.1	0.6	1.8	0	80	160	
$P_{\text{scs}}(\%)$	78.2	92.6	99.6	42.8	91.4	89.5	59.0
$\overline{T}_{\text{eff}}$	36.2	26.9	27.3	54.3	28.3	22.1	15.1
$\overline{R}_{\text{test}}$	9.88	9.96	9.98	9.97	9.98	9.94	9.58
$\overline{E}_{\overline{95}}$ (cm)	6.24	4.91	4.76	4.77	4.79	5.38	5.29

Table 4.7: Numerical evaluation results of the P2P-O agent in test

Agent	λ_{BC} ($N_{\text{demos}} = 100$)			N_{demos} ($\lambda_{BC} = 1$)			PID
	0.5	1	2	100	200	400	
$P_{\text{scs}}(\%)$	92.6	96.2	98.0	71.6	78.4	92.2	83.0
$\overline{T}_{\text{eff}}$	67.3	66.1	65.8	70.5	67.6	66.1	54.9
$\overline{R}_{\text{test}}$	9.88	9.87	9.89	9.89	9.88	9.88	9.21
$\overline{E}_{\overline{95}}$ (cm)	5.43	5.30	5.15	5.06	5.28	5.27	5.51

Table 4.8: Numerical evaluation results of the P&P agent in test

Agent	λ_{BC} ($N_{\text{demos}} = 400$)			N_{demos} ($\lambda_{BC} = 15$)			PID
	5	10	15	150	250	400	
$P_{\text{scs}}(\%)$	80.8	90.0	97.6	86.2	89.2	97.6	73.4
$\overline{T}_{\text{eff}}$	56.0	45.8	43.5	44.1	44.7	43.5	44.0
$\overline{R}_{\text{test}}$	11.88	11.89	11.93	11.78	11.83	11.93	11.76
$\overline{E}_{\overline{95}}$ (cm)	5.29	4.86	5.09	6.07	4.86	5.09	6.05

study, the Demo-agent also presents a decent performance in terms of safety (collision avoidance) and precision (small reaching error). Yet, trained RL agents applied more torque than PID, which can be regulated by reshaping the reward function. It is worth mentioning that the symmetrical partitionable condition for the environment seems to be quite strong for most practical robot manipulation tasks. In practice, only some small regions of the environment likely do not possess such symmetric

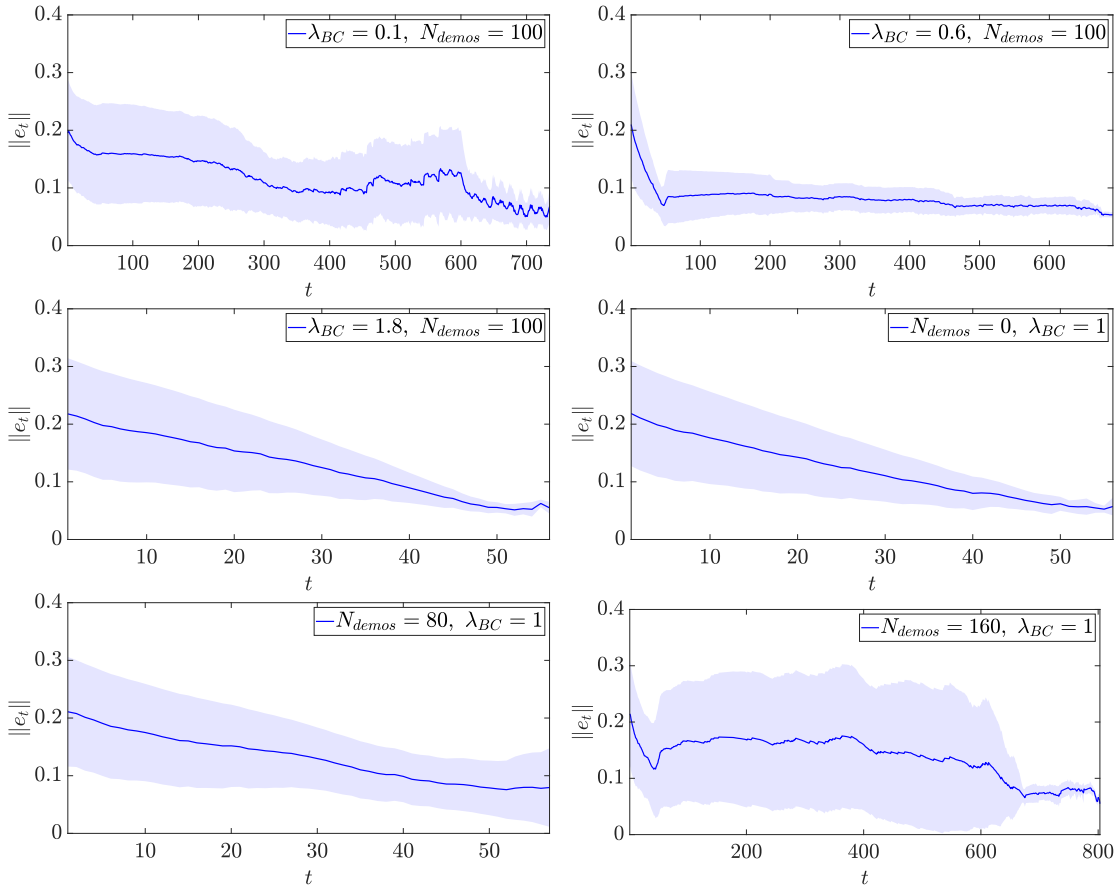


Figure 4.9: The average error $\|e_t\|$ in successful trials over time t in the P2P test.

properties. Nevertheless, the proposed work is devoted to investigating the feasibility and potential of using demonstrations in symmetric environments to improve the sampling efficiency of RL methods with extreme use cases. Automated symmetry detection, generalizability, and verification in practical problems are some possible topics for future work.

4.6 Concluding Remarks

This chapter proposed a novel data aggregation method for model-free RL in robot manipulation and incorporated behavior cloning from abstract symmetric demonstration to improve both the training and test performance of the conventional RL. A conservative PID controller was used as the source of expert knowledge for producing such demonstrations. The symmetric property of the environment was exploited to duplicate the demonstration samples, which further improved the efficiency of the

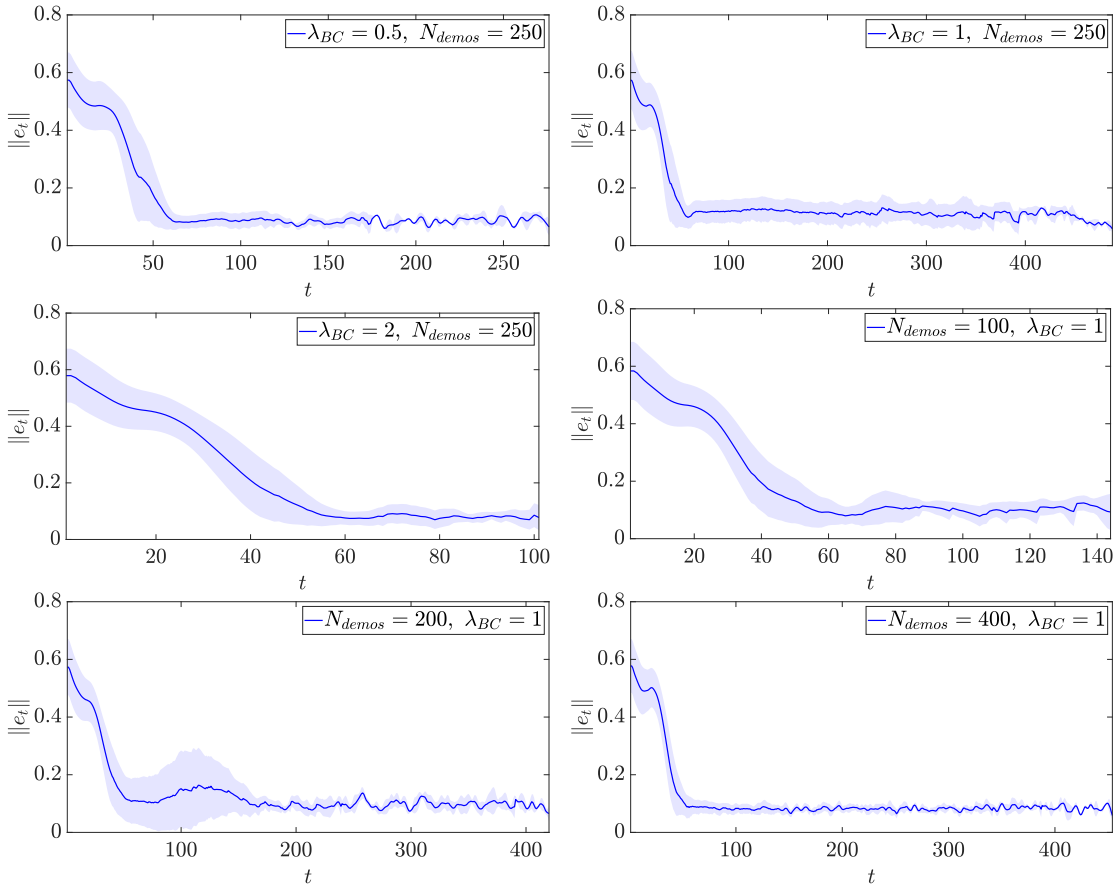


Figure 4.10: The average error $\|e_t\|$ and in successful trials over time t in the P2P-O test.

agent training. The proposed method was validated using a robot manipulator in three essential simulated manipulation tasks. The results showed that the behavior cloning from the demonstration has a positive effect on both the training and deployment of the RL agent. Nevertheless, whether the proposed method is also effective for more complicated manipulation tasks like object pushing or grasping is a question, that will be further investigated in future work.

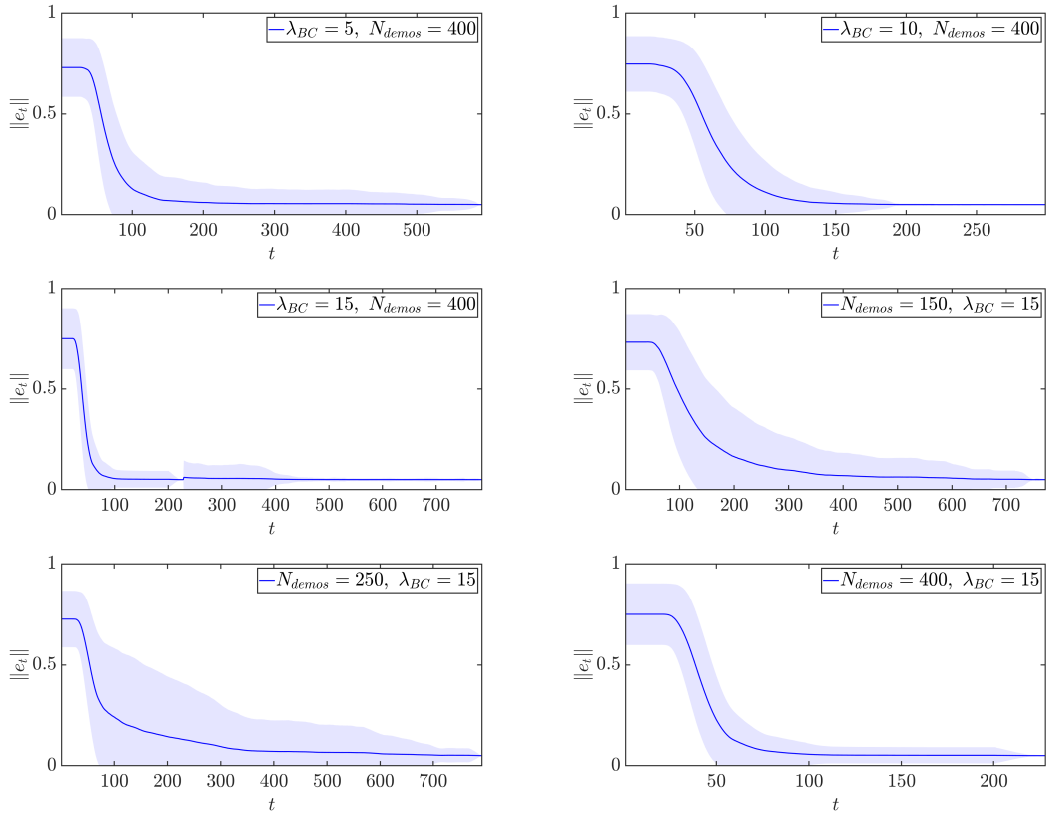


Figure 4.11: The average error $\|e_t\|$ in successful trials over time t in the P&P test.

Chapter 5

Facilitating RL Sim-to-real by Intrinsic Stochasticity of Real-Time Simulation

Simulation is essential to RL before implementation in the real world, especially for safety-critical applications like robot manipulation. Conventionally, RL agents are sensitive to the discrepancies between the simulation and the real world, known as the sim-to-real gap. The application of domain randomization, a technique used to fill this gap, is limited to the imposition of heuristic-randomized models. This chapter investigates the properties of the **I**ntrinsic **S**tochasticity of **R**eal-**T**ime simulation (RT-IS) of off-the-shelf simulation software and its potential to improve RL performance. This improvement includes a higher tolerance to noise and model imprecision and superiority to conventional domain randomization in terms of ease of use and automation. Firstly, analytical studies are conducted to measure the correlation of RT-IS with the utilization of computer hardware and to validate its comparability with the natural stochasticity of a physical robot. The RT-IS feature is exploited in the training of an RL agent. The simulation and physical experiment results verify the feasibility and applicability of RT-IS to robust agent training for robot manipulation tasks. The RT-IS-powered RL agent outperforms conventional agents on robots with modeling uncertainties. RT-IS requires less heuristic randomization, is not task-dependent, and achieves better generalizability than the conventional domain-randomization-powered agents. The findings provide a new perspective on the sim-to-real problem in practical applications like robot manipulation tasks.

5.1 Sim-to-real Challenge

Today, the world finds itself amid an exciting industrial revolution represented by *Industry 4.0*, a paradigm that has pushed manufacturing systems to become more intelligent, reliable, and efficient by utilizing innovative technologies including artificial intelligence (AI), cyber-physical systems, internet of things, and cloud computing [147, 148]. AI technologies provide the tools needed to build intelligent manufacturing systems that are flexible, smart, and easily re-configurable. AI has applications in just about any manufacturing process such as process monitoring [149], process optimization [150], and process control [151]. AI enables manufacturing devices and machines, especially robots, to self-monitor and autonomously respond to different situations and environmental changes [152]. The next generation of intelligent manufacturing requires the robots to be capable of changing their behaviors and modals automatically adapted to the peripheral changes without human intervention or manual reprogramming [5].

Reinforcement Learning (RL) is one of the most promising technologies that can achieve this goal for robotic systems. The RL framework introduces autonomous decision-making based on a feedback and reward setting, where an agent applies actions to an environment according to certain observations. The applied actions are solved subject to the optimization of predefined rewards. Different from the conventional control methods built on heuristics, RL does not usually require prior knowledge of the environment, which is also referred to as *model-free* [153]. Instead, the RL training process takes a trial-and-error approach to learn the best commands for a specific task, such as grasping [154], navigation [155], and Braille typing [156]. This framework allows a robot to learn a feasible control solution automatically from the interaction data without manually programming the robot and task models. To provide sufficient data for the training of RL, simulation of the system is used to save the temporal and hardware resources. Nevertheless, due to the mismatch of the features between the physical systems and their simulation models, the agents trained in simulation do not always reproduce their performance in the real world, which is also referred to as the sim-to-real gap [157] or the *reality gap* [158]. The conventional RL methods are sensitive to the variation between the virtual and the physical robot dynamics [158] caused by the reality gap, which becomes a major barrier preventing the application of RL to practical problems and remains an open question that prevents the wide application of RL to practice.

An inspiring idea to fill the sim-to-real gap is found in robust control, a well-known concept in the conventional control theory to cope with the existence of system uncertainties [159]. Robust control aims at designing a control method that is feasible for all system models that are defined in a closed model set that covers all possible features of the real system. Thus, a similar concept namely *Robust RL* is proposed as a different avenue taken to tackle the issue of uncertainty and disturbance in real applications of RL. The core idea was an amalgamation of the robust control law of H_∞ and reinforcement learning, suggested by [160]. In this approach, the disturbance is modeled as an agent with a non-deterministic but bounded range of *disturbance* trying to destabilize the RL agent. The disturbance model impedes the controller by applying the biggest model discrepancy or input noise. In robust *adversarial* RL, the agent is equipped with another deep learning model which learns the extent of its disturbance output throughout the training process [161]. Various domains of action for the mentioned adversary have also been experimented on, spanning robustness to transition (environment dynamics), disturbance (external unmodeled forces), action (input noise), and observation (transducer uncertainty) [162]. The application of Robust RL formulation has been expanded, in a very diverse set of methods, much further than sim-to-real transfer in recent years. In another work, a robust goal-conditioned reinforcement learning approach for end-to-end robotic control in adversarial and sparse reward environments is introduced [163]. Their system utilizes a mixed adversarial attack scheme to generate diverse perturbations on observations and employs a hindsight experience replay technique to transform failed experiences into successful ones. Also, proposed by [164], an observation-based adversarial reinforcement learning approach for robust lane change of autonomous vehicles is presented, addressing the challenges of perception uncertainty and adversarial observation perturbations. At a more fundamental level, [165] devises an efficient way to exploit perturbed historical data from adversarial environments. This can especially be beneficial in the case of sim-to-real where generating perturbed simulated environments is affordable. Furthermore, a model-based reinforcement learning algorithm is devised by [166] for learning an ϵ-optimal robust policy where the nominal model is unknown. This algorithm considers different forms of uncertainty sets and provides precise characterizations of sample complexity by generative models. This provides an avenue to a range of novel robust RL algorithms. The intuition behind the robust RL concept is astute and it suggests certain guarantees of stability, however, structuring such problems requires two sources of prior knowledge: choosing the domains of

impact for the adversary and designing the extent of said impact. In many cases, this information is not provided before the actual implementation. Mismodeling in this design may lead to over- or under-compensation of the discrepancies and sub-optimal performance. Additionally, parameter tuning the agents equipped with this level of sophistication can be disinclined in the industrial use of the technology.

The existing solutions for RL agents to alleviate the sim-to-real problem mainly include system identification [167], domain adaptation [168], domain randomization [169]. The relation between these three solutions and the reality gap is illustrated in Figure 5.1. *System identification* aims to solve the sim-to-real gap by mathematically estimating a specific pre-structured model based on data [170], e.g., by defining a regression problem [171, 172]. This approach is most effective for tuning time-invariant parameters, however in reality a major problem is noise which is of a different nature. Some recent cases of implementation of machine learning for modeling time-variant parameters, however, can be found [173, 174]. Despite these results, the efficiency of system identification, in this case, is questionable due to the requirement of tremendous data and exhaustive modeling efforts. *Domain adaptation* is inspired by the computer vision field, where the data from a *source domain* is used to improve the performance on a different *target domain* [168]. It has been used as the critical technology of transfer learning [122]. From the perspective of sim-to-real, the source is the simulation environment and the real world serves as the target. This technique focuses on the transformation of feature space either by using statistical-discrepancy methods [175] or adversarial-learning methods [176] such as generative adversarial networks (GAN). Then, the model for the target domain can be generated by adapting the source model utilizing the feature transformation. However, the application of domain adaptation to sim-to-real is hindered by the fact that it requires data from both domains.

Another solution to mitigate the sensitivity of RL is domain randomization, also known as *dynamic variability* [177], which promotes the generalizability of an RL agent by exciting with multiple varying model configurations [178, 179]. The main technical point of domain randomization is to impose heuristic noise to the models or observations of the environment during the training process [180, 181], such that the agents can achieve a balanced performance over various models and observations. The spaces of the perturbed models and the observations are referred to as *domain*s, and the manner of imposing heuristic noise in these domains is referred to as *randomization*. The randomization can be of any two types visual [169] and

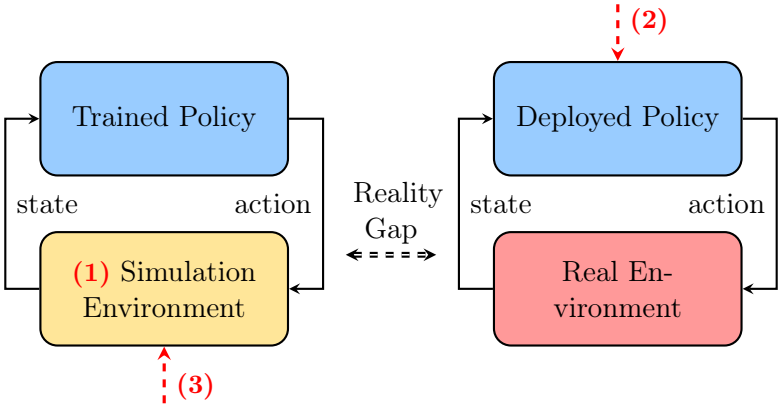


Figure 5.1: The illustration of the *reality gap* and the three common approaches to solving it: (1) system identification, (2) domain adaptation, and (3) domain randomization.

dynamic [119]. Visual randomization is mostly related to computer vision tasks such as object detection [182, 183], object recognition [184], or pose estimation [185]. Dynamic randomization targets robotics and control tasks by randomizing dynamic features such as mass, friction, and measurement noise in observations. The technology of domain randomization can avoid the overfitting of an RL agent to a single environment model and guarantee its generalizability to all perturbed environments [181]. Similar technology is also used to avoid local optimums [138, 180]. However, there exists a trade-off between the extent of randomization and the performance of the agent. Firstly, the randomization sacrifices the training performance of the agent on individual environment models. Thus, large-extent of randomization may not pay off the overall loss of the training performance. Secondly, randomization increases the variance of the trained policy, which makes the test performance unstable and difficult to predict. Thirdly, similar to robust control, large-extent domain randomization may lead to the infeasibility of the agent training, i.e., the failure of policy convergence. Therefore, determining the extent of the heuristic randomization is always a challenge.

Naturally, the approaches to tackle the challenge targeted by this chapter are not limited to the above. For instance, a hierarchical combination of RL (high-level) and precise conventional controller (low-level) is proposed by [186]. The RL agent creates setpoints at a lower rate for a stiff controller in the simulation. The conventional controller can compensate for the disturbances, isolating the RL agent from uncertainties. It has also been shown that decentralizing the controllers minimizes the

effect of uncertainty [187]. It should be noted that delays can form another source of challenge for sim-to-real, besides noise and disturbance [126]. These delays can be caused because of the low update rate of the RL agent on real hardware, the latency of the action execution in the actuators, or both. In [188], adding an estimator to predict the state changes during action selection improved the performance in real-time. Even the mere presence of random delays has shown a general improvement in the robustness of the RL agent [189, 190]. The latter inspires the authors to verify the hypothesis of the benefit of a source of intrinsic noise to robustness from a different perspective. Unlike delays that follow the dynamics of the system or the actuator, action/observation noise is less predictable or possibly not at all. Therefore, a meaningful source of noise is used in the simulation that can be related to the same phenomenon in the physical system.

In the preliminaries of this work, the intrinsic stochasticity of real-time simulation (RT-IS) is found to be promising to provide less-heuristic solutions to domain randomization, although it has not been widely discussed in previous work. Real-time simulation is a special mode of off-the-shelf simulation software, such as PyBullet. The simulation step time in the real-time mode is stochastically changed due to the uncertain resource management of the operating system, which leads to the stochasticity of the entity states of the simulation, such as the movement trajectories of the robot. Such stochasticity is *intrinsic* since it is affected by the internal computing resources, such as CPU, GPU, and $Memory$ usage, similar to the natural stochasticity of physical robotic systems due to the change of temperature, humidity, and lubricant conditions [191].

This naturally leads us to the idea to exploit RT-IS to improve the robustness of a conventional RL agent and reduce the heuristics of a conventional domain-randomized agent. To evaluate the feasibility of this solution, the following three questions should be answered.

1. Does the utility of hardware resources affect RT-IS?

2. Is RT-IS comparable to the physical robot?

3. Can RT-IS be used to improve the robustness of RL and fill the sim-to-real gap?

The answers can help rediscover the functionality of RT-IS which is promising to promote the fidelity of robot simulation [192] and improve the robustness of RL methods [193].

This chapter attempts to fill the sim-to-real gap by answering the three questions above related to RT-IS, which has never been investigated by the existing research work. Specifically, through statistical studies, the cause of RT-IS is investigated in an off-the-shelf simulation environment. The similarity of RT-IS to the stochastic uncertainty of the physical robot is also validated. This provides the practical foundation for utilizing RT-IS to replicate the randomness of real robots and fill the sim-to-real gap. Then, a method is introduced that utilizes this randomization to render robust RL agents. Comparison studies of an essential point-to-point robot manipulation task are conducted to validate the equivalence between RT-IS and domain randomization for RL agent training.

The results of this chapter are promising to provide a more realistic and implementable solution for the practical applications of RL methods to robot manipulation tasks with the existence of sim-to-real gaps, i.e., utilizing RT-IS to train a robust RL agent instead of domain randomization, with fewer computation resources. The remaining part of the chapter is organized as follows. Section 5.2 introduces the experimental setup in simulation and on the physical robot for the data collection for the study of RT-IS. Section 5.3 analyzes the influence of CPU, GPU, and Memory consumption on the intrinsic stochasticity and evaluates the magnitude of stochasticity between the real-time simulation and the real robot. Section 5.4 explains the configuration of the experimental studies to evaluate the effectiveness of RT-IS. The experimental results are evaluated and analyzed in Section 5.5. Finally, Section 5.6 concludes the chapter.

5.2 Data Collection for Analytical Study

This section introduces the data collection process used for the correlation analysis. First, the experimental setup and the data collection in the real-time simulation environment are introduced. Then, the data collection process on the physical robot is introduced. The overall process is illustrated in Figure 5.2. The identical heuristic policy based on a constant-speed controller is applied to both the simulation and the physical robot. The simulation is performed in the real-time mode where the output $\tilde{V}$ is affected by RT-IS. The output of the physical robot V is affected by the physical stochastic uncertainties.

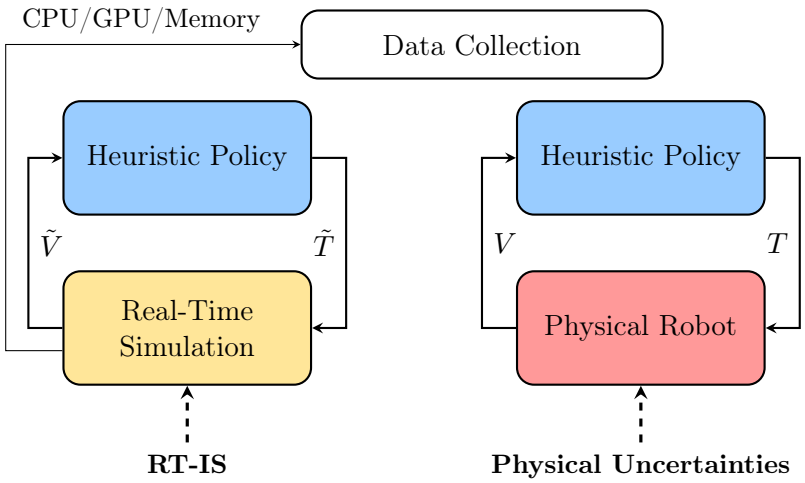


Figure 5.2: The illustration of the experimental setup and the data to be collected, where V and T are the joint velocity and torque of the physical robot, and $\tilde{V}$ and $\tilde{T}$ are those of the virtual robot in the simulation environment.

5.2.1 Real-Time Simulation Experimental Setup

PyBullet, an open-source simulator based on the Bullet physics engine for Python, is chosen as the simulation environment. PyBullet offers parallel simulation functionality, compatibility with Gym and other machine learning libraries, and low-demand and high-speed calculations that are favorable for reinforcement learning research. The stability and viability of PyBullet in different robotic applications (e.g., rigid body motion, locomotion, mobile robots, collision, and contact) are widely recognized [194]. PyBullet supports the real-time simulation mode based on the real-time calling mechanism of the operating system, which provides superior fidelity to reality but brings up stochastic uncertainty of simulation. Nevertheless, the real-time simulation mode of PyBullet is the main interest of this chapter. To run the simulation, a powerful workstation is used capable of training reinforcement learning agents with high computational capabilities. The workstation is equipped with an AMD Ryzen 9 5900X CPU, an Nvidia RTX 3090 GPU, and 64GB (2x32) Corsair DDR4 memory, which is superior to many papers on simulation benchmarks [195, 196]. Moreover, the operating system is Ubuntu 18.04.6 LTS. The purpose of this setup is to validate whether ordinary simulation tools on powerful computers are affected by minor changes in resource consumption.

A virtual model of a six-degree-of-freedom (6DoF) Kinova® Gen3 robot with a Robotiq® 2F-85 gripper is implemented in PyBullet with a Gym environment

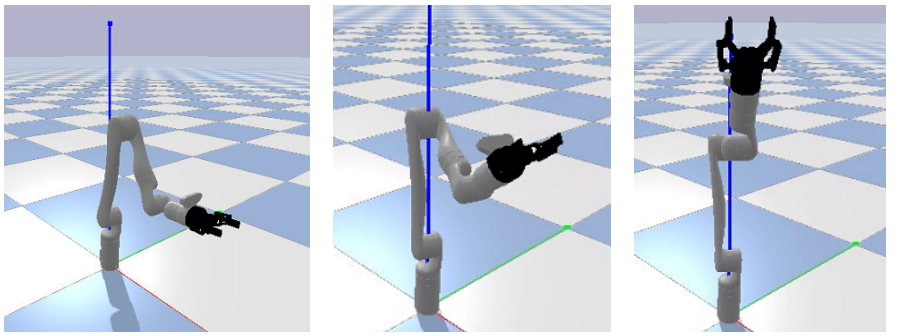
wrapper. The robot model is developed from the official URDF file provided by the manufacturer. The real-time simulation timesteps option is enabled using PyBullet's `setRealTimeSimulation()`. This feature, in contrast with discrete-time increments, does not deal with time as a deterministic parameter. Instead, the model is updated according to the machine's internal clock as the reference for timesteps. While deterministic timesteps are regularly used to take out unpredictability, this feature involves the stochastic nature of digital computation in the simulation. The simulated robot is manipulated in position control mode with a standard constant-coefficient Proportional-Integrator-Derivative (PID) controller. Implementing a common low-order controller guarantees a limited calculation. To restrict other sources of unpredictable behavior, all other kinematic and dynamic equations are solved in the forward direction (from joint space parameters to task space) and hence, needless of any random seed initialization.

5.2.2 Real-Time Simulation Experiment

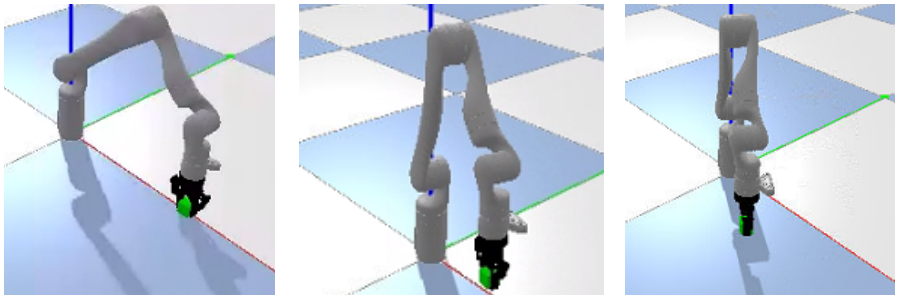
The intrinsic stochasticity in real-time simulation affects the trajectories and forces of the virtual robot models. To capture and measure the stochasticity, the virtual robot runs various robotic tasks repeatedly in simulation, and the trajectories are recorded. Three different tasks are used in data collection, namely point-to-point reaching (P2P), pick-and-place (P&P), and object-pushing (OP). These tasks have been chosen as simple and the most fundamental manipulation tasks that robotic arms are supposed to perform in any application [3]. With these three tasks, the computational burden on the system is gradually increased, and the correlation with hardware usage in multiple scenarios is evaluated. The procedures of the three tasks are illustrated in Figure 5.3, with the following detailed interpretation.

Point-to-Point Reaching (P2P)

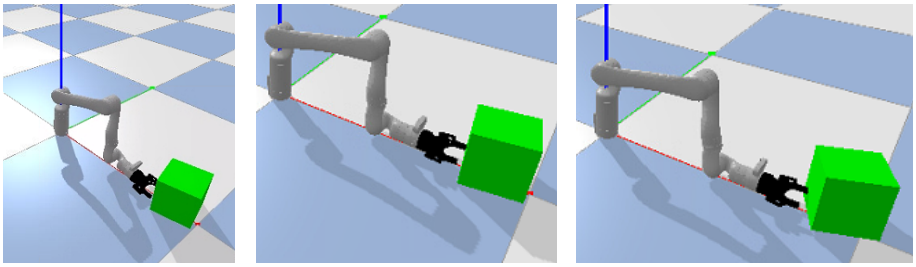
This task involves moving the first three joints simultaneously from the robot's home position, pre-calibrated by the manufacturer, which is consistent between the virtual and the physical system. In this configuration, joint angles are at $\left[0, \frac{\pi}{12}, \frac{23\pi}{18}, 0, \frac{11\pi}{36}, \frac{\pi}{2}\right]$ rad. In each iteration of moving the virtual or the physical robot for this experiment, constant $3 \times 10^{-3}\, deg$ position increment commands are sent per each $1\, ms$ to the first three joints for a trivial motion. Since zero joint acceleration inhibits the magnitude of actuator dynamics involved in the robot's behavior. After some preliminary tests,



(a) The P2P reaching task



(b) The P&P task



(c) The OP task

Figure 5.3: PyBullet simulation environments for the investigation of hardware utility effect on stochasticity.

the joint speeds are calibrated so the same speed is applied to the real robot and simulation. For feedback, all the signals mentioned in Section 5.4 are continuously logged. However, the velocity (V) and torque (T) signals are the most important, for they represent both kinematics and dynamics stochasticity. The next section will investigate if the overall amount of variation in each of these signals has any significant correlation with average resource utility.

Pick-and-Place Task (P&P)

This task follows the same logic, however, the starting position of the robot is defined so that the end-effector stands close to the green cube object in Figure 5.3b. The gripper grasps the object and then the second and third joint positions are moved by a constant speed of 0.3 rad/s for the first 2 seconds. Then, the base joint position moves in a similar manner. Finally, for the last 2 seconds, the second and third joints move toward the target position, then let go of the object while opening the gripper. Here, again, it is intended to keep dynamic complexity in the joint parameter space to a minimum. Instead, the complexity falls within the gripper function, the contact forces, and the dynamics of the object. The collected data is the same as before for this task.

Object Pushing (OP)

For the last experiment, a more complicated pushing task is designed. The manipulator starts from standing behind an object, with an open gripper. Then it starts moving in a straight horizontal line toward the object. The manipulator will keep on pushing after the collision while moving at the same speed and direction. Although this trajectory also seems simple in the task space, the translated 3DoF joint trajectories include accelerations and intricacies. Another detail that adds to the complexity, is the friction force between the object and the floor, which affects the joint torques through the contact force on the gripper. The collected data is the same as before for this task as well.

For each task, 50 trials are executed, and the robot output data including joint angles θ, joint velocities V, joint torques T, and Cartesian position of the end-effector $(X, Y, Z)^T$ are collected. Each task runs for ten seconds (or equivalent simulation time), consisting of 10,000 steps at 1 kHz. The robot data is recorded at the same rate. Only three out of six joints of the robot model are active to simplify the simulation and reduce data requirements. To evaluate the consumption of the computer hardware resources, the computer's real-time clock t^c and the utility of CPU, GPU, and $Memory$ are recorded, inspired by the previous work on the evaluation of robot simulation platforms [195, 194, 197]. The `psutil` Python library is used for collecting the CPU and memory activity, and the `nvidia-smi` Python API is used to collect the GPU utility. While running the simulations, no other tasks were active, so the variations could be explained only by the running simulation. The analysis of the

correlation between the stochastic robot motion and the hardware consumption is discussed in Section 5.3.

5.2.3 Physical Robot Experiment

In this chapter, the RT-IS is measured in a physical robot and compared with that of the real-time simulation. For this purpose, a Kinova® Gen3 robot with a Robotiq® 2F-85 gripper is set up, as in Figure 5.4, equivalent to the simulation models in Section 5.2.1. An off-the-shelf personal computer, equipped with Kinova® Kortex™ Python API, transmits commands and feedback with the hardware at a rate of 1 kHz. Noticeably, this is the highest rate supported by the feedback controller of physical hardware that indicates a response delay of less than 1 ms. The P2P task, the simplest of the tasks mentioned in Section 5.2.2, was assigned to the physical robot to show that even the simplest real-time simulation is projecting sufficient stochasticity. The reason behind implementing only P2P for the data collection on the physical robot is that P2P sufficiently shows the natural stochasticity of the physical robot. Also, P2P only involves the robot and no other external entities in the environment. This eliminates interference from any external dynamics, such as interactions with obstacles. The robot executes the same set of commands as the simulation configuration in Section 5.2.2 while working in the manufacturer's generic position control mode. The velocity (V) and torque (T) signals of the robot joints are continuously recorded since they sufficiently represent the stochasticity of robot dynamics. The data collected from the physical robot is used to compare it with its virtual counterpart, described in Section 5.2.1. The results will be presented in the following section.

5.3 Correlation Analysis of RT-IS

Here, the collected data to assess the correlation between computer hardware and intrinsic stochasticity in real-time simulation is analyzed. The results show that each of the three hardware benchmarks may play significant role, depending on the task nature. Then, a comparable level of stochasticity is observed between the physical robot and its simulation. The overview of the analytical studies is illustrated in Figure 5.5.

Figure 5.4: The physical experiment setup including the Kinova® Gen3 robot and the Robotiq® 2F-85 gripper

5.3.1 Definition of Signals and Variables

The outputs of the robot define the signal $s(t)$, which includes velocity and torque values for the three active joints. Note that the jth repeat of the experiment signal, $s_j(t)$, differs from other repeats due to the RT-IS functionality in PyBullet. Thus, an average signal is defined as,

$$\bar{s}(t) = \frac{1}{N_r} \sum_{k=1}^{N_r} s_k(t), \tag{5.1}$$

Where N_r is the number of trials of the repeated signals. Then, the following deviation signal is defined to represent the extent of stochasticity,

$$\delta_j(t) = s_j(t) - \bar{s}(t). \tag{5.2}$$

In addition to the average signal $\bar{s}(t)$, the variability band of the signal s is defined using the standard deviation of the repeated signal values among different trials $s_j(t)$,

$$\sigma(t) = \sqrt{\frac{1}{N_r} \sum_{j=1}^{N_r} (s_j(t) - \bar{s}(t))^2}. \tag{5.3}$$

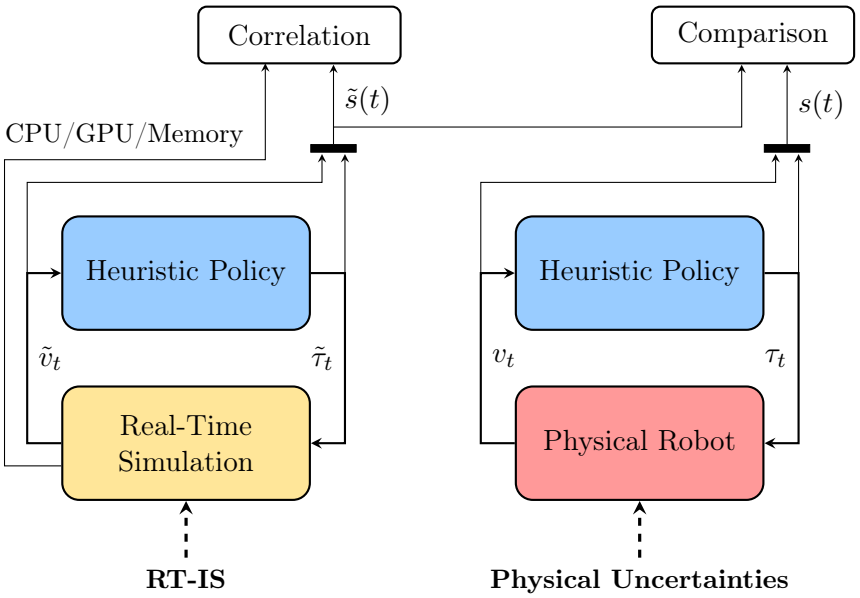


Figure 5.5: The illustration of the analytical studies on (1) the correlation between the simulation data and the hardware resource consumption and (2) the comparison between the stochasticity of the real-time simulation and the physical robot.

Also, Δ_j represents the overall estimate of stochasticity for jth repeat of signal $s(t)$ using the following root-mean-square (RMS) function,

$$\Delta_j = \sqrt{\frac{1}{N_t}\sum_{t=1}^{N_t}[\delta_j(t)]^2}, \tag{5.4}$$

Where N_t is the number of timesteps of each signal $s_j(t)$. Besides, the average consumption of computer hardware $CPU/GPU/Memory$ is calculated for each trial j,

$$CPU_j = \frac{1}{N_t}\sum_{t=1}^{N_t} CPU_j(t), \tag{5.5}$$

$$GPU_j = \frac{1}{N_t}\sum_{t=1}^{N_t} GPU_j(t), \tag{5.6}$$

$$Memory_j = \frac{1}{N_t}\sum_{t=1}^{N_t} Memory_j(t). \tag{5.7}$$

5.3.2 Stochasticity Correlation with Hardware Consumption

Now, Δ_j is a measure of the magnitude of the stochasticity in each trial j for all the signals $s_j(t)$. Then, Pearson Correlation Coefficient (PCC) will show its correlation with the average hardware consumption $CPU_j/GPU_j/Memory_j$ for every trial j. For instance, the correlation between the second actuator's torque Δ_j in P&P simulation and average GPU percentage GPU_j reads,

$$r^{P\&P}_{T_2,GPU} = \frac{\sum_{j=1}^{N_r}(\Delta_j - \overline{\Delta})(GPU_j - \overline{GPU})}{\sqrt{\sum_{j=1}^{N_r}(\Delta_j - \overline{\Delta})^2}\sqrt{\sum_{j=1}^{N_r}(GPU_j - \overline{GPU})^2}}, \tag{5.8}$$

where,

$$\overline{GPU} = \frac{1}{N_r}\sum_{j=1}^{N_r} GPU_j. \tag{5.9}$$

By using the degree-of-freedom $n_{df} = N_r - 2$, and the following t-score,

$$t^{P\&P}_{T_2,GPU} = \frac{r^{P\&P}_{T_2,GPU}\sqrt{n_{df}}}{\sqrt{1-(r^{P\&P}_{T_2,GPU})^2}}, \tag{5.10}$$

the p-value probability can be calculated as,

$$p(t^{P\&P}_{T_2,GPU}, n_{df}) = \frac{\Gamma(\frac{n_{df}+1}{2})}{\sqrt{n_{df}\pi}\,\Gamma(\frac{n_{df}}{2})}\int_{-\infty}^{t^{P\&P}_{T_2,GPU}}(1+\frac{x^2}{n_{df}})dx \tag{5.11}$$

with,

$$\Gamma(z) = \int_0^\infty u^{z-1}e^{-u}du. \tag{5.12}$$

The p-value expresses the probability of the hypothesis that a non-zero $r_{X,Y}$ could be in fact zero for a specific degree-of-freedom, considering the number of samples. Thus, a p-value of less than 5% shows a 95% confidence in the calculated correlation coefficient.

The average Δ_j values for three tasks in the simulation are shown in Table 5.1, which shows that the P2P task has the highest torque stochasticity, while P&P has the highest velocity stochasticity.

Also, for each simulation and signal, Δ_j is plotted against the average hardware consumption (e.g., CPU_j, GPU_j, or $Memory_j$) for that episode. Figure 5.6 is a sample plot for torque stochasticity versus hardware utility consumption correlation

Table 5.1: Average stochasticity ($\overline{\Delta}$) for velocity and torque signals in all three tasks

Joint	Signal	Task		
		P2P	P&P	Push
#1	Velocity	0.011	0.020	0.001
	Torque	2.630	6.947	3.451
#2	Velocity	0.011	0.029	0.007
	Torque	8.631	5.818	2.756
#3	Velocity	0.011	0.027	0.007
	Torque	5.500	4.565	1.546

in the P&P trials. According to the PCC method and p-values, a significant positive effect of GPU usage on stochasticity is remarked (in the black text boxes). PCCs and p-value for the three tasks, P2P, P&P, and OP, are respectively shown in Table 5.2, Table 5.3, and Table 5.4. The results show that, in P2P task, the only significantly important factor is CPU usage. A higher CPU percentage corresponds to more torque and less velocity stochasticity values for all actuators. As previously pointed out, P2P has the highest torque stochasticity. This is a result of moving the two biggest links of the manipulator and exerting higher torque levels on the actuators. Larger torque demand applied to high-speed computational power, forces the CPU to cause a larger magnitude of stochasticity.

Table 5.2: PCCs (and p-values) for P2P simulation stochasticity vs. hardware benchmarks

Joint	Signal	CPU	GPU	Memory
#1	Velocity	**0.439**	-0.128	0.221
		(0.001)	(0.375)	(0.123)
	Torque	**-0.447**	0.252	0.118
		(0.001)	(0.078)	(0.414)
#2	Velocity	**0.439**	-0.129	0.219
		(0.001)	(0.373)	(0.126)
	Torque	**-0.394**	0.245	0.250
		(0.005)	(0.086)	(0.080)
#3	Velocity	**0.441**	-0.130	0.220
		(0.001)	(0.370)	(0.125)
	Torque	**-0.433**	0.243	0.231
		(0.002)	(0.089)	(0.106)

Unlike the previous examination, GPU is proven to be the only important factor in the P&P scenario. Here, higher GPU usage is correlated with higher velocity and torque stochasticity in all three joints. In this scenario, which has the biggest velocity stochasticity among the three, motion is relatively faster and displacements, bigger. Therefore, the most influential factor in stochasticity is the GPU for its role in graphic visualization.

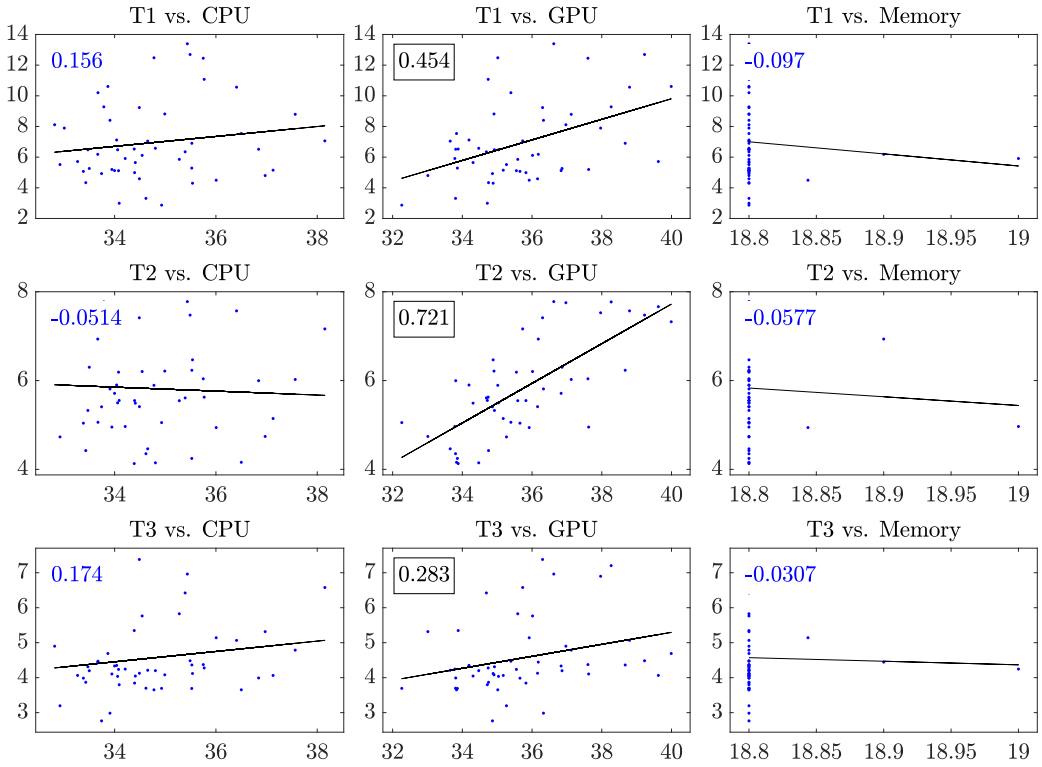


Figure 5.6: Correlation plots for RMS torque stochasticity vs. hardware consumption in P&P simulation

Finally, in the OP task, Memory usage overshadows other components (except for CPU impact on second actuator velocity) and shows significant relation with velocity stochasticity. This relatively complex scenario has in contrast the least amount of stochasticity compared to the other two. Yet, calculations in the OP task simultaneously consist of friction, contact force, object manipulation, and actuator dynamics. Therefore, the required calculation burden in each timestep is relatively bigger and consumes more *Memory*. Consequently, this factor would be the most responsible for stochasticity. It is also remarkable to see that only the velocity stochasticity is showing any significant relation to the hardware utility, due to the smaller forces and

Table 5.3: PCCs (and p-values) for P&P simulation stochasticity vs. hardware benchmarks

Joint	Signal	CPU	GPU	Memory
#1	Velocity	-0.250	**0.546**	-0.105
		(0.080)	**(0.000)**	(0.469)
	Torque	0.156	**0.454**	-0.097
		(0.279)	**(0.001)**	(0.503)
#2	Velocity	-0.159	**0.652**	-0.108
		(0.269)	**(0.000)**	(0.457)
	Torque	-0.051	**0.721**	-0.058
		(0.723)	**(0.000)**	(0.690)
#3	Velocity	-0.157	**0.627**	-0.114
		(0.278)	**(0.000)**	(0.432)
	Torque	0.174	**0.283**	-0.031
		(0.227)	**(0.047)**	(0.833)

torques involved in this scenario.

To sum up, it is concluded that depending on the task nature, each hardware component or benchmark might be influential in determining the magnitude of added stochasticity to virtual simulations. *CPU* tends to have the most important influence on stochasticity in the tasks with higher torques. *GPU* is likely to be influential when higher velocity and faster visualization are involved in the task. Finally, for complex tasks with multiple dynamic elements, *Memory* plays the main role in determining the stochasticity level.

5.3.3 Comparing the Stochasticity: Simulation vs. Real

In the next step, the stochasticity is compared between the real-time simulation and the physical robot. Figure 5.7 shows the normalized average velocity and torque of actuators, $\bar{s}(t)$, for both environments with solid lines. The shaded area corresponds to the variability range, $\sigma(t)$, of stochasticity. It is clear, particularly for torque signals, that simulation is noisier. Still, the power spectral density of all signals is quite similar (Figure 5.8) which indicates similar governing dynamic equations.

The RMS stochasticity value for all signals is also summarized in Table 5.5. It emphasizes the higher stochasticity value in the simulation environment. Considering these results, the intrinsic stochastic functionality of real-time simulation software

Table 5.4: PCCs (and p-values) for OP simulation stochasticity vs. hardware benchmarks

Joint	Signal	CPU	GPU	Memory
#1	Velocity	0.038	-0.098	**0.406**
		(0.797)	(0.502)	**(0.004)**
	Torque	0.087	-0.130	-0.007
		(0.552)	(0.374)	(0.965)
#2	Velocity	**0.394**	-0.059	**0.415**
		(0.005)	(0.688)	**(0.003)**
	Torque	0.109	-0.103	0.013
		(0.457)	(0.481)	(0.928)
#3	Velocity	0.280	-0.042	**0.423**
		(0.051)	(0.773)	**(0.002)**
	Torque	0.125	-0.107	-0.002
		(0.394)	(0.465)	(0.991)

shows potential for improving simulation fidelity. If an RL agent learns to perform well in such stochastic simulation environments, its performance is less susceptible to deterioration when implemented on physical systems with intrinsic stochasticity [179, 181]. This also proposes a measure of randomness for tuning the portion of domain randomization in increasing simulation stochasticity. In many cases, stochastic simulation environments like the real-time PyBullet simulation investigated in this chapter may substitute heuristic domain randomization because of their trivial and unbiased nature.

Table 5.5: The RMS values of the stochasticity ($\overline{\Delta}$) in the P2P task (simulation and real-world)

Joint	Signal	Simulation	Real-World
#1	Velocity	0.629	0.275
	Torque	2.640	0.136
#2	Velocity	0.629	0.263
	Torque	8.592	0.245
#3	Velocity	0.629	0.266
	Torque	5.476	0.209

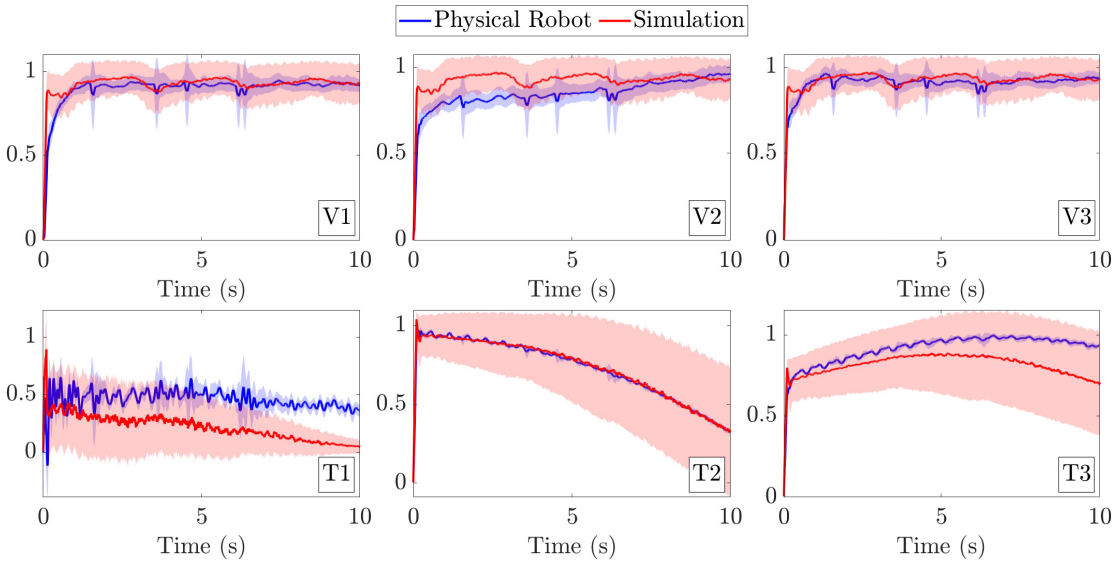


Figure 5.7: Velocity and torque signal (average and deviation) for P2P task

5.4 Training and Test of RT-IS Powered Agents

This section explains how the RT-IS is utilized to enhance the robustness of an RL agent. Firstly, an essential point-to-point (P2P) reaching robot task is formulated as a Markov Decision Process (MDP). Then, the implementation of domain randomization and RT-IS in the training process of an RL agent is explained. Finally, the simulation and experimental configuration for the test studies are laid out.

The overview of the training and test process is illustrated in Figure 5.9. The codebase used for the experiments described in this section is publicly available.[1]

5.4.1 Robot Task Formulation

The main objective of this chapter is to provide RL methods for robotic manipulation tasks with a proposed technology for easy sim-to-real transfer. It should be kept in mind that domain randomization is highly computationally expensive. Thus, a simple but essential robot task, a P2P reaching task, is used as the benchmark of this chapter. A P2P reaching task requires the robot to start from an initial joint position and move until its end-effector reaches a desired Cartesian position. This robot task is formulated as an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_S, \mathcal{P}_R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are respectively the state and action spaces of the robotic system, $\mathcal{P}_S : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

[1] https://github.com/amsoufi/RL_Sim2Real_RealTimeSim

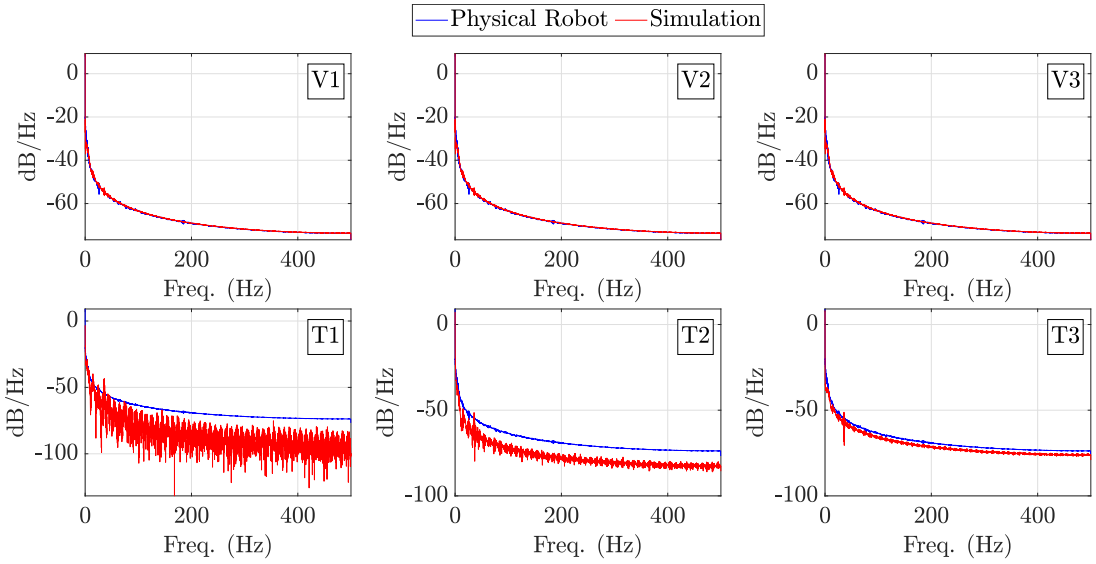


Figure 5.8: Power spectral density of velocity and torque signals for P2P task

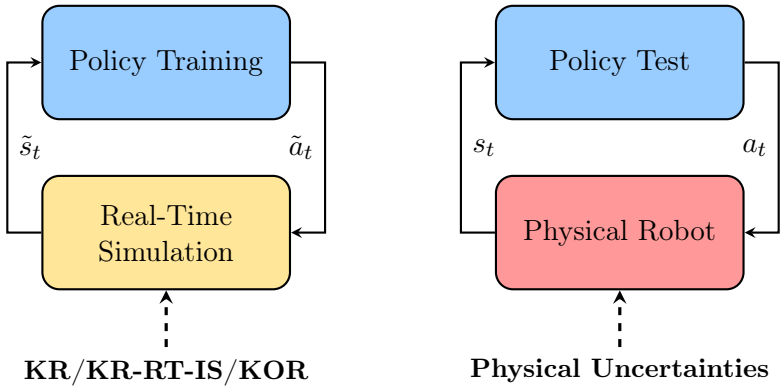


Figure 5.9: The illustration of the training and test of the agents, where $\tilde{s}_t$ and $\tilde{a}_t$ are the state and action of the agent in simulation, at certain time t, and s_t and a_t are those of the agent for the physical robot. Various RL agents are trained with whether the Kinematics Randomization (KR), the Kinematics Randomization Powered by RT-IS (KR-RT-IS), or the Kinematics-Observation Randomization (KOR) activated. The trained agents are directly tested on the physical robot affected by the physical uncertainties to evaluate their generalizability.

represents the unknown state transition model of the system, $\mathcal{P}_R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function determined according to the specific task, and $\gamma \in (0, 1]$ is the discounting factor that balances between the current and the future rewards. The goal of the MDP problem is to solve the optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ such that the accumulated reward is maximized.

For an n-degree-of-freedom (DoF) robot manipulator, the observation of $\mathcal{M}$, $s_t \in \mathcal{S}$, at time t, is defined as

$$s_t = [\, P_t^{\mathrm{e}},\, \Theta_t,\, \sin(\Theta_t),\, \cos(\Theta_t),\, \mathbf{v}_t,\, P_{\mathrm{g}},\, D \,], \tag{5.13}$$

where $P_t^{\mathrm{e}} \in \mathbb{R}^3$ is the Cartesian position of the robot end-effector, $\Theta_t \in \mathbb{R}^n$ is the vector of the angles of the controllable robot joints, $\sin, \cos : \mathbb{R}^n \to \mathbb{R}^n$ are element-wise trigonometric functions of the robot joints, $\mathbf{v}_t \in \mathbb{R}^n$ is a vector of the joint velocities of the controllable joints, $P_{\mathrm{g}} \in \mathbb{R}^3$ is the Cartesian position of the goal, and $D \in \{\, \text{true, false} \,\}$ is a binary signal indicating whether a trajectory has terminated. The action $a_t \in \mathcal{A}$ is selected as

$$a_t = \mathbf{v}_t^{\mathrm{d}}, \tag{5.14}$$

where $\mathbf{v}_t^{\mathrm{d}} \in \mathbb{R}^n$ is the desired or commanded joint velocity vector applied to the active joints at the current time step t. These observations provide enough relevant information for an agent to learn from. The reward function is

$$r_t = -R_1 E_t^{\mathrm{e}} - R_2 \|a_t\|^2 \tag{5.15}$$

where $E_t^{\mathrm{e}} = \|P_t^{\mathrm{e}} - P_{\mathrm{g}}\|$ is the Euclidean distance between the end-effector and target positions, and $R_1, R_2 \in \mathbb{R}^+$ are gain hyperparameters to be heuristically determined. In the reward function (5.15), the term with the reaching error E_t^{e} is for the achievement of the P2P task. Also, the action a_t is penalized to avoid dangerous actions that may be damaging to the real robot, such that the RL agent learns to move the robot at reasonable velocities and smoother trajectories. It is also worth noting that the relative value R_1/R_2, rather than the absolute values of the gains R_1 and R_2 are important. Finally, the terminal condition flag D is determined as

$$D = \begin{cases} \text{true}, & \text{if } t > N \text{ or } E_t^{\mathrm{e}} < \varepsilon \\ \text{false}, & \text{otherwise} \end{cases} \tag{5.16}$$

where N is a positive integer that indicates the maximum length of an episode and $\varepsilon \in \mathbb{R}^+$ is the threshold of the reaching error E_t^{e}.

The proposed approach to sim-to-real transfer can be modeled by this MDP formulation to explain the underlying theory. First, to better identify the differences in the formulation, two existing solutions for the sim-to-real problem are modeled. In *domain adaptation* [198], the stochasticity of the source MDP (the simulation environment) is formulated as a probability distribution $\mathcal{P}_{source}$ for the governing dynamics,

$$s_{t+1} \sim \mathcal{P}_{source}(s_{t+1}|s_t, a_t). \tag{5.17}$$

In the same domain, a stochastic policy π with the learned parameters θ_{source} is trained such that it maximizes the cumulative discounted reward,

$$a_t \sim \pi(a_t|s_t; \theta_{source}). \tag{5.18}$$

After the transfer, a new distribution $\mathcal{P}_{target}$ is identified for the target environment dynamics (the physical system). Consequently, the policy parameters are mapped $\theta_{source} \mapsto \theta_{target}$ to compensate for the influence of the domain shift,

$$\xRightarrow{\text{Transfer}} \begin{cases} s_{t+1} \sim \mathcal{P}_{target}(s_{t+1}|s_t, a_t), \\ a_t \sim \pi(a_t|s_t; \theta_{target}). \end{cases} \tag{5.19}$$

Alternatively, in *domain randomization* [199], the dynamics $\mathcal{P}$ is modeled as a conditioned probabilistic function of the environment characteristics $\psi \in \mathbb{R}^{n_\psi}$ with n_ψ parameters which may include inertia, structural parameters, friction coefficients and time delays in robotics,

$$s_{t+1} \sim \mathcal{P}_\psi(s_{t+1}|s_t, a_t, \psi). \tag{5.20}$$

Note that the policy is also conditioned to the model characteristics,

$$a_t \sim \pi_\psi(a_t|s_t, \psi; \theta). \tag{5.21}$$

ψ is frequently sampled from $p(\psi)$ in between simulation experiences to increase the robustness of policy to multiple configurations of the model. If the target system's characteristics ψ_{target} is sampled enough items during the training, the transfer would

be straight-forward,

$$\xrightarrow{\text{Transfer}} \begin{cases} s_{t+1} \sim \mathcal{P}_{\psi_{target}}(s_{t+1}|s_t, a_t, \psi_{target}), \\ a_t \sim \pi_{\psi_{target}}(a_t|s_t, \psi_{target}; \theta), \end{cases} \tag{5.22}$$

otherwise, further training is necessary.

In the RT-IS approach to domain randomization, the dynamic calculations of the domain-randomized environment follow a stochastic time difference instead of the deterministic formulations. This timestep variability will consequently affect all dynamic calculations. This can be modeled by incorporating uncertainty to both state observation and action, or by conditioning the dynamics to $\tilde{\delta t} \sim p(\tilde{\delta t})$,

$$s_{t+1} \sim \mathcal{P}(s_{t+\tilde{\delta t}}|s_t, a_t, \tilde{\delta t}, \psi) \;\rightarrow\; s_{t+1} \sim \tilde{\mathcal{P}}(s_{t+1}|\tilde{s}_t, \tilde{a}_t, \psi) \tag{5.23}$$

A policy can be trained using the same stochastic signals,

$$a_t \sim \pi(a_t|s_t, \tilde{\delta t}, \psi; \theta) \;\rightarrow\; a_t \sim \pi(a_t|\tilde{s}_t, \psi; \theta). \tag{5.24}$$

After the transfer, time stochasticity still exists in the dynamics. However, it will follow a different distribution $\delta t \sim p(\delta t)$.

$$\xrightarrow{\text{Transfer}} \begin{cases} s_{t+1} \sim \mathcal{P}(s_{t+\delta t}|s_t, a_t, \delta t, \psi_{target}), \\ a_t \sim \pi(a_t|s_t, \delta t, \psi_{target}; \theta). \end{cases} \tag{5.25}$$

It can be argued that if the target stochasticity is present in the source distribution during training, the policy will be robust to the sim-to-real transfer.

5.4.2 Configuration of Domain-Randomized Agents

Domain Randomization is the fundamental approach used to facilitate robust sim-to-real for RL. In brief, domain randomization perturbs the simulation model of the environment and performs agent training on all incorporated models. This argues that utilizing the RT-IS can improve the feasibility and performance of straightforward domain-randomized RL agents. To verify this, multiple RL baseline agents are set up to conduct the studies. The aim is to emphasize that even for basic baseline agents, enabling RT-IS can automatically improve the sim-to-real compatibility.

- *Kinematics-Randomized Agent (KRA):* A KRA is powered by the heuristic randomization in the robot kinematic model and trained in the non-real-time, or deterministic simulation mode. Firstly, an imprecise kinematic model for the robot simulation is built to create the sim-to-real gap manually. Then, the parameters of the robot's kinematic model are sampled according to a specified distribution (randomization). The range of the randomization is heuristically determined. Finally, the KRA is trained over all the randomized models using the domain-randomization algorithm [119].

- *Kinematics Randomized Agent Powered by RT-IS (KRA-IS):* The configuration of KRA-IS is the same as the KRA, except that it is trained in the real-time simulation mode. With KRA-IS, the intention is to show the advantage of RT-IS in the application of sim-to-real in RL.

- *Kinematics-Observation Randomized Agent (KORA):* Similar to KRA, KORA also performs randomization on the kinematic model of the robot. It is trained in the non-real-time simulation mode. Other than that, stochastic noise was manually added to the observation s_t of the agent to impose randomization in the observation domain. The involvement of this agent is to verify whether a wider range of domain randomization contributes to better robustness.

Apart from the above-mentioned domain-randomized agents, the following non-randomized agents are also configured for comparison studies.

- *Non-randomized Agent with Precise Robot Kinematic Model (NA-P):* The NA-P is a conventional Proximal Policy Optimization (PPO) [139] agent without any randomization, trained in a non-real-time simulation environment using a precise robot kinematic model. This agent does not assume the existence of the sim-to-real gap, which serves as the most straightforward problem formulation for robot manipulation.

- *Non-randomized Agent with Imprecise Robot Kinematic Model (NA-I):* The NA-I agent is a conventional RL agent without any randomization, trained in a non-real-time simulation environment using an imprecise robot kinematic model. The imprecise model is the same as the one used to train KRA. The NA-I admits the existence of the sim-to-real gap but attempts to resolve the issue conventionally. It indicates the performance of the conventional RL agents that are sensitive to the sim-to-real gap.

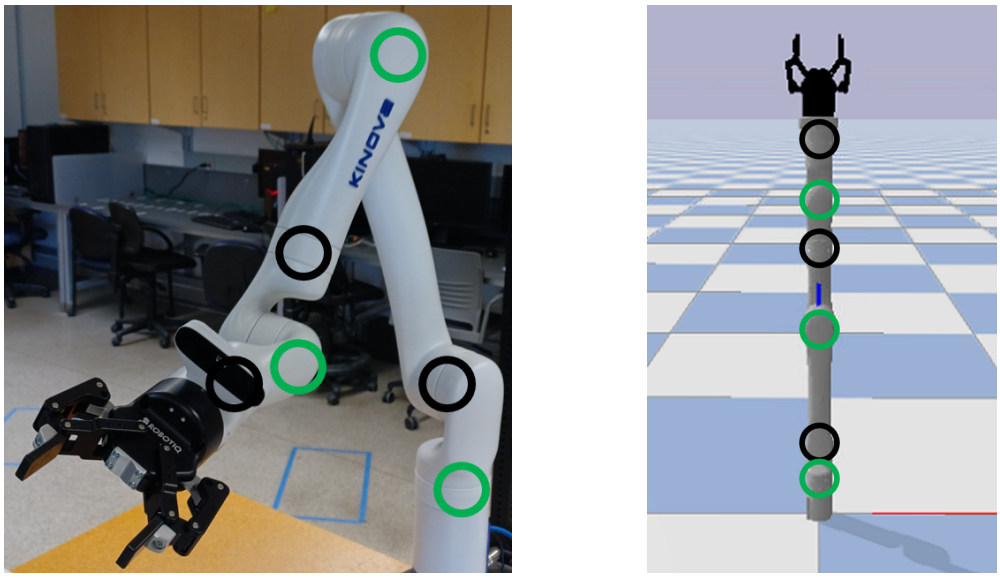
The PPO algorithm is used as the baseline of all agents since it is easy to implement, stable, and sample efficient. All agents are constructed as an actor-critic structure. Both the actor and critic are fully-connected forward neural networks with two hidden layers. The numbers of neurons in the first and the second layers are respectively 128 and 64. The training process of the agents is introduced in the following subsection.

5.4.3 Agent Training

The agents are trained in PyBullet [200] simulation environment which is an open-source platform that is compatible with the Open-AI Gym library and python RL libraries. PyBullet is an off-the-shelf and open-source simulation platform popularly used to simulate mechanical systems. It is powered by the Bullet physical engine and provides a powerful Python Interface that can be seamlessly called by the Open-AI Gym libraries. PyBullet provides a real-time simulation mode, enabled by the *setRealTimeSimulation* method, that steps the simulation using the clock that depends on the operating system. The real-time simulation mode has not attracted much attention in RL since it brings uncertainties to the system's motion. However, believe that this mode is promising to enhance the robustness of conventional RL agents.

In this chapter, a 6-DoF Kinova® Gen3 robot is used, as shown in Figure 5.10a, to conduct the experimental studies. The Kinova® Gen3 lightweight robot has been widely applied to various research and engineering scenarios due to its extendability and versatility. The simulation model of the Kinova® Gen3 robot in PyBullet is shown in the right side of Figure 5.10. The kinematic parameters of the robot are obtained from Kinova® official website [201]. To simplify the problem and to emphasize the effectiveness of domain randomization, only joints $\#1$, $\#3$, and $\#5$ of the robot are active and controllable, while the remaining joints are manually fixed at zero position, as shown in Figure 5.10, to reduce the dimensions of the action and state spaces, which leads to a 3-DoF robot model. Although the DoF of motion of the end effector is thus limited to three, it is sufficient for the P2P reaching task while ensuring the generalizability of the method.

For all training episodes, the base of the robot is spawned at the origin of the environment, $[\,0\;0\;0\,]^\top$, and the robot is initialized at the *HOME* position, i.e., $\Theta_0 = 0$. The target is set at the Cartesian position $P_g = [\,0.4\;\;0.2\;\;0.5\,]^\top$ measured in meters from the base of the robot. The KRA, KRA-IS, and KORA agents are trained in the



(a) Real-robot (b) Simulated-robot

Figure 5.10: The 6-DoF Kinova® Robot and its PyBullet simulation model used for the experimental studies in this chapter. The green and black circles represent the controllable and the manually-fixed joints, respectively .

following manner.

- The KRA and the KRA-IS are both trained by randomizing the domain of the kinematic model. Specifically, the length of both the first and second links of the robot arm is sampled from uniform distribution among 100%±1% of the original model. The range of the kinematic model randomization is kept small but sufficient to cover the possible difference between the simulation environment and the real world.

- The KORA is subject to the same kinematic model randomization as KRA and KRA-IS. In addition, its observation domain is also randomized with noisy signals added. The amplitude of the noisy signal is determined as no more than 5% of the robot joint velocity.

- The NA-I is trained using an imprecise robot kinematic model in simulation. Compared to the original robot kinematic model, the first link is 0.04% shorter and the second link is 0.04% longer. The NA-P is trained using the precise kinematic model of the robot arm.

The hyperparameters of the reward (5.15) are assigned as $R_1 = 2 \times 10^{-5}$, $R_2 =$

10^{-6}. The threshold of the reaching error is determined as $\varepsilon = 0.05\,\text{m}$. The training for all agents was run for 1000 epochs. For each agent, the policy is updated after each epoch which contains 5 episodes. Each episode contains 200 steps, i.e., $N = 200$. The discrete sampling time of the training process is $25\,ms$ which is sufficiently bigger than the actuator delay of $1\,ms$. The configuration of the test study is interpreted in the next subsection. The training results are discussed and analyzed in Section 5.5.

5.4.4 Test Studies

Before implementing the trained agents on the real-world robot, test studies are needed to verify the generalizability and applicability of the agents. There are two main reasons for the test studies, firstly, to ensure that the agents perform the task correctly, and secondly, to test and compare the performance of the trained agents. Testing in simulation ensures that the agents have learned to execute the task safely and reliably. Moreover, the test performance of the agents gives us insight into how they may perform in the real world.

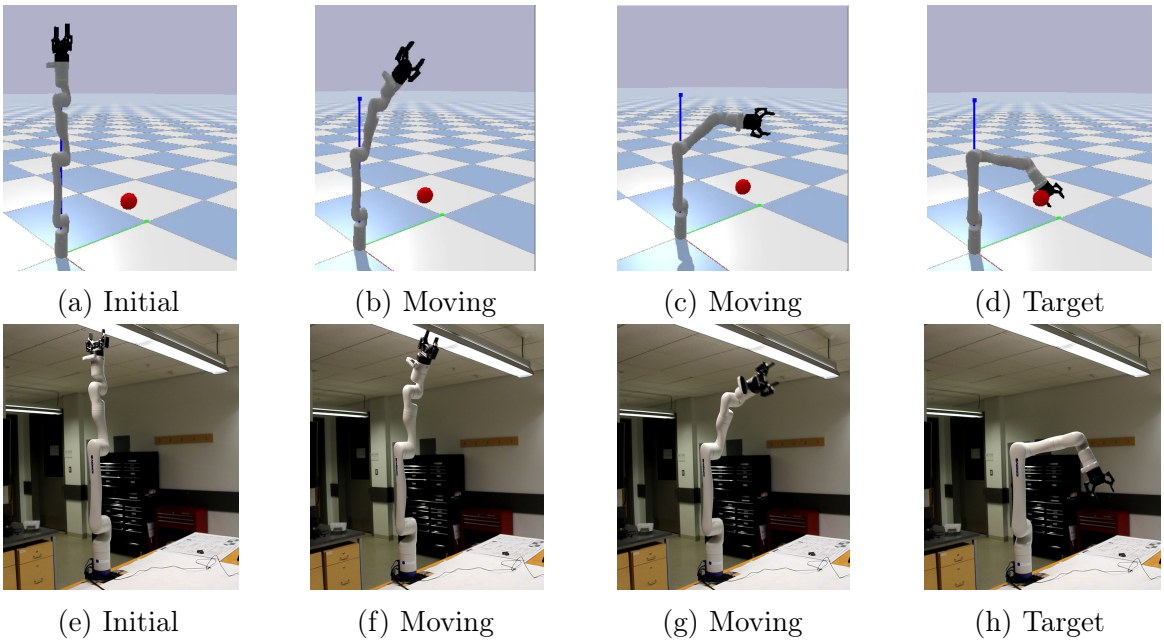
Both tests in the PyBullet simulation and on the real robot platform are conducted. In all test studies, the robot starts from zero joint position. The simulation test is conducted in the non-real-time mode to avoid simulation uncertainties. To perform reasonable result analysis, 500 test trials are executed in the simulated environment. For the hardware test, 50 test trials are recorded for each agent. The examples of test trials in simulation and on the robot hardware are presented in Figure 5.11. The Analysis and discussion of the test results are presented in Section 5.5.

5.5 Result Discussion

This section evaluates the training and test results of the agents introduced in Section 5.4.2. Firstly, the training performance of the agents is presented based on their learning curves and the associated numerical metrics. Then, the results of the agents in both simulation and hardware experiment test studies are discussed.

5.5.1 The Training Performance

The learning curves of the agent, as the interactions increase, offer the main metrics to evaluate the agent training, as shown in Figure 5.12. Based on this, four additional numerical metrics, as follows, are also used.



(a) Initial (b) Moving (c) Moving (d) Target

(e) Initial (f) Moving (g) Moving (h) Target

Figure 5.11: The motion of the robot in simulation (a)-(d) and hardware (c)-(h) test studies. The figures respectively show the robot starting from the initial position ((a) and (e)), moving towards the target position ((b), (c) and (f), (g)), and the robot reaching the target position ((d) and (h)).

- $\mathcal{R}_{\text{ini}}$: the initial reward calculated by averaging the instant reward after the initial 10% of epochs, which shows the performance of the initial policies;
- $\mathcal{R}_{\text{ult}}$: the ultimate reward calculated by averaging the instant reward over the final 10% of epochs, which depicts the performance of the trained policies;
- $\mathcal{T}_{\text{hlf}}$: the half-trained time, a percentage of the episode at which the accumulated reward first exceeds 50% of $\mathcal{R}_{\text{ult}}$
- r_{time}: the ratio of the training time (based on the wall clock time) to the training time of the NA-P agent

The values of the numerical metrics in the training process are shown in Table 5.6. The main phenomenon noticed in Figure 5.12 is how domain randomization affects the training of RL agents. More noise is witnessed on the learning curves of the domain-randomization agents, including KRA, KRA-IS, and KORA. This is because all these agents are trained over various robot and environment models. The attempt to perform training over multiple environmental models means the sacrifice of the stability of the training process. Also, the KRA-IS, dominated by not only

Table 5.6: Numerical Metrics of the Agent Training

Agent	$\mathcal{R}_{\text{ini}} \times 10^3$	$\mathcal{R}_{\text{ult}} \times 10^3$	$\mathcal{T}_{\text{hlf}}$	r_{time}
KRA-IS	-2.33	-1.21	10.2%	2.03
KRA	-2.71	-0.85	21.5%	1.57
KORA	-3.42	-0.91	26.9%	1.54
NA-P	-2.32	-1.09	12.4%	1.00
NA-I	-1.43	-0.84	9.3%	1.54

the manual model randomization but also the RT-IS, achieves the lowest ultimate reward, although the inferiority is insignificant. This is reasonable to understand since it involves the highest extent of randomization over other RL agents. Similar information can also be inferred from Table 5.6 that domain randomization agents KRA-IS, KRA, and KORA have lower ultimate reward $\mathcal{R}_{\text{ult}}$ and reward increment $\mathcal{R}_{\text{inc}}$. Besides, the learning curves of these agents show a slower convergence rate than NA-I and NA-P, which is also reflected by larger values of $\mathcal{T}_{\text{hlf}}$ metric in Table 5.6. Overall, domain randomized agents tend to have inferior training performance than the non-randomized ones due to the sacrifice for generalization. The next subsection will evaluate whether the performance tradeoff paid off for the test studies.

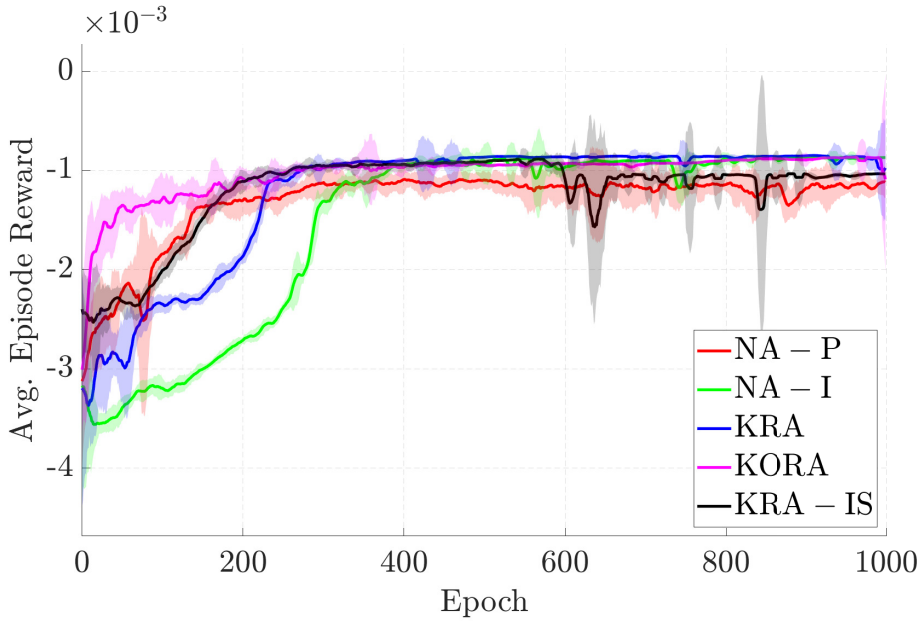


Figure 5.12: Training results of all agents (Average Episode Reward vs. Epoch for three random seeds per agent)

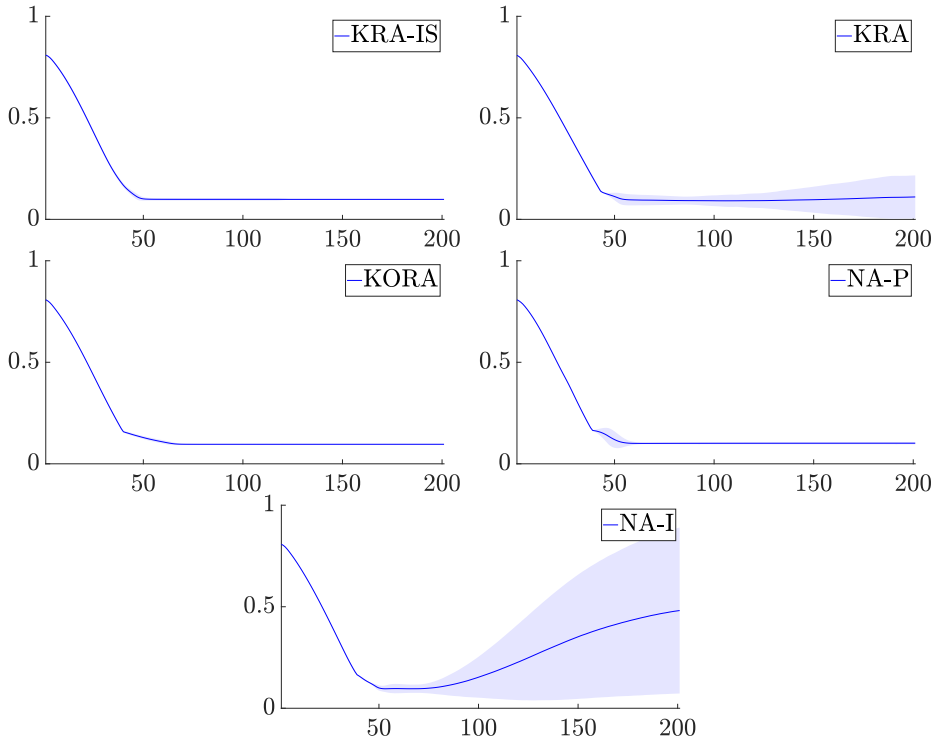


Figure 5.13: The average reaching error and its standard deviation in trials versus time step in the simulation test.

5.5.2 The Test Study in Simulation

The simulation test trials of the robot end-effector reaching error E_t^{e} of the five agents in Section 5.4.3 are illustrated in Figure 5.13. The lines in the figure show the averaged reaching error E_t^{e} over all 500 trials and the shadow region depicts the standard deviation over all trials. It is noticed that the NA-I agent achieves the worst test performance reflected by its large average error (the dark blue line) and large variance (the shadow blue region). This indicates the bad robustness of the conventional RL agent with the existence of kinematic model deviations. Meanwhile, the NA-P agent shows a small ultimate reaching error with a very small variance. This can be recognized as a baseline representing the highest possible test performance of a conventional RL agent since it is trained and tested on an identical model. It is also noticed that both KRA-IS and KORA agents achieve similar test performance to NA-P, which indicates that both of them are sufficiently robust to the model deviations to provide an equivalent effect to the high-performance baseline NA-P. Besides, the KRA agent, with the less extent of randomization compared to KRA-IS and KORA, has a larger variance and a slightly bigger ultimate reaching error.

Note that KRA-IS and KORA are subject to the same extent of randomization in the kinematic domain. Their difference is that KRA-IS has additional randomization in the time domain driven by the RT-IS, while KORA is additionally randomized in the observation domain. This verifies that the robustness of KRA-IS is due to the additional randomization exerted on the time domain. Thus, the simulation test study conducted in this subsection indicates the superior robustness of the KRA-IS agent compared to the conventional RL agent (NA-I) and the conventional domain-randomized RL agent (KRA). Its advantage over the enhanced domain-randomized RL agent KORA will be reflected by the hardware experiment study in the next subsection.

5.5.3 The Test Study on Robot Hardware

The test study on the robot hardware is confined by the limited time resource and the safety requirements. Therefore, 50 trials are performed for the hardware test, and two numerical metrics are defined, namely the ultimate reaching error E_N^{e} and the success rate ρ_{s} to evaluate the test performance. The ultimate reaching error E_N^{e} is the value of E_t^{e} at the ultimate time $t = N$, in meters. The success rate is defined as $\rho_{\mathrm{s}} = N_s/N_{\mathrm{ttl}}$ in percentage, where N_s is the number of trials of which the ultimate error $E_N^{\mathrm{e}} < \varepsilon$, and $N_{\mathrm{ttl}} = 50$ is the total number of the trials. The values of the numerical metrics of the 50 test trials on robot hardware are presented in Table 5.7.

Table 5.7 shows the obvious advantage of the KRA-IS over the rest of the agents in terms of both the success rate and the ultimate reaching error. It achieves the highest success rate of 94% and the lowest reaching error of 0.0367 m. Note that its performance is followed by NA-P, the conventional RL agent trained on the precise robot model, which is expected to present good test performance on the real robot, with a decent success rate of 80% and reaching an error of 0.0432 m. The fact that NA-P loses performance compared to KRA-IS indicates that the latter has better generalizability than the conventional RL agents. Even though the model used to train NA-P is from Kinova®and represents the most precise identification of the robot system, sim-to-real uncertainties still exist, which are not reflected by the simulation test study. This comparison study shows that KRA-IS has superior robustness compared to the conventional domain-randomization approach.

It is expected that the NA-I trained using an imprecise robot model performs the worst among all agents with zero success rate and the largest ultimate error E_N^e.

This reflects the conventional RL agent's lack of robustness. It is worth mentioning that the mismatch was included in the randomization range of KRA and KORA (Section 5.4.2), and both agents worked relatively better than NA-I. Nevertheless, it is surprising that the domain-randomized agents, KRA and KORA, also achieve zero success rate, even though they have smaller ultimate errors than the NA-I. This means that the ultimate reaching errors of all their trials are larger than the predefined threshold of 0.05 m. This is believed to be due to the improper extent of randomization, which makes the agents not robust enough to the sim-to-real uncertainties. Also, it is noticed that KORA, which has additional randomization in the observation domain than KRA, has a smaller ultimate error 0.069 m than the latter 0.092 m. This validates the superiority of KORA over KRA due to its better generalizability. It should be noted that fine assembly may need additional calibration or hybrid visual servoing compared to the chosen 0.05 m success threshold, but it clearly shows the reliability of an RL algorithm application in motion planning.

Table 5.7: Numerical Metrics of the Test on Hardware

Agent	KRA-IS	KRA	KORA	NA-P	NA-I
ρ_{s}	94	0	0	80	0
E_N^{e}	0.037	0.092	0.069	0.043	0.153

Therefore, it is concluded that involving randomization or stochasticity helps improve the robustness of the agent against the sim-to-real uncertainties. Naturally, the stochasticity of the simulation hardware is case-specific and depends on the configuration of the setup. Therefore, to further guarantee the efficiency of RT-IS, using a wider range of computational hardware is necessary. However, this was unfortunately unavailable to the authors at this point. In addition, more comprehensive task definitions for manipulation scenarios are also needed to better support the conclusion here. However, to what extent the model domains should be randomized is a tricky problem and is difficult to determine. If not properly determined, the randomized agents could show inferior performance, like KRA and KORA. However, utilizing the RT-IS greatly improves the robustness of the agent. The presented comparison study indicates that it is even more robust than the conventional RL agent trained using the precise robot model. Since the real-time simulation mode mainly randomizes the time domain, it is also concluded that the time domain randomization is more

effective than the domains of kinematics and observation, a feature also present in methods such as in [189].

5.6 Summary and Implications

This chapter investigates the stochastic property of the intrinsic stochasticity of real-time simulation (RT-IS) and explores the feasibility of utilizing it to facilitate the transfer of an RL robot controller from simulation to the real world. Significance tests in multiple robot tasks have verified their correlation with the hardware computation components. The variability range and the RMS stochasticity value are also assessed and quantified as opposed to those of a real-world robot. At this stage, it was concluded that the computational hardware utility is correlated to the real-time simulation stochasticity. Then, the stochastic real-time simulation (RT-IS) was used in an RL framework for an essential point-to-point (P2P) task. Three basic domain randomized agents are designed for this experiment, one of which is powered by RT-IS. Compared to the other agents, the RT-IS-powered agent achieves the highest success rate and lowest reaching errors. This shows that by enabling RT-IS in RL training, a straightforward domain-randomized model can readily be transferred to real environments. Future work in this regard should be conducted to verify this phenomenon for more advanced methods and baselines. At this stage, it is shown that stochastic simulation improves generalizability and eases the sim-to-real. The conclusions can be summarized in these two points:

1. Considering both stages of the experiments, taking advantage of the simulation's computational hardware is beneficial to the robustness of RL agents. This is a physically plausible source of noise in simulation engines that emulates the physical setup's noise. The correlation between computational hardware and RT-IS suggests a plausible origin for the stochasticity in the simulation environment. It also shows a promise for future work to manipulate the hardware resources such that RT-IS will be exploited at a favorable stochasticity level.

2. Even independent from the role of computational hardware, RT-IS can help train RL agents to be less sensitive to sim-to-real, with fewer heuristics, no task-dependency, and better generalizability. Activating this feature is straightforward and does not require additional design or manipulation by an expert.

> To the best of the authors' knowledge, this level of automation for such improvement stands out from the related literature.

The price of such generalizability is the decreased training performance, which, however, is a worthwhile trade-off in most practical applications. Also, while the proposed method successfully produces a robust RL agent, it is important to note that using real-time simulations for training RL methods can be more time-consuming than using non-real-time simulations. This is because non-real-time simulations allow for the specification of time steps that can significantly speed up a simulation by huge magnitudes. More efficient methods to train RL agents powered by RT-IS will be investigated in future work. The application of these agents to more complicated robot manipulation tasks should be explored.

Chapter 6

Action-Free Transformer-Based Belief Model for Context-Adaptive Meta-RL

Along the same line as the previous two chapters, this chapter presents an independent but complementary module in the larger outlook described in Chapter 1. As pointed out previously in this dissertation, three challenges can be identified in adaptive RL for robotics, namely *sample-efficiency*, *sim-to-real*, and *generalization*. The previous two chapters presented practical remedies to mitigate the first two concerns. Here in this chapter, a novel variational belief model for task inference in a Bayes-adaptive meta-reinforcement learning framework is presented for improved adaptability in robot manipulation, tackling the third challenge. The presented method is **C**ontext **R**epresentation via **A**ction-**F**ree **T**ransformer encoder-decoder (CRAFT), a belief model that infers latent task structure from action-free sequences of observations and rewards, unlike conventional approaches that rely on full trajectories. This design supports modular, policy-independent inference and enables robust task adaptation through amortized variational inference. Experiments on the MetaWorld ML-10 benchmark demonstrate improved adaptation, long-context representations, and generalization to unseen environments.

6.1 Introduction

Reinforcement learning (RL) is traditionally framed as the problem of finding an optimal policy that maximizes expected return in a Markov decision process (MDP). However, real-world applications introduce complexities beyond this formulation, especially in unknown environments where balancing exploration and exploitation becomes critical. This challenge is magnified in robotics as data collection is costly, and

sample efficiency is paramount.

Biological systems demonstrate adaptability by generalizing experience across tasks, i.e., a capability also desirable in robotic agents. While standard RL assumes known rewards and transitions, practical applications involve dynamic or unknown structures. Meta-reinforcement learning (meta-RL) addresses this by enabling agents to "*learn how to learn*" capturing task structures for better generalization.

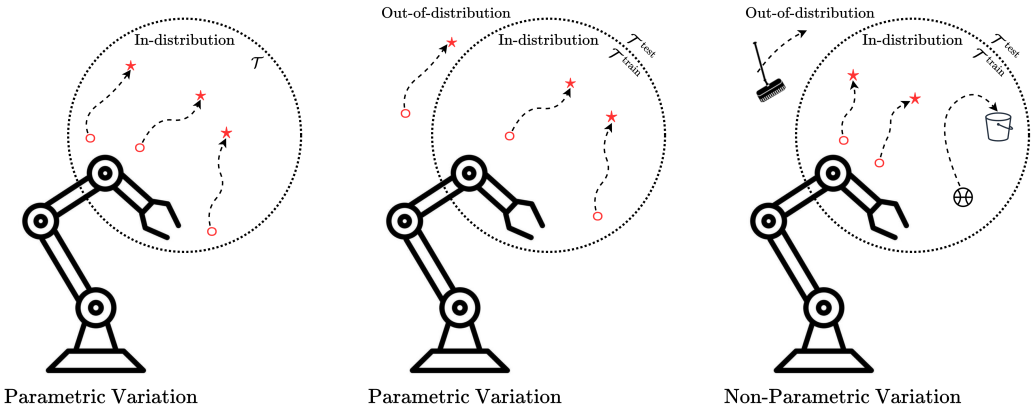


Figure 6.1: An example of parametric and non-parametric variations in meta-RL

Although Bayes-optimal agents offer ideal exploration-exploitation trade-offs, they are intractable in complex domains. Posterior sampling and latent variable meta-learning attempt to approximate this behavior, encoding task uncertainty into structured representations to reduce reliance on extensive interactions. Yet, most meta-RL approaches struggle with significant task variability, often assuming smooth parametric changes between tasks, which limits their adaptability in real-world robotic scenarios involving non-parametric variations [202]. A schematic of different approaches in context accumulation for exploration in meta-RL is depicted in Figure 6.2.

Algorithms like the Model-Agnostic Meta-Learning (MAML) [44] and RL^2 [203] focus on fast adaptation, but they often falter in handling long-term dependencies and generalization across unseen tasks. Recent advances in transformer architectures have shown promise in this context. Transformers, originally introduced for Natural Language Processing (NLP), excel in sequence modeling, offering superior capabilities over RNNs in capturing long-term dependencies. As an example, TrMRL [204] leverages transformer architecture to derive an intraepisodic understanding of the environment to enhance generalization but disregards the variational belief inference in conventional Bayes-adaptive meta-RL [202] that provides meaningful feature

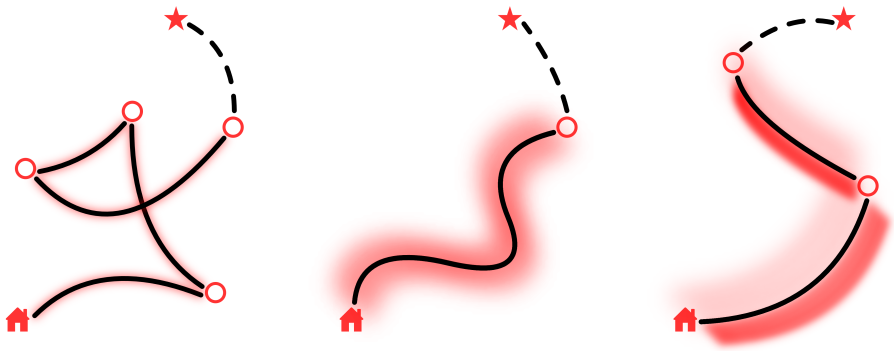


Figure 6.2: Exploration approaches to meta-RL: posterior sampling (left), Bayes-optimal (middle), Bayes-adaptive (right)

extraction on the task distribution.

The attention mechanism in Transformers allows for efficient encoding of sequential experiences without the vanishing gradient issues of RNNs. This ability facilitates modeling of meta-features within or along with the policy and will improve adaptation and robustness. Moreover, transformers remove the need for handcrafted structures, facilitating scalable and flexible architecture in high-dimensional, dynamically changing environments. TrMRL extends this by modeling memory reinstatement via associating multi-layer working memories. This structure attempts to minimize Bayes' risk by reducing the condition to only the most recent interactions.

To address current limitations, a framework is proposed in this chapter for extracting long-term interepisodic features for Bayes-adaptive exploration. Also, CRAFT isolates the agent's performance from task history by removing the condition of the belief model on the actions. This abstraction will allow learned behaviors to be reused and recombined in every stage of the training. Leveraging structured latent spaces and variational inference, the method facilitates strategic exploration and robust generalization, evaluated in a robotic simulation benchmark. This integration of probabilistic task embeddings and meta-learning techniques aims to ameliorate performance limitations and advance autonomy in RL-driven robot manipulation.

The remainder of this chapter is structured as follows: Section 6.2 provides an overview of related work in RL, meta-learning, and Bayesian inference. Section 6.3 shows the details of CRAFT method, including the inference mechanisms decomposition strategy. Section 6.4 presents experimental results demonstrating the efficacy of the approach in robotic control tasks. Finally, Section 6.5 discusses the implications of the findings and outlines future research directions.

6.2 Background and Related Work

In this section, the general objective and formulation of meta-RL are first presented. Then, the context-adaptive meta-RL paradigm and specifically Bayes-adaptive meta-RL are discussed in detail, while pointing out several related works. Finally, previous applications of transformers in meta-RL are discussed.

6.2.1 Meta-RL General Outline

Meta-RL is concerned with finding a policy, after seeing $n \in N$ tasks (each denoted by $\mathcal{T}_{\text{i}}$, $1 \leq i \leq N$) and their corresponding environments during training, that is optimal in the sense that it is close enough to the optimal single-task policy for all of the tasks in training or test. The proximity here is not always intended to be an averaging or trade-off in performance; rather, it can be measured in learning distance, i.e., the number of shots or experiences that policy requires to adapt to a new task. Therefore, a meta-policy is presumably adaptable with a few-shot training to achieve optimal performance in each $\mathcal{T}_{\text{i}}$, as shown in Figure 6.3. In general, these tasks can be different stages of one single time-variant environment, multiple parameterically different time-invariant environments, or any two different environments sharing some fundamental dynamic or semantic features [2].

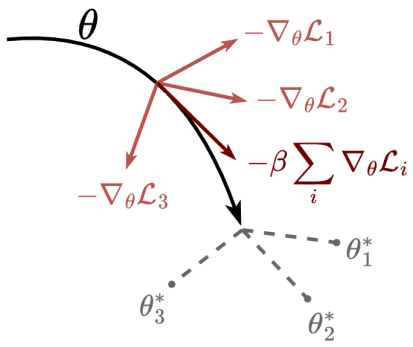


Figure 6.3: Meta-training versus adaptation updates in MAML

MAML is a gradient-based optimization problem to formulate proximity after a few adaptation steps. With the loss function for each task $\mathcal{T}_{\text{i}}$ defined as

$$\mathcal{L}_{\mathcal{T}_{\text{i}}}(\pi_\theta) = -\mathbb{E}_{(s_t,a_t)\sim\pi_\theta}\left[\sum_{t=0}^{N_\tau} \gamma^t r_{t+1})\right], \tag{6.1}$$

a new parameter defining interim policy improvement for $\mathcal{T}_i$ can be calculated in the meta-training stage with

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\pi_\theta). \tag{6.2}$$

The resulting interim policy $\pi_{\theta_i'}$ will then be used to collect new trajectory experiences and losses for updating the policy in the meta-training stage through

$$\theta \rightarrow \theta - \alpha^{\text{meta}} \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}(\pi_{\theta_i'}). \tag{6.3}$$

Both α and α^{meta} are learning rate hyperparameters of MAML. By excluding other tasks from the task distribution or adding a new task, θ_i^* or θ^*, in general, can be trained after a few further iterations of meta-training. π_{θ^*} is deemed the optimal policy for task $\mathcal{T}$.

Meta-RL faces several challenges that hinder its adoption and effectiveness. One major challenge is the issue of sample efficiency. Meta-RL algorithms typically require a large number of interactions with the environment to learn useful meta-policies. However, collecting such a vast amount of data can be time-consuming and resource-intensive, especially in complex and high-dimensional environments [47].

Another challenge in meta-RL is the problem of generalization across tasks and domains. Meta-RL algorithms aim to learn meta-policies that can adapt to new tasks or environments with minimal additional training. However, achieving robust generalization remains a significant issue. Meta-learned policies often struggle to transfer knowledge to tasks or domains that differ significantly from the training distribution, resulting in poor performance and limited applicability. Yu et al. [3] has shown that state-of-the-art meta-RL algorithms usually struggle in adaptation to an unseen robotic manipulation task after being exposed to ten or even 45 tasks in meta-training. This indicates that knowledge sharing among the training tasks is not easily generalizable to the test domain.

Addressing these challenges requires developing effective methods to capture variations. Incorporating domain adaptation techniques and exploring approaches that can extract structured knowledge from an unstructured training might enhance the generalization capability.

6.2.2 Context-Adaptive Meta-Reinforcement Learning

Context-adaptive meta-RL refers to a class of methods in which agents adapt to new tasks by conditioning the policy on an inferred latent representation from the *context*. Context is an umbrella term that might include past transitions or any task-related information available. Instead of relying solely on gradient-based parameter updates at test time, these methods learn a mapping from the raw information in trajectories to a latent task domain during training, which is then used for improved adaptation. The conditioned policy will ideally be enabled to make task-aware decisions. Prominent examples include PEARL [47], which infers a variational latent for conditioning an off-policy actor-critic agent.

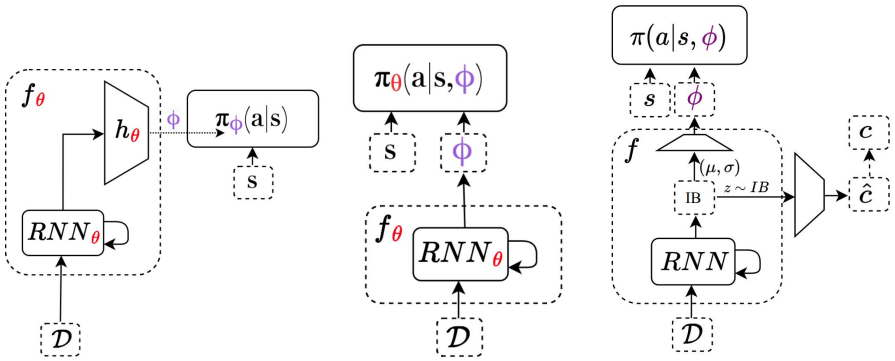
Among the various context-adaptive strategies, state augmentation, technically concatenation, which embeds the learned representation directly into the observation space of the agent, stands out for its architectural simplicity and empirical robustness. As opposed to using hypernetworks [205], which directly tune the parameters of policy networks according to the context representation, state augmentation allows the context-conditioned policy to retain the Markov property and scale more effectively with task complexity [2]. State-augmented policies are particularly effective in continuous control environments and are compatible with both on-policy and off-policy training regimes. These methods strike a balance between inference flexibility and representational capacity, making them well-suited for meta-RL settings. Figure 6.4 shows the architectural differences among the three context-adaptive designs pointed out in this chapter.

In addition to offering strong empirical results, context-adaptive approaches facilitate interpretability and modularity in policy design. By decoupling task inference from policy optimization, these methods enable more transparent analysis of task representations and provide a solution for incorporating structured priors under uncertainty [206]. As such, context-adaptive meta-RL forms a crucial foundation for more complex frameworks, including Bayes-adaptive meta-RL, which build on the core idea of conditioning on task inference to achieve generalization.

6.2.3 Off-Policy Context-Adaptive Meta-RL

In this section, a few off-policy meta-RL frameworks will be discussed that use some version of context embedding in their models.

Rakelly et al. [47] proposed PEARL that uses a learned latent space z to condition



(a) Hypernetwork (b) RL^2 (c) Task Inference

Figure 6.4: Context-adaptive meta-RL baselines from Beck et al. [2]

the meta-policy on each task. Therefore, the former policy $\pi_\theta(a|s)$ will be redefined as $\pi_\theta(a|s,z)$. Here, z will be estimated by training a probabilistic model called *inference network* $q_\phi(z|\mathbf{c})$, where c is a context variable consisting of task-specific independently sampled transitions. Hence, for a task $\mathcal{T}_\text{i}$, $\mathbf{c}$ is a set of transitions $\mathbf{c}_{1:N}^{\mathcal{T}_\text{i}}$ where

$$\mathbf{c}_n^{\mathcal{T}_\text{i}} = (s,a,r,s')_n^{\mathcal{T}_i}. \tag{6.4}$$

The main advantage of PEARL is its adaptability to off-policy training regimes with minimal storage overhead. In the presence of a new task $\mathcal{T}$ at inference time, the context $c^\mathcal{T}$ will be accumulated in the form of transitions during the collection of the new experiences and $q_\phi(z^\mathcal{T}|\mathbf{c}^\mathcal{T})$ will estimate a latent vector corresponding to $\mathcal{T}$. $\pi_\theta(a|s,z^\mathcal{T})$ will be conditioned to the acquired knowledge about the dynamics of $\mathcal{T}$ and make decisions accordingly. Although the introduction of $q_\phi(z|\mathbf{c})$ will enable the automatic task encoding feature for this method, the downside of PEARL remains in disregarding the temporal structure of the context, which limits its application to hierarchical tasks and partially-observable MDPs.

Nam et al. [207] present an extension to PEARL's approach, SiMPL, that enables meta-learning on long-horizon, sparse-reward tasks using offline datasets. This approach involves extracting reusable skills and a skill prior from the offline data, meta-training a high-level policy to compose these skills into long-horizon behaviors, and rapidly adapting the meta-trained policy to solve unseen target tasks. Experimental results on continuous control tasks demonstrate a considerable advantage compared with PEARL [47].

Unlike PEARL, SiMPL uses offline transition datasets without rewards or task

labels to extract short, reusable skills for solving unseen long-horizon tasks. A high-level policy, guided by a skill prior from offline data, explores the learned skill space, while a probabilistic encoder infers task information from context. The high-level policy selects latent skill vectors z, which condition the low-level policy to produce actions based on the current state and selected skill.

Another notable off-policy meta-RL method, TIGR [208], is a task-inference-based meta-RL method that adapts the Bayesian meta-RL formulation, regularly present in on-policy-based meta-RL algorithms, by introducing Gaussian variational autoencoders (VAE) and gated recurrent units (GRUs) to capture multi-modal task representations in its task inference model. While off-policy meta-RL methods, predominantly PEARL-based approaches, handle parametric adaptation quite well, they still struggle when faced with non-parametric variations in task distributions. These methods rely on structured priors and efficient latent space encodings but lack the flexibility to adapt to non-stationary and qualitatively distinct task variations.

6.2.4 Bayesian Approach To Context-Adaptive Meta-RL

Meta-RL aims to develop agents capable of rapid adaptation to new tasks by leveraging prior experiences across a distribution of related Markov Decision Processes. A key challenge in meta-RL is efficiently balancing exploration and exploitation when the underlying task dynamics are initially unknown. This challenge is even more important when dealing with on-policy algorithms that rely on the most recent interactions with a task and are prone to achieving sub-optimal performance because of the local minima issues or forgetting in the long run. Bayesian meta-reinforcement learning (Bayesian meta-RL) provides a principled framework to address this challenge by maintaining and updating a belief distribution over the latent MDP structure by defining a recent online history, thereby enabling approximate Bayesian optimal behavior. This design also solves the intractable nature of posterior sampling present in the off-policy algorithms.

A prominent model-free meta-RL algorithm, RL^2, relies on recurrent networks that implicitly encode task information through sequential interactions [203]. Memory-based methods such as RL^2 process past states, rewards, and actions as auxiliary inputs, allowing the agent to adapt its policy within a task dynamically. This recurrent structure aligns closely with Bayesian formulations, where past experiences inform a belief distribution over task parameters [209]. The Bayesian perspective re-

frames meta-learning as an inference problem, where the agent maintains a posterior belief over task dynamics and updates it continuously based on observed transitions and rewards.

Variational Bayes-Adaptive Deep RL (VariBAD) [202] extends this concept by integrating variational inference into model task uncertainty explicitly. The agent learns a stochastic latent representation of the task, conditioned on past recent experiences, which serves as an inductive bias for exploration-exploitation trade-offs. Unlike memory-based approaches that implicitly infer task dynamics through recurrence, VariBAD encodes the belief in a structured latent space, enabling a more explicit and interpretable task representation.

For a regular stochastic episodic MDP corresponding to the task $\mathcal{T}_i$,

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_R, \mathcal{P}_S, \mathcal{P}_{S_0}, \gamma, H) \tag{6.5}$$

with state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor γ, and maximum episode length of H, $\mathcal{P}_S^i$ and $\mathcal{P}_R^i$ represent conditional probabilistic distributions $p(s'|s,a)$ and $p(r|s,a)$ respectively defined on the state domain $\mathcal{S}$ and the reward domain $\mathcal{R}$. The successive state and the instant reward are recognized as sampling from $s' \sim p(s'|s,a)$ and $r \sim p(r|s,a)$. Similarly, the Bayes-Adaptive MDP (BAMDP) framework formalizes optimal decision-making under uncertainty by augmenting the state space with a belief distribution over task dynamics [210, 211]. In this setting, the agent's objective is to maximize expected return while simultaneously refining its belief over the task. Formally, the belief update follows Bayes' rule, integrating observed transitions into the posterior distribution over the reward and transition functions of the $\mathcal{M}^i$, the MDP representing the task $\mathcal{T}_i$, as

$$b_t(\mathcal{P}_R^i, \mathcal{P}_S^i) = p(\mathcal{P}_R^i, \mathcal{P}_S^i | \tau_{0:t}) \tag{6.6}$$

with $\tau_{0:t} = \{s_0, a_1, r_1, s_1, \cdots, s_{t-1}, a_t, r_t, s_t\}$ denotes the agent's trajectory up to timestep t. The belief-augmented BAMDP enables structured exploration by guiding the agent toward actions that optimally reduce uncertainty while maximizing reward. However, exact Bayesian inference is intractable in complex environments, necessitating approximate solutions.

By integrating the belief in the state definition, a hyper-state space $\mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$ is obtained, where $\mathcal{B}$ is the belief space, and the hyper-state at timestep t is therefore $s_t^+ = (s_t, b_t) \in \mathcal{S}^+$. The resulting dynamic transition hyper-functions $\mathcal{P}_S^+$ and $\mathcal{P}_R^+$ is

derived as

$$\begin{aligned}\mathcal{P}_S^+(s_{t+1}^+|s_t^+, a_{t+1}, r_{t+1}) &= \mathcal{P}_S^+(s_{t+1}, b_{t+1}|s_t, a_{t+1}, r_{t+1}, b_t) \\ &= \mathcal{P}_S^+(s_{t+1}|s_t, a_{t+1}, b_t)\ p(b_{t+1}|s_t, a_{t+1}, r_{t+1}, s_{t+1}, b_t) \\ &= \mathbb{E}_{b_t}\left[\mathcal{P}_S(s_{t+1}|s_t, a_{t+1})\right]\ \delta(b_{t+1} = p(\mathcal{P}_R, \mathcal{P}_S|\tau_{0:t+1})).\end{aligned} \tag{6.7}$$

The consequent state of the original MDP s_t is found via deterministic Bayes' rule applied to the current posterior distribution of the transition function and the belief distribution. Similarly, the reward hyper-function is the expected reward under the current posterior, after the state transition, over reward functions

$$\begin{aligned}\mathcal{P}_R^+(r_{t+1}|s_t^+, a_t, s_{t+1}^+) &= \mathcal{P}_R^+(r_{t+1}|s_t, b_t, a_t, s_{t+1}, b_{t+1}) \\ &= \mathbb{E}_{b_{t+1}}\left[\mathcal{P}_R(r_{t+1}|s_t, a_t, s_{t+1})\right].\end{aligned} \tag{6.8}$$

These two dynamic hyper-functions result in defining a BAMDP [210]:

$$\mathcal{M}^+ = (\mathcal{S}^+, \mathcal{A}, \mathcal{P}_R^+, \mathcal{P}_S^+, \mathcal{P}_{S_0}^+, \gamma, H^+),$$

which is a special case of a context-adaptive MDP formed by augmenting Markov states by the posterior beliefs in a partially observable MDP [212]. The optimization objective of the agent is now to maximize the following for BAMDP

$$\mathcal{J}^+(\pi) = \mathbb{E}_{b_0, \mathcal{P}_{S_0}^+, \mathcal{P}_S^+, \pi}\left[\sum_{t=0}^{H^+-1} \gamma^t \mathcal{P}_R^+(r_{t+1}|s_t^+, a_t, s_{t+1}^+)\right], \tag{6.9}$$

where the scope of the learning objective is extended over the meta-horizon H^+, distinct from the original MDP horizon H. These two parameters can naturally coincide; however, longer memory for BAMDP helps maintain the belief under less uncertainty. In other words, remembering a *few* more shots, a few more experiences in the case of RL, will narrow down the belief distribution, so it is reasonable to assume $H^+ = n_H \times H$. In this formulation, the exploration-exploitation switch is triggered by the agent when it deems information-seeking actions to be sufficient.

VariBAD [202] applies variational inference to estimate the belief distribution over tasks. Rather than using an explicit belief update, they use an amortized inference network conditioned on previous transitions to generate a variational posterior latent vector. This gives a more flexible online adaptation capability to VariBAD, and as a result, a greater sample efficiency compared to conventional gradient-based meta-RL

methods like MAML.

6.2.5 Transformers in Meta-RL

Meta-learning approaches in Bayesian meta-RL can be categorized into recurrent models, gradient-based methods, and transformer-based architectures. Traditional methods such as RL^2 use RNNs to encode past experiences into hidden states, allowing agents to infer task dynamics based on episodic memory. However, these approaches often suffer from information bottlenecks and fail to model long-range dependencies effectively. Note that this is usually the case for the BAMDP hyper-episodes of length H^+.

Gradient-based meta-learners, such as MAML, optimize model parameters to allow fast adaptation through a few gradient updates. While effective in certain settings, these methods are computationally expensive and struggle with hierarchical robotic tasks with complex temporal dependencies. Moreover, they typically rely on a fixed initialization, like MAML, which limits their ability to generalize effectively across highly diverse environments.

Recent advancements in transformer-based reinforcement learning have demonstrated how RL can be formulated as a sequence modeling problem, paving the way for architectures such as Decision Transformer (DT) [213] and Online Decision Transformer (ODT) [214]. Decision Transformer defines RL by casting it as a conditional sequence modeling task, leveraging autoregressive modeling to compensate for partial observability and output actions based on the history of interactions for desired future returns. This eliminates the need for explicit value functions or policy gradients, making RL more interpretable and stable. Expanding on this, ODT extends the capabilities of the Decision Transformer by adding online fine-tuning into the framework. These developments indicate how transformers might enhance memory-based meta-learning. Prompt-DT [215] evolves the original DT by adding a prompt token to *label* the task and providing an implicit MDP definition for few-shot generalization. The prompt used in Prompt-DT is a combination of several short demonstrations, each containing only a few timesteps. Although this design falls under the multi-task RL paradigm that takes advantage of external information provided for each task, it is a remarkable example of generalizable policy generation with transformers in RL.

Other transformer-based architectures, such as TrMRL [204] and HTrMRL [216], provide an alternative approach by leveraging self-attention mechanisms to process

entire sequences in parallel. Unlike RNN-based methods, transformers do not require sequential processing, allowing for more efficient training. TrMRL introduces an episodic memory mechanism, refining task representations at multiple layers. The memory reinstatement mechanism in TrMRL reuses previously learned representations for faster adaptation in meta-learning tasks. The hierarchical structure of HTrMRL enables the agent to capture intra-episode and inter-episode relationships, facilitating mode detection and definition generalization across tasks.

Notably, a recent work, AMAGO [217], is an off-policy transformer-based context-adaptive meta-RL algorithm. AMAGO uses long-sequence transformers, trained over full trajectories in parallel. They tried to address two major issues in training large transformer models, the memory bottlenecks and planning horizon limitation, validated in goal-conditioned tasks with sparse rewards. Like TrMRL and HTrMRL, they leverage the parallelism transformers for inter-episode, i.e., inter-task, generalization. While AMAGO combines sequence modeling and off-policy learning, its reliance on dense replay data and goal relabeling introduces unnecessary challenges in context-adaptive meta-RL. Moreover, AMAGO does not explicitly address task uncertainty with amortized inference. Compared to models that infer latent task distributions, AMAGO prioritizes direct goal adaptation with relabeling, offering a complementary parallel approach within transformer-based meta-RL.

Considering the mentioned body of work with transformers in meta-RL, transformers can improve upon previous architectures in Bayesian meta-reinforcement learning. Improvements may involve better uncertainty handling, capturing complex multi-scale dependencies, or enabling agents to generalize more efficiently. Compared to RNN-based models and gradient-based meta-learning techniques, transformer architectures such as TrMRL and HTrMRL hold promise for a more robust and scalable solution for temporally challenging and diverse robotic RL applications.

6.3 Design and Architecture of CRAFT

This section describes how CRAFT is learning a belief model in context-adaptive meta-RL, emphasizing architectural choices that support generalizable and decomposable task representations. The design leverages a transformer encoder-decoder to maintain a posterior belief about the task from action-free trajectories, isolates inference from policy optimization, and supports Bayesian task inference via amortized variational learning.

6.3.1 Learning the Belief Representation

Whenever, in context-adaptive meta-reinforcement learning, the task inference module is explicitly separated from the policy, such as in TIGR [208] or VariBAD, additional loss functions are typically required to promote meaningful representation learning within the inference model. This is because, in the absence of policy supervision signals, the task representation must be learned from auxiliary objectives, such as reconstructing rewards, transitions, or other relevant statistics of the task environment.

Task Decomposition

The representation learned by the inference model, often a latent belief or embedding, serves as a compact and informative summary of the current task. This representation should not only differentiate between tasks but ideally capture internal task structure in a form that is decomposable either temporally or structurally. In meta-RL, such decomposability is critical for generalization and transfer. These subcomponents of tasks are commonly referred to as *skills*, temporally extended, goal-directed policies that serve as reusable building blocks across multiple tasks [218]. Formally, each task $\mathcal{T}_i$ can be expressed as a sequence of skills

$$\mathcal{T}_i = \{\kappa_j\}_{j=1}^{N_\kappa^i}, \quad \kappa_j : \mathcal{S} \times \Theta_\kappa \to \mathcal{U}, \tag{6.10}$$

where $\mathcal{S}$ denotes the state space, $\mathcal{U}$ the action space, and Θ_κ the parameter space of the skill policies. Skills κ_j may be shared across different tasks, enabling modular policy construction and improved sample efficiency in adaptation.

Task representation learning can be supervised using either privileged information or unsupervised reconstruction-based signals. For example, TIGR uses explicit task labels as ground truth supervision. While such privileged supervision can enforce structure, it is often unavailable or too rigid in practical applications. An alternative, more general approach is to learn the task embedding by reconstructing observed transitions and rewards. Although less constrained, this strategy encourages the model to reflect the underlying dynamics and optimality characteristics of the task, which are more aligned with real-world scenarios where task labels are absent.

Gradient Isolation

A defining feature of task-inference-based meta-RL algorithms like VariBAD, unlike approaches such as PEARL [47], is the deliberate prevention of policy gradients from backpropagating through the task inference model. This design choice is motivated by several important considerations:

I. *Stable and Modular Belief Inference:* if policy gradients are allowed to flow into the inference model, the resulting latent embeddings may become entangled with the current policy. This coupling leads to task representations that are biased toward short-term policy performance rather than general task structure. By isolating gradient flow, the inference model remains a standalone module that encodes consistent and reusable task representations, independent of the policy's current behavior.

II. *Amortized Variational Inference Stability:* in VariBAD, the task inference model is trained using amortized variational inference to maximize the Evidence Lower Bound (ELBO)

$$\log p(\tau) \geq \mathbb{E}_{q_\phi(z|\tau)}[\log p(\tau|z)] - D_{\mathrm{KL}}(q_\phi(z|\tau)\|p(z)). \tag{6.11}$$

Without gradient isolation, the encoder $q_\phi(z|\tau)$ might overfit to policy-specific features, distorting the latent belief distribution in favor of optimizing the current return rather than encoding generalizable task features. Preventing this gradient feedback ensures that the learned latent space remains well-calibrated and policy-agnostic.

III. *Reduced Instability in Policy Optimization:* reinforcement learning policy updates are known to be noisy and high-variance. If these unstable gradients influence the inference model, they can introduce non-stationarity into the belief space, making training less stable and degrading the quality of the learned task representations. Isolating the inference model from these gradients helps preserve a stable and reliable task embedding throughout learning.

In summary, decoupling the learning dynamics between the policy and the inference model supports the formation of robust and transferable internal representations of tasks. This structural separation is critical for achieving sample-efficient adaptation in context-adaptive meta-RL, particularly in environments with high task diversity or limited supervision.

6.3.2 The Action-Free Belief Model Design

A central component of context-adaptive meta-reinforcement learning is the belief model, which maintains a latent representation of the current task based on interaction history. To be effective, this model must extract meaningful temporal patterns that characterize both the task dynamics and reward structures. Most frameworks incorporate some form of temporal modeling to capture these features from ongoing interactions.

For example, PEARL circumvents sequential modeling by decomposing each episode $\tau_{0:t}^{\mathcal{T}_k} = \{(s_i, a_i, r_i, s_{i+1})\}_{i=0}^{t-1}$ into independent interaction tuples. These tuples are stored in task-specific replay buffers. During inference, N_c time-invariant context samples $\{\mathbf{c}_i^{\mathcal{T}_k}\}_{i=1}^{N_c}$ are drawn at random from experiences with task $\mathcal{T}_k$ and used to infer a posterior over the latent task variable. While this formulation is effective in off-policy training, it imposes two limitations. First, it implicitly assumes the Markov property across all tasks, which limits its applicability to environments with long-range temporal dependencies or partial observability. Second, PEARL allows policy gradients to backpropagate through the belief model. This coupling between the inference and control components may lead to overfitting, where the task representation becomes tailored to short-term policy performance rather than capturing generalizable features of the environment.

Alternatively, BAMDP approaches such as VariBAD treat the task context as entire sequences of interaction—either single episodes $\tau_{0:H}$ or multi-episode sequences $\tau_{0:H^+}$. In BAMDPs, the posterior belief over the unknown transition and reward functions is typically formulated as $b_t = p(\mathcal{P}_S, \mathcal{P}_R \mid \tau_{0:t})$. This formalism explicitly encodes uncertainty over environment dynamics and rewards and supports exploration strategies based on belief updating.

To further decouple task inference from decision-making, and following the rationale in Section 6.3.1, the posterior belief can be conditioned only on the observed states and rewards

$$b_t^{\text{action-free},\mathcal{T}_k} = p(\mathcal{P}_R, \mathcal{P}_S \mid \tau_{0:t}^{\text{action-free},\mathcal{T}_k}), \tag{6.12}$$

where the context $\tau_{0:t}^{\text{action-free},\mathcal{T}_k}$ omits actions entirely and may consist of multiple

episodes drawn from the same task $\mathcal{T}_k$. This context is defined as

$$\begin{aligned}\tau_{0:t}^{\text{action-free},\mathcal{T}_k} = \{&s_0^1, r_1^1, s_1^1, \dots, r_H^1, s_H^1,\\ &s_0^2, r_1^2, s_1^2, \dots, r_H^2, s_H^2,\\ &\dots,\\ &s_0^{n_\tau}, r_1^{n_\tau}, s_1^{n_\tau}, \dots, r_t^{n_\tau}, s_t^{n_\tau}\},\end{aligned} \tag{6.13}$$

where H is the maximum episode length, $n_\tau \leq n_H$ denotes the number of episodes in the context, and $H^+ = n_H \times H$ is the maximum context horizon.

In this formulation, the belief-augmented transition and reward functions presented in Equation (6.7) and Equation (6.8) for the BAMDP are updated as

$$\begin{aligned}\mathcal{P}_S^+(s_{t+1}^+ \mid s_t^+, a_{t+1}, r_{t+1}) &= \mathcal{P}_S^+(s_{t+1}, b_{t+1} \mid s_t, a_{t+1}, r_{t+1}, b_t)\\ &= \mathcal{P}_S^+(s_{t+1} \mid s_t, a_{t+1}, b_t) \cdot p(b_{t+1} \mid s_{t+1}, b_t)\\ &= \mathbb{E}_{b_t}\left[\mathcal{P}_S(s_{t+1} \mid s_t, a_{t+1})\right] \cdot \delta\left(b_{t+1} = p(\mathcal{P}_R, \mathcal{P}_S \mid \tau_{0:t+1}^{\text{action-free}})\right),\end{aligned} \tag{6.14}$$

and

$$\begin{aligned}\mathcal{P}_R^+(r_{t+1} \mid s_t^+, a_t, s_{t+1}^+) &= \mathcal{P}_R^+(r_{t+1} \mid s_t, b_t, a_t, s_{t+1}, b_{t+1})\\ &= \mathbb{E}_{b_{t+1}}\left[\mathcal{P}_R(r_{t+1} \mid s_t, s_{t+1})\right].\end{aligned} \tag{6.15}$$

This formulation naturally supports learning from demonstrations or agent-agnostic data sources, as belief updates no longer require knowledge of the agent's actions. In many robotic tasks, especially goal-oriented environments using high-level controllers, reward functions primarily depend on outcome success, rather than the precise action taken. This is particularly true in meta-RL benchmarks like Metaworld [3], where high-level Cartesian actions are mapped to joint motions by a robust controller. Since the influence of the action is implicitly reflected in the state transitions, this design enables the inference model to operate effectively even when actions are absent from the input sequence. Therefore, it is safe to assume that Assumption 1 holds for the environments in the field of interest of this chapter.

Assumption 1. *Consider episodic stochastic MDP: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_R, \mathcal{P}_S, \mathcal{P}_{S_0}, \gamma, H)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{P}_S(s'|s,a)$, reward function $\mathcal{P}_R(r|s,a,s')$, discount factor γ, and maximum trajectory length of H.*

A Bayes-adaptive belief maintained on the dynamic functions, conditioned on a

full trajectory $\tau_{0:t} = \{s_0, a_1, r_1, \ldots, s_{t-1}, a_t, r_t, s_t\}$*, is defined as*

$$b_t(\mathcal{P}_R, \mathcal{P}_S) = p(\mathcal{P}_R, \mathcal{P}_S \mid \tau_{0:t}).$$

This belief can be approximated by

$$b_t^{\textit{action-free}} = p(\mathcal{P}_R, \mathcal{P}_S \mid \tau_{0:t}^{\textit{action-free}}) \approx p(\mathcal{P}_R, \mathcal{P}_S \mid \tau_{0:t})$$

assuming a sufficiently long trajectory ($t \geq t_{\min}$)*, and if*

$$\mathcal{P}_R(r|s, a, s') \approx \tilde{\mathcal{P}}_R(r|s_0, \ldots, s_{-1}, s, s') \quad \textit{and} \quad \mathcal{P}_S(s'|s, a) \approx \tilde{\mathcal{P}}_S(s'|s_0, \ldots, s_{-1}, s).$$

However, this action-free design introduces several new challenges that place greater demands on the capacity and stability of the sequence model:

I. *Hierarchical and Phase-Aware Inference:* The belief model must not only infer the overall task identity but also distinguish different phases or skills within a task. This is particularly important during adaptation, where the context window is updated continuously while the belief and policy models remain frozen.

II. *Long-Term Dependency Modeling:* Removing actions reduces the amount of structured supervision available to the belief model, which now must infer latent task features from more ambiguous and flexible input. In this setting, attention-based models with autoregressive and long-range dependency capabilities are more suitable than memory-based recurrent networks.

III. *Scalability with Longer Contexts:* The action-free input sequence is more compact, allowing longer trajectories across multiple episodes to be included in the context. While this can improve task inference performance, it also necessitates robust sequence models that can prevent forgetting and capture dependencies across extended temporal horizons.

Due to these requirements, a transformer-based belief model is adopted in the subsequent design, as detailed in the following section.

6.3.3 Transformer Encoder-Decoder for Task Inference

This section introduces the integration of a transformer encoder-decoder architecture into the belief model for the Bayes-adaptive meta-RL framework. It begins by reviewing the foundational components of transformer architectures, followed by a detailed

explanation of CRAFT design, the associated loss formulations, and the use of rotary positional embeddings for better temporal reasoning.

Transformers Foundations

Transformers [219] offer a sequence modeling framework based on self-attention, allowing for efficient parallel processing of tokens and long-range dependency capture. The core mechanism employs learned projections of input embeddings into query ($\mathbf{q}$), key ($\mathbf{k}$), and value ($\mathbf{v}$) vectors. The scaled dot-product attention is defined as

$$\text{Attn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{softmax}\left(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d_k}}\right)\mathbf{v}, \tag{6.16}$$

where d_k is the dimensionality of the keys.

A transformer encoder applies self-attention across the entire sequence, enabling each token to attend to all others, while a transformer decoder includes a causal mask to ensure that predictions depend only on current and past tokens. These features make transformers well-suited for autoregressive modeling and sequence-to-sequence tasks. In the context of meta-reinforcement learning, transformers are adopted to process trajectories of varying lengths, capturing both temporal and structural information about the task without relying on recurrent states.

Rotary Positional Embedding

Positional embedding is an integral part of a transformer architecture, also present in CRAFT, to make the attention mechanism aware of the temporal relations among the tokens. For instance, in self-attention, rather than multiplying transformation matrices $\mathbf{W}_{\{q,k,v\}}$ on raw token embeddings $\mathbf{E}$, the following process is adopted in the conventional Additive positional embedding [219],

$$\begin{aligned} \mathbf{q}_m &= f_q(\mathbf{E}_m, m) \\ \mathbf{k}_n &= f_k(\mathbf{E}_n, n) \\ \mathbf{v}_n &= f_v(\mathbf{E}_n, n), \end{aligned} \tag{6.17}$$

with,

$$f_{\{q,k,v\}}(\mathbf{E}_i, i) := \mathbf{W}_{\{q,k,v\}}(\mathbf{E}_i + \mathbf{p}_i). \tag{6.18}$$

Here, $\mathbf{p}$ is commonly chosen as a sinusoidal function of both i and the position of each element of the token, as in,

$$\begin{cases} \mathbf{p}_{i,2t} & = \sin(m/10000^{2t/d}) \\ \mathbf{p}_{i,2t+1} & = \cos(m/10000^{2t/d}) \end{cases}, \tag{6.19}$$

where i is the token position in the context and $t \in 0, \ldots, \frac{d}{2} - 1$ with d representing the length of the token.

In this work, the rotary positional embedding (RoPE) [220] is used for its superiority in retaining information on the relative position of the tokens present in the dot product process of the attention mechanics. In other words, the dot product of two token embeddings is preferred to be only dependent on $m - n$, the relative position or distance between tokens $\mathbf{E}_m$ and $\mathbf{E}_n$.

$$\langle f_q(\mathbf{E}_m, m), f_k(\mathbf{E}_n, n) \rangle = g(\mathbf{E}_m, \mathbf{E}_n, m - n). \tag{6.20}$$

The following transformation in RoPE formulation satisfies this property,

$$f_{\{q,k\}}(\mathbf{E}_m, m) = \mathbf{R}^d_{\Theta,m} \mathbf{W}_{\{q,k\}} \mathbf{E}_m \tag{6.21}$$

where,

$$\mathbf{R}^d_{\Theta,m} = \begin{pmatrix} \mathbf{R}^d_{\theta_1,m} & 0 & \cdots & 0 \\ 0 & \mathbf{R}^d_{\theta_2,m} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{R}^d_{\theta_{d/2},m} \end{pmatrix} \tag{6.22}$$

is a quasi-diagonal matrix with 2-dimensional rotation matrices $\mathbf{R}^d_{\theta_i,m}$ on the diagonal,

$$\mathbf{R}^d_{\theta_i,m} = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}. \tag{6.23}$$

The rotations are derived using similar angles as in the conventional additive sinusoidal positional embedding $\Theta = \{\theta_i = 10000^{-2i/d}, i \in [0, 1, ..., d/2 - 1]\}$. A schematic of the computation process in RoPE is shown in Figure 6.5. The dot product of two token embeddings after RoPE transformation will be

$$\begin{aligned}\mathbf{q}_m^\intercal \mathbf{k}_n &= \langle f_q(\mathbf{E}_m, m), f_k(\mathbf{E}_n, n) \rangle \\ &= (\mathbf{R}^d_{\Theta,m}\mathbf{W}_q\mathbf{E}_m)^\intercal(\mathbf{R}^d_{\Theta,n}\mathbf{W}_k\mathbf{E}_n) \\ &= \mathbf{E}_m^\intercal \mathbf{W}_q^\intercal R^d_{\Theta,n-m}\mathbf{W}_k\mathbf{E}_n \end{aligned} \tag{6.24}$$

where $\mathbf{R}^d_{\Theta,n-m} = (\mathbf{R}^d_{\Theta,m})^\intercal \mathbf{R}^d_{\Theta,n}$.

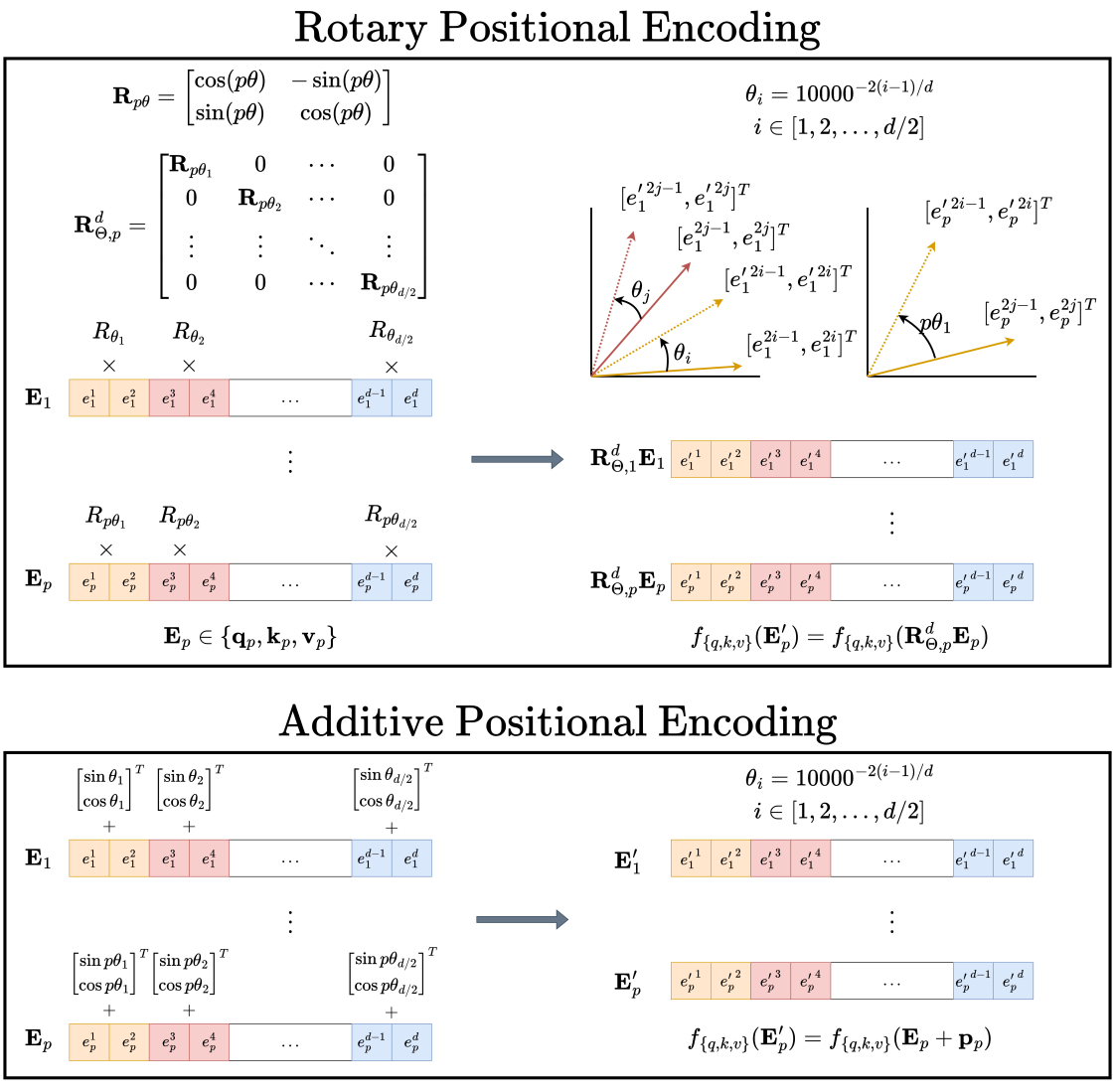


Figure 6.5: Contrastive illustration of conventional additive and rotary positional encoding methods mechanisms

Consequently, the self-attention in the general form can be rewritten as

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_m = \frac{\sum_{n=1}^{N} \text{sim}(\mathbf{q}_m, \mathbf{k}_n)\mathbf{v}_n}{\sum_{n=1}^{N} \text{sim}(\mathbf{q}_m, \mathbf{k}_n)}. \tag{6.25}$$

In the case of conventional self-attention in transformers [219], $\text{sim}(\mathbf{q}_m, \mathbf{k}_n) = \exp(\mathbf{q}_m^\intercal \mathbf{k}_n / \sqrt{d})$. The same principle applies to cross-attention as well. Incorporating this type of positional embedding makes the belief model aware of not only the positions of a state or reward, but also the distance between tokens. This will result in an improved ability to identify recurrences in tasks.

Implementation in CRAFT

The architecture shown in Figure 6.6 uses a transformer encoder-decoder to extract compact and informative representations from action-free trajectories, aligning with the structure of belief updates in Bayes-Adaptive Markov Decision Processes (BAMDP). Borrowing from Equation (6.12), the posterior belief over task identity is defined as

$$b_t^{\mathcal{T}_k} = p(\mathcal{P}_R, \mathcal{P}_S | \tau_{0:t}^{\text{action-free}, \mathcal{T}_k}),$$

where $\tau_{0:t}^{\text{action-free}, \mathcal{T}_k}$ includes only sequences of states and rewards across one or multiple episodes. This eliminates the dependency on actions and better reflects the structure of tasks in domains like robotics, where actions often correspond to high-level commands with minimal influence on reward assignment.

This architecture is chosen for its ability to combine two complementary inference mechanisms: the transformer encoder's self-attention extracts structural features from observed state transitions, effectively modeling the dynamic nature of the environment; meanwhile, the cross-attention in the transformer decoder captures how reward signals depend on these dynamics, defining the notion of optimality specific to the task at hand. This separation enables the belief model to represent both parametric variations (e.g., reward shaping or goal/position variations) and non-parametric variations (e.g., topology or temporal/causal structure) in a unified and flexible framework.

Excluding action tokens from the context serves two main purposes. First, it ensures that the inferred belief is disentangled from the agent's decision process, maintaining the task definition as a property of the environment rather than a byproduct of a particular policy. Second, it facilitates the use of action-free trajectories, such as demonstrations from human teleoperation or scripted controllers, for warm-

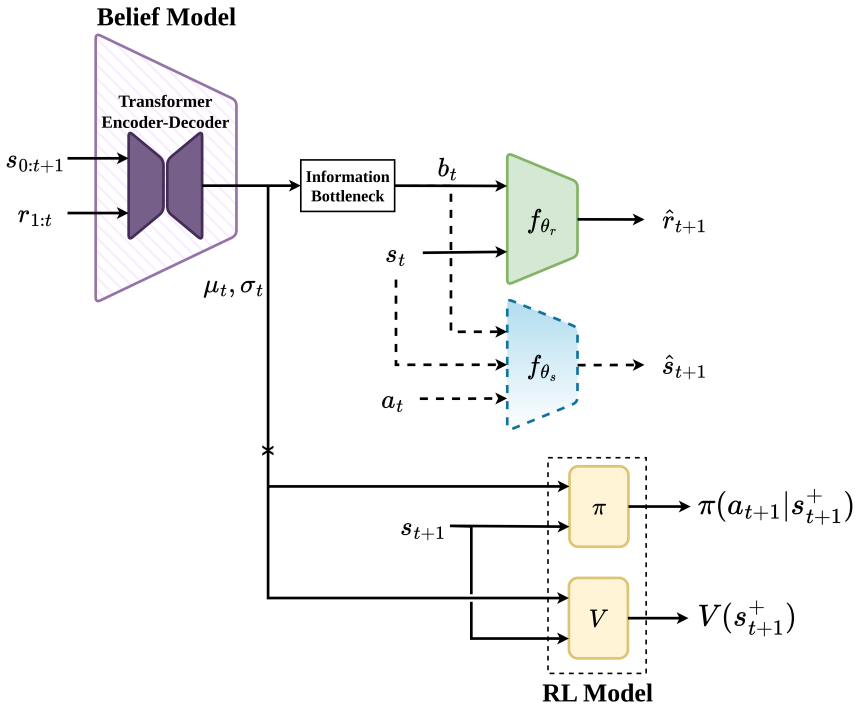


Figure 6.6: General overview of CRAFT, the proposed action-free belief model

starting or pre-training the inference model. This capability extends the applicability of CRAFT to scenarios where actions are unavailable or irrelevant, and opens opportunities to leverage existing datasets [221, 222] for task representation learning. To clarify the mentioned use case, actions might be unavailable in video-only demonstration datasets or teleoperation, and they can be irrelevant when downstream control differs and we seek task-state patterns rather than actuator specifics. Hence, action-free belief simplifies reuse across embodiments and sensing modalities.

I. *Transformer Encoder on States:*

The encoder receives a sequence $\{s_0, s_1, \ldots, s_{t+1}\}$, and embeds them into

$$\{\mathbf{E}^s_0, \mathbf{E}^s_1, \ldots, \mathbf{E}^s_{t+1}\}$$

via a learnable transformation. The encoder then applies self-attention to cap-

ture temporal dependencies among the state tokens

$$\begin{aligned}[\mathbf{u}_i^s]_{i=0:t+1} &= \text{Causal SelfAttn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \\ &= \text{Causal SelfAttn}([f_q(\mathbf{E}_i^s)]_{i=0:t+1}, [f_k(\mathbf{E}_i^s)]_{i=0:t+1}, [f_v(\mathbf{E}_i^s)]_{i=0:t+1}),\end{aligned} \tag{6.26}$$

where $f_{\{q,k,v\}}(\mathbf{E}^s) = \mathbf{W}_{\{q,k,v\}}\mathbf{E}^s$. The inclusion of s_{t+1} ensures that each transition is fully represented up to the most recent outcome.

Causal SelfAttn is the self-attention mechanism with a causal mask where each token $\mathbf{E}_{i+1}^s$, $0 \leq i \leq t$, can only attend to $\{\mathbf{E}_j^s\}_{j \leq i}$. Therefore, the undesirable attention weights will be suppressed to make the representation $\mathbf{u}_i^s$ a causal autoregressive estimation

$$\mathbf{u}_{t+1}^s \approx p(s_{t+1}|s_0, \ldots, s_t). \tag{6.27}$$

The state-based encoder outputs $[\mathbf{u}_i^s]_{i=0:t+1}$ are then passed to the subsequent decoder module and will be used as queries and keys.

II. *Transformer Decoder on Rewards:*

The reward sequence $\{0, r_1, \ldots, r_t\}$, after going through an embedding transformation and mapping to $\{\mathbf{E}_0^r, \mathbf{E}_1^r, \ldots, \mathbf{E}_t^r\}$, is processed with an autoregressive causal self-attention similar to the transformer encoder

$$\begin{aligned}[\mathbf{u}_i^r]_{i=0:t} &= \text{Causal SelfAttn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \\ &= \text{Causal SelfAttn}([f_q(\mathbf{E}_i^r)]_{i=0:t}, [f_k(\mathbf{E}_i^r)]_{i=0:t}, [f_v(\mathbf{E}_i^r)]_{i=0:t}),\end{aligned} \tag{6.28}$$

to estimate

$$\mathbf{u}_t^r \approx p(r_t|0, r_1, \ldots, r_{t-1}). \tag{6.29}$$

III. *Belief Formation in the Transformer Decoder:*

The decoder finally attends to both the state and the reward encodings through cross-attention, fusing dynamic information and temporal optimality structure

to generate task-relevant representations

$$\begin{aligned}[h_i]_{i=1:t} &= \text{Causal CrossAttn}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \\ &= \text{Causal CrossAttn}([f_q(\mathbf{u}_i^s)]_{i=0:t+1}, [f_k(\mathbf{u}_i^s)]_{i=0:t+1}, [f_v(\mathbf{u}_i^r)]_{i=0:t}), \quad (6.30)\end{aligned}$$

The causal mask here is a one-step shifted version of the ones used for the self-attention in the previous stages. This is because the cross-attention mechanism receives two sequences with different lengths, $\{\mathbf{u}_0^s, \mathbf{u}_1^s, \ldots, \mathbf{u}_{t+1}^s\}$ for queries and keys, and $\{\mathbf{u}_0^r, \mathbf{u}_1^r, \ldots, \mathbf{u}_t^r\}$ for values, to estimate

$$h_t \approx p(r_{t+1}|s_0, s_1, \ldots, s_{t+1}, 0, r_1, \ldots, r_t) \quad (6.31)$$

This design is motivated by the idea that the amortized inference of a belief relies on the approximation of the reward function, which is the most authentic definition of the tasks that share state and action spaces. It is also assumed that this can be done only with state transitions and independently of the actions. The mentioned assumption is stronger, especially in environments where outcome-based signals are dominant, such that

$$b_t = p(\tilde{\mathcal{P}}_R|\tau_{0:t}^{\text{action-free}}) \approx p(\tilde{\mathcal{P}}_R(r_{t+1}|s_t, s_{t+1})). \quad (6.32)$$

These representations h_t are further compressed via a transformation parametrized by ψ to a latent posterior distribution $\mathcal{N}(\mu_\psi(h_t), \sigma_\psi(h_t))$. All the previous stages describing the internal structure of the transformer encoder-decoder are visualized in Figure 6.7.

Finally, to create an information bottleneck for regulating variational inference, the belief is sampled from this distribution

$$b_t \sim \mathcal{N}(\mu_\psi(h_t), \sigma_\psi(h_t)) \quad (6.33)$$

Now, as an example, consider a trajectory with two transitions

$$\tau_{0:2}^{\text{action-free}} = \{s_0, r_1, s_1, r_2, s_2\}. \quad (6.34)$$

The encoder self-attention block receives $\{s_0, s_1, s_2\}$ and outputs $\{\mathbf{u}_0^s, \mathbf{u}_1^s, \mathbf{u}_2^s\}$, while the decoder self-attention block receives $\{0, r_1\}$ and outputs $\{\mathbf{u}_0^r, \mathbf{u}_1^r\}$. Using the latent representation h_1 calculated by the masked cross-attention of the transformer decoder,

a decoder f_{θ_r} estimates the current reward $\hat{r}_2$ by a belief $b_1 \sim \mathcal{N}(\mu_\psi(h_1), \sigma_\psi(h_1))$ and the previous state s_1. Another decoder f_{θ_s} is responsible for estimating $f_{\theta_s}(s_2|s_1, a_1, b_1)$. The gradients of the loss of these estimations backpropagate though the belief model for additional guidance in the updates.

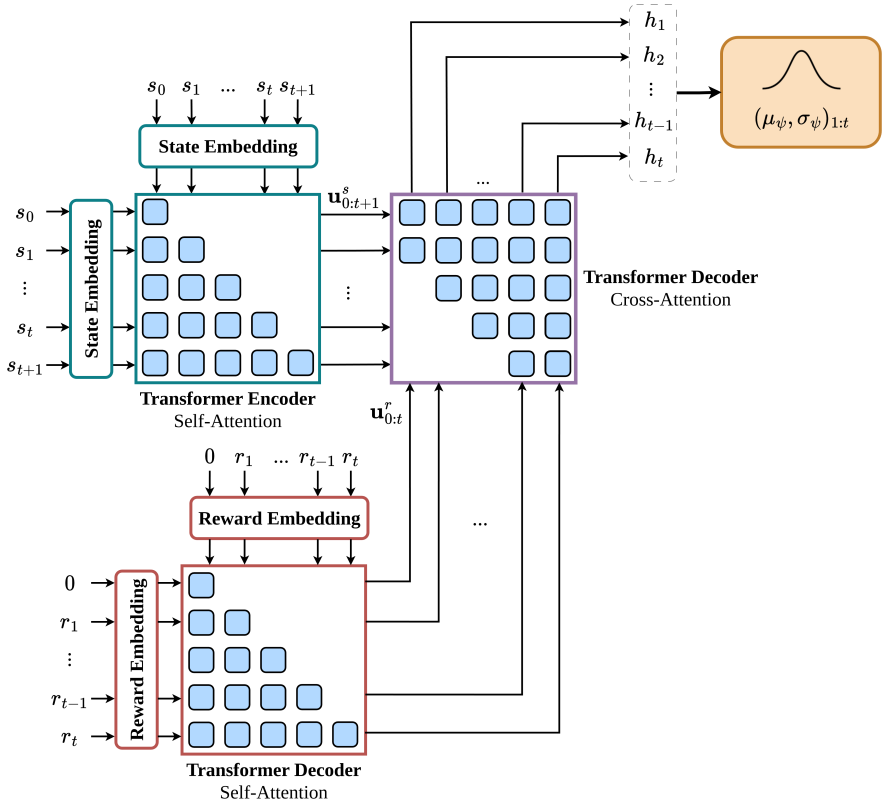


Figure 6.7: The internal structure of the presented belief model based on the causal transformer encoder-decoder

The resulting latent representation distribution, (μ_1, σ_1), summarizes the task-relevant latent distribution, representing belief and uncertainty, and is useful for downstream prediction and control in the RL agent.

Bayesian Inference and Evidence Lower-Bound

The belief model outputs a posterior $q_\phi(b_t|\tau^{\text{action-free}}_{0:t+1})$ over the latent distribution $b_t \sim p(b_t)$. The posterior b_t is used to condition dynamics and reward predictors f_{θ_r} and f_{θ_s}. The latent belief is derived from the transformer decoder's cross-attention representations $\{h_1, \ldots, h_t\}$, transformed into Gaussian parameters:

$$q_\phi(b_t|\tau_{0:t}) \sim \mathcal{N}(\mu_\psi(h_t|\tau^{\text{action-free}}_{0:t+1}), \sigma_\psi(h_t|\tau^{\text{action-free}}_{0:t+1})). \tag{6.35}$$

The reward and transition models are conditioned on this belief and used to reconstruct the next-step reward and state, respectively. The predictive decoders are defined as:

$$f_{\theta_r}(s_t, b_t) \to \hat{r}_{t+1}, \quad f_{\theta_s}(s_t, a_t, b_t) \to \hat{s}_{t+1}. \tag{6.36}$$

To learn this structure, the evidence lower bound (ELBO) on the marginal trajectory likelihood is optimized:

$$\begin{aligned}\mathcal{L}_{\text{ELBO},t} = \mathbb{E}_{q_\phi(b_t|\tau_{0:t+1}^{\text{action-free}})}\left[\log p_{\theta_r,\theta_s}(r_{t+1}, s_{t+1}|s_t, a_t, b_t)\right] \\ - \text{KL}\Big(q_\phi(b_t|\tau_{0:t+1}^{\text{action-free}})\|p_{\theta_r,\theta_s}(b_t)\Big),\end{aligned} \tag{6.37}$$

with the prior $p_{\theta_r,\theta_s}(b_t)$ set to $\mathcal{N}(0, I)$. In detail, the objective can be derived as,

$$\mathcal{L}_{\text{ELBO},t} = \mathbb{E}_{q_\phi(b_t|\tau_{0:t+1})}\big[\mathcal{L}_{\text{Recon}}^{\text{S}} + \mathcal{L}_{\text{Recon}}^{\text{R}} + \mathcal{L}_{\text{Regularization}}\big] \tag{6.38}$$

$$\begin{aligned}= \mathbb{E}_{q_\phi(b_t|\tau_{0:t+1})}\Big[&\log p_{\theta_r}(r_{t+1}|s_t, b_t) \\ &+ \log p_{\theta_s}(s_{t+1}|s_t, a_t, b_t) \\ &- \text{KL}\Big(q_\phi(b_t|\tau_{0:t+1})\|\mathcal{N}(0, I)\Big)\Big]\end{aligned} \tag{6.39}$$

This ensures that the latent representation b_t captures sufficient information to predict both reward and state transitions, thereby supporting fast and generalizable adaptation. By optimizing this objective, the model learns to infer latent task features from action-free contexts and propagate uncertainty into predictive modules, in alignment with BAMDP principles.

The combination of information bottleneck regularization and cross-attentive encoding supports modular design and interpretability. The separation between inference and decision-making also improves policy robustness across diverse task distributions.

6.4 Experiments and Findings

This section presents experimental results for CRAFT belief inference model, built upon a transformer encoder-decoder architecture and designed for adaptive meta-RL dealing with multi-task robotic environments. The experiments evaluate the ability of the model to infer task structure and adapt policies across diverse tasks in a conventional simulation benchmark designed for such an objective. The section is organized into environment setup, implementation details, performance results,

qualitative insights, and discussion.

6.4.1 Environments and Training Pipeline

CRAFT is evaluated using the MetaWorld benchmark [3], a simulated open-source suite consisting of 50 robotic manipulation tasks designed to train and evaluate meta-RL and multi-task RL algorithms. Unlike previous benchmarks that focus on narrow parametric variations, such as different positions or velocities, MetaWorld not only includes parametric variations, but also offers diverse non-parametric changes across tasks. This benchmark also enables generalization to entirely new held-out tasks. This feature makes it a suitable platform to evaluate general-purpose adaptation, particularly before deployment in physical systems.

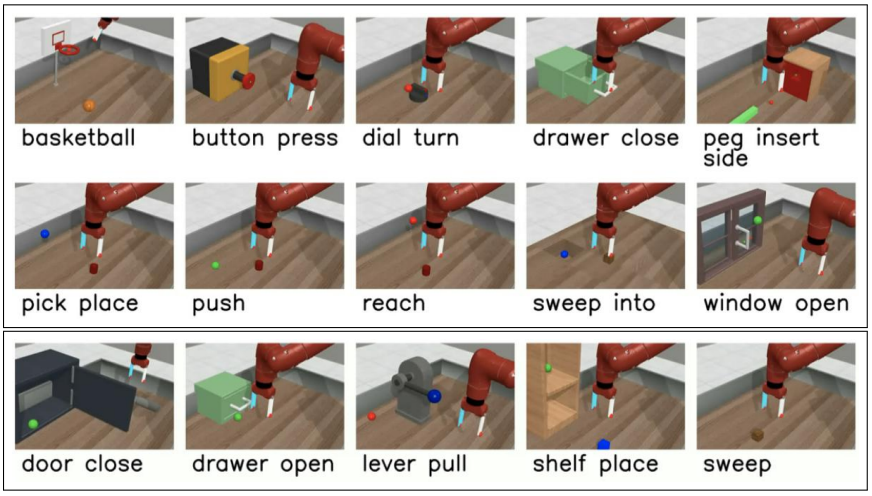


Figure 6.8: Training (upper box) and test (lower box) environments of the ML-10 experiment in MetaWorld for meta-RL benchmarking Yu et al. [3]

From this suite, the ML-10 scenario (Figure 6.8) is adopted for the following implementation. ML-10 provides $n_{\text{train}} = 10$ training tasks and $n_{\text{test}} = 5$ disjoint test tasks, and each task comes with $n_{\text{variations}} = 50$ different parametric variants. All these fifteen tasks differ entirely in terms of entities, the corresponding dynamics, and objectives. In each epoch of meta-training, a subset of training environments is sampled $\mathcal{T}^k \sim \mathcal{T}^{\text{train}}$, regardless of their identity or parametric configuration. Then, parallelized across the tasks in $\mathcal{T}^k$, a series of consecutive n_H episodes of interaction is executed synchronously in each environment. The goal here is to accumulate contextual raw information in each meta-episode and update the belief online, following the Bayes-adaptive exploration approach in Equations (6.12) and (6.13).

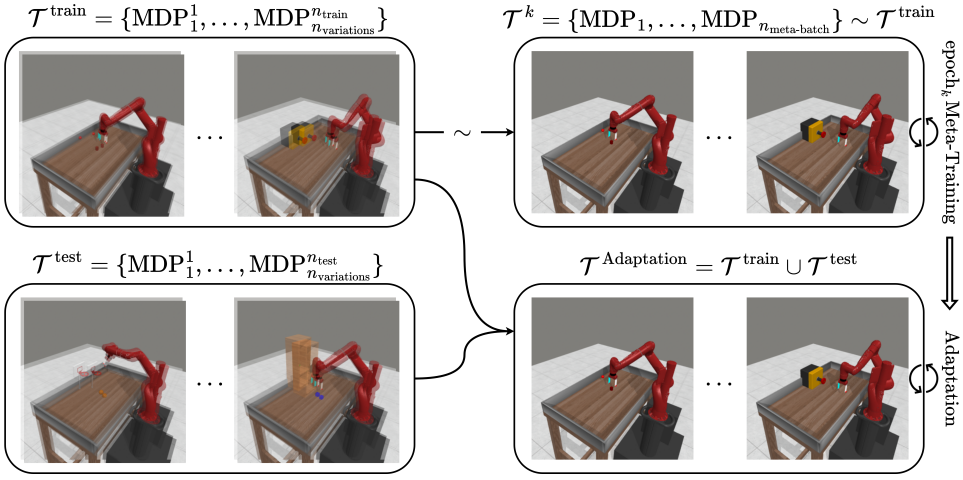


Figure 6.9: Distribution of task variations in MetaWorld ML-10

Following some isolated update iterations of the RL agent, i.e., actor and critic networks, evaluation is then performed on all variants of both the training and test sets $\mathcal{T}^{\text{train}} \cup \mathcal{T}^{\text{test}}$. Figure 6.9 illustrates how the environments are sampled in meta-training and adaptation phases. It should be noted that the RL updates need a separate buffer for keeping all the action and belief information in the interactions. To quantify adaptation to each task, the average return and success rate of all its variations are computed. This separation enables detailed analysis of generalization to familiar and unfamiliar environments under both identity and parametric diversity.

In addition to computing the performance metrics over each n_H-episode epoch, the success rate and return of the final episode in each sequence are also logged. This provides a direct measure of how accumulated context over n_H episodes contributes to performance. By comparing the statistics from the average episodes and final episodes within a block, the effect of belief update on exploration can be quantified.

The implementation in this experiment uses an on-policy PPO agent [139]. As pointed out previously, the policy uses an online buffer that stores the most recent n_H episodes. After each meta-episode, the policy is updated for n_{steps}^{π} iterations. In parallel, an offline buffer keeps contextual trajectories of interactions with the training set $\tau_{0:H^+}^{\mathcal{T}_k}$ from all epochs. The belief model is updated after every n_H episodes using this stored data. Again, it is worth mentioning that the belief model is completely isolated from the PPO policy, i.e., there is no shared gradient flow. This design choice avoids instability from entangled updates and ensures modularity.

CRAFT is compared against several on-policy meta-RL baselines. These include

MAML and RL^2, widely known meta-RL baselines, as well as VariBAD, the Bayes-adaptive meta-RL method with memory-based belief model, and SDVT [223]. SDVT aimed to extend VariBAD by improving generalization to non-parametric task variations. To do so, SDVT proposed a more rigorous task decomposition into subtasks by learning a Gaussian mixture VAE updated by minimizing categorical and occupancy losses, and a virtual training strategy. Hypothetical latent representations, i.e., subtask composition in their work, are introduced to the RL agent in virtual training.

6.4.2 Training Implementation Details

To support reproducibility and clarity, the full training framework is outlined in Algorithm 3. It summarizes the rollout process and shows how the belief model and PPO policy are trained concurrently but independently, using separate buffers populated during meta-rollouts. In regard to baselines, for MAML and RL^2, the implementations from [224] are used[1]. The implementations from the SDVT paper are also adopted for their algorithm and VariBAD[2]. Finally, the ML-10 benchmark is borrowed from the well-maintained Metaworld simulation repository[3].

The detailed hyperparameter configuration used throughout the experiments is provided in Table 6.1, reporting relevant configurations for the belief inference module, the transformer architecture, and the reinforcement learning agent.

6.4.3 Results and Analysis

Here, the findings of the aforementioned experiments are explained in two categories of quantitative and qualitative comparisons.

Quantitative Findings

The quantitative evaluation focuses on two performance metrics, return and success rate, across all task variations in the MetaWorld ML-10 benchmark. The average values of these metrics are plotted against the number of training interaction frames in Figure 6.10 and Figure 6.11. Results are reported separately for training and test environments, ensuring that generalization can be meaningfully assessed. Table 6.2 and Table 6.3 present more detailed numerical findings, showing the post-training

[1] https://github.com/rainx0r/metaworld-algorithms
[2] https://github.com/suyoung-lee/SDVT
[3] https://github.com/Farama-Foundation/Metaworld

Algorithm 3: Belief Inference from Action-Free Transformer Model

Input: Training and test tasks $\mathcal{T}^{\text{train}}, \mathcal{T}^{\text{test}}$;
Transformer: encoder f^s, decoder f^r, n_{head} attention heads, n_{block} transformer blocks, n_{FFN} units in feedforward network, inference head f^{ψ}, reward and state predictors $f_{\theta_r}, f_{\theta_s}$;
Rollout Horizon H, n_H rollouts in meta-episode, policy π_ζ, value function V_ν;
Belief buffer $\mathcal{B}_{\text{belief}}$, rollout buffer $\mathcal{B}_{\text{RL}}$, ELBO batch size n_{ELBO}
for *each meta-episode* $k = 1$ *to* n_{meta} **do**
  Sample tasks from the training tasks $\mathcal{T}^{\text{meta-batch}} \sim \mathcal{T}^{\text{train}}$;
  Reset hidden state h_0, $z_0 = (\mu_\psi(h_0), \sigma_\psi(h_0))$, $\mathcal{B}_{\text{RL}}$, and $\tau^{\text{action-free}}$;
  for $t = 0$ *to* $n_H \times H - 1$ **do**
    if $t \mod H = 0$ **then**
      Reset tasks and sample $s_0 \sim \mathcal{P}_{S_0}^{\text{meta-batch}}(.)$;
      Append $(\varnothing, s_0)$ to $\tau^{\text{action-free}}$;
      $s_t \leftarrow s_0$
    Sample actions $a_t \sim \pi_\zeta(.|s_t, z_t)$ Execute the actions, observe next states s_{t+1}, and receive rewards r_{t+1};
    Store $(s_t, z_t, r_{t+1}, a_{t+1})$ in $\mathcal{B}_{\text{RL}}$;
    Append (r_{t+1}, s_{t+1}) to $\tau^{\text{action-free}}$;
    Embed $[s_0, \ldots, s_{t+1}] \subset \tau^{\text{action-free}}$ and apply Rotary Positional Encoding $\rightarrow [\mathbf{E}'^s_0, \ldots, \mathbf{E}'^s_{t+1}]$;
    Apply causal self-attention:
    $\mathbf{u}^s_t = \text{FFN}(\text{Multi-Head}(\text{CausalSelfAttn}))(f^s_q(\mathbf{E}'^s), f^s_k(\mathbf{E}'^s), f^s_v(\mathbf{E}'^s))$;
    Embed $[0, r_1, \ldots, r_t] \subset \tau^{\text{action-free}}$ and apply Rotary Positional Encoding $\rightarrow [\mathbf{E}^r_0, \ldots, \mathbf{E}^r_t]$;
    Apply causal self-attention:
    $\mathbf{u}^r_t = \text{FFN}(\text{Multi-Head}(\text{CausalSelfAttn}))(f^r_q(\mathbf{E}'^r), f^r_k(\mathbf{E}'^r), f^r_v(\mathbf{E}'^r))$;
    Fuse via causal cross-attention:
    $h_t = \text{FFN}(\text{Multi-Head}(\text{CausalCrossAttn}))(f^r_q(\mathbf{u}^s), f^r_k(\mathbf{u}^s), f^r_v(\mathbf{u}^r))$;
    Update $z_t = (\mu_\psi(h_t), \sigma_\psi(h_t))$;
    Sample $b_t \sim \mathcal{N}(\mu_\psi(h_t), \sigma_\psi(h_t))$;
  Store $\tau^{\text{action-free}}_{0:t+1}$ in $\mathcal{B}_{\text{belief}}$;
  `/* Update */`
  Sample trajectories $\tau^{1:n_{\text{ELBO}}} \sim \mathcal{B}_{\text{belief}}$ and infer belief $\{\{b_t(\tau^i_{0:t+1})\}_{t=0:H^+-1}\}^{i=1:n_{\text{ELBO}}}$;
  Optimize $\mathcal{L}_{\text{ELBO}} = \sum_\tau \sum_{t=0}^{H^+-1} \mathcal{L}_{\text{ELBO},t}$ with:
  $\mathcal{L}_{\text{ELBO},t} = \mathbb{E}_{q_\phi(b_t|\tau_{0:t+1})}\left[\beta^{\text{S}}\mathcal{L}^{\text{S}}_{\text{Recon}} + \beta^{\text{R}}\mathcal{L}^{\text{R}}_{\text{Recon}} + \beta^{\text{KL}}\mathcal{L}_{\text{Regularization}}\right]$;
  Optimize PPO $\mathcal{L}_\pi$, $\mathcal{L}_V$ to update π_ζ, V_ν;

generalization performance in each environment for CRAFT and other Bayes-adaptive

baselines.

The transformer-based belief model consistently outperforms the baseline algorithms across many environments. The performance advantage is most pronounced in training environments. Although CRAFT uses less information and a less constrained model, it achieves comparable, and in some cases even better, results on the

Table 6.1: Action-Free Transformer Belief Model and RL Hyperparameters

Category	**Description**	**Value**
Belief Model	Optimizer	Adam
	Learning rate	1×10^{-3}
	Buffer size (meta-episodes)	1000
	n_{ELBO}	10
	ELBO KL coefficient (β^{KL})	0.1
	ELBO reward coefficient (β^{R})	10
	ELBO transition coefficient (β^{S})	200
	Number of updates per epoch	10
	Reward reconstruction objective	MSE
	State reconstruction objective	MSE
Transformer	d_q, d_k, d_v	256
	n_{block}	1
	n_{head}	4
	n_{FFN}	512
	State embedding size	32
	Reward embedding size	16
	Latent dimension (d_h)	5
	Reward decoder network	[64, 64, 32]
	State decoder network	[64, 64, 32]
PPO	Optimizer	Adam
	Learning rate	7×10^{-4}
	γ	0.99
	Clip factor	0.1
	τ	0.9
	Entropy Coefficient	1e−3
	Mini-batch size	10
	Architecture	$[256, 256]$
	Policy activation	tanh
	Policy state embedding size	64
	Policy belief embedding size	64

test environments that involve unfamiliar dynamics and temporal dependencies.

Another finding is that in the early stages of training, when the belief model is immature, the performance of memory-based methods is better compared to CRAFT. However, after the first few evaluations, the action-free transformer model begins to produce meaningful guidance for the context-adaptive PPO more effectively than the baselines. This leads to higher success rates and returns in later stages, where it seems the action-free transformer models have not yet even reached their full potential. This can be attributed to the complexity of training transformer models; they are slightly slower but eventually effective, even when consisting of only a small number of blocks.

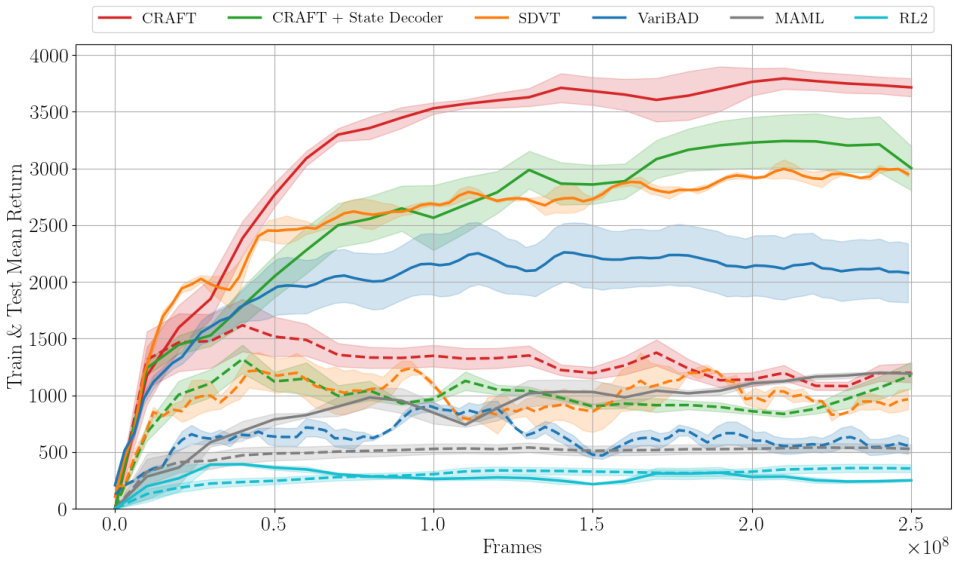


Figure 6.10: Average return of different methods in evaluation on training and test environments of the MetaWorld ML-10

During many stages of training, CRAFT achieves higher final-episode return and success compared to the mean over the n_H episodes in a meta-episode, denoted as $R_f - \overline{R}_{H^+}$ and $S_f - \overline{S}_{H^+}$, respectively. The corresponding values are plotted against the number of meta-training interaction frames in Figures 6.12 and 6.13. As can be seen, this relative advantage remains in the later stages of training for the action-free methods, suggesting that the policy can still benefit from continued exploration and training. This indicates that accumulated experience within the meta-episode contributes meaningfully to improved performance, reflecting effective belief updates in the BAMDP framework. In contrast, other baseline methods show weaker improvement—or even degradation in some cases—between early and final episodes. The

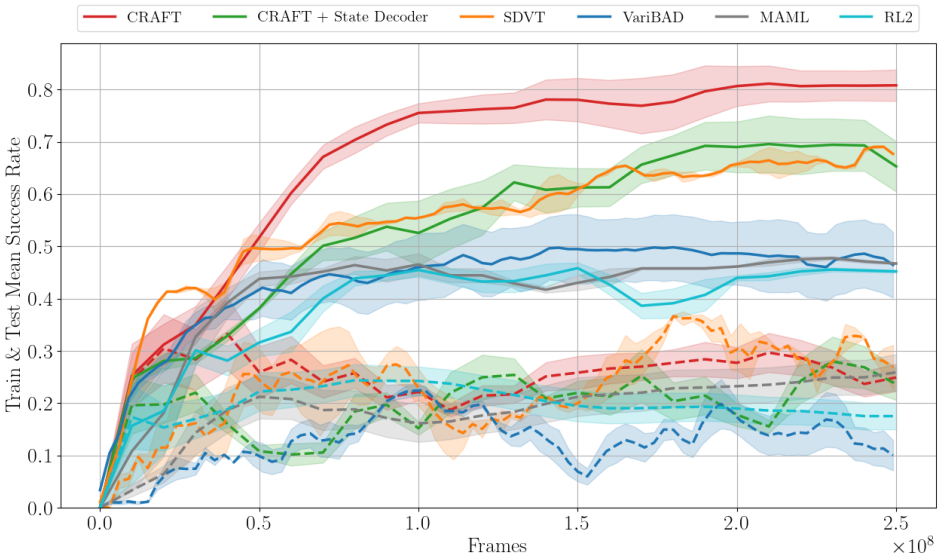


Figure 6.11: Average success rate of different methods in evaluation on training and test environments of the MetaWorld ML-10

ability of transformer-based models to better extract information from long contexts compared to memory-based sequence models contributes to this result. It should also be noted that CRAFT, particularly when a state decoder is included during training, shows the most noticeable benefit during intermediate training stages, when the belief model is still developing and the policy is actively exploring. This further supports the value of gradient isolation in transformer-based context encoder training, which enables informative task inference independently and provides a strong prior for the adaptation process.

Qualitative Findings

To further analyze the structure of the learned latent space, the pairwise projections of the five-dimensional task belief vectors are plotted. A 2-D plot is generated for each pair of the latent space dimensions, resulting in ten distinct subplots covering all one-vs-one combinations shown in Figure 6.14. Some task environments cluster closely in certain latent dimensions while remaining dispersed in others, suggesting that different latent axes are responsible for encoding distinct aspects of task variation. A 3-D projection of the latent vectors is illustrated in Figure 6.15. By finding the means and variances of the 3-D UMAP projections computed for the latent representations of each of the 15 tasks [4], the estimated distributions are visualized as ellipsoids,

Table 6.2: Return comparison among methods across MetaWorld ML-10 tasks

Task	**CRAFT**	**CRAFT + State Decoder**	SDVT	**VariBAD**
Reach	$\mathbf{3696.9 \pm 136.0}$	3641.1 ± 162.0	3016.8 ± 216.5	3588.7 ± 220.5
Push	$\mathbf{3000.2 \pm 833.0}$	2360.2 ± 783.5	2445.6 ± 250.2	889.7 ± 614.1
Pick-Place	$\mathbf{1528.7 \pm 434.0}$	1029.1 ± 370.9	1180.0 ± 4.6	382.7 ± 266.4
Door Open	3705.0 ± 770.4	3625.8 ± 477.7	$\mathbf{4339.0 \pm 96.6}$	2797.8 ± 1082.3
Drawer Close	4734.1 ± 97.0	4749.0 ± 64.7	$\mathbf{4777.9 \pm 9.4}$	4694.9 ± 3.1
Button Press Topdown	2977.0 ± 620.2	2665.0 ± 244.0	$\mathbf{3134.4 \pm 270.1}$	1993.9 ± 613.4
Peg Insert Side	$\mathbf{2455.5 \pm 722.4}$	1381.9 ± 467.0	1451.6 ± 172.0	774.6 ± 540.7
Window Open	3692.2 ± 833.5	$\mathbf{4289.6 \pm 43.4}$	4219.9 ± 101.7	3081.2 ± 780.5
Sweep	$\mathbf{3332.7 \pm 921.4}$	2293.3 ± 735.3	2820.2 ± 227.0	1527.8 ± 1050.8
Basketball	$\mathbf{2545.5 \pm 781.8}$	1346.4 ± 502.7	1722.6 ± 110.9	868.9 ± 609.2
Drawer Open	2143.4 ± 313.3	2043.6 ± 40.3	$\mathbf{2144.3 \pm 84.0}$	1589.5 ± 352.7
Door Close	791.0 ± 359.2	318.3 ± 72.6	$\mathbf{1185.7 \pm 53.0}$	261.8 ± 4.8
Shelf Place	$\mathbf{409.9 \pm 123.3}$	232.0 ± 77.4	354.3 ± 50.3	61.1 ± 43.2
Sweep Into	$\mathbf{893.2 \pm 324.3}$	504.5 ± 141.1	754.8 ± 145.7	359.0 ± 223.2
Lever Pull	$\mathbf{346.7 \pm 39.0}$	324.6 ± 18.5	292.6 ± 26.9	337.5 ± 2.9
Train Avg.	**3166.8**	2738.1	2910.8	2060.03
Test Avg.	916.82	684.6	**946.4**	521.8
Total Avg.	**2416.8**	2053.6	2256.0	1547.3

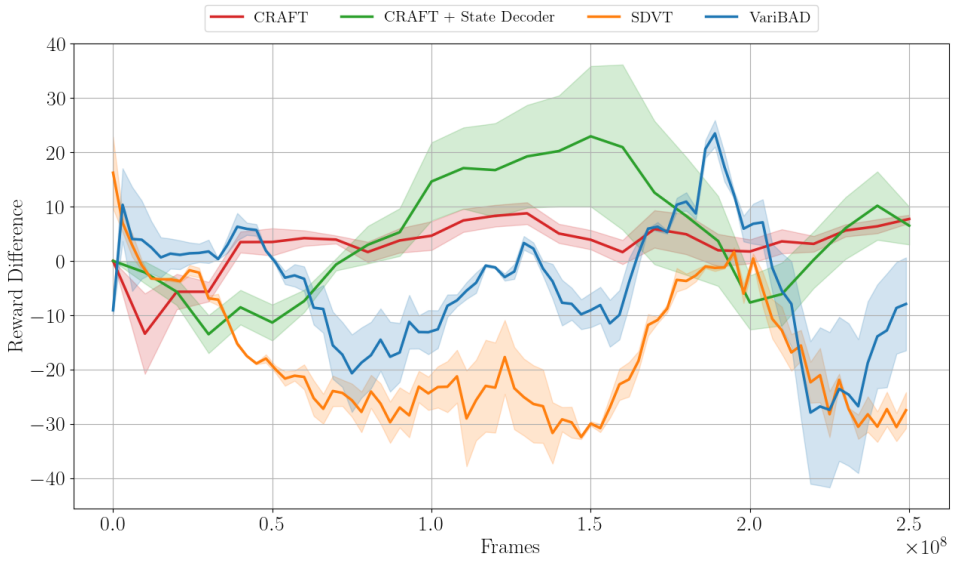


Figure 6.12: The difference between the return in the last episode of an adaptation meta-episode and the average ($R_f - \overline{R}_{H^+}$)

indicating the variability region of each task in the latent space.

Figure 6.16 shows the average latent vector in all evaluation episodes of all varia-

Table 6.3: Success rate comparison among methods across MetaWorld ML-10 tasks

Task	CRAFT	CRAFT + State Decoder	SDVT	VariBAD
Reach	0.401 ± 0.073	0.413 ± 0.061	0.363 ± 0.002	$\mathbf{0.487 \pm 0.015}$
Push	$\mathbf{0.555 \pm 0.158}$	0.491 ± 0.165	0.497 ± 0.078	0.150 ± 0.106
Pick-Place	0.394 ± 0.116	0.227 ± 0.101	$\mathbf{0.410 \pm 0.064}$	0.078 ± 0.054
Door Open	0.798 ± 0.223	0.812 ± 0.148	$\mathbf{0.997 \pm 0.002}$	0.501 ± 0.353
Drawer Close	$\mathbf{1.000 \pm 0.000}$	0.995 ± 0.004	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$
Button Press Topdown	0.918 ± 0.089	$\mathbf{0.946 \pm 0.033}$	0.943 ± 0.035	0.795 ± 0.145
Peg Insert Side	$\mathbf{0.530 \pm 0.171}$	0.175 ± 0.061	0.210 ± 0.021	0.103 ± 0.073
Window Open	0.821 ± 0.194	0.990 ± 0.011	$\mathbf{1.000 \pm 0.000}$	0.797 ± 0.134
Sweep	$\mathbf{0.774 \pm 0.217}$	0.561 ± 0.187	0.713 ± 0.071	0.377 ± 0.266
Basketball	$\mathbf{0.664 \pm 0.196}$	0.238 ± 0.163	0.503 ± 0.016	0.170 ± 0.120
Drawer Open	0.428 ± 0.203	$\mathbf{0.529 \pm 0.005}$	0.427 ± 0.071	0.261 ± 0.122
Door Close	0.200 ± 0.133	0.014 ± 0.014	$\mathbf{0.223 \pm 0.071}$	0.020 ± 0.004
Shelf Place	0.007 ± 0.007	0.000 ± 0.000	$\mathbf{0.013 \pm 0.009}$	0.000 ± 0.000
Sweep Into	0.303 ± 0.134	0.364 ± 0.121	$\mathbf{0.523 \pm 0.059}$	0.213 ± 0.151
Lever Pull	$\mathbf{0.007 \pm 0.007}$	0.004 ± 0.004	0.000 ± 0.000	0.000 ± 0.000
Train Avg.	**0.690**	0.580	0.660	0.450
Test Avg.	0.190	0.180	**0.237**	0.100
Total Avg.	0.520	0.450	**0.530**	0.330

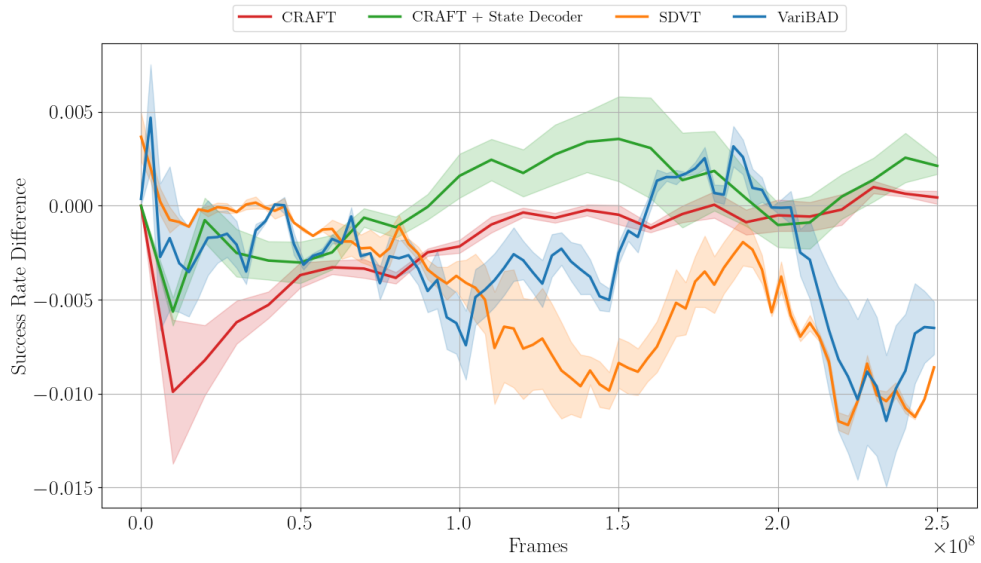


Figure 6.13: The difference between the success rate in the last episode of an adaptation meta-episode and the average ($S_f - \overline{S}_{H^+}$)

tions of each environment. Some tasks consistently exhibit large values along specific

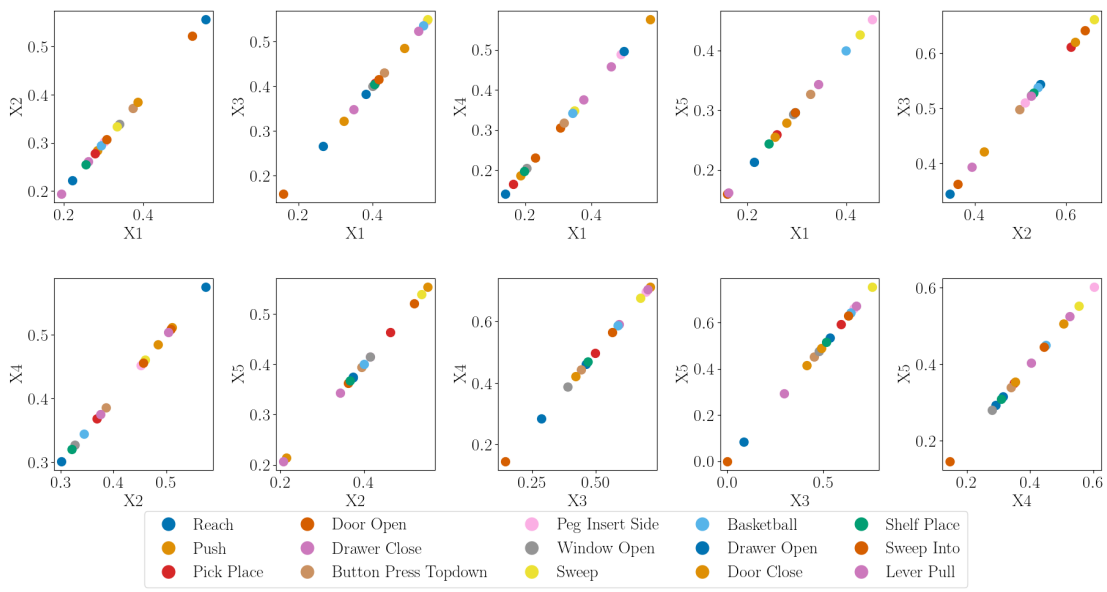


Figure 6.14: Learned 5-dimensional latent representations associated with different tasks in the MetaWorld ML-10 benchmark. Each pair of two elements of the average task belief vector is plotted against each other.

latent dimensions, effectively dominating those axes. For instance, the *lever-pull* task is represented by a very low value along axis one and a large value on axis three, while the *door-open* task shows the highest value on axis one compared to all others. In general, conceptually similar tasks tend to share these dominant dimensions, indicating that the model has learned a structured and interpretable belief space. This organization reflects the transformer encoder-decoder's ability to disentangle task-specific features. CRAFT was able to extract meaningful representations without access to action information, supporting its role as a reliable inference module in adaptive meta-RL.

6.4.4 Summary and Discussion

The experiments presented in this chapter evaluate CRAFT, the proposed transformer-based belief model for action-free variational task inference in robot manipulation with meta-reinforcement learning. Using a well-maintained benchmark consisting of goal-oriented robotic manipulation tasks, the model is assessed on both adaptation performance and quality of learned task representations.

CRAFT consistently demonstrated strong adaptation performance in training environments, outperforming established baselines such as RL^2, MAML, VariBAD, and

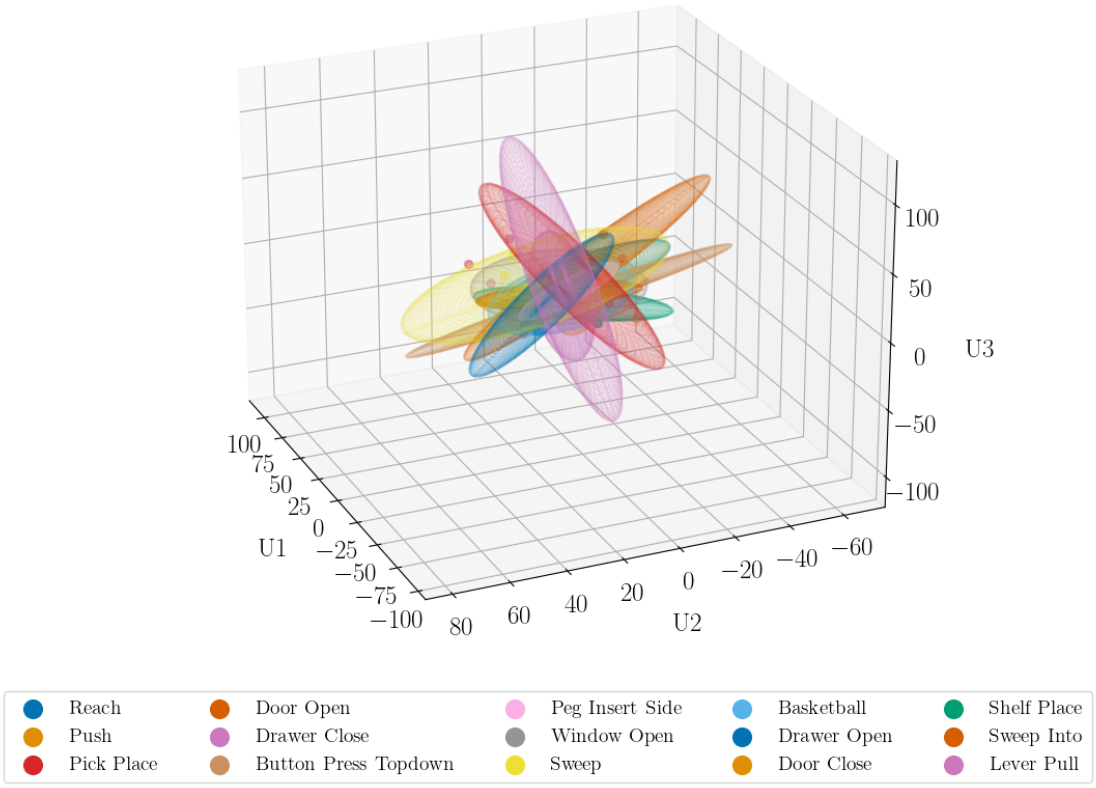


Figure 6.15: 3-D projection of latent representations associated with different tasks in the MetaWorld ML-10 benchmark. The projection is computed using the UMAP method [4]. An approximate Gaussian 3-D ellipsoid is illustrated for each task.

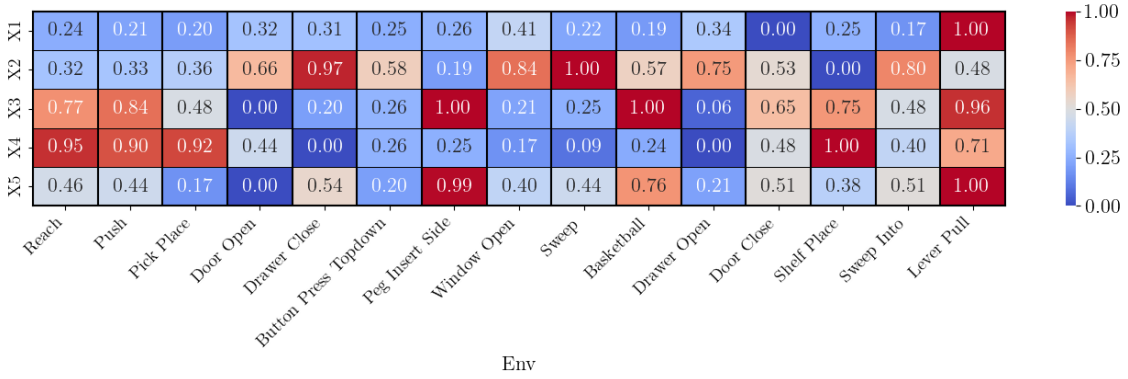


Figure 6.16: The numerical values of the average task belief vector across all tasks in the MetaWorld ML-10 benchmark

SDVT by a considerable margin. This means that without action information, Bayes-adaptive sampling of sequences of states and rewards offers sufficient structure to

support effective task inference and policy adaptation. While this method maintains competitive performance on generalization and adaptation to test environments, additional virtual training used in SDVT proves beneficial for unseen tasks by training the policy on overlapping belief representations. As a result, SDVT achieves comparable results in hold-out environments, and its virtual training approach can be an effective addition to CRAFT. Nonetheless, the model maintains a significant and consistent advantage in training tasks, reflecting efficient context inference and adaptation in familiar settings.

The analysis of final-episode performance as opposed to meta-episode average highlights the effectiveness of the proposed action-free method in active exploration and leveraging accumulated context. This advantage persists throughout training, especially in intermediate and later stages, suggesting that belief refinement in this model can continue to support improved adaptation. Hence, it was indicated that transformer-based models demonstrate a stronger capacity to extract information from longer sequences compared to memory-based baselines.

Also, based on the ablation study of two different belief models, the inclusion of a state decoder during training, although beneficial in exploration, overregulates the latent representation and harms the adaptation performance. That is since the context encoder is action-free, the state decoder serves as an unnecessary information bottleneck for its use of actions, which are deemed privileged information in this case. Additionally, it was qualitatively confirmed that the learned latent space shows coherent structure across tasks.

In summary, CRAFT provides a robust and flexible alternative for creating task inference modules in meta-reinforcement learning, especially where action logging is infeasible or undesirable. Its generalization capabilities and action-agnostic architecture make it suitable for real-world deployment in adaptive control systems.

6.5 Conclusions and Outlook

This section summarizes the chapters objectives, method, and results, followed by the individual future implications of the presented work in this chapter for a possible research outlook.

6.5.1 Summary and Contributions

This chapter introduced an encoder-decoder transformer model designed for task inference from action-free trajectories in the context of meta-reinforcement learning. Motivated by the challenge of context inference under partial observability, the method leverages relatively long sequences of solely states and rewards to infer a latent belief over tasks without relying on access to the actions executed by the agent.

The transformer, equipped with rotary positional encodings, uses causal self-attention amongst states and rewards in the encoder and decoder blocks, respectively. These two modules extract the necessary dynamic information approximating the transition and optimality properties associated with the task. Finally, in the shifted causal cross-attention mechanism in the decoder, these state and reward dynamic features are fused to create a variational representation of the understanding of the task. Samples from the latent distributions are then used for learning by optimizing an evidence lower bound objective on the future reward prediction loss. To complete the Bayes-adaptive MDP formulation, the latent distribution is then used to augment the state observations of a PPO RL agent.

Evaluation is conducted on a robotic manipulation benchmark, MetaWorld ML-10, comparing CRAFT with state-of-the-art Bayes-adaptive and other conventional baselines. Results indicated that the action-free transformed encoder-decoder achieves better performance and more effective belief update throughout a meta-episode. Specifically, the adaptation prowess of this belief model is observed in variations of training environments, and to a lesser degree in completely unfamiliar test environments. Since transformers are good at handling long sequences, the belief update advantage was considerably bigger compared to memory-based baselines. This was more emphasized in intermediate and later training stages, where accumulated context enabled improved exploration guidance.

Additionally, ablation studies revealed that including a state decoder for the belief model improved belief update advantage but hindered adaptation due to reliance on privileged action information in training. It should also be noted that training the RL agent with virtual samples in the latent space, as used in SDVT, can enhance adaptation to unseen environments to the same level as CRAFT. This is believed to be because the RL agent was introduced to smoother overlapping sub-regions of the task latent space. The integration of this virtual training regime into CRAFT is straightforward and possibly promising, yet it was not explored in this work. Finally,

this chapter presented a flexible, scalable approach for task inference in the absence of action information.

6.5.2 Future Directions

The method presented here improves on some adaptation metrics and shows a strong quality of task representation. To do so, a major contributing factor to the dynamic definition of an MDP, the actions, was removed from the belief model, and this lack of information was compensated for by the suitable design of a transformer encoder-decoder that captures useful features from a less informative combination of sequences. The assumption under which this redesign was based, i.e., the minimal role of action penalties in reward definition, is present in goal-oriented robotic tasks. However, the same approach is limited in application by the foundational concepts of what is deemed *success* in other domains. Additionally, transformers, even a single-layer architecture as used for this belief model, are notorious for introducing computational overhead, particularly during inference.

Considering these limitations, future research is now possible based on the flexible design proposed and validated here. The abundance of action-free robot manipulation trajectories has energized pre-training and semi-supervised learning in offline reinforcement learning from image observations [225, 226]. This shows the importance of using numerous datasets of recorded demonstrations to facilitate generalizability in robotic manipulation. The proposed action-free belief model, trained independently from the RL agent, holds promise as a tool for providing RL agents with auxiliary task representations to support adaptation to new tasks.

Considering the mentioned line of research, two possibilities may emerge from using the inherent flexibility of CRAFT in this chapter:

1. Pre-training the belief model with action-free video recordings of various robot manipulation tasks, which requires more training investment because of the transformer architecture, and then online adaptation to a new task in a similar environment.

2. Cross-modal adaptation to environments with different state observation modes.

The former possibility is quite straightforward in terms of implementation. However, the latter needs further explanation. As an example, consider a robotic environment expressed by two different MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ per Equation (6.5). These

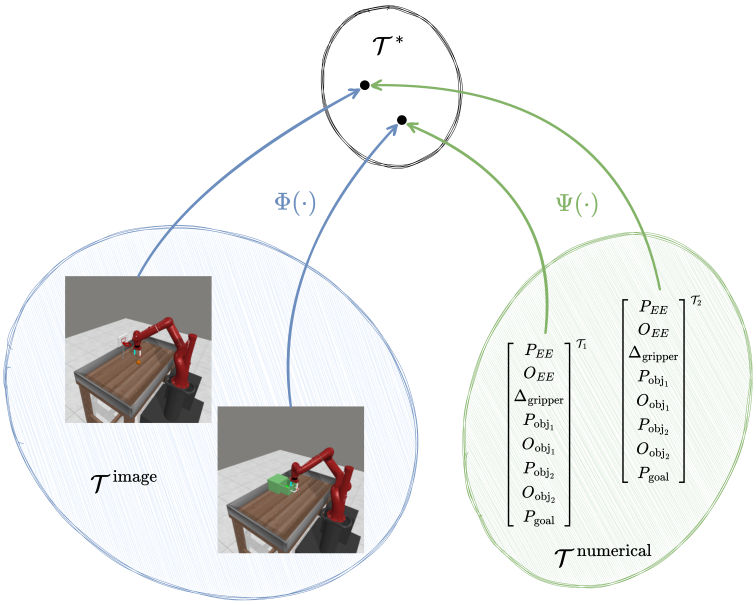


Figure 6.17: A conceptual shared task latent representation for MDPs with different state observation modalities

two MDPs share the same action space ($\mathcal{A}$), dynamic functions ($\mathcal{P}_S, \mathcal{P}_R$), discount factor ($\gamma$), and episode length ($H$). However, the two MDPs differ in the state space domains $\mathcal{S}_1$ and $\mathcal{S}_2$ and optimal policies π_1^* and π_2^*, respectively. For instance, $\mathcal{M}_1$ can receive state observations in the numerical space from sensor data, versus $\mathcal{M}_2$ receiving workspace images as observations. Assuming the essential Markov property holds for both MDPs, it is possible to infer a shared latent domain where $\Psi(\mathcal{S}_1) \cong \Phi(\mathcal{S}_2)$, as conceptualized in Figure 6.17, such that $\Psi(\mathcal{M}_1) \simeq \Phi(\mathcal{M}_2)$. These transformed MDPs can be governed with a shared optimal policy $\pi^* \simeq \pi^*(\Psi(\cdot)) \simeq \pi^*(\Phi(\cdot))$.

If learning the mappings $\Psi(\cdot)$ and $\Phi(\cdot)$, and their inverse functions $\Psi^{-1}(\cdot)$ and $\Phi^{-1}(\cdot)$, is tractable, it is possible to train an action-agnostic, and hence agent-agnostic, foundational transformer encoder-decoder from offline data in the source domain and fine-tune it with additional adapters to enable adaptation in the target domain. This means the proposed belief model can be trained on an MDP with numerical state observations. By learning observation adapters q^{belief} and q^{policy}, and fine-tuning the RL agent consisting of value function V_ν and π_ζ, the context-adaptive meta-RL pipeline can acclimate to visual state observations online as outlined in Figure 6.18.

To reiterate, the inherently less structured design of CRAFT enables its flex-

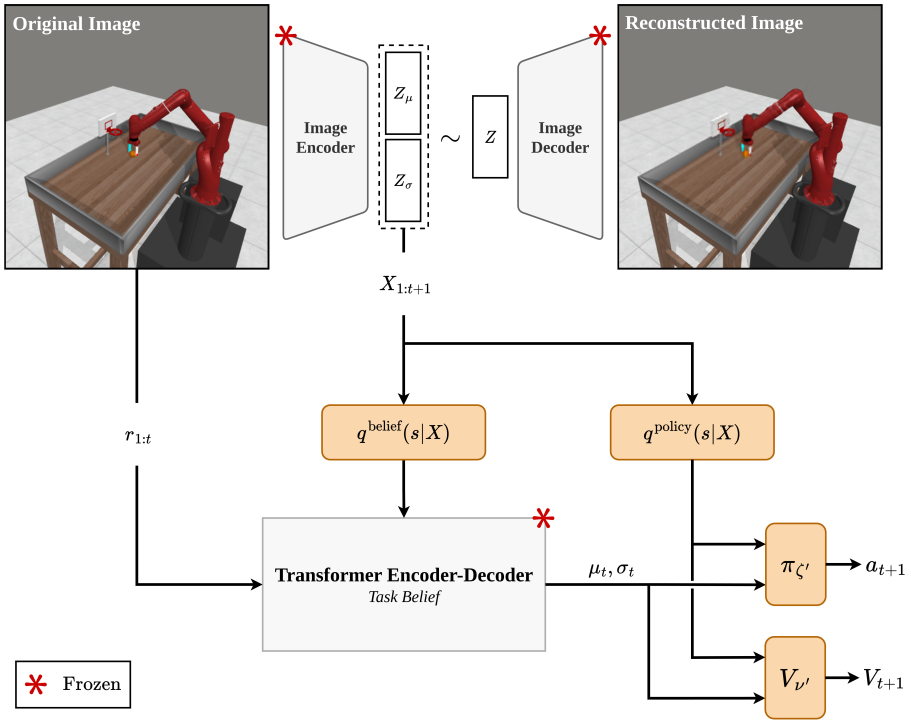


Figure 6.18: Outline of a possible research on cross-modal adaptation scenario

ible application to multi-task robotic manipulation via offline pre-training or cross-modal adaptation. Nonetheless, action-free task inference with a transformer encoder-decoder enables flexible adaptation in robotic manipulation environments where actions may be unavailable, expensive to track, or noisy.

Chapter 7

Concluding Remarks

7.1 Synopsis

The overarching theme of this dissertation has been to explore the fundamental mechanisms necessary to grant ML-based agents for robot arm manipulation with the capability to adapt efficiently and robustly to changes in their tasks and operating environments. Three interlocking contributions, Demo-EASE, RT-IS, and Action-agnostic Belief Model, form a framework for sample-efficient, real-world-deployable, and task-adaptive reinforcement learning in robotics.

Each component was inspired by real-world constraints: data scarcity and limited workspace in learning from demonstration, failure of direct sim-to-real policy deployment, and the challenge of inter-task generalizability. These were respectively addressed by:

- Exploiting symmetry and task structure through abstraction and online adaptive trade-off between imitation and learning in Demo-EASE,
- Leveraging intrinsic noise in real-time simulators, a previously almost overlooked feature, via RT-IS,
- Complete decoupling of action dependency in context inference with transformer-based models in CRAFT for improved adaptation and facilitation of pre-training with action-free observations from a heterogeneous domain.

Together, these solutions show that adaptability in robotics is not only possible in the form of reactive adjustment, but it can be built into the agent architecture or training regime proactively. It was achieved by introducing modular learning

pipelines and flexible inference components. This work also emphasizes the importance of compositionality for each contribution operates as a plug-in mechanism that can be integrated into a broader adaptive robotics framework. This opens the door to scalable solutions where system components can be improved independently and implemented collectively for enhanced robotic autonomy.

7.2 Practical Significance

The practical contributions of this work can be seen in the potential to reduce impediments in the real-world deployment of reinforcement learning for robotics. Learning from demonstration is bottlenecked by the effort required to collect consistent, yet diverse, expert data. Demo-EASE relaxes this limitation by enabling the reuse of abstract demonstrations across state space sub-regions through the definition of *General Abstract Symmetry*.

Sim-to-real transfer, known for facing the *reality gap*, will benefit from RT-IS, which circumvents the need for exhaustive manually-tuned domain randomization. RT-IS enables automated noise introduction, looser modeling precision thresholds, and robust results.

Finally, the CRAFT belief model supports more flexible interaction by removing the action conditioning in belief posteriors. It can also provide a capability to use observations from different domains to bootstrap the belief model as a pre-trained auxiliary to an RL agent deployed in a new task, possibly also interacting in a different domain. This is critical for many real-world multi-task robot applications, such as surgical robotics, collaborative assembly, and assistive robots in unstructured domestic environments. Furthermore, the model's compatibility with partial observability conditions makes it well-suited for robotic applications in search and rescue, remote inspection, or field robotics.

Taken together, the three components introduced in this dissertation reflect a more general shift in robot manipulation: from monolithic, task-specific systems to modular, adaptable pipelines that can cope with a variety of uncertainties and changes. By addressing the three challenges of efficiency, deployability, and generalization, this dissertation aimed to step on a path toward practical robot intelligence.

7.3 Limitations and Future Work

While this dissertation introduces novel components for building a foundation, they are limited in their definitions, and several avenues remain open for future research.

First, Demo-EASE depends on the presence of symmetry in the workspace. Future work could incorporate unsupervised symmetry detection or relational reasoning models to extend applicability. In general, graph-based geometry learning could help generalize the notion of symmetry beyond mirroring to include spatial or temporal similarities.

Second, RT-IS assumes that simulator-induced stochasticity sufficiently approximates real-world variation. This assumption was verified, and RT-IS proved to be powerful in the transfer between the simulator and the specific robot used in this dissertation. However, with every new setup, there is room for doubt about this assumption. To improve transferability, an active learning approach that augments simulator noise profiles using online real-world feedback could be explored. Additionally, the randomization and noise incorporated in the experiments provided here are all concerned with robot low-level control and dynamics. RT-IS has not been adopted or tested in other robotic-related sim-to-real issues, e.g., regarding computer vision.

Third, the transformer in CRAFT, even a single-layer architecture, introduces computation overhead, particularly during inference. Possible solutions include state-space transformer models [227], distillation-based approaches [228], efficient online inference [229], and sparse attention mechanisms [230] to ensure scalability to more diverse and long-horizon tasks. Embedding priors into the transformer's positional encoding or architecture could further reduce data dependence and improve inference quality. This belief model was introduced for and validated on robot manipulation tasks, on the assumption that rewards in robotics are highly dependent on the states, and to a considerably lesser degree on the actions. This assumption could be violated in other domains of RL and meta-RL applications. Therefore, the performance of this method is open to exploration in these cases.

In summary, this dissertation advances the state of the art in adaptive learning for robotic manipulation. By targeting core challenges in efficiency, sim-to-real transfer, and context-aware generalization, it aims for robust and scalable autonomous systems. The hope is that the modularity and generality of the proposed methods inspire not only future research but also practical adoption across diverse fields of robotics, from industrial automation to assistive care and beyond.

Bibliography

[1] Zengjie Zhang. *Towards Safe Human-Robot Collaboration Oriented to Collision Handling*. PhD thesis, Technische Universität München, 2021.

[2] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

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

[4] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[5] Amir M. Soufi Enayati, Zengjie Zhang, and Homayoun Najjaran. A methodical interpretation of adaptive robotics: Study and reformulation. *Neurocomputing*, 512:381–397, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.09.114. URL https://www.sciencedirect.com/science/article/pii/S0925231222012073.

[6] Kashish Gupta. *Reinforcement learning in complex environments with locally trained naïve agents*. PhD thesis, University of British Columbia, 2021.

[7] Kashish Gupta and Homayoun Najjaran. Exploiting abstract symmetries in reinforcement learning for complex environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3631–3637. IEEE, 2022.

[8] Amir M Soufi Enayati, Zengjie Zhang, Kashish Gupta, and Homayoun Najjaran. Exploiting symmetry and heuristic demonstrations in off-policy rein-

forcement learning for robotic manipulation. *arXiv preprint arXiv:2304.06055*, 2023.

[9] Zengjie Zhang, Jayden Hong, Amir M Soufi Enayati, and Homayoun Najjaran. Using implicit behavior cloning and dynamic movement primitive to facilitate reinforcement learning for robot motion planning. *IEEE Transactions on Robotics*, 2024.

[10] Amir M Soufi Enayati, Ram Dershan, Zengjie Zhang, Dean Richert, and Homayoun Najjaran. Facilitating sim-to-real by intrinsic stochasticity of real-time simulation in reinforcement learning for robot manipulation. *IEEE Transactions on Artificial Intelligence*, 5(4):1791–1804, 2023.

[11] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[12] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[13] Jiang Hua, Liangcai Zeng, Gongfa Li, and Zhaojie Ju. Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning. *Sensors*, 21 (4):1278, 2021.

[14] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

[15] Klaus Schwab. *The fourth industrial revolution*. Currency, 2017.

[16] Nasser Jazdi. Cyber physical systems in the context of industry 4.0. In *2014 IEEE international conference on automation, quality and testing, robotics*, pages 1–4. IEEE, 2014.

[17] Hang Su, Wen Qi, Jiahao Chen, and Dandan Zhang. Fuzzy approximation-based task-space control of robot manipulators with remote center of motion constraint. *IEEE Transactions on Fuzzy Systems*, 30(6):1564–1573, 2022. doi: 10.1109/TFUZZ.2022.3157075.

[18] ZM Bi, Yingzi Lin, and WJ Zhang. The general architecture of adaptive robotic systems for manufacturing applications. *Robotics and Computer-Integrated Manufacturing*, 26(5):461–470, 2010.

[19] Sachin Chitta, E Gil Jones, Matei Ciocarlie, and Kaijen Hsiao. Mobile manipulation in unstructured environments: Perception, planning, and execution. *IEEE Robotics & Automation Magazine*, 19(2):58–71, 2012.

[20] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2): 398–400, 2021.

[21] Toshio Fukuda and Takanori Shibata. Hierarchical control system in intelligent robotics and mechatronics. In *Proceedings of IECON'93-19th Annual Conference of IEEE Industrial Electronics*, pages 33–38. IEEE, 1993.

[22] George Edwards, Joshua Garcia, Hossein Tajalli, Daniel Popescu, Nenad Medvidovic, Gaurav Sukhatme, and Brad Petrus. Architecture-driven self-adaptation and self-management in robotics systems. In *2009 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 142–151. IEEE, 2009.

[23] Tønnes F Nygaard, Charles P Martin, David Howard, Jim Torresen, and Kyrre Glette. Environmental adaptation of robot morphology and control through real-world evolution. *Evolutionary Computation*, pages 1–21, 2021.

[24] Stefano Nolfi, Josh Bongard, Phil Husbands, and Dario Floreano. Evolutionary robotics. In *Springer handbook of robotics*, pages 2035–2068. Springer, 2016.

[25] Stefano Nolfi and Domenico Parisi. Learning to adapt to changing environments in evolving neural networks. *Adaptive behavior*, 5(1):75–98, 1996.

[26] Dario Floreano and Francesco Mondada. Evolution of plastic neurocontrollers for situated agents. In *Proc. of The Fourth International Conference on Simulation of Adaptive Behavior (SAB), From Animals to Animats*, page null. ETH Zürich, 1996.

[27] Evert Haasdijk, Nicolas Bredeche, and Agoston E Eiben. Combining environment-driven adaptation and task-driven optimisation in evolutionary robotics. *PloS one*, 9(6):e98466, 2014.

[28] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E Gusz Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015.

[29] Stavros Tripakis and Karine Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In *International Symposium on Formal Methods*, pages 233–252. Springer, 1999.

[30] Jing Yuan. Adaptive control of robotic manipulators including motor dynamics. *IEEE Transactions on Robotics and Automation*, 11(4):612–617, 1995.

[31] Hang Su, Yingbai Hu, Hamid Reza Karimi, Alois Knoll, Giancarlo Ferrigno, and Elena De Momi. Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results. *Neural Networks*, 131:291–299, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.07.033. URL https://www.sciencedirect.com/science/article/pii/S0893608020302744.

[32] Mei Liu, Jialiang Fan, Yu Zheng, Shuai Li, and Long Jin. A simultaneous learning and control scheme for redundant manipulators with physical constraints on decision variable and its derivative. *IEEE Transactions on Industrial Electronics*, 69(10):10301–10310, 2022. doi: 10.1109/TIE.2022.3165279.

[33] Zhengtai Xie, Long Jin, Xin Luo, Zhongbo Sun, and Mei Liu. Rnn for repetitive motion generation of redundant robot manipulators: An orthogonal projection-based scheme. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):615–628, 2022. doi: 10.1109/TNNLS.2020.3028304.

[34] Nicholas Roy and Sebastian Thrun. Motion planning through policy search. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2419–2424. IEEE, 2002.

[35] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[36] Maopeng Ran, Juncheng Li, and Lihua Xie. Reinforcement-learning-based disturbance rejection control for uncertain nonlinear systems. *IEEE Transactions on Cybernetics*, pages 1–13, 2021. doi: 10.1109/TCYB.2021.3060736.

[37] Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[38] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3876–3881. IEEE, 2014.

[39] Fangzhou Xiong, Biao Sun, Xu Yang, Hong Qiao, Kaizhu Huang, Amir Hussain, and Zhiyong Liu. Guided policy search for sequential multitask learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):216–226, 2018.

[40] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*, pages 817–828. PMLR, 2018.

[41] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[42] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[43] Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. Fast online adaptation in robotics through meta-learning embeddings of simulated priors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5269–5276. IEEE, 2020.

[44] 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.

[45] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, 2020.

[46] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR, 2019.

[47] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

[48] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

[49] Xi Chen, Ali Ghadirzadeh, Mårten Björkman, and Patric Jensfelt. Meta-learning for multi-objective reinforcement learning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 977–983. IEEE, 2019.

[50] Kashish Gupta, Debasmita Mukherjee, and Homayoun Najjaran. Extending the capabilities of reinforcement learning through curriculum: A review of methods and applications. *SN Computer Science*, 3(1):1–18, 2022.

[51] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[52] George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems*, 22:1015–1023, 2009.

[53] Juraj Holas and Igor Farkaš. Advances in adaptive skill acquisition. In *International Conference on Artificial Neural Networks*, pages 650–661. Springer, 2021.

[54] Manfred Eppe, Sven Magg, and Stefan Wermter. Curriculum goal masking for continuous deep reinforcement learning. In *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 183–188. IEEE, 2019.

[55] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in Neural Information Processing Systems*, 32:12623–12634, 2019.

[56] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[57] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

[58] David Martınez, Guillem Alenya, Tony Ribeiro, Katsumi Inoue, and Carme Torras. Relational reinforcement learning for planning with exogenous effects. *Journal of Machine Learning Research*, 18(78):1–44, 2017.

[59] David Martínez, Guillem Alenya, and Carme Torras. Relational reinforcement learning with guided demonstrations. *Artificial Intelligence*, 247:295–312, 2017.

[60] Tobias Lang, Marc Toussaint, and Kristian Kersting. Exploration in relational domains for model-based reinforcement learning. *The Journal of Machine Learning Research*, 13(1):3725–3768, 2012.

[61] Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4051–4058. IEEE, 2020.

[62] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

[63] Hamid Eghbal-zadeh, Florian Henkel, and Gerhard Widmer. Context-adaptive reinforcement learning using unsupervised learning of context variables. In

NeurIPS 2020 Workshop on Pre-registration in Machine Learning, pages 236–254. PMLR, 2021.

[64] Alessandro Achille, Giovanni Paolini, Glen Mbeng, and Stefano Soatto. The information complexity of learning tasks, their structure and their distance. *Information and Inference: A Journal of the IMA*, 10(1):51–72, 2021.

[65] Konstantinos Chatzilygeroudis, Vassilis Vassiliades, Freek Stulp, Sylvain Calinon, and Jean-Baptiste Mouret. A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, 36 (2):328–347, 2019.

[66] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[67] Marie Charbonneau, Valerio Modugno, Francesco Nori, Giuseppe Oriolo, Daniele Pucci, and Serena Ivaldi. Learning robust task priorities of qp-based whole-body torque-controllers. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE, 2018.

[68] Yueyue Liu, Zhijun Li, Huaping Liu, and Zhen Kan. Skill transfer learning for autonomous robots and human–robot cooperation: A survey. *Robotics and Autonomous Systems*, 128:103515, 2020.

[69] Dennis Mronga and Frank Kirchner. Learning context-adaptive task constraints for robotic manipulation. *Robotics and Autonomous Systems*, 141:103779, 2021.

[70] 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.

[71] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[72] Bruno Siciliano and Luigi Villani. A passivity-based approach to force regulation and motion control of robot manipulators. *Automatica*, 32(3):443–447, 1996.

[73] Zengjie Zhang, Marion Leibold, and Dirk Wollherr. Integral sliding-mode observer-based disturbance estimation for euler–lagrangian systems. *IEEE Transactions on Control Systems Technology*, 28(6):2377–2389, 2019.

[74] Seul Jung, Tien C Hsia, and Robert G Bonitz. Force tracking impedance control of robot manipulators under unknown environment. *IEEE Transactions on Control Systems Technology*, 12(3):474–483, 2004.

[75] Fotios Dimeas and Nikos Aspragathos. Online stability in human-robot cooperation with admittance control. *IEEE transactions on haptics*, 9(2):267–278, 2016.

[76] Tzyh-Jong Tarn, Yunying Wu, Ning Xi, and Alberto Isidori. Force regulation and contact transition control. *IEEE Control Systems Magazine*, 16(1):32–40, 1996.

[77] Yongchao Wang, Zengjie Zhang, Cong Li, and Martin Buss. Adaptive incremental sliding mode control for a robot manipulator. *Mechatronics*, 82:102717, 2022.

[78] John C Doyle. Structured uncertainty in control system design. In *1985 24th IEEE Conference on Decision and Control*, pages 260–265. IEEE, 1985.

[79] Zengjie Zhang, Yongchao Wang, and Dirk Wollherr. Safe tracking control of euler-lagrangian systems based on a novel adaptive super-twisting algorithm. *IFAC-PapersOnLine*, 53(2):9974–9979, 2020.

[80] Luigi Palmieri and Kai O Arras. A novel rrt extend function for efficient and smooth mobile robot motion planning. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 205–211. IEEE, 2014.

[81] Tao Zhang, Yi Zhu, and Jingyan Song. Real-time motion planning for mobile robots by means of artificial potential field method in unknown environment. *Industrial Robot: An International Journal*, 2010.

[82] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[83] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4238–4245. IEEE, 2018.

[84] S Saravana Perumaal and N Jawahar. Automated trajectory planner of industrial robot for pick-and-place task. *International Journal of Advanced Robotic Systems*, 10(2):100, 2013.

[85] Feryal MP Behbahani, Ruth Taunton, Andreas AC Thomik, and A Aldo Faisal. Haptic slam for context-aware robotic hand prosthetics-simultaneous inference of hand pose and object shape using particle filters. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 719–722. IEEE, 2015.

[86] Rafiq Ahmad and Peter Plapper. Safe and automated assembly process using vision assisted robot manipulator. *Procedia Cirp*, 41:771–776, 2016.

[87] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

[88] Takahiro Takeda, Yasuhisa Hirata, Zhidong Wang, and Kazuhiro Kosuge. Hmm-based error detection of dance step selection for dance partner robot-ms dancer. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5631–5636. IEEE, 2006.

[89] Christoffer B Kristensen, Frederik A Sørensen, Hjalte B Nielsen, Martin S Andersen, Søren P Bendtsen, and Simon Bøgh. Towards a robot simulation framework for e-waste disassembly using reinforcement learning. *Procedia Manufacturing*, 38:225–232, 2019.

[90] Guoting Chang and Dana Kulić. Robot task error recovery using petri nets learned from demonstration. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6. IEEE, 2013.

[91] Nicole Mirnig, Gerald Stollnberger, Markus Miksch, Susanne Stadler, Manuel Giuliani, and Manfred Tscheligi. To err is robot: How humans assess and act toward an erroneous social robot. *Frontiers in Robotics and AI*, 4:21, 2017.

[92] Thomas B Sheridan. Human–robot interaction: status and challenges. *Human factors*, 58(4):525–532, 2016.

[93] Ali Ghorbandaei Pour, Alireza Taheri, Minoo Alemi, and Ali Meghdari. Human–robot facial expression reciprocal interaction platform: case studies on children with autism. *International Journal of Social Robotics*, 10(2):179–198, 2018.

[94] Fernando Alonso-Martin, Maria Malfaz, Joao Sequeira, Javier F Gorostiza, and Miguel A Salichs. A multimodal emotion detection system during human–robot interaction. *Sensors*, 13(11):15549–15581, 2013.

[95] S Aswath, Chinmaya Krishna Tilak, Amal Suresh, and Ganesha Udupa. Human gesture recognition for real-time control of humanoid robot. In *proceedings of International Conference on Advances in Engineering and Technology (Singapore*, page null, 2014.

[96] Takayuki Kanda, Hiroshi Ishiguro, Michita Imai, and Tetsuo Ono. Body movement analysis of human-robot interaction. In *IJCAI*, volume 3, pages 177–182. Citeseer, 2003.

[97] Emrah Akin Sisbot, Luis F Marin, and Rachid Alami. Spatial reasoning for human robot interaction. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2281–2287. IEEE, 2007.

[98] Kerstin Dautenhahn, Michael Walters, Sarah Woods, Kheng Lee Koay, Chrystopher L Nehaniv, A Sisbot, Rachid Alami, and Thierry Siméon. How may i serve you? a robot companion approaching a seated person in a helping context. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 172–179, 2006.

[99] Zheng Wang, Angelika Peer, and Martin Buss. Fast online impedance estimation for robot control. In *2009 IEEE International Conference on Mechatronics*, pages 1–6. IEEE, 2009.

[100] Wesley P Chan, Chris AC Parker, HF Machiel Van der Loos, and Elizabeth A Croft. Grip forces and load forces in handovers: implications for designing human-robot handover controllers. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 9–16, 2012.

[101] Manuel Giuliani, Mary Ellen Foster, Amy Isard, Colin Matheson, Jon Oberlander, and Alois Knoll. Situated reference in a hybrid human-robot interaction system. In *Proceedings of the 6th International Natural Language Generation Conference (INLG 2010)*, page null, 2010.

[102] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[103] Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 1942–1951. PMLR, 2019.

[104] Mohammed AH Ali and Musa Mailah. Path planning and control of mobile robot in road environments using sensor fusion and active force control. *IEEE Transactions on Vehicular Technology*, 68(3):2176–2195, 2019.

[105] Maolin Jin, Sang Hoon Kang, Pyung Hun Chang, and Jinoh Lee. Robust control of robot manipulators using inclusive and enhanced time delay control. *IEEE/ASME Transactions on Mechatronics*, 22(5):2141–2152, 2017.

[106] Rahul Gautam, Rahul Kala, et al. Motion planning for a chain of mobile robots using a* and potential field. *Robotics*, 7(2):20, 2018.

[107] Zhijun Li, Ting Zhao, Fei Chen, Yingbai Hu, Chun-Yi Su, and Toshio Fukuda. Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator. *IEEE/ASME Transactions on Mechatronics*, 23(1):121–131, 2017.

[108] Delowar Hossain and Genci Capi. Multiobjective evolution of deep learning parameters for robot manipulator object recognition and grasping. *Advanced Robotics*, 32(20):1090–1101, 2018.

[109] Diana Löffler, Nina Schmidt, and Robert Tscharn. Multimodal expression of artificial emotion in social robots using color, motion and sound. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 334–343, 2018.

[110] Klaus Weber, Hannes Ritschel, Ilhan Aslan, Florian Lingenfelser, and Elisabeth André. How to shape the humor of a robot-social behavior adaptation based on reinforcement learning. In *Proceedings of the 20th ACM international conference on multimodal interaction*, pages 154–162, 2018.

[111] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[112] LarsPeter Hansen and Thomas J Sargent. Robust control and model uncertainty. *American Economic Review*, 91(2):60–66, 2001.

[113] Han Hu, Kaicheng Zhang, Aaron Hao Tan, Michael Ruan, Christopher Agia, and Goldie Nejat. A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain. *IEEE Robotics and Automation Letters*, 6(4):6569–6576, 2021.

[114] James L Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 803–808. IEEE, 2005.

[115] Haitham Bou Ammar, Eric Eaton, Matthew E Taylor, Decebal Constantin Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, page null, 2014.

[116] Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 544–551, 2008.

[117] Mark W Spong. On the robust control of robot manipulators. *IEEE Transactions on Automatic Control*, 37(11):1782–1786, 1992.

[118] Fabio Muratore, Felix Treede, Michael Gienger, and Jan Peters. Domain randomization for simulation-based policy optimization with transferability assessment. In *Conference on Robot Learning*, pages 700–713. PMLR, 2018.

[119] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 The International Conference on Robotics and Automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[120] Gang Tao. *Adaptive control design and analysis*, volume 37. John Wiley & Sons, 2003.

[121] Dan Zhang and Bin Wei. A review on model reference adaptive control of robotic manipulators. *Annual Reviews in Control*, 43:188–198, 2017.

[122] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[123] Matthew E Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15–15, 2011.

[124] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pages 1015–1022, 2007.

[125] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv preprint arXiv:2003.04960*, 2020.

[126] 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.

[127] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018.

[128] Zengjie Zhang, Kun Qian, Björn W Schuller, and Dirk Wollherr. An online robot collision detection and identification scheme by supervised learning and bayesian decision theory. *IEEE Transactions on Automation Science and Engineering*, 18(3):1144–1156, 2020.

[129] Junkai Ren, Yujun Zeng, Sihang Zhou, and Yichuan Zhang. An experimental study on state representation extraction for vision-based deep reinforcement learning. *Applied Sciences*, 11(21):10337, 2021.

[130] Muhammad Mudassir Ejaz, Tong Boon Tang, and Cheng-Kai Lu. Vision-based autonomous navigation approach for a tracked robot using deep reinforcement learning. *IEEE Sensors Journal*, 21(2):2230–2240, 2020.

[131] Chengshu Li, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pages 603–616. PMLR, 2020.

[132] Zhimin Hou, Jiajun Fei, Yuelin Deng, and Jing Xu. Data-efficient hierarchical reinforcement learning for robotic assembly control applications. *IEEE Transactions on Industrial Electronics*, 68(11):11565–11575, 2020.

[133] Xintong Yang, Ze Ji, Jing Wu, Yu-Kun Lai, Changyun Wei, Guoliang Liu, and Rossitza Setchi. Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[134] Freek Stulp, Evangelos A Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on robotics*, 28(6):1360–1370, 2012.

[135] Anuj Mahajan and Theja Tulabandhula. Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*, 2017.

[136] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467, 2008.

[137] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[138] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[139] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[140] 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.

[141] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[142] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3061–3071. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/fedus20a.html.

[143] Matthew E Taylor, Halit Bener Suay, and Sonia Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 617–624, 2011.

[144] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[145] Jiang Hua, Liangcai Zeng, Gongfa Li, and Zhaojie Ju. Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning. *Sensors*, 21 (4), 2021. ISSN 1424-8220. doi: 10.3390/s21041278. URL https://www.mdpi.com/1424-8220/21/4/1278.

[146] Joshua Achiam. Spinning Up in Deep Reinforcement Learning, 2018. URL https://spinningup.openai.com/en/latest/.

[147] Ray Y Zhong, Xun Xu, Eberhard Klotz, and Stephen T Newman. Intelligent manufacturing in the context of industry 4.0: a review. *Engineering*, 3(5): 616–630, 2017.

[148] Zengjie Zhang, Ram Dershan, Amir M. Soufi Enayati, Marjan Yaghoubi, Dean Richert, and Homayoun Najjaran. A high-fidelity simulation platform for industrial manufacturing by incorporating robotic dynamics into an industrial simulation tool. *IEEE Robotics and Automation Letters*, 7(4):9123–9128, 2022. doi: 10.1109/LRA.2022.3190096.

[149] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, and Jorge Barbosa. Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges. *Computers in Industry*, 123:103298, 2020.

[150] Cristina Morariu, Octavian Morariu, Silviu Răileanu, and Theodor Borangiu. Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems. *Computers in Industry*, 120:103244, 2020.

[151] Keene Chin, Tess Hellebrekers, and Carmel Majidi. Machine learning for soft robotic sensing and control. *Advanced Intelligent Systems*, 2(6):1900171, 2020.

[152] J Norberto Pires, Amin S Azar, Filipe Nogueira, Carlos Ye Zhu, Ricardo Branco, and Trayana Tankova. The role of robotics in additive manufacturing: review of the am processes and introduction of an intelligent system. *Industrial Robot: the international journal of robotics research and application*, 2021.

[153] Onder Tutsoy and Duygun Erol Barkana. Model free adaptive control of the under-actuated robot manipulator with the chaotic dynamics. *ISA transactions*, 118:106–115, 2021.

[154] Hongxu Zhang, Fei Wang, Jianhui Wang, and Ben Cui. Robot grasping method optimization using improved deep deterministic policy gradient algorithm of deep reinforcement learning. *Review of Scientific Instruments*, 92(2):025114, 2021.

[155] P Zieliński and U Markowska-Kaczmar. 3d robotic navigation using a vision-based deep reinforcement learning model. *Applied Soft Computing*, 110:107602, 2021.

[156] Alex Church, John Lloyd, Raia Hadsell, and Nathan F Lepora. Deep reinforcement learning for tactile robotics: Learning to type on a braille keyboard. *IEEE Robotics and Automation Letters*, 5(4):6145–6152, 2020.

[157] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274, 2013.

[158] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

[159] Ian R Petersen and Roberto Tempo. Robust control of uncertain systems: Classical results and recent developments. *Automatica*, 50(5):1315–1335, 2014.

[160] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural Computation*, 17(2):335–359, 2005. doi: 10.1162/0899766053011528.

[161] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2817–2826. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/pinto17a.html.

[162] Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022. ISSN 2504-4990. doi: 10.3390/make4010013. URL https://www.mdpi.com/2504-4990/4/1/13.

[163] Xiangkun He and Chen Lv. Robotic control in adversarial and sparse reward environments: A robust goal-conditioned reinforcement learning approach. *IEEE Transactions on Artificial Intelligence*, pages 1–10, 2023. doi: 10.1109/TAI.2023.3237665.

[164] Xiangkun He, Haohan Yang, Zhongxu Hu, and Chen Lv. Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach. *IEEE Transactions on Intelligent Vehicles*, 2022.

[165] Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. Robust reinforcement learning using offline data. *arXiv preprint arXiv:2208.05129*, 2022.

[166] Kishan Panaganti and Dileep Kalathil. Sample complexity of robust reinforcement learning with a generative model. In *International Conference on Artificial Intelligence and Statistics*, pages 9582–9602. PMLR, 2022.

[167] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[168] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250, 2018. doi: 10.1109/ICRA.2018.8460875.

[169] 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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[170] Enrique Jimenez-Vazquez, Julieta Ayala-Rodriguez, David Navarro-Duran, and Edgar Lopez-Caudana. Model approximation of an arm of the nao™ robot using system identification. In *2021 Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, pages 1–6, 2021. doi: 10.1109/ICA-SYMP50206.2021.9358430.

[171] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. doi: 10.1109/TRO.2008.2006706.

[172] Feng Ding, Xuehai Wang, Qijia Chen, and Yongsong Xiao. Recursive least squares parameter estimation for a class of output nonlinear systems based

on the model decomposition. *Circuits, Systems, and Signal Processing*, 35(9): 3323–3338, 2016.

[173] Kaicheng Niu, Mi Zhou, Chaouki T Abdallah, and Mohammad Hayajneh. Deep transfer learning for system identification using long short-term memory neural networks. *arXiv preprint arXiv:2204.03125*, 2022.

[174] Hamid Khodabandehlou and Mohammed Sami Fadali. Nonlinear system identification using neural networks and trajectory-based optimization. *arXiv preprint arXiv:1804.10346*, 2018.

[175] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[176] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C. Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2884–2890, 2021. doi: 10.1109/ICRA48506.2021.9561731.

[177] Harry A. Pierson and Michael S. Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821 – 835, 2017. ISSN 01691864.

[178] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi: 10.1109/ICRA.2018.8460528.

[179] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6202–6209, 2020.

[180] 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. *arXiv preprint arXiv:1706.01905*, 2017.

[181] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

[182] Leon Eversberg and Jens Lambrecht. Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization. *Sensors*, 21(23):7901, 2021.

[183] Enric Moreu, Kevin McGuinness, Diego Ortego, and Noel E O'Connor. Domain randomization for object counting. *arXiv preprint arXiv:2202.08670*, 2022.

[184] Hang Du, Haidong Hu, and Yingzi He. Structural components recognition method for malfunctioned satellite based on domain randomization. In *2021 China Automation Congress (CAC)*, pages 1979–1983, 2021. doi: 10.1109/CAC53003.2021.9727819.

[185] Frederik Hagelskjaer and Anders Glent Buch. Parapose: Parameter and domain randomization optimization for pose estimation using synthetic data. *arXiv preprint arXiv:2203.00945*, 2022.

[186] Asad Ali Shahid, Dario Piga, Francesco Braghin, and Loris Roveda. Continuous control actions learning and adaptation for robotic manipulation through reinforcement learning. *Autonomous Robots*, 46(3):483–498, 2022.

[187] Asad Ali Shahid, Jorge Said Vidal Sesin, Damjan Pecioski, Francesco Braghin, Dario Piga, and Loris Roveda. Decentralized multi-agent control of a manipulator in continuous task learning. *Applied Sciences*, 11(21):10227, 2021.

[188] Simon Ramstedt and Chris Pal. Real-time reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

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

[190] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. Hierarchical reinforcement learning for precise

soccer shooting skills using a quadrupedal robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1479–1486. IEEE, 2022.

[191] H Harry Asada and Vijay Kumar. Special issue on stochasticity in robotics and bio-systems, 2011.

[192] Karel J Keesman and Karel J Keesman. *System identification: an introduction*, volume 2. Springer, 2011.

[193] Athanasios S. Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017. doi: 10.1007/s10846-017-0468-y.

[194] Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. Comparing popular simulation environments in the scope of robotics and reinforcement learning. *arXiv preprint arXiv:2103.04616*, 2021.

[195] Angel Ayala, Francisco Cruz, Diego Campos, Rodrigo Rubio, Bruno Fernandes, and Richard Dazeley. A comparison of humanoid robot simulators: A quantitative approach. In *2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 1–6. IEEE, 2020.

[196] Lenka Pitonakova, Manuel Giuliani, Anthony Pipe, and Alan Winfield. Feature and performance comparison of the v-rep, gazebo and argos robot simulators. In *Annual Conference Towards Autonomous Robotic Systems*, pages 357–368. Springer, 2018.

[197] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4397–4404. IEEE, 2015.

[198] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning, 2017.

[199] Fabio Muratore, Felix Treede, Michael Gienger, and Jan Peters. Domain randomization for simulation-based policy optimization with transferability assessment. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 700–713. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/muratore18a.html.

[200] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[201] Kinova kortex api github repository, 2023. URL https://github.com/kinovarobotics/ros_kortex. Accessed: October 2023.

[202] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *arXiv preprint arXiv:1910.08348*, 2019.

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

[204] Luckeciano C Melo. Transformers are meta-reinforcement learners. In *international conference on machine learning*, pages 15340–15359. PMLR, 2022.

[205] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Conference on Robot Learning*, pages 1478–1487. PMLR, 2023.

[206] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

[207] Taewook Nam, Shao-Hua Sun, Karl Pertsch, Sung Ju Hwang, and Joseph J Lim. Skill-based meta-reinforcement learning. *arXiv preprint arXiv:2204.11828*, 2022.

[208] Zhenshan Bing, Lukas Knak, Long Cheng, Fabrice O Morin, Kai Huang, and Alois Knoll. Meta-reinforcement learning in nonstationary and nonparametric environments. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[209] Pedro A Ortega, Jane X Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

[210] Michael O'Gordon Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. University of Massachusetts Amherst, 2002.

[211] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, Aviv Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[212] Anthony R Cassandra, Leslie Pack Kaelbling, and James A Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 2, pages 963–972. IEEE, 1996.

[213] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[214] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pages 27042–27059. PMLR, 2022.

[215] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR, 2022.

[216] Gresa Shala, André Biedenkapp, and Josif Grabocka. Hierarchical transformers are efficient meta-reinforcement learners. *arXiv preprint arXiv:2402.06402*, 2024.

[217] Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. *arXiv preprint arXiv:2310.09971*, 2023.

[218] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[219] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[220] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[221] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[222] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[223] Suyoung Lee, Myungsik Cho, and Youngchul Sung. Parameterizing non-parametric meta-reinforcement learning tasks via subtask decomposition. *Advances in Neural Information Processing Systems*, 36:43356–43383, 2023.

[224] Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, KR Zentner, Ryan Julian, JK Terry, Isaac Woungang, et al. Meta-world+: An improved, standardized, rl benchmark. *arXiv preprint arXiv:2505.11289*, 2025.

[225] Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022.

[226] Qinqing Zheng, Mikael Henaff, Brandon Amos, and Aditya Grover. Semi-supervised offline reinforcement learning with action-free trajectories. In *International conference on machine learning*, pages 42339–42362. PMLR, 2023.

[227] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[228] Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. Knowledge distillation via the target-aware transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10915–10924, 2022.

[229] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, et al. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15. IEEE, 2022.

[230] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.