UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title

Data-Efficient Learning and Generalization for Industrial Robotic Systems

Permalink

https://escholarship.org/uc/item/625827hs

Author

Wu, Zheng

Publication Date

2024

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

Data-Efficient Learning and Generalization for Industrial Robotic Systems

By

Zheng Wu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Associate Professor Koushil Sreenath
Professor Pieter Abbeel

Spring 2024

Data-Efficient Learning and Generalization for Industrial Robotic Systems

Copyright 2024
by
Zheng Wu

Abstract

Data-Efficient Learning and Generalization for Industrial Robotic Systems

by

Zheng Wu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Autonomous robots have swiftly revolutionized several industries, enhancing manufacturing processes by streamlining assembly lines for heightened efficiency, revolutionizing agriculture with automated planting and harvesting, and refining logistics through the optimization of warehousing and delivery systems. These advancements underscore the groundbreaking impact of robotics on productivity and innovation across a diverse range of sectors. However, the rapid advancement in autonomous robotics faces a significant challenge: the heavy dependency on massive data and the daunting task of generalizing learned behaviors to new tasks and environments. This challenge represents more than a mere technical obstacle; it is a critical bottleneck that constrains the widespread adoption and effectiveness of autonomous robots, stifling innovation and practical deployment across a myriad of industrial applications.

In this dissertation, we study the problem of data-efficient learning and generalization for industrial autonomous robots. Our goal is to develop algorithms that enable robots to learn from limited data and generalize the learned behaviors to novel tasks and environments effectively. The core idea is to leverage proper task knowledge and assumptions, embedding them into the algorithmic designs to significantly enhance their data efficiency. Our research endeavors are dedicated to three principal aspects: data-efficient reward learning, data-efficient policy learning, and data-efficient policy generalization. These approaches are meticulously applied across a wide array of industrial scenarios, such as autonomous vehicles, robotic assembly, and robotic palletization, showcasing their versatility and effectiveness in enhancing robotic efficiency and adaptability in real-world applications.

This dissertation unfolds in three distinct parts, offering a comprehensive examination of data-efficient learning and generalization for autonomous robots. Part I lays the foundation with an in-depth exploration of data-efficient reward learning, employing inverse reinforcement learning (Chapter 2) and representation learning (Chapter 3) to uncover efficient ways to infer reward signals from limited data. Part II shifts focus to the nuances of data-efficient

policy learning. Chapter 4 introduces a data-efficient reinforcement learning (RL) policy specifically designed for robotic palletization, enhancing data efficiency by narrowing the exploration space via learned action space masking. In Chapter 5, we propose a novel skill representation method, namely motion primitives (MP), alongside a data-efficient framework for learning MP-based insertion skills directly from human demonstrations. Concluding with Part III, the dissertation advances into the realm of data-efficient policy generalization across diverse tasks. In Chapter 6, a novel zero-shot policy generalization approach is presented, capitalizing on the compositional structure of task representations to enable seamless adaptation to new tasks without the need for additional data.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

I extend my first and deepest gratitude to my Ph.D. advisor, Prof. Masayoshi Tomizuka, whose invaluable guidance and unwavering support have been pivotal throughout my doctoral journey. More than a mentor, Prof. Tomizuka has been a guiding light, offering me the freedom to explore my research interests while providing the wisdom to steer my efforts when necessary. His commitment to academic excellence and genuine concern for my personal and professional growth have profoundly shaped me. Under his mentorship, I have evolved not only as a researcher but also as a person, aspiring to mirror his remarkable blend of dedication, kindness, and zest for life. My transformative journey under Prof. Tomizuka's guidance is something I deeply cherish, and for which I am immensely grateful.

My heartfelt appreciation also goes to Prof. Koushil Sreenath and Prof. Pieter Abbeel for their crucial contributions to my dissertation committee, and to Prof. Mark Muller, Prof. Francesco Borrelli, Prof. Anil Aswani, along with Prof. Sreenath again, for their roles in my qualifying exam committee. Their lectures and advice have laid a robust foundation in controls and robotics that inspired my research path.

I am profoundly thankful for the opportunity to intern at Google X during two remarkable summers, where I worked on industrial robotic manipulation projects with Dr. Wenzhao Lian and Dr. Stefan Schaal. This experience introduced me to some of the field's most challenging problems and significantly influenced my research at UC Berkeley, enriching my academic journey with practical insights and inspiring new research directions.

I extend my gratitude to Prof. Cewu Lu, Prof. Jiajun Wu, and Prof. Josh Tenenbaum for their mentorship during my undergraduate years, which ushered me into the fascinating world of research and set me on the path to pursuing my Ph.D.

Collaboration has been a cornerstone of my work, and I am grateful to all my co-authors, including Liting Sun, Changhao Wang, Shiyu Jin, Xinghao Zhu, Lingfeng Sun, Xiang Zhang, Jianyu Chen, Te Tang, Wei Zhan, Hengbo Ma, Yichuan Li, Yun-Hui Liu, Wenzhao Lian, Stefan Schaal, Vaibhav Unhelkar, Wanqiao Xu, Zheqing Zhu, Jalaj Bhandari, and Daniel Jiang, for their insightful discussions and contributions.

My time at the Mechanical System Controls (MSC) lab has been enriched by every member I've had the pleasure to meet and work with, including Zining Wang, Jiachen Li, Yeping Hu, Zhuo Xu, Saman Fahandezhsaadi, Yujiao Chen, Chen Tang, Jessica Leu, Huidong Gao, Yiyang Zhou, Hengbo Ma, Jinning Li, Catherine Weaver, Wu-Te Yang, Chenran Li, Akio Kodaira, Qiyang Qian, Ran Tian, Chenfeng Xu, Wei-Jer Chang, Jen-Wei Wang, Keita Kobashi, Yuxin Chen, Boyuan Liang, Yixiao Wang, John Viljoen, and many others for their friendship and support.

Lastly, my deepest love and gratitude go to my parents, Yuquan Wu and Zhanglan Yang, and my wife, Mengxi Li, for your unconditional love, support, and encouragement. Your unwavering presence, through both the good times and the challenging ones, has been my steadfast source of strength and motivation. My achievements are a testament to your love and belief in me, for which I am forever grateful.

Chapter 1

Introduction

1.1 Background and Motivations

Industrial robots are at the vanguard of a transformative movement within the automation industry, marking a paradigm shift towards more flexible and intelligent manufacturing systems. The advent of learning methods, particularly Reinforcement Learning (RL) [105] and Learning from Demonstrations (LfD) [6], has been instrumental in this transition, empowering robots with the capacity to operate in environments characterized by fewer constraints. These methodologies enable robotic systems to adapt to new tasks and environments through a process of learning, rather than being programmed with explicit instructions for each possible scenario. As such, the application of learning methods in robotics facilitates a significant expansion in the range of tasks that robots can perform, moving beyond repetitive, predefined tasks to more complex and dynamic operations [58].

However, the methodologies underpinning RL and LfD demand extensive data to effectively imbue robots with the requisite skills for autonomous operation. This data-intensive nature presents a formidable barrier to the application of these technologies, especially when considering the deployment and transferability of learned skills across varied environments, a task marked by significant challenges [107, 80]. Moreover, the acquisition of real-world data on physical robotic systems entails considerable expense, limiting the feasibility of widespread data collection and, by extension, the broad application of these learning methods in diverse settings [12].

To this end, this dissertation addresses the challenge of enhancing data efficiency in the context of industrial robotics from three distinct but interconnected perspectives: reward learning, policy learning, and policy generalization. By dissecting the robot learning problem into these specific components, this thesis seeks to construct a holistic framework that not only optimizes the acquisition and utilization of data but also ensures the robustness and adaptability of learned behaviors across diverse industrial scenarios. Reward learning focuses on the formulation and refinement of objectives that guide the robot's actions, laying the groundwork for effective decision-making. Policy learning then builds upon this foundation,

developing strategies that enable the robot to act in a manner that achieves these objectives efficiently. Finally, policy generalization addresses the challenge of transferring these learned strategies to new and varied environments, ensuring the robot's performance remains consistent even when faced with unfamiliar tasks or conditions. Together, these focal areas form the cornerstone of this dissertation's approach to advancing the deployment of data-efficient, learning-based industrial robotic systems. In this section, the necessary background of this dissertation will be provided.

Reinforcement Learning

Reinforcement Learning (RL) is a subfield of Machine Learning where an agent learns to make decisions by taking actions in an environment to achieve some goals. The fundamental principle of reinforcement learning is the concept of agents learning from the consequences of their actions, rather than from being taught explicitly. This learning process involves discovering which actions yield the most reward by trying them out and assessing the results.

The typical reinforcement learning scenario involves an agent, the environment, and the interaction between the two. The agent takes actions in the environment, which is then responded to by the environment through the provision of rewards and the presentation of new states. This interaction is often modeled as a Markov Decision Process (MDP), comprising a set of states (S), a set of actions (A), a transition function ($P(s_{t+1}|s_t, a_t)$) that models the dynamics of the environment, and a reward function ($R(s, a)$). The agent's objective is to maximize the cumulative reward it receives over time. The cumulative reward is often defined as a discounted sum of future rewards:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right] \tag{1.1}$$

where γ is the discount factor, $0 \leq \gamma \leq 1$, which determines the present value of future rewards. Two common approaches for learning the optimal policy are value-based methods, which involve learning the value function, and policy-based methods, which involve learning the policy directly.

The key components of an RL system include:

- **Agent**: The learner or decision maker.
- **Environment**: The system with which the agent interacts.
- **State**: A representation of the current situation of the agent within the environment.
- **Action**: An intervention the agent makes in the environment.
- **Reward**: A feedback signal from the environment indicative of the success of an action.
- **Policy**: A strategy employed by the agent, mapping states to actions.

- **Value Function**: A function that estimates the expected return from states or state-action pairs under a particular policy.
- **Model** (optional): A representation of the environment's dynamics used for planning.

Learning from Demonstrations

Learning from Demonstration (LfD) stands as a cornerstone in the evolution of robotic learning, enabling robots to acquire new skills by imitating human demonstrations. This methodology circumvents the limitations of traditional programming by directly leveraging the nuanced capabilities of human teachers, thus facilitating the transmission of complex task knowledge to robots. The core premise of LfD is predicated on the observation and replication of human behavior, through which robots discern and adopt the necessary actions to accomplish a given task.

The mathematical foundation of LfD can be delineated through various problem formulations, each corresponding to a distinct learning strategy within the LfD paradigm. Two primary methodologies prevalent in LfD research are behavior cloning and inverse reinforcement learning.

Behavior Cloning Behavior cloning (BC) conceptualizes the learning process as a direct mapping from observed states to actions, akin to supervised learning. Given a dataset of demonstration trajectories $D = \{(s_1, a_1), (s_2, a_2), \ldots, (s_N, a_N)\}$, where s_i denotes the state and a_i the corresponding action at the i-th step of a demonstration, the objective of BC is to learn a policy π_θ that minimizes the discrepancy between the actions predicted by the policy and those performed by the human demonstrator. The optimization problem is typically formulated as:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} L(\pi_\theta(s_i), a_i), \tag{1.2}$$

where L is a loss function measuring the difference between the policy's output and the actual action taken by the demonstrator.

Inverse Reinforcement Learning Inverse Reinforcement Learning (IRL) seeks to infer the underlying reward function that the demonstrator is optimizing. The foundational assumption of IRL is that the demonstrator's actions are rational and aimed at maximizing a cumulative reward. Given a set of demonstrations, the goal of IRL is to find a reward function R such that the demonstrated behavior is optimal under this reward. Formally, this can be expressed as:

$$\max_R \mathbb{E}_{(s,a)\sim D}[\log P(a|s, R)], \tag{1.3}$$

where $P(a|s, R)$ denotes the probability of the demonstrator choosing action a in state s under the reward function R. The expectation is taken over the distribution of state-action pairs in the demonstration dataset.

1.2 Thesis Outline

This dissertation contains three parts. Part I delves into data-efficient reward learning, analyzing how we can derive meaningful objectives from trajectory data or multi-modal sensory inputs. Part II shifts the focus towards data-efficient policy learning. Finally, Part III tackles the intricacies of achieving data-efficient policy generalization, ensuring robust applicability across a diverse array of tasks. Figure 1.1 illustrates the outline of this dissertation. We provide more detailed descriptions of these components as follows.

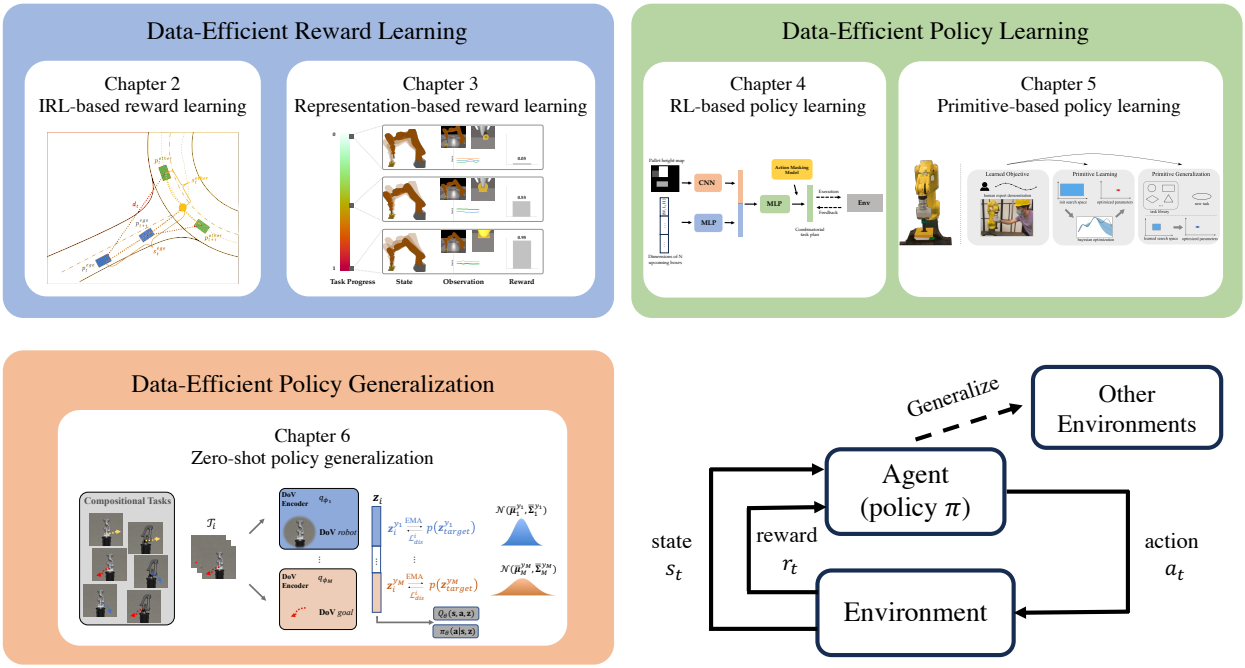


Figure 1.1: Dissertation outline. This dissertation focuses on three important aspects of data-efficient learning and generalization for industrial robotic systems. In Part I, Chapters 2-3 focus on data-efficient reward learning. In Part II, we investigates data-efficient policy learning algorithms. Finally Part III studies data-efficient policy generalization.

Part I: Data-Efficient Reward Learning

Reward learning is pivotal as it enables robots to discern valuable insights into desired behaviors directly from expert demonstrations, bypassing the often complex and labor-intensive process of manually designing reward functions. Moreover, this foundational understanding of reward structures greatly benefits the subsequent policy learning stage, enhancing the efficiency and effectiveness of developing robust policies. This part of dissertation discusses methodologies developed for data-efficient reward learning.

Efficient Inverse Reinforcement Learning for Human Driving Trajectories

In Chapter 2, we present a data-efficient inverse reinforcement learning (IRL) algorithm in continuous domain to efficiently learn reward functions from human driving trajectories. Correctly inferring the reward functions of human drivers is crucial for enabling more naturalistic and transparent interactions between the autonomous vehicles and humans. This algorithm sets itself apart by incorporating a novel continuous-domain trajectory sampler, allowing directly learning reward functions in the continuous domain while considering the uncertainties in demonstrated trajectories from human drivers. By leveraging prior insights into vehicle kinematics and motion planning, we crafted an efficient sampler that precisely estimates the partition term crucial for accurate reward function retrieval. Our extensive testing across various driving scenarios — both non-interactive and interactive — demonstrates the algorithm's superior performance over conventional IRL methods. It shines in deterministic and probabilistic metrics alike, including feature count deviation, mean Euclidean distance, and the likelihood of demonstrations, while showcasing remarkable generalization capabilities and a faster convergence rate. Part of this work was published in [119].

Learning Dense Rewards for Contact-Rich Manipulation Tasks

In Chapter 3, we explore learning dense reward functions from robot's high-dimensional observations, such as images and tactile feedback, for contact-rich manipulation tasks. In robotics, the success of reinforcement learning (RL) relies heavily on the availability of high-quality, dense reward signals. These dense rewards often need to be hand-crafted by a human and require significant domain expertise and trial-and-error. In contact-rich manipulation tasks, due to the discontinuous dynamics and high-dimensional observation spaces, the challenge of reward specification is further amplified; thus, making the adoption of RL methods for robotic manipulation difficult in practice. Prior research leverages generative adversarial learning and inverse reinforcement learning (IRL) techniques to learn rewards from continuous, high-dimensional observation space. However, this class of methods inevitable inherits the instabilities associated with adversarial training and fail to utilize multi-modal observations. In this chapter, we introduce a novel reward learning algorithm through the len of representation learning. Our approach learns a mapping from the high-dimensional observation space to a latent measure of task progress, and thus derive dense rewards from the measured task progress. We demonstrate the effectiveness and efficiency of our approach on two contact-rich manipulation tasks, namely, peg-in-hole and USB insertion. The experimental results indicate that the policies trained with the learned reward function achieves better performance and faster convergence. The content of this chapter is based primarily on [121].

Part II: Data-Efficient Policy Learning

The successful inference of reward functions is a critical precursor to the development of efficient policy learning methodologies. Nonetheless, the pursuit of data efficiency within policy learning introduces its own set of unique challenges and complexities. In Part II, we delve into policy learning techniques specifically designed to enhance data efficiency.

Efficient Reinforcement Learning of Task Planners for Robotic Palletization

Chapter 4 focuses on developing data-efficient reinforcement learning (RL) policy for tasking planning in robotic palletization systems. Confronted with the substantial challenge of a vast action space, which is a significant impediment to efficiently apply out-of-the-shelf RL methods, this chapter introduces a novel method of utilizing supervised learning to iteratively prune and manage the action space effectively. By reducing the complexity of the action space, our approach not only accelerates the learning phase but also ensures the effectiveness and reliability of the task planning in robotic palletization. To assess the efficacy of our proposed methodology, we conducted extensive experiments and compared our results with existing RL-based solutions. Experimental results indicate that our proposed method outperforms other baselines and is more sampling-efficient. Part of this work was published in [118].

A Framework to Learn and Adapt Primitive-Based Skills for Industrial Insertion Tasks

In Chapter 5, we jump out of the framework of reinforcement learning (RL) and introduce a novel approach to learn and adapt industrial insertion policies based on motion primitives (MP). Distinct from RL, our MP-based methodology benefits from the incorporation of specific task priors, achieving greater sample efficiency. Our proposed framework is designed to efficiently learn and adapt MP-based insertion skills from demonstrations, employing black-box function optimization to refine primitive parameters by drawing on prior experiences. Human demonstrations serve as a dense source of rewards, facilitating the targeted learning of parameters. The efficacy of our approach is empirically validated across a series of eight peg-hole and connector-socket insertion tasks, demonstrating that our framework can acquire new insertion skills in less than an hour and adapt to novel insertion tasks in as little as fifteen minutes when implemented on a physical robot. Part of this work was published in [122].

Part III: Data-Efficient Policy Generalization

In addition to efficient policy learning, achieving data-efficient policy generalization is essential for the deployment of industrial robots. This allows them to swiftly adapt to new tasks and environments with minimal data. This segment of the dissertation delves into methodologies designed to accomplish data-efficient policy generalization.

Zero-Shot Policy Transfer with Disentangled Task Representation

Chapter 6 introduces a novel zero-shot policy generalization algorithm for reinforcement learning (RL) agents, enabling them to generalize over unseen tasks. This chapter specifically targets the transfer of policies across compositional tasks — novel combinations of existing task elements. Utilizing the concept of task compositionality, we propose a meta-RL algorithm that employs disentangled task representation. This approach allows for the generalization of policies to unseen compositional tasks by inferring their representations through the achieved disentanglement, eliminating the need for additional exploration. Our methodology is rigorously evaluated through three simulated tasks and a complex real-world robotic insertion challenge. The experimental results validate our algorithm's ability to effectively generalize policies to novel compositional tasks in a zero-shot fashion. The content of this chapter is based primarily on [123].

Lastly, we conclude by summarizing the key contributions of this dissertation and exploring promising directions for future research in Chapter 7.

Part I

Data-Efficient Reward Learning

Chapter 2

Efficient Inverse Reinforcement Learning for Human Driving Trajectories

2.1 Introduction

Although rapid progresses have been made in autonomous driving recently, many challenging problems remain open, particularly when interactions between autonomous vehicles (AVs) and humans are considered. Advanced techniques in reinforcement learning (RL) and optimization such as search-based methods[89, 34, 66], gradient-based approaches [65, 14] and nested optimization [90] have significantly boosted our capability in finding the optimal trajectories of autonomous vehicles that maximize the specified reward/cost functions. However, to enable more naturalistic and transparent interactions between the AVs and humans, another question is of equal importance: *what should we optimize*? A mis-specified reward function can cause severe consequences. The autonomous vehicles might be either too conservative or too aggressive, neither of which are desirable or safe in real traffic. More importantly, optimizing for reward functions that are misaligned with human expectations will make the behavior of AVs non-transparent to humans, which will eventually degrade the trust from human.

Thus, it is desired to extract what human drivers are optimizing from real traffic data. Inverse reinforcement learning (IRL) [49, 77] have been widely explored and utilized for acquiring reward functions from demonstrations, assuming that the demonstrations are (sub-)optimal solutions of the underlying reward functions. Many examples have proved the effectiveness of such IRL algorithms. For instance, [1] proposed an IRL algorithm based on expected feature matching and evaluated it on the control of helicopters. [63] proposed an online IRL algorithm to analyze complex human movement and control high-dimensional robot systems. [125] used RL and IRL to develop planning algorithm for autonomous cars in traffic and [26] designed a deep IOC algorithm based on neural networks and policy

optimization, which enabled the PR2 robots to learn dish placement and pouring tasks. In [102] and [103], a courteous cost function and a hierarchical driving cost for autonomous vehicles were learned respectively via IRL to allow autonomous vehicles to generate more human-like behaviors while interacting with humans. [76] extensively studied the feature selection in IRL for autonomous driving.

Typically, acquiring humans' driving costs from real traffic data has to satisfy a set of requirements:

- The trajectories of vehicles are continuous in a high-dimensional and spatiotemporal space, thus the IRL algorithm needs to scale well in high-dimensional continuous space;
- The trajectories of the vehicles satisfy the vehicle kinematics and the IRL algorithms should take it into consideration while learning reward functions;
- Uncertainties exist in real traffic demonstrations. The demonstrations in naturalistic driving data are not necessarily optimal or near-optimal, and the IRL algorithms should be compatible with such uncertainties;
- The features adopted in the reward functions of the IRL algorithms should be highly interpretable and generalizable to improve the transparency and adaptability of the AVs' behaviors.

Unfortunately, it is not trivial to satisfy all the above requirements simultaneously. First of all, most existing IRL algorithms, for instance, apprenticeship learning [1], maximum-entropy IRL [140] and Bayesian IRL [86], are defined within the discrete Markov Decision Process (MDP) framework with relatively small-scale state and action spaces and suffer from the scaling problem in large-scale continuous-domain applications with long horizons. The continuous-domain IOC algorithm proposed in [61] effectively addressed such issue by considering only the local shapes of the reward functions via Laplace approximation. However, it is applicable to deterministic problems where all expert demonstrations are required to be optimal or sub-optimal, which is practically impossible to satisfy via naturalistic driving data, particularly when the features are not known in advance. Moreover, the Laplace approximation requires the calculation of both the gradients and the inverse of the Hessians of the reward functions at the expert demonstrations. When the demonstrations contains long-horizon trajectories, both variables are time-consuming to obtain. The deep IOC algorithm in [26] can also work in continuous domain. However, its features are not interpretable, and lead to poor generalization in different scenarios. In [47], a sampling-based IRL algorithm was also proposed to address this issue, but it only considered the uncertainties caused by the noises in control and can hardly capture the uncertainties induced by the sub-optimality of different driving maneuvers.

In terms of application domain, this chapter is closely related to [59] which also utilize inverse reinforcement learning to learn the reward functions from real driving trajectories. In the forward problem at each iteration, it directly solves the optimization problem and use the

optimal trajectories to represent the expected feature counts. However, such optimization process might be quite time-consuming, especially when the driving horizon is long.

Thus, in this chapter, we propose an efficient sampling-based maximum entropy IRL (SMIRL) algorithm that satisfies all the above mentioned requirements in autonomous driving. In the algorithm, we explicitly leverage our prior knowledge on efficiently generating feasible long-horizon trajectory samples which allow the autonomous vehicles to interact with the environment, including human drivers. More specifically, in terms of problem formulation, we adopt the principle of maximum entropy. In terms of efficient trajectory sampling for the estimation of the partition term with maximum entropy, we integrate our previous non-conservative and defensive motion planning algorithm with a sampling method to efficiently generate feasible and representative long-horizon trajectory samples. We compare the performance of the proposed SMIRL algorithm with the other two IRL algorithms, i.e., the ones in [61] and [59] on real human driving data extracted from the INTERACTION dataset. Three sets of evaluation metrics are employed, including both the deterministic metrics such as mean Euclidean distance (MED) and feature count deviation and the probabilistic metric such as the likelihood of the ground-truth trajectories. The experimental results showed that our proposed SMIRL can achieve more accurate prediction performance on the test set in both non-interactive and interactive driving scenarios.

2.2 The Method

Maximum-entropy IRL

Let x and u denote, respectively, the states and actions of vehicles. The dynamics of the vehicle, $f(\cdot)$, can then be described as:

$$x_{k+1} = f(x_k, u_k). \tag{2.1}$$

A driving trajectory in spatial-temporal domain, denoted as ξ, contains a sequence of states and actions, i.e., $\xi = [x_0, u_0, x_1, u_1, ..., x_{N-1}, u_{N-1}]$ where N is the length of the planning horizon. Given a set of demonstrations $\Xi_D = \{\xi_i\}$ with i=$1, 2, \cdots, M$, with the principle of maximum-entropy [140], the IRL problem aims to recover the underlying reward function from which the likelihood of the demonstrations can be maximized, assuming that the trajectories are exponentially more likely when they have higher cumulative rewards (Boltzman noisily-rational model [74]):

$$P(\xi, \theta) \propto e^{\beta R(\xi,\theta)} \tag{2.2}$$

where the parameter vector θ specifies the reward function R. β is a hyper-parameter that describes how close the demonstrations are to perfect optimizers. As $\beta \to \infty$, the demonstrations approach to perfect optimizers. Without loss of generality, we set β=1 in this work.

Note that in this work, we assume that all the agents/demonstrations share the same dynamics defined in Equation 2.1. We also assume the reward function underlying the given

demonstration set is roughly consistent. Namely, we do not consider scenarios where human drivers change their reward functions along the demonstrations. We also do not specify the diversity of reward functions among different human drivers. Hence, the acquired reward function is essentially an averaged result defined on the demonstration set.

We adopt linear-structured reward function with a selected feature space $\mathbf{f}(\cdot)$ defined over the trajectories ξ, i.e.,

$$R(\xi, \theta) = \theta^T \mathbf{f}(\xi) \tag{2.3}$$

Hence, the probability (likelihood) of the demonstration set becomes

$$P(\Xi_D|\theta) = \prod_{i=1}^{M} \frac{e^{\beta R(\xi_i,\theta)}}{\int_{\tilde{\xi}\in\Phi_{\xi_i}} e^{\beta R(\tilde{\xi},\theta)} d\tilde{\xi}} = \prod_{i=1}^{M} \frac{1}{Z_{\xi_i}} e^{\beta R(\xi_i,\theta)} \tag{2.4}$$

where Φ_{ξ_i} represents the space of all trajectories that share the same initial and goal conditions as in ξ_i. Our goal is to find the optimal θ^* which maximizes the averaged log-likelihood of the demonstrations, i.e.,

$$\theta^\star = \arg\max_\theta \frac{1}{M} \log P(\Xi_D|\theta) = \arg\max_\theta \frac{1}{M} \sum_{i=1}^{M} \log P(\xi_i|\theta). \tag{2.5}$$

From Equation 2.4 and Equation 2.5, we can see that the key step in solving the optimization problem in (2.5) is the calculation of the partition factors Z_{ξ_i}. In sampling-based methods, Z_{ξ_i} for each demonstration is approximated via the sum over samples in the sample set $\{\tau_m^i\}, m = 1, 2, ..., K$:

$$Z_{\xi_i} \approx \sum_{m=1}^{K} e^{\beta R(\tau_m^i,\theta)}. \tag{2.6}$$

Thus, the objective function in Equation 2.5 becomes:

$$\begin{aligned} L(\theta) &= \frac{1}{M} \sum_{i=1}^{M} \log P(\xi_i|\theta) = \frac{1}{M} \sum_{i=1}^{M} \log \frac{e^{\beta R(\xi_i,\theta)}}{\sum_{m=1}^{K} e^{\beta R(\tau_m^i,\theta)}} \\ &= \frac{1}{M} \sum_{i=1}^{M} \{\beta R(\xi_i, \theta) - \log \sum_{m=1}^{K} e^{\beta R(\tau_m^i,\theta)}\}. \end{aligned} \tag{2.7}$$

The derivative is thus given by:

$$\nabla_\theta L = \frac{\beta}{M} \sum_{i=1}^{M} (\mathbf{f}(\xi_i) - \widetilde{\mathbf{f}}(\xi_i)) \tag{2.8a}$$

$$\widetilde{\mathbf{f}}(\xi_i) = \sum_{m=1}^{K} \frac{e^{\beta R(\tau_m^i,\theta)}}{\sum_{m=1}^{K} e^{\beta R(\tau_m^i,\theta)}} \mathbf{f}(\tau_m^i) \tag{2.8b}$$

where $\widetilde{\mathbf{f}}(\xi_i)$ defines the expected feature counts over all samples given θ.

Note that an additional l_1 regularization over the parameter vector θ is introduced in the training process to compensate for possible errors induced via the selected set of features.

The Sampler

From Equation 2.6, we can see that an efficient sampler is extremely important for solving the aforementioned optimization problem in (2.5). Many sampling-based planning algorithms have been widely explored by researchers [34, 66, 53]. In this section, we integrate the sampler from [34] with our previous work on non-conservative defensive motion planning [127] to efficiently generate samples to estimate Z in (2.4).

We represent maps via occupancy grids and feasible trajectory samples are generated using decoupled spatio-temporal approaches. As shown in Figure 2.1, at each time step, the sampler includes three steps: 1) global path sampling via discrete elastic band (ED), 2) path smoothing via pure pursuit control, and 3) speed sampling via optimization and polynomial curves.

Step I - Path Sampling via Discrete ED

The objective of the first step is to generate collision-free paths. The key insight is that by constraining all samples to be safe, we have intrinsically considered the safety constraints of the optimal problem that the demonstrators try to solve. Moreover, it can also reduce the sample space to approximate Z, thus improving the efficiency of the inverse learning algorithm. To account for moving objects, it considers the sweep-volume of each object of interest within a temporal prediction horizon [34]. As shown in Step I in Figure 2.1, a path τ consists of a sequence of elastic nodes (the blue shaded area), i.e., τ=$\{\text{node}^i, \text{IN}^i, \text{OUT}^i\}, i$=$1, 2, \cdots, T$. For each elastic node, we calculate the weighted sum of three types of forces: the contraction force (both left and right) from neighboring spatial nodes, the repulsive force from obstacles, and the attraction force to drive the vehicle towards desired lane centers. Moreover, collision check is conducted for the IN and OUT edges. Then a graph search method is utilized to find a set of collision-free paths that satisfy $\{\mathcal{T}_I\}$=$\{\tau : \text{collision(IN)}$=$0, \text{Force(node)}\leq F_{\text{threshold}}\}$. The $F_{\text{threshold}}$ is a hyper-parameter describing the threshold of the sample set.

Step II - Path Smoothing

All the samples in Step I are piece-wise linear but non-smooth paths, which is not feasible for the vehicle kinematics and thus not suitable to estimate Z. Hence, in Step II, a pure-pursuit tracking controller is employed to smoothen the paths in $\{\mathcal{T}_I\}$ and generate $\{\mathcal{T}_{II}\}$.

Step III - Speed Sampling

This step will generate speed samples for each path sample in Step I. First, a suggested speed profile is generated by finding the time-optimal speed plan under physical constraints (e.g., the acceleration/deceleration limits). Second, local speed curve sampling based on polynomials is performed to explore the neighborhood of the suggested speed profile. The key insight behind such a two-step speed sampling approach is to reduce the exploration space of the speed profile based on the prior knowledge on human drivers, namely they tend

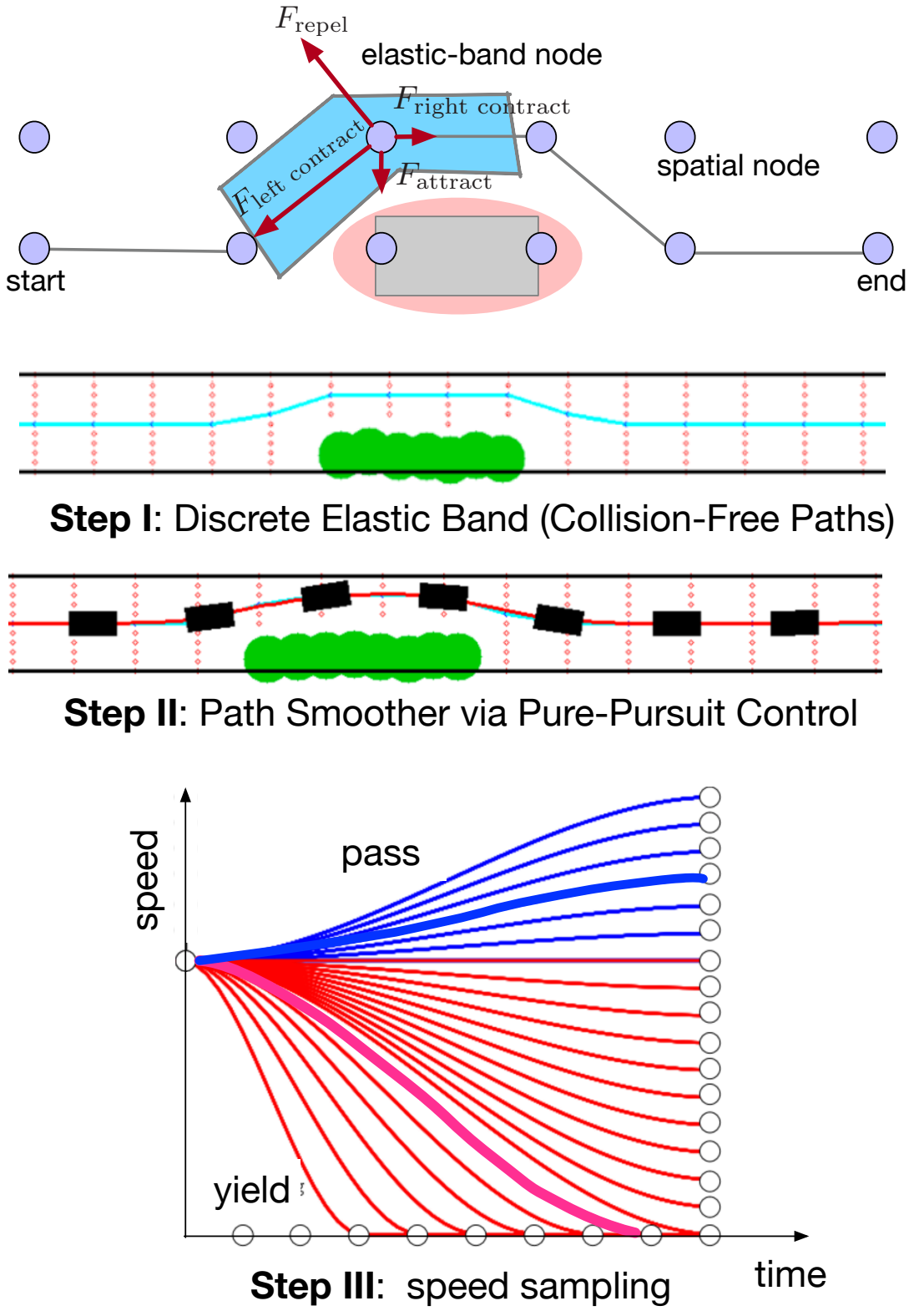


Figure 2.1: The overview of the sampling process.

to pursue time-optimal speed plans. To account for discrete driving decisions in interactive scenarios, we will find one suggested speed profile under each decision and conduct separate local speed curve search around them as in [127]. For instance, as shown in Step III in Figure 2.1, when the ego vehicle and another vehicle are driving simultaneously towards an intersection from crossing directions, we will generate suggested speed profile (thick curves in Figure 2.1) and local speed samples under both the "yield" (red) and the "pass" (blue) decisions. Third-order polynomials are employed for speed curve samples. Hence, via Step III, a spatio-temporal trajectory set $\{\mathcal{T}_{III}\}$ is generated.

Re-Distribution of Samples

Note that in Equation 2.4, the probability of a trajectory ξ is evaluated via the normalization term Z which is estimated via the samples in $\{\mathcal{T}_{III}\}$. However, the samples in $\{\mathcal{T}_{III}\}$ are not necessarily uniformly distributed in the selected feature space $\mathbf{f}(\xi)$, which will cause biased evaluation of probabilities, as pointed in [11]. To address this problem, we propose to use Euclidean distance in the feature space as a similarity metric for re-distributing the samples. As shown in Figure 2.2, the re-distribution process takes two steps: 1) with an initial sample set $\{\mathcal{T}_{III}\}$ ([11](a)), we uniformly divide the feature space into discrete bins, and 2) re-sample within each bin to make sure that each bin has roughly equal number of samples.

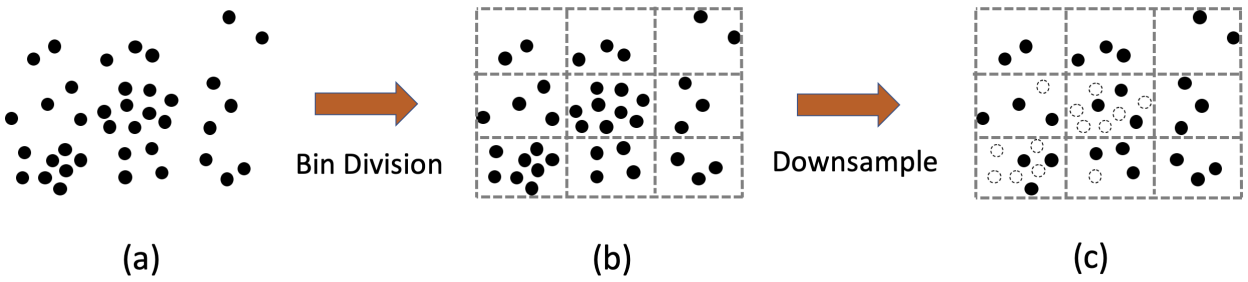


Figure 2.2: Re-distribution of samples.

Summary of the SMIRL algorithm

The proposed SMIRL algorithm can thus be summarized in Algorithm 1.

2.3 Experiments on Driving Behavior

We apply the proposed SMIRL to learn the human driving behavior from real traffic data. Experiments on two types of driving scenarios are conducted: one focusing on independent

Algorithm 1 The Proposed Sampling-based Maximum Entropy IRL for Driving

Result: optimized reward function parameters $\theta^\star$

Input: The demonstration dataset $\mathcal{D}_M = \{\xi_i\}_{i=1:M}$, the convergence threshold ϵ and the learning rate α.

Initialize θ_0, $k = 0$ and compute expected expert feature count $\overline{\mathbf{f}}(\mathcal{D}_M) = \frac{1}{M}\sum_{i=1}^{M}\mathbf{f}(\xi_i)$

Generate the sample set $\mathcal{D}_s^0 = \{\tau_m^i\}_{m=1:K,i=1:M}$ using the sampler in Chapter **??**

Re-distribute the samples according to their similarities as discussed in Chapter **??**, and generate a new sample set $\mathcal{D}_s$

Compute the initial expected feature count over all samples $\widetilde{\mathbf{f}}_0(\mathcal{D}_s) = \frac{1}{M}\sum_{i=1}^{M}\widetilde{\mathbf{f}}_0(\xi_i) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{m=1}^{K}\frac{\exp R(\tau_m^i,\theta_0)}{\sum_{m=1}^{M}\exp R(\tau_m^i,\theta_0)}\mathbf{f}(\tau_m^i)$

while $\|\overline{\mathbf{f}}(\mathcal{D}_M) - \widetilde{\mathbf{f}}_k(\mathcal{D}_s)\|_2 \geq \epsilon$ **do**

  Update θ_k using gradient decent, i.e., $\theta_{k+1} = \theta_k + \nabla_{\theta_k} L = \theta_k + \alpha(\overline{\mathbf{f}}(\mathcal{D}_M) - \widetilde{\mathbf{f}}(\mathcal{D}_s))$

  Compute the expected feature count based on θ_{k+1} over all samples $\widetilde{\mathbf{f}}_{k+1}(\mathcal{D}_s) = \frac{1}{M}\sum_{i=1}^{M}\widetilde{\mathbf{f}}_{k+1}(\xi_i) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{K}\sum_{m=1}^{K}\frac{\exp R(\tau_m^i,\theta_{k+1})}{\sum_{m=1}^{M}\exp R(\tau_m^i,\theta_{k+1})}\mathbf{f}(\tau_m^i)$

  $k = k + 1$

end

$\theta^* = \theta_k$

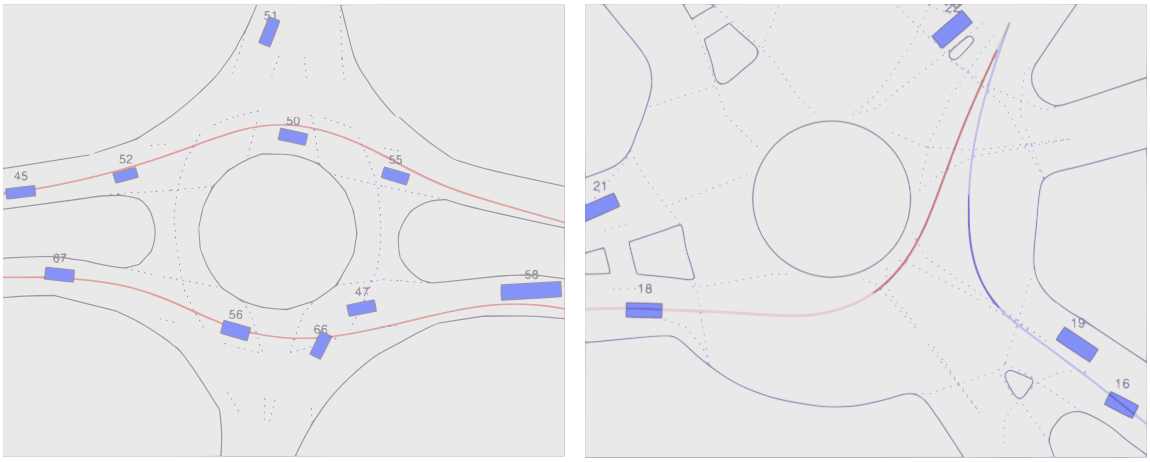
driving behavior (i.e., non-interactive driving (NID)) and the other on interactive driving (ID) behavior at merging traffic. The NID scenario aims to recover humans' preference when they are driving freely, and the ID case aims to capture how human drivers interact with others in merging traffic.

Dataset

We select training data from the INTERACTION dataset [129, 128] with the NID trajectories from the subset DR_USA_Roundabout_SR and ID trajectories from the subset DR_USA_Roundabout_FT, as shown in Figure 2.3.

In the NID scenario, we select 113 driving trajectories which travel across the roundabout horizontally, as demonstrated by the red lines in Figure 2.3(a). 80 trajectories are used as training data and the remaining 33 as test data. In the ID scenario, we selected 233 pairs of interactive driving trajectories with two vehicles: an ego vehicle trying to merge into the roundabout and an interacting vehicle that is already in the roundabout (interactive trajectories at different locations of the roundabout are contained). An illustrative example is shown in Figure 2.3(b) where the blue and red lines represent, respectively, the trajectories of the ego vehicle and the interacting vehicle. The solid segments of the red and blue lines in Figure 2.3(b) indicate the time period when the two vehicles are interacting with each other, and we use them for learning. 150 pairs of trajectories are for training and the other

83 for testing. The sampling time of all trajectories is $\Delta t = 0.1s$.



(a) The NID scenario with DR_USA_Roundabout_SR

(b) The ID scenario with DR_USA_Roundabout_FT

Figure 2.3: Two roundabout scenarios in INTERACTION dataset.

Feature Selection

Recall that in (2.3), we assume that the reward function is a linear combination of a set of specified features. The goal of such a feature space is to explicitly capture the properties that humans care when they are driving. We categorize the features into two types: non-interactive features and interactive features.

Non-interactive features

Speed To describe human's incentives to drive fast and the influence of traffic rules, we define the speed feature as

$$f_v(\xi) = \frac{1}{N}\sum_{i=1}^{N}(v_i - v_{\text{desired}})^2 \tag{2.9}$$

where v_{desired} is the speed limit.

Longitudinal and lateral accelerations Accelerations are related to not only the power consumption of the vehicle, but also to the comfort of the drivers. To learn human's prefer-

ence on them, we include both of them as features:

$$f_{a\_lon}(\xi) = \frac{1}{N}\sum_{i=1}^{N} a_{lon}^2 \tag{2.10}$$

$$f_{a\_lat}(\xi) = \frac{1}{N}\sum_{i=1}^{N} a_{lat}^2 \tag{2.11}$$

Longitudinal jerk We also have included longitudinal jerk as a feature since it can describe the comfort level of human's driving behavior. It is defined as:

$$f_{jerk}(\xi) = \frac{1}{N}\sum_{i=1}^{N} (\frac{a_t - a_{t-1}}{\Delta t})^2 \tag{2.12}$$

Interactive features

To capture the mutual influence between interactive drivers, we define two interactive features.

Future distance The future distance d is defined as the minimum spatial distance of two interactive vehicles within a predicted horizon τ_{predict} assuming that they are maintaining their current speeds (in this chapter, we use τ_{predict}=1s). As demonstrated in Figure 2.4, between time t and $t+\tau$, the blue vehicle drives from p_t^{ego} to $p_{t+\tau}^{\text{ego}}$ and the green one drives from p_t^{other} to $p_{t+\tau}^{\text{other}}$. At time t, their spatial distance is demonstrated via d_t. Hence, The feature of future distance d is given by

$$f_{dist}(\xi) = \frac{1}{N}\sum_{i=1}^{N} e^{-\min_{\tau\in[0,\tau_{\text{predict}}]} d\left(t_i+\tau\right)}. \tag{2.13}$$

Future interaction distance Different from $f_{dist}(\xi)$, the feature related to future interaction distance is defined as the minimum distance between their distances to the collision point, i.e.,

$$f_{\text{int\_dist}}(\xi)=\frac{1}{N}\sum_{i=1}^{N} e^{-\min_{\tau\in[0,\tau_{\text{predict}}]} |s^{\text{ego}}(t_i+\tau)-s^{\text{other}}(t_i+\tau)|} \tag{2.14}$$

where $s^{\text{ego}}(t_i+\tau)$ and $s^{\text{other}}(t_i+\tau)$ represent, respectively, the longitudinal distances of the blue and green vehicles to the shared collision point (the orange circle in Figure 2.4) at time $t_i+\tau$. Again, we assume the vehicles maintain their speeds at t_i through the horizon τ_{predict}.

Note that the above features all have different physical meanings and units. Hence, to assure fair comparison of their contributions to the reward function, we normalize all of them to be within (0,1) before the learning process by dividing them by their own maximum values on the dataset.

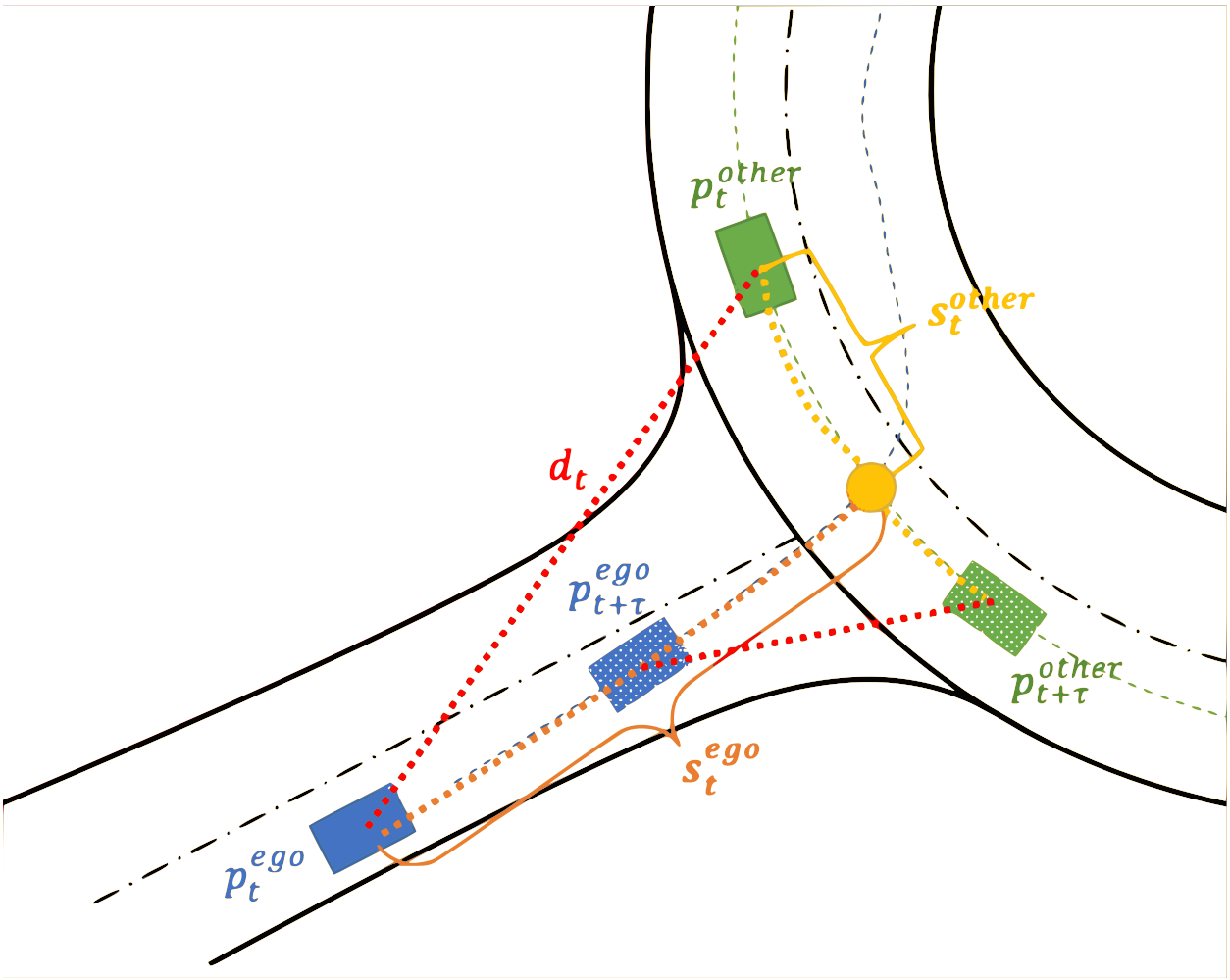


Figure 2.4: The illustration of the proposed interactive features. The blue and green rectangles represent the ego and the other vehicle, respectively. The orange circle is the conflict point of the two vehicles on their routes. The spatial distance d at t is demonstrated via red dotted lines. The longitudinal distances to the collision point for both vehicles at t are shown by the yellow dot curves.

Baseline Methods

To validate the effectiveness and efficiency of our proposed method, we compare our method with three other representative IRL algorithms: the continuous-domain IRL (CIOC) in [61], the optimization-approximated IRL (Opt-IRL) in [59], and the guided cost learning (GCL) algorithm in [26]. CIOC, Opt-IRL and our method are model-based, while GCL is model-free (i.e., deep learning based). All of them are based on the principle of maximum entropy, but differ in the estimation of Z.

- CIOC estimates Z in a continuous domain via Laplace approximation. Specifically, the reward at an arbitrary trajectory $\tilde{\xi}$ can be approximated by its second-order Taylor expansion at a demonstration trajectory $\hat{\xi}_D$, i.e.,

$$R(\theta, \tilde{\xi}) \approx R(\theta, \hat{\xi}_D) + (\tilde{\xi} - \hat{\xi}_D)^T \frac{\partial R}{\partial \xi_D} + (\tilde{\xi} - \hat{\xi}_D)^T \frac{\partial^2 R}{\partial \xi_D^2} (\tilde{\xi} - \hat{\xi}_D). \tag{2.15}$$

  This simplifies $Z = \int_{\tilde{\xi}} e^{\beta R(\theta, \tilde{\xi})} d\tilde{\xi}$ as a Gaussian integral. For more details, one can refer to [61].

- Opt-IRL estimates $Z = \int_{\tilde{\xi}} e^{\beta R(\theta, \tilde{\xi})} d\tilde{\xi}$ via the optimal trajectory ξ_{opt}. Namely at each training iteration, with the updated θ, an optimal trajectory ξ_{opt} can be obtained by minimizing the updated reward function, and $Z \approx e^{\beta R(\theta, \xi_{opt})}$ is utilized as an approximation.

- Different from model-based IRL, GCL parameterizes the reward function as well as the policy via two deep neural networks which are trained together. It uses samples of the policy network to estimate Z in each iteration. Note that the key difference between GCL and the model-based IRL is that GCL does not need manually crafted features, but automatically learns features via the neural networks.

Evaluation Metrics

We employ three metrics to evaluate the performance of different IRL algorithms: 1) feature deviation from the ground truth as in [1], 2) mean Euclidean distance to the ground truth as in [102] and [103], and 3) the likelihood of the ground truth. Definitions of the three metrics are given below.

Feature Deviation

Given a learned reward function, we can correspondingly generate a best predicted future trajectory for every sample in the test set. The feature deviation (FD) between the predicted trajectories and the ground-truth trajectories is defined as follows:

$$\mathcal{E}_{FD} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{N_i} \frac{|\mathbf{f}(\xi_i^{gt}) - \mathbf{f}(\xi_i^{pred})|}{\mathbf{f}(\xi_i^{gt})}. \tag{2.16}$$

where M is the number of trajectories in the test set, and N_i is the length of the i-th trajectory.

Mean Euclidean Distance (MED)

The mean Euclidean distance is defined as:

$$\mathcal{E}_{MED} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{N_i} ||\xi_i^{gt} - \xi_i^{pred}||_2. \tag{2.17}$$

Probabilistic Metrics

We also evaluate the likelihood of the ground-truth trajectories given the learned reward functions through different IRL algorithms. Recalling (2.4), the likelihood of a ground-truth demonstration ξ in the test set is given by

$$P(\xi|\theta, \{\mathcal{T}\}) = \frac{\exp\left(R(\xi,\theta)\right)}{\exp\left(R(\xi,\theta)\right) + \sum_{i=1}^{M} \exp\left(R(\tau_i,\theta)\right)}, \tag{2.18}$$

where $\{\mathcal{T}\}$ is the set of samples generated via our proposed approach. In our proposed method, CIOC and Opt-IRL, $R(\xi,\theta)=\theta^T\mathbf{f}(\xi)$, and in GCL, it is given via a neural network. Among the four IRL algorithms, the one that generates the highest likelihood on the ground-truth trajectories wins.

Experiment Conditions

Among the four different IRL algorithms (i.e., ours vs the three baselines), the CIOC, Opt-IRL and ours are model-based and need manually selected features. GCL, on the other hand, learns to extract features. Hence, for first three approaches, we used the four mentioned features in the non-interactive case (*speed* (v_des), *longitudinal acceleration* (a_lon), *lateral acceleration* (a_lat) and *longitudinal jerk* (j_lon)) and six features in the interactive case (*speed* (v_des), *longitudinal acceleration* (a_lon), *lateral acceleration* (a_lat), *longitudinal jerk* (j_lon), *future distance* (fut_dis), and *future interaction distance* (fut_int_dis)). For the GCL algorithm, no manual features were utilized, and the inputs were directly trajectories.

Other hyper-parameters of the proposed algorithm is set as follows: $F_{\text{threshold}} = 1.0$ with weights on different forces as $w_{\text{contract}} = w_{\text{repel}} = w_{\text{attract}} = 1/3$, i.e., no preference was introduced during the path sampling process over the smoothness of the paths, deviations from the reference lane, and the distance to obstacles.

2.4 Results and Discussion

The performances of the four algorithms, i.e., ours, CIOC, Opt-IRL and GCL, are evaluated based on the metrics in Section 2.3. We conducted two types of tests: one on test sets in the same environment (seen) as the training sets, and one on completely new test sets in a new merging environment (unseen).

Performance on Test Sets in Seen Environments

The results of the four IRL algorithms under the non-interactive and interactive driving scenarios are listed in Table 2.1 and Table 2.2, respectively.

Table 2.1: A summary of the IRL algorithms in the non-interactive driving scenario.

	a_lon	j_lon	v_des	a_lat	MED	Win Count	Log Likelihood
Ours	0.16±0.12	0.20±0.15	0.09±0.04	0.09± 0.03	0.21± 0.06	33	-238.98
Opt-IRL	0.19± 0.19	0.32± 0.19	0.13± 0.06	0.11± 0.03	0.29± 0.09	0	-398.93
CIOC	0.48± 0.42	0.23± 0.17	0.10± 0.07	0.06± 0.05	0.23± 0.09	0	-662.16
GCL	—	—	—	—	3.73± 1.95	0	-1377.65

Table 2.2: A summary of the IRL algorithms in the interactive driving scenario.

	a_lon	j_lon	a_lat	v_des	fut_dis	fut_int_dis	MED	Win Count	Log Likelihood
Ours	0.15± 0.24	0.54± 0.19	0.19± 0.24	0.034± 0.026	0.012± 0.0078	0.032± 0.045	0.066± 0.038	63	-515.97
Opt-IRL	0.69± 1.04	0.55± 0.40	0.20± 0.23	0.083± 0.11	0.021± 0.018	0.043± 0.066	0.14± 0.16	4	-802.01
CIOC	0.42± 0.77	0.69± 0.26	0.26± 0.23	0.064 ± 0.10	0.023± 0.012	0.045± 0.10	0.14± 0.14	9	-595.27
GCL	—	—	—	—	—	—	1.53± 1.16	0	-1196.75

Table 2.3: The learned weights of reward function using our proposed approach (NID refers to non-interactive driving scenario and ID refers to interactive driving scenario)

	a_lon	j_lon	a_lat	v_des	fut_dis	fut_int_dis
NID	1	0.363	0.001	0.14	-	-
ID	1	0.007	0.002	0.102	0.007	0.022

Feature Deviation

Feature deviation was evaluated among the three model-based approaches, i.e., ours, CIOC and Opt-IRL. We can see that compared to the other two algorithms, the proposed SMIRL algorithm achieved relatively smaller $\mathcal{E}_{FD}$ on most of the features in non-interactive scenario, except for the feature *a_lat*. In the interactive scenario, the proposed algorithm achieved smaller $\mathcal{E}_{FD}$ on all features. Moreover, the variations of $\mathcal{E}_{FD}$ are consistently smaller across different features and scenarios. This means that the proposed algorithm learned a reward function that can better capture the general driving preference in the dataset, and thus achieve more stable performance. The learned weights via our proposed approach are given in Table 2.3. We can see that the contribution of the feature *a_lat* is relatively small in both scenarios, particularly in the non-interactive case. Therefore, the proposed algorithm generated a relatively large feature deviation on *a_lat* in the non-interactive case.

MED

MED was evaluated among all four IRL algorithms. In Table 2.1 and Table 2.2, we can see that our method achieved smaller MEDs and variances compared to the CIOC, Opt-

IRL and GCL in both driving scenarios. This indicates that the our algorithm can find a reward function that better explains human driving behaviors, i.e., generates more human-like trajectories. An interesting finding is that GCL, as a deep learning based method, did not necessarily achieve better performance than the model-based IRL algorithms. There might be multiple reasons leading to such an outcome. First, the training sets we utilized were too small for deep learning based models to converge well. Second, the data from real traffic contained noises which might deteriorate the performance of GCL, particularly when the amount of data was not sufficient. Such outcomes also further verified the data efficiency and better robustness of model-based IRL algorithms in the presence of data noises.

Probabilistic Metric

The results of the four methods in terms of the probabilistic metric are also shown in Table 2.1 and Table 2.2. We can see that in both scenarios, the ground-truth trajectories in the test sets all have higher likelihood using the reward function retrieved by our approach compared to those by the other three algorithms, which demonstrates the effectiveness of our proposed approach.

The learned weights in Table 2.3 indicate that humans care more about longitudinal accelerations in both non-interactive and interactive scenarios. During interaction, human drivers pay more attention to future interaction distance, and less on longitudinal jerks and lateral accelerations.

Performance on Test Sets in Unseen Environments

To validate the robustness and generalization ability of our proposed method, we also conducted a comparison among the four IRL algorithms on new test sets in an unseen environment in the training sets. More specifically, we trained all four algorithms using the roundabout merging in the training data and directly tested the trained models on other unseen merging scenarios with different road structures. Both interactive and non-interactive scenarios were tested. The results were given in Table 2.4 and Table 2.5. It is shown that all model-based IRL algorithms, including ours, can generalize better compared to GCL. Both the MED and likelihood did not degrade as significantly as the GCL algorithm.

Table 2.4: Generalization results of different IRL algorithms under the MED metric. The results are in meters.

	Seen NID	Unseen NID	Seen ID	Unseen ID
Ours	0.21	0.74	0.066	0.072
Opt-IRL	0.29	0.89	0.14	0.17
CIOC	0.23	0.90	0.14	0.16
GCL	3.73	46.70	1.53	4.69

Table 2.5: Generalization results of different IRL algorithms under the probabilistic metric.

	Seen NID	Unseen NID	Seen ID	Unseen ID
Ours	-238.98	-399.85	-515.97	-571.60
Opt-IRL	-398.93	-472.51	-802.01	-870.72
CIOC	-662.16	-1153.74	-595.27	-621.13
GCL	-1377.65	-3140.24	-1196.75	-2898.64

Computation Complexity

We also compared the convergence speed of the four IRL algorithms. The time cost is summarized in Table 2.6. We can see that the convergence speed of our method is significantly faster than that of CIOC, Opt-IRL and GCL. Such a superior convergence speed benefits from two reasons. First, the sampling method in our proposed approach is efficient (around 1 minute to generate all samples for the entire training set). Second, the sampling process is *one-shot* in the algorithm through the training process. On the contrary, in each iteration, Opt-IRL needs to solve an optimization problem through gradient descent to find the optimal trajectory given the updated reward function. Such procedure can be extremely time-consuming, particularly when the planning time horizon is long. Thus, Opt-IRL might suffer from the scaling problem with long planning horizon and large training set. GCL also adopts a sampling-based method. However, it needs to re-generate all the samples in every training iteration, while our method only needs to generate all samples once. As for CIOC, the main computation load comes from the computation of gradient and hessian introduced via the Laplace approximation as shown in (2.15). Besides, we also find the stability of the training performance for CIOC is quite sensitive to data noise and the selection of feature sets. Numerical computation issues, particularly for the hessian calculation, could happen and influence the learning performance in the presence of large data noise and/or miss-specified feature sets. As a comparison, our proposed method is much less sensitive to either noise and feature selection, which is also a significant advantage.

Table 2.6: The time cost of the three algorithms for both non-interactive and interactive scenarios. Results are in minutes.

	Ours	CIOC	Opt-IRL	GCL
Non-interactive	6	60	1800	40
Interactive	5	90	1260	30

Table 2.7: Experiment results of the non-interactive scenario with and without the step of sample re-distribution.

	a_lon	j_lon	v_des	a_lat	MED	Win Count	Log Likelihood
w/ re-distribution	0.16± 0.12	0.20± 0.15	0.09± 0.04	0.09± 0.03	0.21± 0.06	33	-238.982
w/o re-distribution	0.18± 0.10	0.30± 0.16	0.12± 0.05	0.11± 0.04	0.26± 0.08	0	-259.064

Table 2.8: Experiment results of the interactive scenario with and without the step of sample re-distribution.

	a_lon	j_lon	a_lat	v_des	fut_dis	fut_int_dis	MED	Win Count	Log Likelihood
w/ sample re-distribution	0.14 ± 0.24	0.53 ± 0.18	0.19 ± 0.23	0.032 ± 0.026	0.012 ± 0.0074	0.027 ± 0.044	0.072 ± 0.043	76	-515.965
w/o sample re-distribution	0.23 ± 0.53	0.55 ± 0.18	0.19 ± 0.23	0.031 ± 0.028	0.012 ± 0.0062	0.027 ± 0.045	0.067 ± 0.041	0	-557.307

The Effect of Sample Re-Distribution

We investigated the effect of sample re-distribution step (in Chapter **??**) for the learning performance. The experiment results with and without the re-distribution step for the non-interactive and interactive scenarios are shown in Table 2.7 and Table 2.8, respectively. We can see that with the procedure of sample re-distribution, the proposed SMIRL algorithm can learn a better reward function in terms of the three categories of metrics: feature count deviations, MED and probabilistic metric of the ground-truth demonstrations in the test set. Most significant improvements are the results on "Win Count": the re-distribution of samples can help better evaluate the probabilities of ground-truth demonstrations. Such results are consistent with the conclusion from [11], namely a uniformed distribution of samples via appropriate similarity function can help generate more accurate probabilistic predictions using the Boltzmann noisily-rational model in Equation 2.2.

2.5 Conclusion

In this chapter, we proposed a sampling-based maximum entropy inverse reinforcement learning (IRL) algorithm in continuous domain to efficiently learn human driving behaviors. By explicitly leveraging prior knowledge on vehicle kinematics and motion planning, an efficient sampler was designed to estimate the intractable partition term when retrieving the reward function. Such benefits were verified by experiments on both non-interactive and interactive driving scenarios using the INTERACTION dataset. Comparing to the other three popular IRL algorithms, the proposed algorithm achieved better results in terms of both deterministic metrics such as feature count deviation and mean Euclidean distance, and probabilistic metrics such as the likelihood of demonstrations in the test set. Moreover, the proposed IRL algorithm shows better generalization ability and converges significantly faster than the baseline methods.

In this work, the effectiveness of the proposed approach for learning driving cost functions was verified in an offline manner, i.e., the ground-truth trajectories and the optimal trajectories from the learned cost functions were compared. In the future, we would like to extend to online experimental verifications where real human drivers can interact with robot vehicles driven by the learned cost functions in simulators. Also, the proposed IRL algorithm in this work was designed specifically for mobile robot systems such as ground vehicles. Extension to general robotic systems with higher dimensions such as robot manipulators will also be considered. Furthermore, the Euclidean distance metric used in the re-sampling step is not necessarily the best metric, and we will explore better metrics in future works.

Chapter 3

Learning Dense Rewards for Contact-Rich Manipulation Tasks

3.1 Introduction

Reinforcement learning (RL) has shown promising performance on a variety of complex tasks, ranging from playing video games to vision-based robotic control [105, 73, 48]. However, the success of RL techniques relies heavily on the availability of high-quality, dense reward signals. Learning policies on tasks with sparse rewards, such as Montezuma's revenge [73], remains notoriously challenging. In robotics, these dense rewards often need to be hand-crafted by a human. This *reward engineering* typically requires significant domain expertise as well as trial-and-error. In contact-rich manipulation tasks, due to the discontinuous dynamics and high-dimensional observation spaces, the challenge of reward specification is further amplified; thus, making the adoption of RL methods for robotic manipulation difficult in practice.

The goal of this chapter is to reduce the effort involved in reward specification for contact-rich manipulation tasks, such as peg-in-hole and connector insertion [55]. Inverse reinforcement learning (IRL) techniques, which learn reward functions from expert demonstrations, offer one such alternative to circumvent the challenge of reward engineering [1]. However, classical IRL techniques require hand-engineered states or features, making their application to tasks with high-dimensional and continuous observation spaces (such as camera images and tactile feedback) challenging [1, 86, 140]. More recently, by leveraging generative adversarial learning [31], IRL techniques have been developed to address continuous [26, 29], high-dimensional observation spaces [99, 72] and successfully demonstrated on robotic manipulation tasks. However, despite their encouraging performance, this class of methods inevitable inherits the instabilities associated with adversarial training [91] and fail to utilize multi-modal observations.

To learn reward functions for contact-rich manipulation tasks while avoiding the challenges of generative adversarial learning, we propose a novel approach **Dense Rewards**

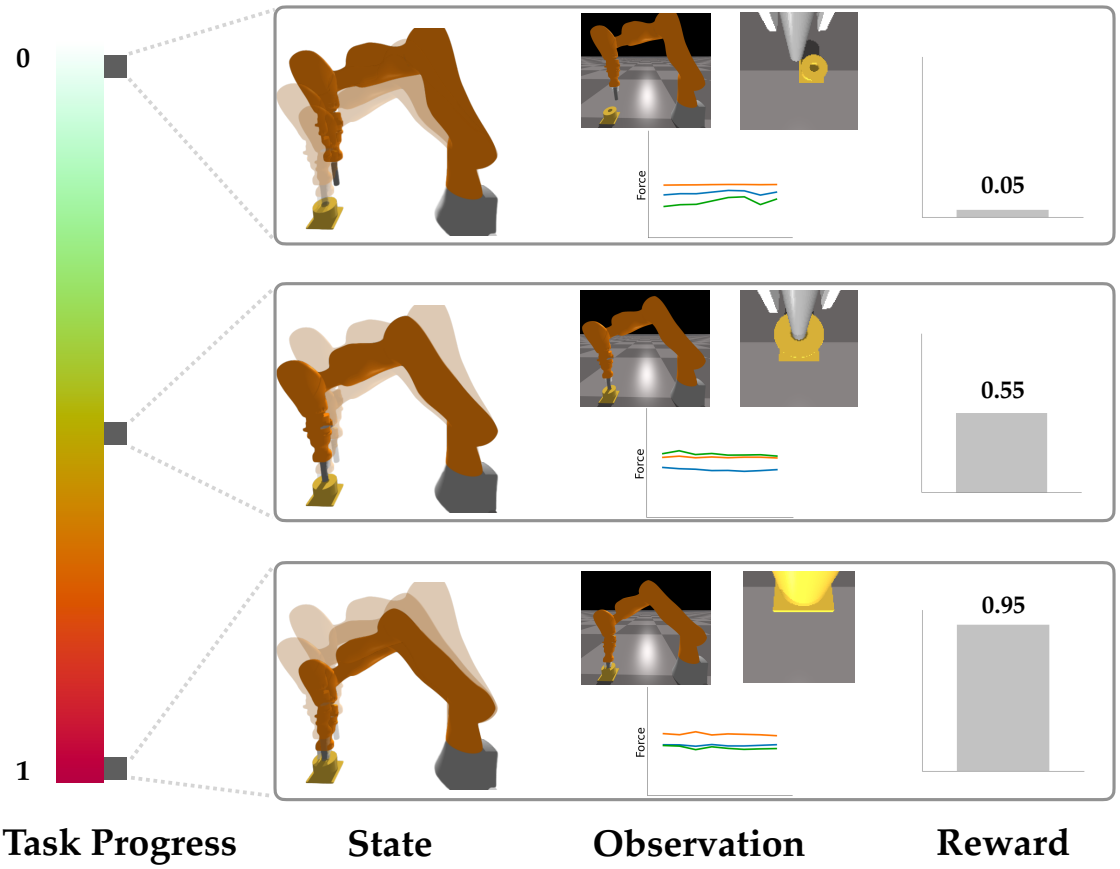


Figure 3.1: We provide a reward learning approach (DREM) based on the notion of *task progress*. For instance, in the figure, the robot is performing a peg-in-hole task. As the robot moves closer to the peg, the learned reward signal increases, thereby facilitating sample efficient training of RL algorithms. DREM arrives at this reward function algorithmically by learning a mapping from robot's multi-modal observations to task progress in a self-supervised manner.

for Multimodal Observations (DREM). Several contact-rich manipulations tasks can be specified through sparse reward signals, such as a goal state or a Boolean measure of task success (e.g., a peg being inside a hole, electricity passing through a connector and socket pair). While these readily available sparse reward signals are often insufficient for sample efficient high-dimensional RL, they can help design dense reward signals when coupled with expert demonstrations or, even, self-supervision. Our approach, thus, combines a sparse reward signal, self-supervision, and (optionally) observation-only expert demonstrations to extract a dense reward function. This dense reward can then be used with any appropriate RL technique to arrive at the robot policy.

Given the sparse measure of task success and optional expert demonstrations, DREM learns a mapping from the high-dimensional observation space to a latent measure of *task progress*. For an arbitrary robot observation, this latent measure aims to provide its task-specific distance to the goal state, thereby serving as a proxy for the dense reward signal. Intuitively, our approach builds on the insight that a state that is closer to the task goal (in a task-specific metric) should be assigned a higher reward than one that is farther, as shown in Figure 3.1. Our core contribution is to realize this insight computationally for high-dimensional, multimodal, and continuous observation spaces. DREM achieves this by first creating a dataset of observations and their corresponding task progress (latent space labels) through an efficient, self-supervised sampling process. This dataset is then used to train an encoder-decoder network, attain a latent space, and obtain the dense reward function.

We evaluate the proposed approach to reward learning on two representative contact-rich manipulation tasks: peg-in-hole and USB insertion. In these evaluations, DREM learns the reward using only *one* expert demonstration. The learned reward function is then used to generate robot policies using Soft Actor-Critic, a model-free RL algorithm [38]. Experimental results show that the policies trained with our learned reward achieves better performance and faster convergence compared to baseline reward functions, including those obtained by recent reward learning approaches.

3.2 Related Work

In this section, we provide a brief review of research on reward learning, with emphasis on applications in contact-rich manipulation. Rewards provide a mechanism for humans to specify tasks to robots [105]. However, reward specification is challenging and can lead to unanticipated behavior [98, 4]. Consequently, approaches to reward learning have attracted increasing attention in the last two decades [16, 39, 95], with inverse reinforcement learning (IRL) being the prominent paradigm [6]. IRL aims to recover a reward function given expert demonstrations [8, 77, 1]. However, as many rewards may explain the demonstrations equally well, additional objective criteria that help identify the *correct* reward have been explored [9, 86, 140, 40]. For instance, [140] utilize the principle of maximum entropy to learn the reward.

Recently, deep variants of IRL have also been developed with the goal of addressing tasks

with larger state, observation, and action spaces [26, 29, 30]. For instance, Finn et al. [26] provide guided cost learning, which uses neural networks to parameterize the reward function. [29] provide a reward learning approach robust to changes in environment dynamics. These IRL techniques and others have been successfully demonstrated on a diverse set of problems, such as modeling driving behaviors [140, 59, 119], robotic manipulation [26, 75], and grasping [47]. Despite the steady success of IRL techniques, several open challenges remain. First, most techniques require hand-engineered features (or states) while efficient learning of rewards from high-dimensional observations remains difficult. Second, state-of-the-art IRL techniques utilize adversarial optimization [26, 29, 30], thereby inheriting the associated training instabilities [91]. Consequently, we seek to develop a reward learning approach that can reason over high-dimensional observations spaces without utilizing adversarial training.

There are a few related attempts along this line of research. Specifically, [117] re-framed imitation learning within the standard reinforcement learning setting using expert policy support estimation. However, they only evaluated their method on a low-dimensional state space. [13] proposed to use reward sketching provided by annotators to learn a dense model from the image input. The major drawback of their work is that their method requires many execution trajectories and the corresponding annotations, which are expensive to acquire. By contrast, in our proposed method, the data used to train the reward model is self-generated by the robot. The sampling process only takes a single execution trajectory, much cheaper than [13].

Many research efforts have been devoted in recent years to learning contact-rich manipulation skills, such as peg insertion, block packing, etc. However, most of the works use either images [62, 96, 138] or haptic feedback [46, 108, 104] as the policy input. There are only a few works that exploit image and force together. Specifically, [24] combined vision and force to learn to play Jenga, [60] proposed to learn a multi-modal representation of sensor inputs and utilize the representation for policy learning on peg insertion tasks, and the authors in [51] achieved motion generation for manipulation with multi-modal sensor feedback using manipulation graphs. However, all the previous works focus on policy learning for manipulation tasks, while our method tries to learn the reward function that can be seamlessly integrated into policy learning algorithms for manipulation. To the best of our knowledge, our work is the first attempt to learn dense rewards from multi-modal inputs.

3.3 Problem Statement

Tasks of Interest

Motivated by near-term applications in assembly, we focus on contact-rich robotic manipulation tasks. In particular, we consider tasks that can be suitably modeled as discrete-time Markov decision processes (MDPs) [84]. Briefly, MDPs describe sequential decision-making problems and are parameterized via the tuple $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$, where $\mathcal{S}$ corresponds to

the state space, $\mathcal{A}$ corresponds to the action space, $\mathcal{T}$ denotes the transition model, and $R(s) \rightarrow \Re$ represents the real-valued reward of state s.

In contact-rich manipulation tasks, the state space is composed of high-dimensional, continuous observations available from robot sensors. The robot observation is typically multi-modal, e.g., visual (through cameras), tactile (through force-torque sensors), and proprioceptive (through measurements of joint angles and velocities). The action space models the actuation capability of the robot. Prior works have explored a variety of action spaces across the joint space, the operational space, and their combinations [71]. Our problem is agnostic to the specification of the action space. In our analysis, without loss of generality, we model actions as the end-effector twist (linear and angular velocity) of the robot.

The transition model represents the physics of the environment. Due to the task being contact-rich, the model is highly non-linear, potentially discontinuous, and difficult to specify analytically. This motivates the need for model-free RL techniques. The objective of an agent solving the MDP is to arrive at a policy $\pi(s) \rightarrow a$ that maximizes its expected cumulative reward. Several contact-rich manipulation tasks encountered in the manufacturing domain can be suitably described by the above MDPs, such as peg-in-hole, connector insertion, and gear assembly [55].

Inputs and Outputs

Assembly tasks are typically categorized by their *repeatability* and presence of a *goal* state s_G, which can be used to arrive at a sparse reward (i.e., a positive value at s_G and zero otherwise). We denote this sparse goal reward as R_G. However, as motivated in Section 3.1, this sparse signal is often inadequate for learning the policy for high-dimensional continuous MDPs, requiring prohibitive amounts of exploration. Instead, to achieve sample efficiency, RL techiques require a richer reward signal that can guide exploration and result in faster convergence to the optimal policy.

Hence, our problem focuses on learning a dense reward function (denoted as R), which when provided as the reward signal to RL techniques results in the optimal policy for the MDP, $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{T}, R_G)$. To solve this problem, we assume access to the MDP state and action specifications, the sparse reward function R_G, a training environment (i.e., robot's real or simulated environment), and (optionally) expert observation-only demonstrations[1], i.e., state sequences from an expert performing the task. Further, we delimit our scope to manipulation tasks where task progress is monotonic (i.e., the optimal policy does not induce cycles in the state space). In relation to RL, the two rewards R_G and R can be viewed as extrinsic (specified by the human designer) and intrinsic (self-supervised) reward signals of the robot, respectively [100].

[1]Acquiring measurements of actions is challenging when collecting demonstrations from human experts [6], motivating the focus on observation-only demonstrations.

3.4 Learning Rewards for Multimodal Observations

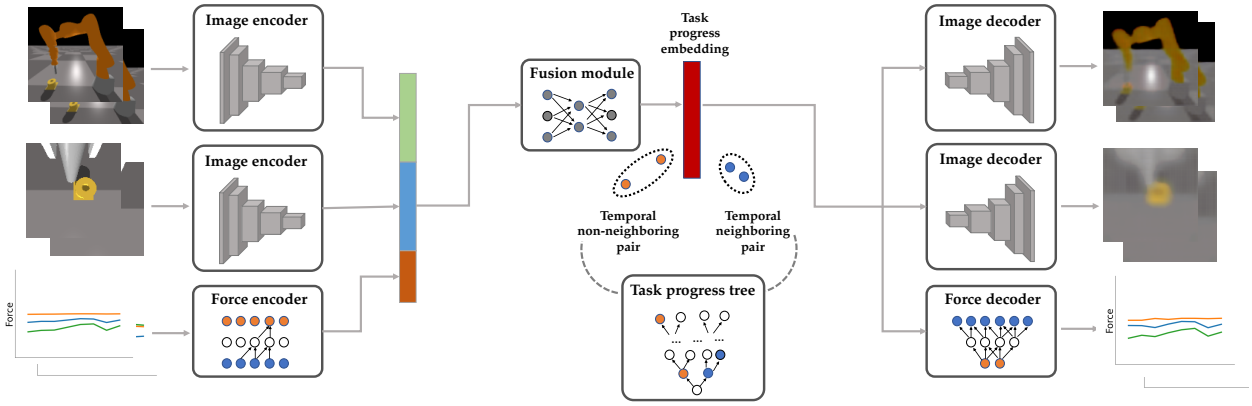


Figure 3.2: DREM takes three sensor signals: RGB images from a fixed camera, RGB images from a wrist-mounted camera, and 6-axis force data of the last 8 frames from a F/T sensor. The latent embedding is learned using a reconstruction loss (defined between the input and the decoded output) and a triplet loss (defined on the temporal neighboring and non-neighboring pairs as detailed in Section 3.4, where the data pairs are sampled from the task progress tree in Section 3.4).

The core idea of our method is to learn a continuous variable $p \in [0, 1]$ from the observation to represent the current progress towards completing the given task. We name it the *task progress* variable. When $p = 0$, it indicates the robot is at the initial state and $p = 1$ indicates the task is finished. If task progress is learned for each observation, it can then be directly used as the reward for policy learning algorithms, i.e., $R(s_t) = p(s_t)$. We propose to learn the function $R(s)$, such that on a nominal task execution trajectory, the task progress variable is aligned with the temporal positions of the states observed. Namely, for any nominal task execution trajectory $\tau = [s_0, s_1, ..., s_{N-1}]$ and s_{N-1} coincides with the goal state s_G, we have

$$t_2 - t_1 > \epsilon \quad \rightarrow \quad R(s_{t_2}) > R(s_{t_1}) \tag{3.1}$$

where $0 \leq t_1 < t_2 \leq N - 1$ and $\epsilon > 0$. Our approach uses this insight for learning an embedding from the observation space $\mathcal{S}$ to a latent space $\mathcal{H}$, $h_\phi : \mathcal{S} \rightarrow \mathcal{H}$, in which the task progress can be measured as the distance between any state s_t and the goal state s_G. Once the embedding is learned, the reward function can be easily derived using the ratio of distance in the latent space:

$$R(s_t) = 1 - \frac{\text{dist}(h_\phi(s_t), h_\phi(s_G))}{\text{dist}(h_\phi(s_0), h_\phi(s_G))}. \tag{3.2}$$

Our proposed framework, titled DREM, is composed of two stages. First, we adopt a novel backward sampling process, i.e., sampling from the goal state to the feasible start region,

to create a dataset of observations s and their temporal positions $p(s)$. Second, we use the generated data to learn the embedding that satisfies the constraints defined in Equation (3.1) and Equation (3.2). It is worth noting that a *single* observation-only expert trajectory is used in the backward sampling process and the embedding is learned in a self-supervised manner.

Backward Sampling

To learn the embedding h_ϕ, state observations with different temporal positions are required as the training data. One straightforward solution is to collect many (tens to hundreds of) expert demonstrations and record the corresponding observations for each state. However, such data collection can be extremely time-consuming, making the adoption of our method difficult in practice. Thus, we propose a novel backward sampling process (i.e., sampling from the goal state to start region) to tackle the data generation problem. The similar idea was also explored in [28] for policy learning with increasing difficulty and [126] for learning generalizable shape descriptor for kit assembly.

Our proposed sampling process is based on the insight that for most manipulation problems the variance of goal state is much smaller than the start state. For instance, consider the peg-in-hole task where goal state corresponds to the peg being inside the hole, while the start state corresponds to the robot holding the peg in free space. For this task, the goal state is significantly more constrained than the start state (which has much larger variance), thereby making backward sampling more efficient compared to forward sampling. We demonstrate the efficacy of the backward sampling method with a single observation-only (without action annotations) expert demonstration available.

We achieve the backward sampling process by constructing a state tree, named as the *task progress tree*, where each node represents a state observation and the node's depth represents its temporal position (in reverse order). A general framework to build task progress trees is detailed as follows:

1. Add goal state s_G to an empty seed set Q^0 with maximum capacity N; assign the current depth d to 0.

2. For each of the state s_q in Q^d, sample M random actions $\{a_k, k = 1, 2, ..., M\}$ and apply each of them to s_q to get the next state observations $\{s_{k|q}^{d+1}, k = 1, 2, ..., M\}$. Add the M next state observations into the tree as the child nodes of the parent node s_q.

3. For each child node sampled in Step 2, compute an approximate progress measure $f(s_{k|q}^{d+1})$. Intuitively, the progress measure $f(\cdot)$ indicates whether the state observation is progressing away from the goal state. It can either be task-agnostic such as measuring the information gain on the distribution of visited states [97], or task-specific such as using one or multiple observation-only expert trajectories as a reference. The latter option is adopted in our implementation and detailed below.

4. Select N samples from the $M \cdot N$ child nodes in Step 2 as the new seed set Q^{d+1} based on the progress measure in Step 3; increment d to $d+1$.

5. Repeat Steps 2-4 until the number of the states in Q^d that are in the start region is above a threshold $N \cdot \delta$, where $\delta \in (0, 1)$. For instance, for the peg-in-hole task, the start region is defined as the set of states with robot's end effector position being above a pre-defined height threshold.

As mentioned above, a single observation-only expert trajectory is used to construct the task-specific progress measure in our implementation (Step 3). Specifically, given the expert trajectory $\xi = \{s_e^0, s_e^1, ..., s_e^{N-1}\}$, we simplify Steps 2-4 by setting $N = 1$. At depth d, we assign Q^d as a single element set, containing the state closest to s_e^{N-d-1} (the $d-$th state in reversed order of the expert trajectory ξ) measured by robot's end-effector pose difference. The obtained task progress tree, consists of state observation nodes along with their depths, equivalently, temporal positions (in reversed order). These states and temporal positions are then used as training data to train the desired embedding h_ϕ.

Multi-Modal Representation Learning

We aim to learn the embedding from multi-modal inputs that satisfy Equation (3.2) using the data generated in Section 3.4. Figure 3.2 illustrates our representation learning model.

Multi-Modality Encoder

Three sources of sensory data are used as the inputs of our multi-modality encoder: RGB images from a fixed camera, RGB images from a wrist-mounted camera, and force-torque (F/T) readings from a wrist-mounted F/T sensor. We adopt a similar encoder architecture as used in [60]. For 128x128x3 RGB images, we use a 6-layer Convolutional Neural Network, followed by a fully-connected layer to transform each image to a 64 dimensional vector. For F/T readings, we take the last 8 readings from the 6-axis F/T sensor as a 8x6 time-series and perform 4-layer causal convolution [79] to transform the F/T readings into a 32 dimensional vector. The three vectors are concatenated and passed as input to a 2-layer Multi-Layer Perceptron to produce the final 128 dimensional hidden vector.

Multi-Modality Decoder

The decoder takes the fused hidden representation as input and tries to reconstruct the multi-modal sensor input. For reconstructing RGB images, we use 7-layer transposed convolution to upsample the hidden vector to the original 128x128x3 size. For F/T readings, we use 3-layer 1-d transpose convolution to convert the data back to 8x6 size. The loss function is defined between the original sensor input and reconstructed output. We use Sigmoid loss for RGB images and L_2 loss for F/T readings.

Task Progress Latent Representation Learning

Our goal is to obtain a latent space, where the distance measure between any state latent representation $h_\phi(s_t)$ and goal state latent representation $h_\phi(s_G)$ reflects the current *task progress* (reward), as shown in Eq (3.2). The supervision signal comes only from the corresponding temporal positions of the states. This is achieved by enforcing the prior that for any state pair (s_{t_1}, s_{t_2}) , when $t_1 < t_2$, s_{t_2} should be closer to the goal state s_G than s_{t_1} in terms of the distance measure in the latent space. Let $g(s_{t_1}, s_{t_2})$ be the distance gap between (s_{t_1}, s_G) and (s_{t_2}, s_G) in the latent space, i.e., $g(s_{t_1}, s_{t_2}) = dist(h_\phi(s_{t_1}), h_\phi(s_G)) - dist(h_\phi(s_{t_2}), h_\phi(s_G))$. We train the desired latent representation by defining the triplet loss as:

$$L_{\text{triplet}} = \begin{cases} \max[0, \beta_1(t_2 - t_1) - g(s_{t_1}, s_{t_2})], \\ \qquad\qquad \text{if } \epsilon < (t_2 - t_1); \text{and} \\ \max[0, g(s_{t_2}, s_{t_1})] + \max[0, g(s_{t_1}, s_{t_2}) - \beta_2], \\ \qquad\qquad \text{if } 0 \leq (t_2 - t_1) \leq \epsilon, \end{cases}$$

where $\epsilon, \beta_1, \beta_2$ are hyper-parameters in our model. We set $\epsilon = 4$, $\beta_1 = 0.04$, $\beta_2 = 0.04$. ϵ is the temporal margin we set to distinguish temporally neighboring pairs from non-neighboring pairs. For temporally neighboring pairs, we enforce them to have similar distances w.r.t. the goal state in the latent space. For non-neighboring pairs, we enforce states whose temporal positions are closer to the goal state should also be closer to the goal state in terms of the distance measure in the latent space. The distance metric in the latent space is defined as:

$$\text{dist}(h_\phi(s_t), h_\phi(s_G)) = 1 - \langle \overline{h}_\phi(s_t), \overline{h}_\phi(s_G) \rangle, \tag{3.3}$$

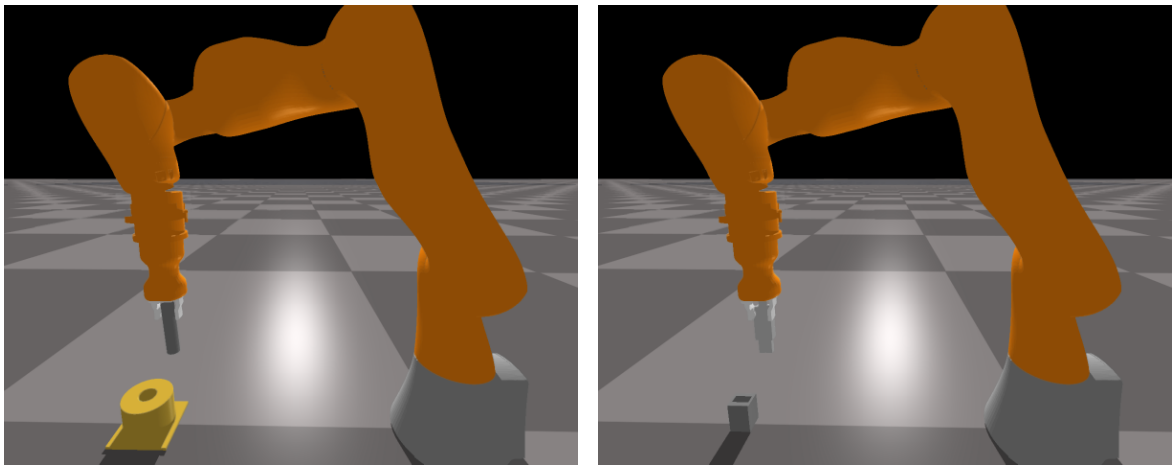
where $\langle \cdot, \cdot \rangle$ denotes the dot product and $\overline{h}$ denotes L_2 normalization. Intuitively, the distance from any state to the goal state is represented by $1 - cos(\theta)$, where θ denotes the angle between the two corresponding embedding vectors.[2]

The whole network is trained using a weighted sum of the reconstruction loss and triplet loss, $L = L_{\text{triplet}} + \alpha L_{\text{reconstruction}}$. We set $\alpha = 0.2$ during training. To prevent the data imbalance issue, we collect the same number of temporal neighboring pairs and non-neighboring pairs when sampling from the task progress tree. Once the embedding is trained, we obtain the reward function according to Equation (3.2).

3.5 Experiments

We aim to investigate two questions in the experiments. First, we examine if the learned reward realizes our insight, namely, states that are closer to the task goal should have higher rewards. This allows us to evaluate the effectiveness of our learned reward. Second, we study how the learned reward can be applied to policy learning in acquiring contact-rich

[2]We explored other distance metrics, e.g., the Euclidean distance, using which our method also works. We empirically found policies trained with rewards using the chosen metric achieves the fastest convergence.



(a) The peg-in-hole task (b) The USB insertion task

Figure 3.3: The two manipulation tasks used in our experiments.

manipulation skills. Our study is based on the evaluation in two common contact-rich manipulation tasks: peg-in-hole and USB insertion (shown in Figure 3.3).

Experimental Setup

All the experiments are conducted in Isaac Gym [68], a GPU-accelerated simulator based on NVIDIA PhysX and Flex simulation backends, which provides high physics fidelity for contact-rich simulation. We defer validating our approach on real-robots to future work. The tasks are performed using a 7-DoF robotic arm modeled on the Kuka LBR IIWA robot. Three types of sensor observations are served as the input to the robot: 128x128 RGB images from a fixed camera, 128x128 RGB images from a wrist-mounted camera, and 6-axis F/T readings from a wrist-mounted F/T sensor. For the peg-in-hole task, we use a round peg with a clearance of 2.4mm. For the USB insertion task, the clearance is 1.0mm. The simulated robot is torque controlled and the torque command is computed via the following control law:

$$\tau = -J^T K_D(\dot{x} - \dot{x}_{ref}), \tag{3.4}$$

where τ denotes the 7-dimension joint torque, J the Jacobian, and K_D the gain matrix. $\dot{x}$ is the current end-effector twist, and $\dot{x}_{ref}$ is the twist command (action) computed by an RL agent or a scripted policy. The joint torques are sent to the robot at 100Hz and the policies are queried at 5Hz.

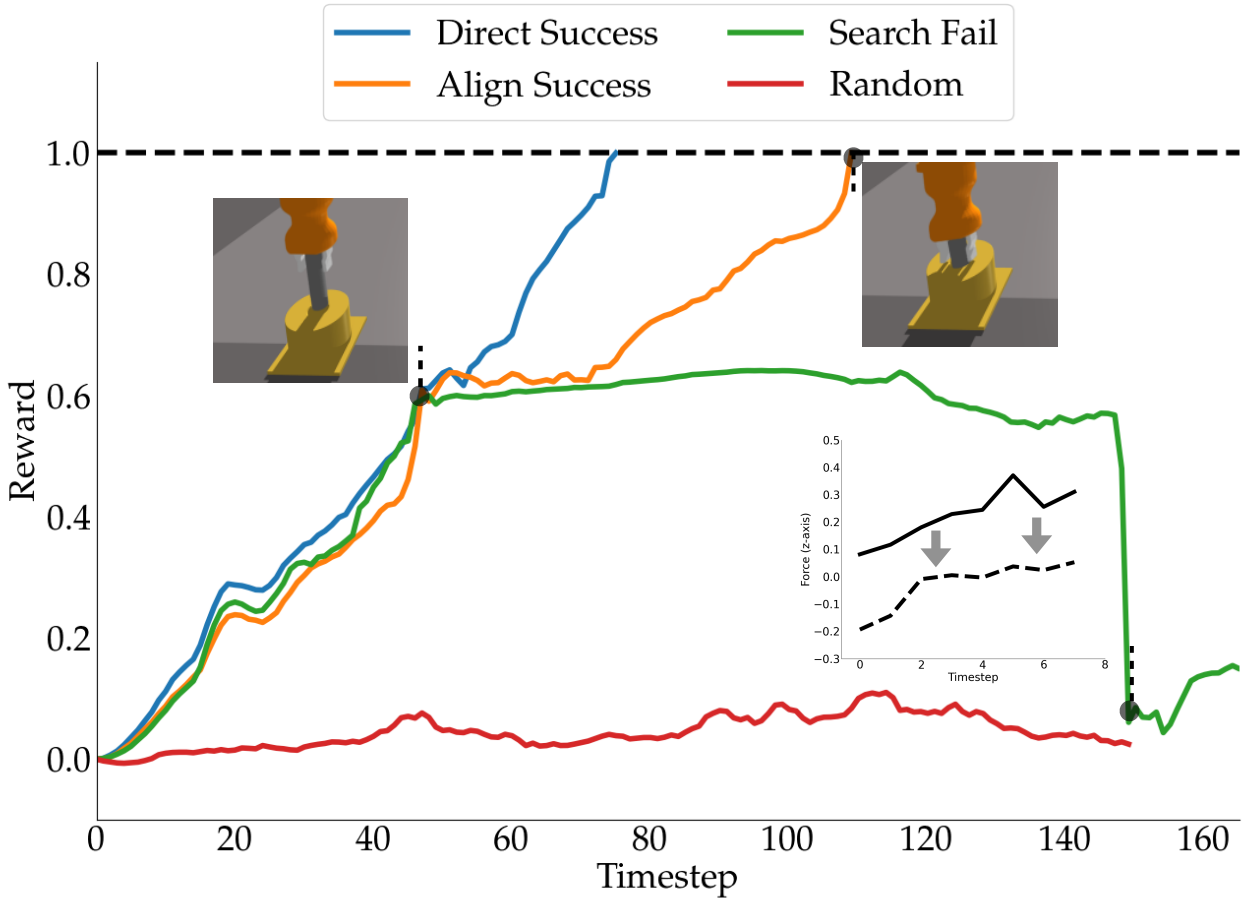


Figure 3.4: Visualization of the learned reward function on four different scripted policies for peg-in-hole task. The results for USB insertion are similar and thus omitted.

Implementation Details

In the backward sampling process for both tasks, we set the number of random actions $M = 500$. We randomly perturb the colors of the objects (peg & hole and USB male & USB female) in the scene to prevent overfitting to certain colors. To train the latent embedding model, we adopted the network architecture detailed in Section 3.4. We trained the model on a Titan 2080Ti GPU for $20,000$ iterations using Adam [56] with learning rate $lr = 2e^{-4}$.

Learned Reward Examination

In this experiment, we want to examine the rationality of the learned reward. Specifically, we want to test if the output of our reward model assigns sensible values to the visited states under various policies. To achieve this, we test our learned reward model with four different scripted policies on the peg-in-hole task listed below.

1. Imitate the expert demonstrator by directly inserting the peg into the hole (`Direct Success`).

2. Approach the hole with a small translation error until contact → Align the peg with the hole center → Insert the peg into the hole (`Align Success`).

3. Approach the hole with a small translation error until contact → Move away from the hole center while the peg is in contact with the hole → The peg loses contact with the hole (`Search Fail`).

4. Random policy (`Random`).

The results of the learned reward model on the four scripted policies are shown in Figure 3.4. In `Direct Success`, the reward increases (approximately) linearly from 0 to 1 along with the number of timesteps in the trajectory. The reward in `Align Success` first increases similarly as in `Direct Success`, until the peg makes contact with the hole surface. Then it roughly stays the same while attempting to align with the hole center, and continues increasing to 1 when the peg is aligned and being inserted into the hole. For `Search Fail`, the reward behaves similarly as in `Align Success` at the beginning (increases and then plateaus). The reward then abruptly decreases to a small value (below 0.1). That is exactly the timestep when the peg loses contact with the hole surface (the force in z-axis rapidly changes from a large positive value to around 0). The reward of `Random` just fluctuates around a small value close to 0. The above observations demonstrate the interpretability of our learned reward model and its potential to provide denser supervision for policy learning.

RL Policy Learning with Learned Reward

The policy learning experiments aim to demonstrate the effectiveness of our learned reward, and provide a rigorous comparison with other reward design methods. While our approach to reward learning is agnostic to the RL algorithm being used, we use Soft Actor-Critic (SAC) in our evaluations [38]. `DREM` refers to our learned reward model introduced in Section 6.3, with all three modalities as input. We compare it with four baseline methods: `Sparse` represents the sparse reward, i.e., the binary success indicator at the end of episode; `Engineered` represents a handcrafted reward $-\exp(\kappa||x - x_G||_2)$, i.e., a function of the L_2 distance between the current robot's end-effector pose and the target pose; `VICE-RAQ` represents the state-of-the-art reward learning algorithm on high-dimensional (image) input [99]; `Image learned` represents our reward model with only images as input. For all baseline experiments, we use RGB images as the input of the policy, while for `DREM` we use RGB images and force data as the input. We adopted a similar neural network in Section 3.4 as the policy network. The experimental results are shown in Figure 3.5.

It can be observed that our learned reward (`DREM` and `Image learned`) outperforms the other reward designs in both peg-in-hole and USB insertion, indicating that our proposed reward learning framework is able to learn reward functions that boosts RL training. The

fact that DREM achieves faster convergence and a higher success rate than Image learned demonstrates the advantages of multi-modal input (force-torque signals in our case), i.e., enabling a more efficient and robust policy. Sparse does not achieve any success within the training budget, showing difficulty of the two contact-rich tasks. Engineered performs satisfactorily at peg-in-hole but much worse at the more difficult USB insertion, as it requires a tighter fit and a strict orientation alignment. It should be noted that VICE-RAQ does not succeed at the two tasks experimented with; the success rate remains 0 after 150k training steps. We hypothesize that only using images, adversarial training is easily stuck in a local optimal as similar images got similar rewards even though they might be in different contact states. We constructed a significantly easier task by configuring the peg to be close to the hole initially (aligned right above the hole center), to avoid training getting stuck due to frequent contact states. A 0.10 success rate is achieved after 50k training steps; however, if the peg is initially misaligned with the hole center, though close, training fails to achieve any success.

3.6 Conclusion

In summary, we propose an approach to learn dense rewards from multi-modal observations (DREM), with particular emphasis on contact-rich manipulation tasks. A novel backward sampling method is proposed to generate requisite training data through exploration, guided by a single expert demonstration. Dense rewards are learned from the high-dimensional observation space without adversarial training. We evaluate DREM on two assembly tasks and demonstrate its efficacy through comparisons with other learned and handcrafted reward functions.

Our work also motivates several future directions. For instance, currently DREM is geared towards *monotonic* tasks, i.e., where task progress is monotonic in the observation space. While this assumption is valid for a range of assembly tasks (due to the presence of defined start and goal states), extension of DREM to non-monotonic tasks remains of interest. Further, the experiments indicate that task-specific logical concepts (such as in-contact or losing-contact) underlie our learned reward. Hence, another interesting direction is to formalize the problem of recovering such logical concepts from dense rewards to enable multi-task learning and consequently further improve sample efficiency.

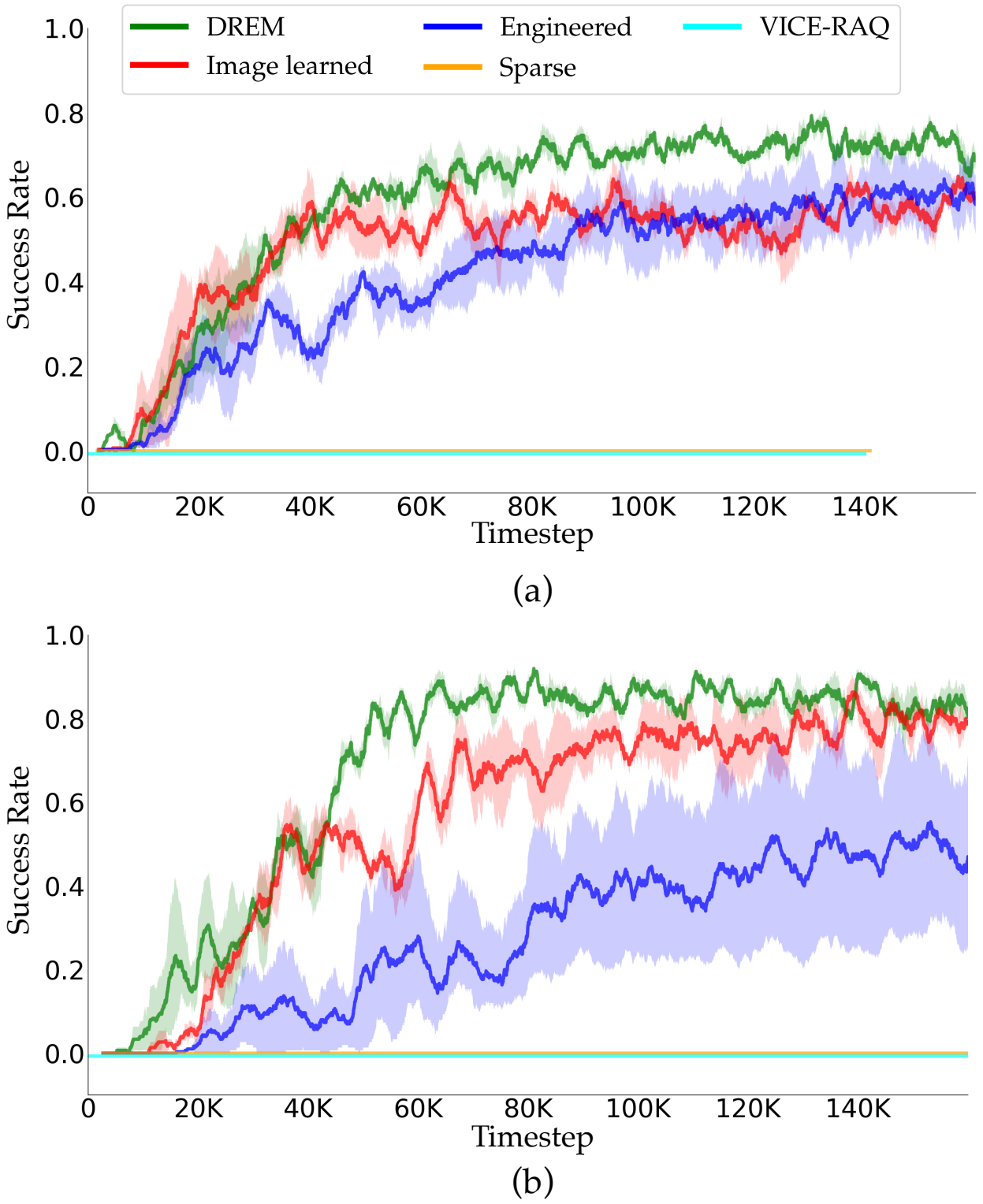


Figure 3.5: Results of the policy learning experiments on peg-in-hole (a) and USB insertion (b). Each method is run with three different random seeds for each task.

Part II

Data-Efficient Policy Learning

Chapter 4

Efficient Reinforcement Learning of Task Planners for Robotic Palletization

4.1 Introduction

The demand for efficient palletization in warehouses and logistics centers has grown significantly in recent years. This increase is largely driven by the expansion of global trade and the rise of e-commerce, which require rapid and precise handling of goods. Traditional, manual methods of palletization are often unable to meet these demands due to their labor-intensive nature and potential for inconsistency. Consequently, the development of robotic systems for palletization has become crucial. These robotic systems, as illustrated in Fig 4.1, are complex and composed of various integral components, including perception systems for item identification and localization, task planning modules for decision-making, trajectory planning and control modules for collision-free and precise execution, etc. In this study, our focus is primarily on the task planning module of robotic palletization systems, which functions as determining the optimal picking order from a stream of boxes, deciding how each box should be rotated, and identifying the precise location for placing each box on the pallet, all in an online and dynamic fashion.

The task planning for robotic palletization in most industrial environments can be conceptually framed as a variant of the online 3D Bin Packing Problems (BPP), where the inventory of items is predetermined, yet their sequence of arrival remains unpredictable [115]. Furthermore, in our work we address a more realistic and challenging problem setting that reflects common practice in robotic palletization systems by including a "buffer" area. This consideration allows the robot to utilize an auxiliary space for storing up to N pending items, thereby expanding its operational capabilities beyond merely handling the immediate one. While many research efforts have been made towards addressing online 3D BPP [52, 36, 116, 115], most of the studies rely on hand-coded heuristic methods. In contrast to

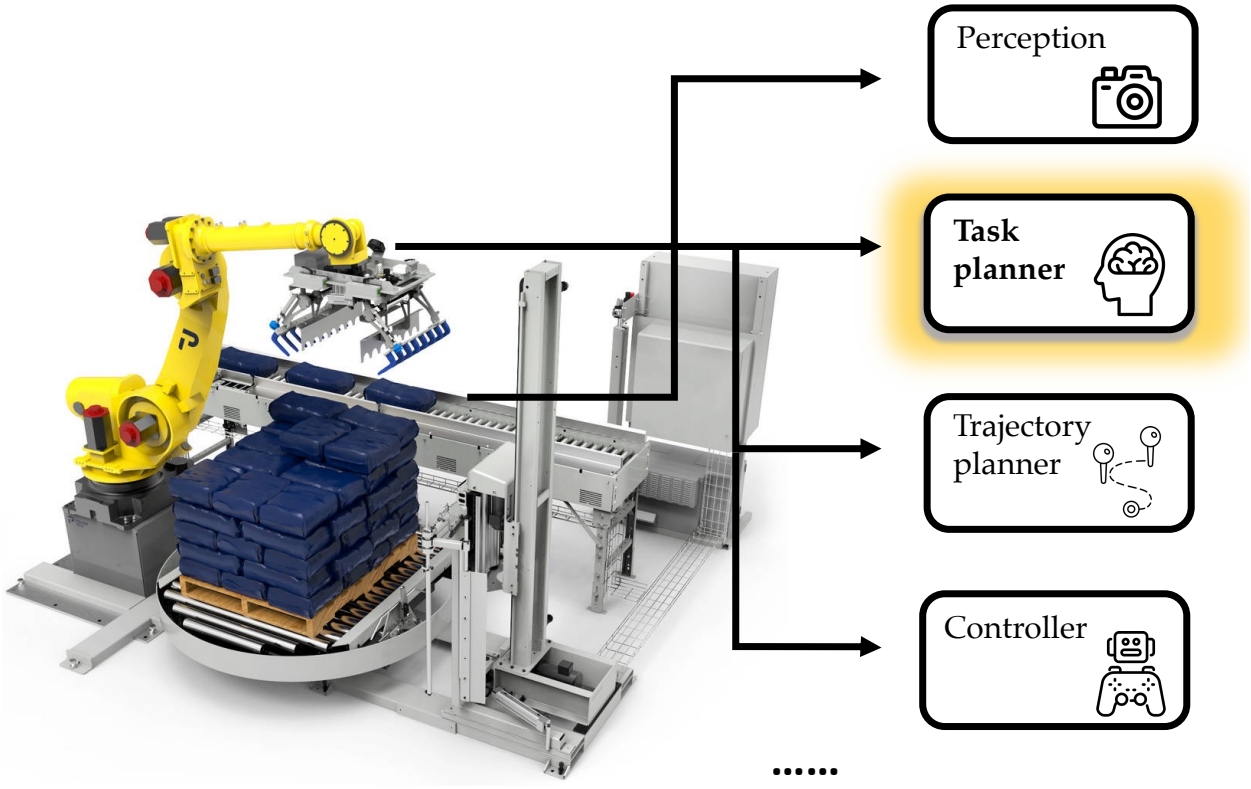


Figure 4.1: A typical robotic palletization system is composed of various modules, including perception, task planner, trajectory planner, controller, etc. Task planning is the main focus of our work.

these approaches, our work aims to statistically learn an optimal and physically feasible palletization task planner. While there are several existing works that use RL to solve online 3D BPP [134, 132, 133], these works are characterized by two primary limitations. Firstly, during policy training, these studies either neglect the aspect of packing stability [134, 132] or assess stability through heuristic-based analyses [133], rendering the policies derived from such methods less viable for deployment in real-world systems where systematic uncertainties are present. Secondly, the problem settings in these works typically overlook the existence of the buffer area, thus potentially limiting their applicability and effectiveness in more complex scenarios.

In this work, we aim to provide an effective solution to our specific online 3D BPP problem. Due to the combinatorial nature of the action space of the problem, applying RL to solve online 3D BPP suffers from the problem of large action space. This challenge is further exacerbated with the introduction of the "buffer" area. One commonly used solution to the problem of RL with large action space is through "invalid action masking" [112, 10, 124], which identifies and masks out the *invalid* actions and directs the policy to exclusively sam-

ple valid actions during the learning phase. Our work adopts a similar approach, pinpointing valid actions as task plans that guarantee the stability of intermediate item stacks on the pallet under gravitational forces throughout the training process. Accurately estimating action masks (i.e., valid actions) for varying states during RL training is crucial, albeit challenging. To address this, we introduce a methodology that learns to estimate action masks through supervised learning (Section 4.3). Specifically, leveraging the image-like characteristics of observation and action mask data, we employ a semantic segmentation paradigm [15, 69] to train the action masking model. The training dataset is compiled through an offline data collection phase, utilizing a physics engine to verify the stability of specific placements across various pallet configurations. Moreover, to ameliorate the potential issue of distribution shift inherent in our approach, we propose a DAgger-like framework [88] designed to iteratively refine the action masking model, thereby enhancing the RL policy's performance (Section 4.3).

To assess the efficacy of our proposed methodology, we conducted extensive experiments and compared our results with existing RL-based online 3D BPP solutions. Experimental results indicate that our proposed method outperforms other baselines and is more sampling-efficient. We also demonstrate the effectiveness of our proposed iterative framework through experimental evaluations. Lastly, our learned task planner is deployed in a real-world robotic palletizer to demonstrate the robustness and applicability of our approach in real-world settings.

4.2 Related Work

In this section, we provide a brief review of research on 3D Bin Packing Problems (BPP) under both the offline and online settings.

Offline 3D BPP

The problem of 3D bin packing, a complex variant of the classical bin packing problem, involves efficiently packing objects of various sizes into a finite number of bins or containers with fixed dimensions in three dimensions. The offline version of the problem, where all information about the objects to be packed is known in advance, has been particularly challenging due to its NP-hard nature. Early strategies were heavily reliant on heuristic methods, as initially demonstrated by [70] that extended 2D BPP algorithms to 3D context. This is followed by many research efforts that propose various heuristic and meta-heuristic algorithms to approximate the solutions [23, 17, 50]. More recently, [41, 131] propose to use deep reinforcement learning (DRL) to learn the policy for offline 3D BPP. Our work differs from these works by focusing on online 3D BPP instead of offline version.

Online 3D BPP

In contrast to offline scenarios, online 3D bin packing problems involve making packing decisions without knowledge of future items, presenting unique challenges in achieving optimality and efficiency. Research in this domain has focused on developing adaptive heuristics and algorithms that can handle the sequential nature of item arrivals. Notably, Karabulut et al. [52] proposes deep-bottom-left (DBL) heuristics and Ha et al. [36] extends [52] by employing a strategy that orders empty spaces using the DBL method and positions the current item into the earliest suitable space. [116] introduces a Heightmap-Minimization technique aimed at reducing the volume expansion of items in a package when viewed from the loading perspective. Aside from heuristics-based methods, there are also research attempts utilizing DRL to solve the problem of online 3D BPP [134, 132]. DRL for online 3D BPP typically suffers from the problem of large action space, making the RL algorithms hard to optimize. [134] address this challenge by implementing straightforward heuristics to reduce the action space, yet their demonstrations are confined to scenarios with limited discretization resolution. Conversely, another study by [132] seeks to alleviate this issue by devising a packing configuration tree that employs more intricate heuristics to identify a subset of feasible actions, marking a progression from their prior work. Our research is distinct from these approaches in two critical aspects. Firstly, the scenario we investigate is notably more complex and mirrors real-world conditions more closely. Specifically, we introduce a "buffer" area, enabling the agent to select any item from a queue of upcoming boxes and rotate it along any axis, thereby significantly expanding the action space. Furthermore, our model accounts for translational and rotational uncertainties that occur when a robot places a box onto the pallet—factors that are overlooked by [132, 134] but are prevalent in actual physical systems. Secondly, from a methodological perspective, our approach diverges from the reliance on varying heuristics to trim the action space. Instead, we introduce a supervised learning-based pipeline to develop an action masking mechanism.

4.3 Our Approach

In this section, we provide detailed description of our proposed method. Initially we specify the problem setting of the online 3D Bin Packing Problem (BPP) addressed in this study. Then a comprehensive description of our action masking pipeline, along with its integration into reinforcement learning (RL), is presented. Furthermore, we introduce the iterative adaptation of our proposed algorithm, illustrating the progressive refinement of our approach.

Problem Formulation

The online 3D BPP can be formulated as a Markov decision process (MDP), denoted by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ signifies the state space, $\mathcal{A}$ denotes the action space, $\mathcal{P}$ represents the transition probability between states, and $\mathcal{R}$ encapsulates the real-valued

rewards. The objective of RL is to find a policy, $\pi(a|s)$, that can maximize the accumulated discounted reward $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$, where γ represents the discounted factor. In the following, we details the components of the MDP in this study.

State In the process of palletization, the policy π evaluates two primary sources of information: the present configuration of the pallet and the dimensions of the forthcoming boxes. Consequently, we model the state at any given time t as $s_t = \{\mathcal{C}_t, \mathrm{d}_t\}$, where $\mathcal{C}_t$ denotes the current configuration of the pallet, and d_t encapsulates the dimensions of the forthcoming boxes. Following [134], we employ a *height map* to depict the pallet configuration. This involves discretizing the pallet into an $l_p \times w_p$ matrix, where each element signifies the discretized height at the corresponding location on the pallet. The variable d_t is conceptualized as a list of N 3-dimensional tuples, with each tuple specifying the (*length*, *width*, *height*) of a box. Here, N represents the capacity of the "buffer" area, which is ascertained by the perception system and typically ranges between 3 and 5.

Action At each timestep, the policy π is tasked with executing three sequential decisions: selecting a box from the N boxes in the upcoming queue (buffer area), rotating the box to the desired orientation, and positioning the box on the pallet. We restrict our consideration to axis-aligned orientations of the boxes, yielding 6 possible orientations. Furthermore, the front-left-bottom (FLB) corner of the box can be positioned in any grid cell among the $l_p \times w_p$ cells available on the pallet. Consequently, the action space is of a combinatorial nature, characterized by a dimensionality of $N \times 6 \times l_p \times w_p$, i.e., action $a \in \mathbb{R}^{N \times 6 \times l_p \times w_p}$. In a typical industrial context, with $N = 5$ and $l_p = w_p = 25$ (at a 1-inch discretization resolution), this translates to an action space with $18,750$ dimensions. The extensive size of the action space introduces additional complexity to the RL problem, complicating the optimization process [22].

Reward We conceptualize the accumulated reward as a measure of the *space utilization* of the current pallet, contingent upon the *stability* of the items positioned on it. The reward function R is formally defined as follows:

$$R(s_t) = \mathbb{1}(s_t) \cdot \frac{V_{total}}{V_{max}} \tag{4.1}$$

Here, $\mathbb{1}(s_t)$ serves as an indicator function that validates the physical stability of the boxes on the pallet at state s_t. V_{total} refers to the cumulative volume of the boxes currently situated on the pallet, while V_{max} signifies the theoretical maximum volume the pallet can accommodate. Given considerations for safety during transport, it is assumed that the maximum height of the boxes on the pallet should not surpass $h_p = 25$ inches, thus defining $V_{max} = l_p \cdot w_p \cdot h_p = 25^3$. Consequently, we can delineate a dense reward at each timestep as $r_d(s_t) = \mathbb{1}(s_t) \cdot \frac{V_t}{V_{max}}$, where V_t denotes the volume of the most recently placed item.

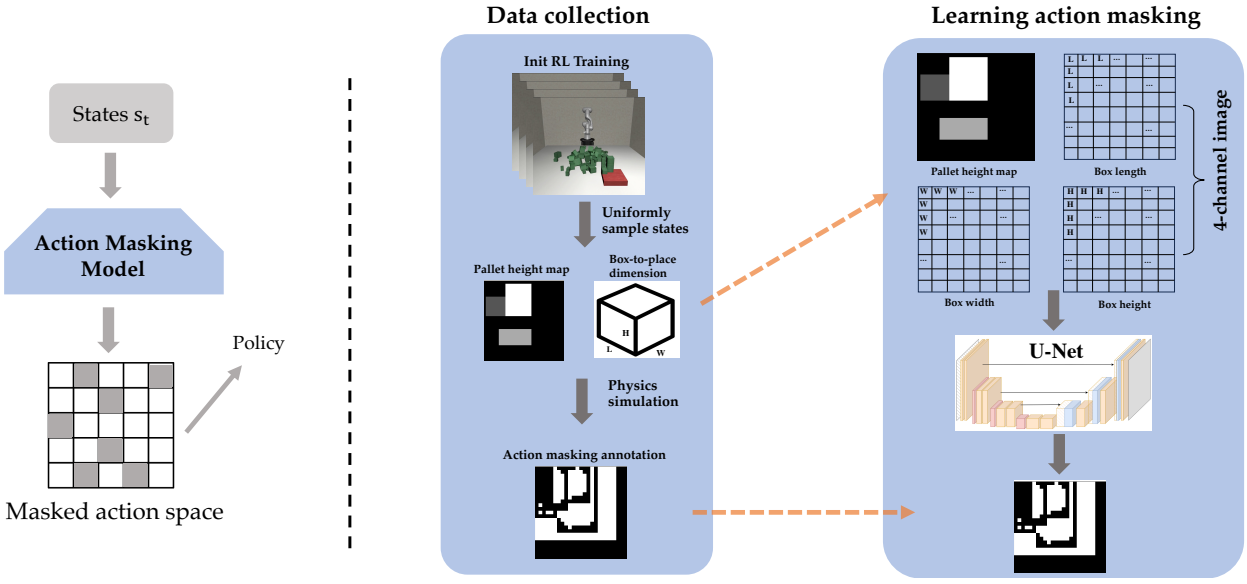


Figure 4.2: An overview of our action masking learning process. The methodology unfolds in three phases: data collection, for gathering relevant training data; learning the action masking model, where a U-net architecture learns to distinguish stable from unstable placements; and embedding the learned action masking model into RL training, integrating the model to dynamically reduce the action space and enhance RL optimization.

Action Space Masking via Supervised Learning

The principal challenge in applying RL to address the online 3D BPP resides in the expansive dimensionality of the action space, which complicates the optimization of RL algorithms. To counteract this issue, we embrace the strategy of "invalid action masking" [112, 10, 124], which serves to alleviate the aforementioned challenge. This section elucidates our methodology for developing the action masking model through supervised learning. The fundamental premise behind action space masking is the exclusion of invalid placements that could culminate in unstable pallet configurations. While prior studies have implemented similar strategies to exclude unstable placements and thereby simplify the learning process[133, 134], their approaches to action masking rely predominantly on heuristic algorithms. We posit that heuristic-based action masking exhibits several limitations: (i) The necessity for multiple hyperparameters in most action masking methods poses a challenge in tuning these parameters to accommodate diverse scenarios, such as variations in box sizes and weights. ii) Heuristic methods, particularly those involving the analysis of the center-of-mass (CoM), fail to account for uncertainties inherent in palletization execution, necessitating further elaboration. For instance, consider a heuristic that uses CoM analysis to prevent tipping by restricting box placements that significantly shift the CoM. However, this doesn't consider real-world uncertainties like slight placement inaccuracies due to robot sensor errors,

leading to unexpected CoM shifts and potentially unstable configurations despite heuristic approval. This example underscores the limitations of CoM-based heuristics, highlighting their lack of accommodation for practical uncertainties in robot execution, which questions their reliability in real-world scenarios. In response, our work investigates the feasibility of adopting supervised learning to construct the action masking model. Figure 4.2 delineates the framework of our proposed methodology for learning action masking. Specifically, the learning process is segmented into the following phases:

Data Collection The action masking models eliminate unstable placements by learning to infer the stability of sampled box placements during RL training process. Specifically, we aim to identify a set of feasible placements, or action masks $g_t \in \mathbb{R}^{l_p \times w_p}$, based on the configurations of the pallet $\mathcal{C}_t$ and the (rotated) dimensions of the boxes b_t, formulated as $g_t = f_\theta(\mathcal{C}_t, b_t)$. The collection of data samples consisting of $(\mathcal{C}_t, b_t)$ pairs along with their corresponding g_t is essential for the learning of the action masking model f_θ. However, ensuring that the collected data accurately reflects the RL training distribution poses a significant challenge. To address this, we initially employ a heuristic-based method to develop an initial RL policy π_o, from which we systematically sample and store $(\mathcal{C}_t, b_t)$ pairs for subsequent offline data collection.

In the data collection phase, our goal is to generate the corresponding annotations g_t given the sampled RL states $(\mathcal{C}_t, b_t)$. Specifically, g_t represents the placement stability on every discretized bin of the pallet for a given pallet state $\mathcal{C}_t$ and box dimension b_t. To achieve this, we leverage the physics simulation, i.e., MuJoCo [110], to evaluate the stability of every potential placement on the pallet, thereby generating g_t. The dataset thus compiled, denoted as $\mathcal{D} = \{(\mathcal{C}_t, b_t, g_t)\}$, forms the basis for the subsequent learning stage of the action masking model.

Learning the Action Masking Model As illustrated in Figure 4.2, g_t functions as a binary image corresponding in size to $\mathcal{C}_t$, with pixels valued at 1 indicating stable placements and those valued at 0 signifying unstable ones. Leveraging the image-like properties of both the input and the output, we transform the learning challenge into a well-established computer vision task known as semantic segmentation [15, 69, 87]. To this end, we generate three single-channel images mirroring the dimensions of $\mathcal{C}_t$, each channel encoding the length, width, and height of the box, respectively. These images are then combined with $\mathcal{C}_t$ to produce a 4-channel image, which serves as the input, with g_t acting as the target output. This input-output pair is subsequently processed by a conventional U-net [87] architecture for training purposes. The resulting neural network, regarded as the trained action masking model f_θ, is subsequently integrated into the RL training regimen.

Embedding the Learned Action Masking Model into RL Training We employ the developed action masking model to trim the action space during RL training, enhancing the efficiency of the learning process. At each timestep, with the pallet configuration $\mathcal{C}_t$ and the

dimensions of the box to be placed b_t given, the model f_θ identifies a viable subset $\hat{g}_t$ from the total available placements ($l_p \cdot w_p$ placements). The RL policy π is then restricted to selecting placements solely from this feasible subset $\hat{g}_t$. This strategy effectively narrows the action space, significantly facilitating the RL training. We demonstrate the effectiveness of the action masking model for RL training in Section 6.4.

Iterative Action Masking for RL Training

While being effective, the learned action masking model can still be inaccurate in certain states, particularly for the out-of-distribution (OOD) states encountered during the RL training process. Such discrepancies arise because during RL training the agent might explore states that diverge significantly from those within the training dataset $\mathcal{D} = (\mathcal{C}_t, b_t, g_t)$, leading to a misalignment between the distributions of the training data and the actual scenarios the model faces.

Inspired by DAgger [88], we introduce an iterative enhancement pipeline for refining the learning of the action masking model. This process begins with the application of a heuristic-based method for initial action masking during RL training, yielding the initial policy π_0. Concurrently, we systematically collect state samples during π_0 training, assembling dataset $\mathcal{D}_0$ via offline collection as detailed in Section 4.3. Utilizing $\mathcal{D}_0$, we then train the neural network-based action masking model f_θ^0, as depicted in Section 4.3. This model, f_θ^0, is subsequently integrated into a new RL training cycle to derive an enhanced policy π_1. This cycle of training, data collection to form $\mathcal{D}_i$, and model updating to f_θ^i is repeated, each iteration aiming to embed the updated action masking model in the subsequent RL training phase. This iterative process continues until the RL policies cease to show significant improvements, thereby optimally aligning the action masking model with the encountered state distributions. The pseudo code of the proposed method is presented in Algorithm 2.

Algorithm 2 Iterative action masking model learning for RL training

Result: Best policy π_i on testing.

Input: Initial policy π_0 and dataset $\mathcal{D}_0$ obtained from RL training with heuristics-based action masking.

Initialize $\mathcal{D} \leftarrow \mathcal{D}_0$
Train action masking model f_θ^0 with $\mathcal{D}$
for $i = 1$ **to** N **do**
    RL training with f_θ^i to obtain policy π_i
    Collect dataset $\mathcal{D}_i$ by uniformly sampling from π_i training
    Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
    Train f_θ^i with $\mathcal{D}$
end
return *best* π_i *on testing*

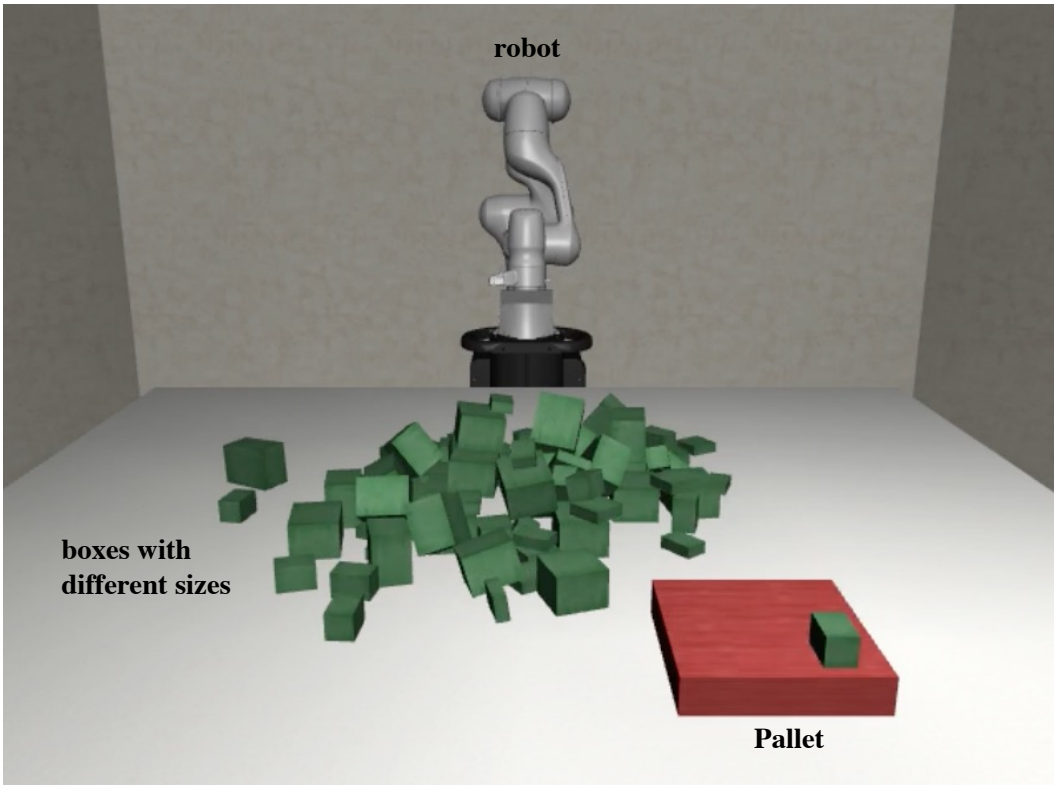


Figure 4.3: Visualization of our simulated palletization environment in MuJoCo [110]. Although 80 boxes are displayed for illustrative purposes, the robot is programmed to perceive and interact with only N boxes within the buffer area. The arrangement of the boxes is randomized and unknown, shuffled anew for each RL episode.

4.4 Experimental Validations

In this section, we present a thorough quantitative analysis of our proposed approach. Our evaluation is structured to address the following key questions: 1) Does the method outlined in Section 4.3 successfully learn to identify the feasible subset of placements? 2) Does the integration of the learned action masking model enhance the RL training process, culminating in a superior policy? 3) Does the iterative action masking learning strategy described in Section 4.3 contribute to further enhancements in the RL agent's ultimate performance?

Experimental Setup

To explore the task planning challenges associated with the palletization problem, we developed a simulated palletization task in MuJoCo [110], reflecting a realistic logistic scenario, as depicted in Figure 4.3. This simulation environment features 80 boxes of 5 distinct sizes,

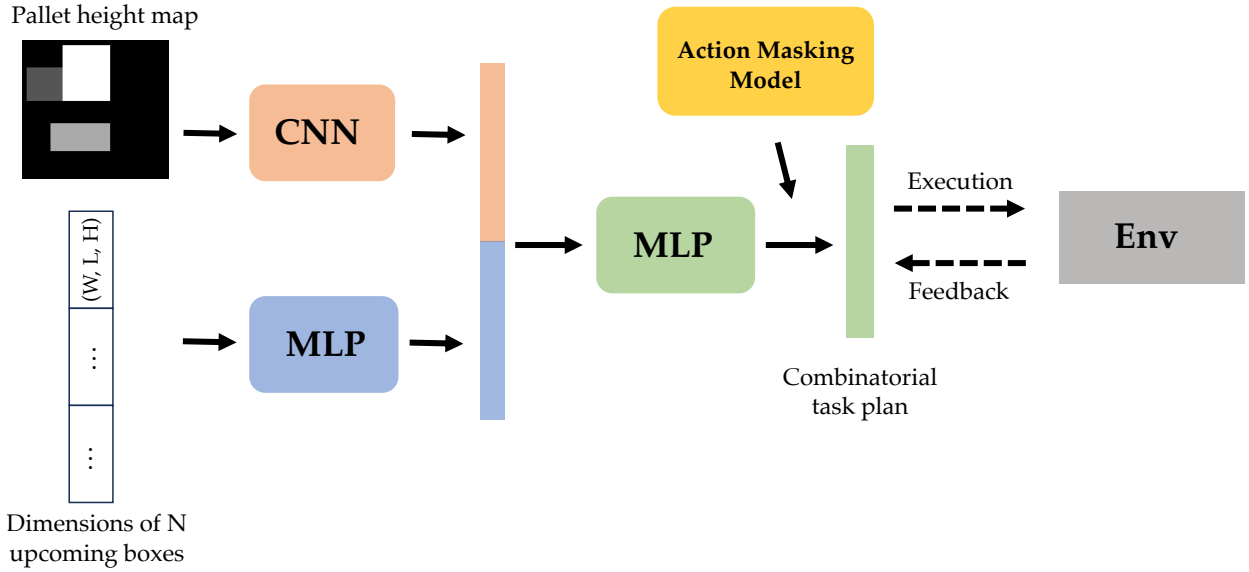


Figure 4.4: The policy network architecture adopted in our study. We use a CNN to encode the height map and a MLP to encode the dimensions of forthcoming boxes. The resulting embeddings are concatenated and serve as the input to the policy network. The action masking model, if it exists, helps the policy network ignore infeasible actions during learning.

with dimensions (in inches) of $10 \times 8 \times 6$, $9 \times 6 \times 4$, $6 \times 6 \times 6$, $6 \times 4 \times 4$, $4 \times 4 \times 4$. The pallet is specified to have dimensions of 25×25 inches, with a stipulation that the maximum height of stacked boxes cannot exceed 25 inches. For our experiments, we applied a discretization resolution of 1 inch to both the boxes and the pallet.

At each planning step, the task planner receives the current pallet configuration, represented as a height map, and the dimensions of N forthcoming boxes in the buffer, then generates a task plan. This plan involves selecting one of the N boxes, orienting it as desired, and placing it on the pallet, in accordance with the methodology described in Section 4.3. To expedite the learning process, the actual 'pick-rotate-place' actions by the robot are bypassed; instead, the chosen box is immediately positioned at the planner's determined goal pose. To account for uncertainties inherent in physical execution, each box placement is subjected to random translational noise on the xy plane $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{0.05})$ and rotational noise around the z-axis $\delta_r \sim \mathcal{N}(0, 5)$, measured in inches and degrees, respectively. The sequence in which the 80 boxes are presented to the policy is randomized and unknown at the start of each RL training episode, and the weights of the boxes are also randomized to further enhance the simulation's fidelity to real-world variability.

Implementation Details and Baselines

Our objective is to validate the efficacy of the learned action masking model within the RL training framework. To this end, we implemented two baseline methodologies, `NoMask` and `HeuristicsMask`, for comparison against our proposed `LearnedMask` method, detailed in Section 4.3. The implementation specifics are outlined as follows:

`NoMask` In the `NoMask` approach, action masking is not applied during the RL training process. Although this method can be utilized with various RL algorithms, we have chosen to implement PPO [94] for all experiments conducted in this study. During the training, the policy receives as input the discretized pallet height map alongside the dimensions of N forthcoming boxes, subsequently generating a combinatorial task plan. To process this information, a Convolutional Neural Network (CNN) is used to encode the 2D height map into a vector, while a Multi-Layer Perceptron (MLP) encodes the dimensions of the incoming boxes. These vectors are then concatenated and supplied to the policy as input. The architecture of the policy is depicted in Figure 4.4 for further clarification.

`HeuristicsMask` The `HeuristicsMask` configuration is similar to `NoMask`, with the distinction of incorporating a heuristics-based action masking strategy during RL training. Specifically, we employ the heuristic criteria outlined in Zhao et al. [134] to determine the feasible subset $\hat{g}_t$ at each timestep. According to these criteria, a placement location is deemed feasible if it satisfies any of the following conditions: 1) more than 60% of the base area of the box to be placed, along with all four of its bottom corners, receive support from the boxes already on the pallet; 2) more than 80% of the base area and at least three of the four bottom corners are supported; or 3) more than 95% of the base area is supported.

`LearnedMask` `LearnedMask` (Ours) also shares the same settings as `NoMask`, except that our proposed learning-based action masking is performed during RL training, as described in Section 4.3.

Experimental Results

In this section, we provide evaluation results under different experimental settings, aiming to address the three questions posed at the outset of Section 6.4.

Learned Action Masking Model as Accurate Stability Predictor

A fundamental approach to assessing the effectiveness of the learned action masking model involves evaluating the accuracy of its predictions. As outlined in Section 4.3, the action masking model functions as an estimator, predicting the stability of specific placements and essentially operates as a semantic segmentation model. Therefore, we employ the standard Intersection-over-Union (IoU) metric to gauge the action masking model's performance

in stability prediction. The performance metrics for both our learned action masking model and the heuristic-based action masking, as introduced in Section 4.4, are reported for comparison. The heuristic-based action masking achieves an IoU of 76.6%, whereas our learned action masking attains an IoU of 89.2%, markedly surpassing the heuristic approach. This result demonstrates that the learned action masking model can provide a more accurate action space mask for policy sampling than the heuristic-based method, potentially enhancing RL training outcomes.

RL Policy Performance Improvement with Learned Action Masking

Intuitively, a precisely calibrated action masking model can significantly reduce the exploration space inherent in the original RL problem, thereby simplifying the optimization process. To empirically verify this, we conducted experiments by comparing the policy efficacy of our proposed `LearnedMask` model against two baselines: `NoMask` and `HeuristicsMask`, as elaborated in Section 4.4. We adopted the *space utilization* as defined in Section 4.3 as the reward. Figure 4.5 shows the performance of the three methods under the setting when the buffer size $N = 1$ as training proceeds, across 5 random seeds. Observations reveal that both `LearnedMask` and `HeuristicsMask` significantly surpass `NoMask` in performance, thereby affirming the hypothesis that optimization within the original action space of RL is inherently challenging. Action masking thus serves to effectively narrow the problem domain, substantially enhancing policy performance. Moreover, `LearnedMask` not only converges more rapidly but also achieves superior performance relative to `HeuristicsMask`, indicative of the premise that a more precise action masking model correlates with improved policy learning.

In a bid to comprehensively assess policy performance across variable configurations, we tabulated the average space utilization metrics for all three methods under varying buffer sizes, specifically, $N = 1, 3, 5$. These results were derived from executing the corresponding policy across 20 iterations and computing the mean reward from these episodes. Notably, during these test episodes, the arriving order of the boxes is random and unknown to the policy, identical to the training phase. The outcomes of this analysis are encapsulated in Table 4.1. As depicted, `LearnedPrune` consistently outperforms `HeuristicsPrune` across all experimental conditions, thereby evidencing the robustness of our algorithm. Additionally, an increase in N correlates with an enhancement in policy performance, a logical deduction considering the expanded decision-making vista afforded by a larger buffer. Nonetheless, in practical applications, the feasible value of N is constrained by various factors, including the limitations of the perception system and physical spatial constraints. Within the context of most industrial palletization scenarios, a buffer size of $N = 5$ represents a pragmatic upper limit.

Iterative Policy Improvements with Iterative Action Masking Learning

Here we aim to investigate if the iterative action masking learning algorithm in Section 4.3 can further improve the performance of RL policy over the iterations. Figure 4.6 delineates

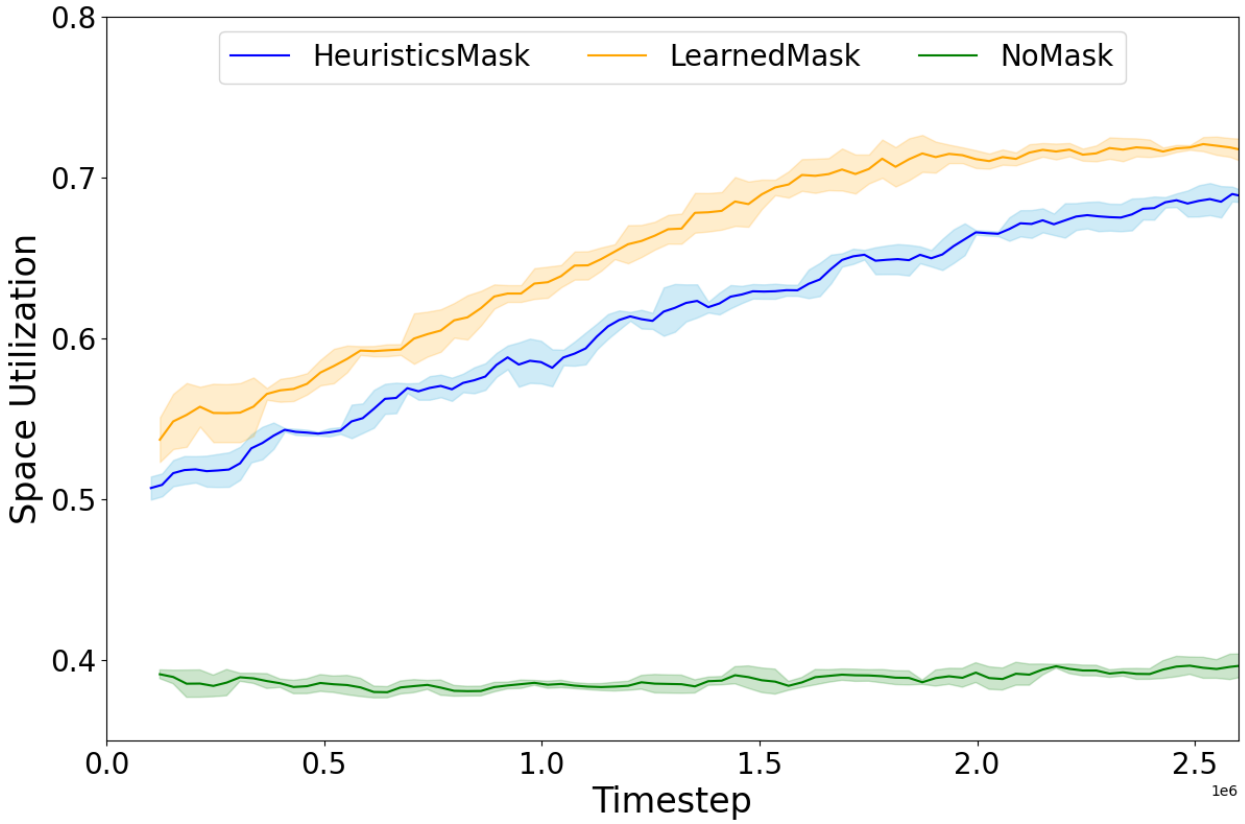


Figure 4.5: Learning curve of the three methods when buffer size $N = 1$. Results are averaged over 5 random seeds. Our method (`LearnedMask`) converges faster and achieves better space utilization compared to the baseline methods.

Table 4.1: Average space utilization of the obtained policies under different buffer size N.

Buffer size	$N = 1$	$N = 3$	$N = 5$
`NoMask`	38.8%	37.6%	38.1%
`HeuristicsMask`	69.4%	71.9%	72.7%
`LearnedMask`	**72.1%**	**75.1%**	**76.2%**

the relationship between policy performance and the number of iterations, denoted by T. Specifically, $T = 0$ denotes the baseline scenario of RL training underpinned by heuristic action masking, represented as `HeuristicsMask`, while $T = 1$ signifies the employment of `LearnedMask`. Through empirical investigation, we observed a notable enhancement in performance during the initial iterations ($T = 2, 3$), with improvements of 1.0% and 0.3% over their immediate predecessors, respectively. However, the policy performance "saturates"

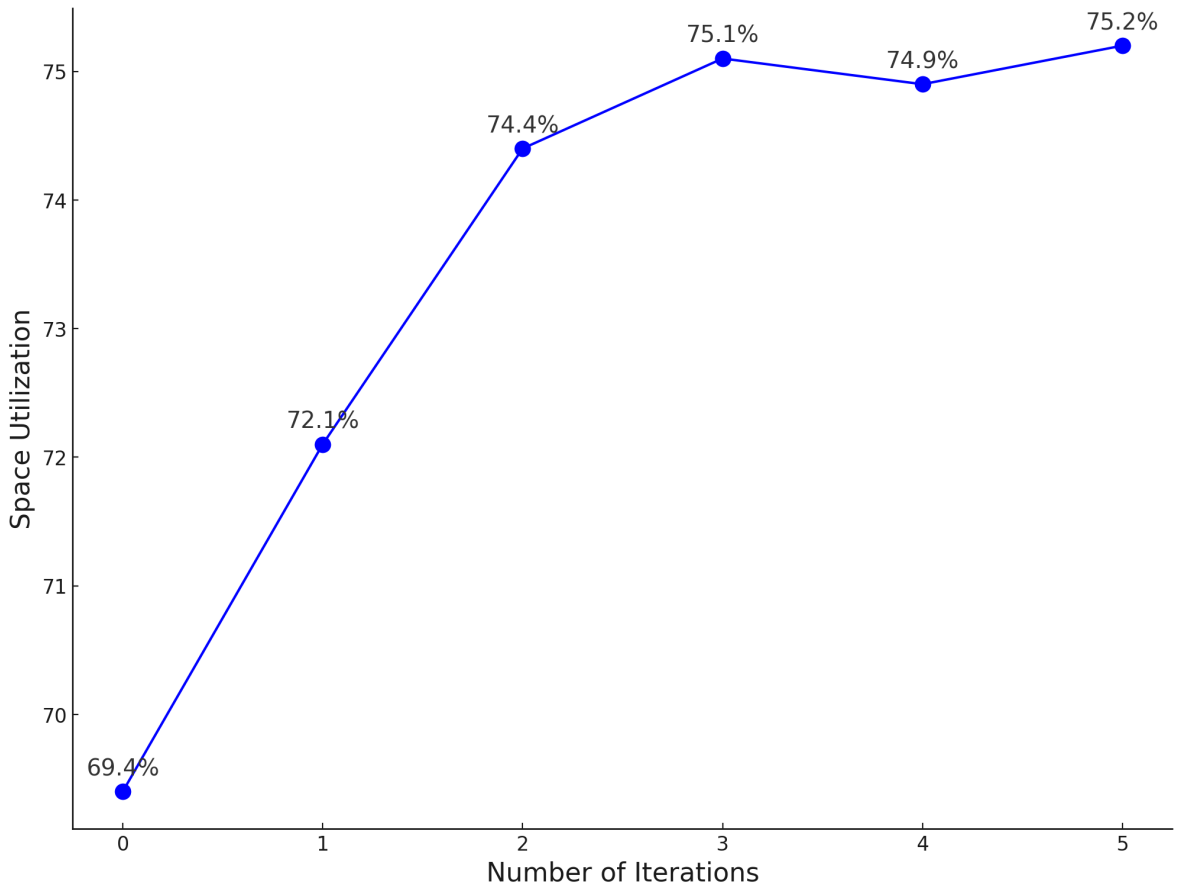


Figure 4.6: The policy performance along with the number of iterations T. The first two iterations ($T = 2, 3$) significantly improves the performance, while the incremental improvement in performance begins to diminish in later iterations ($T = 4, 5$).

in subsequent iterations ($T = 4, 5$). We hypothesize that this phenomenon is attributed to the substantial mitigation of the initial distribution *misalignment* issue within the first two iterations ($T = 2, 3$), which directly contributes to the incremental improvements of the derived policies. Consequently, beyond this point, the performance is no longer impeded by the *misalignment* challenge, rendering additional iterations ineffectual in further elevating the policy's performance.

Real-World Deployment

We further tested our RL-based palletization task planner in a real-world physical system. Due to limitations in space and hardware, it was not feasible to implement our algorithm on an industrial-scale robot handling large boxes. Therefore, we utilized a Franka Panda

robotic arm as a prototype for our physical system setup, depicted in Figure 4.7(a). The robot is equipped with a suction-type gripper for box manipulation and an Intel Realsense RGB-D camera mounted in-hand for box detection and localization on the conveyor. Space constraints restrict the camera's perception to a single box at a time, resulting in an upcoming box count $N = 1$. The boxes, showcased in Figure 4.7(b), come in 5 different sizes and contain various items to simulate the diversity in weights found in real palletization tasks.

Our methodology involved training the task planner in a simulated environment tailored to the physical system's specifications regarding pallet and box dimensions. This trained planner was then directly applied to the real-world setup. The robotic system demonstrated proficiency by successfully stacking 35 boxes on the pallet, achieving a space utilization rate of 72.0% in a single episode. The final pallet configuration, viewed from different angles, is presented in Figure 4.8. The ability to stack 35 boxes while maintaining compactness and stability showcases the effectiveness and robustness of our learned palletization planner policy.

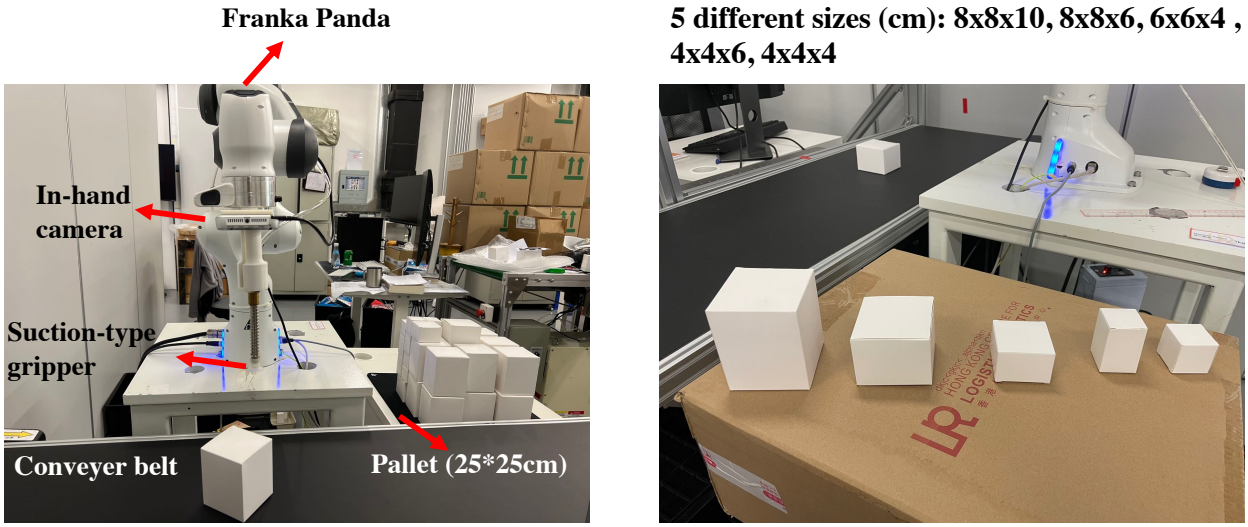


Figure 4.7: (a) Real-world experiment setup. (b) The boxes (of 5 different sizes) used in the real-world experiment.

4.5 Conclusion

This study introduced a novel reinforcement learning-based approach for robotic palletization task planning, emphasizing the significance of iterative action masking learning to manage and prune the action space effectively. Our methodology combines the precision of supervised learning with the adaptive capabilities of reinforcement learning, showcasing

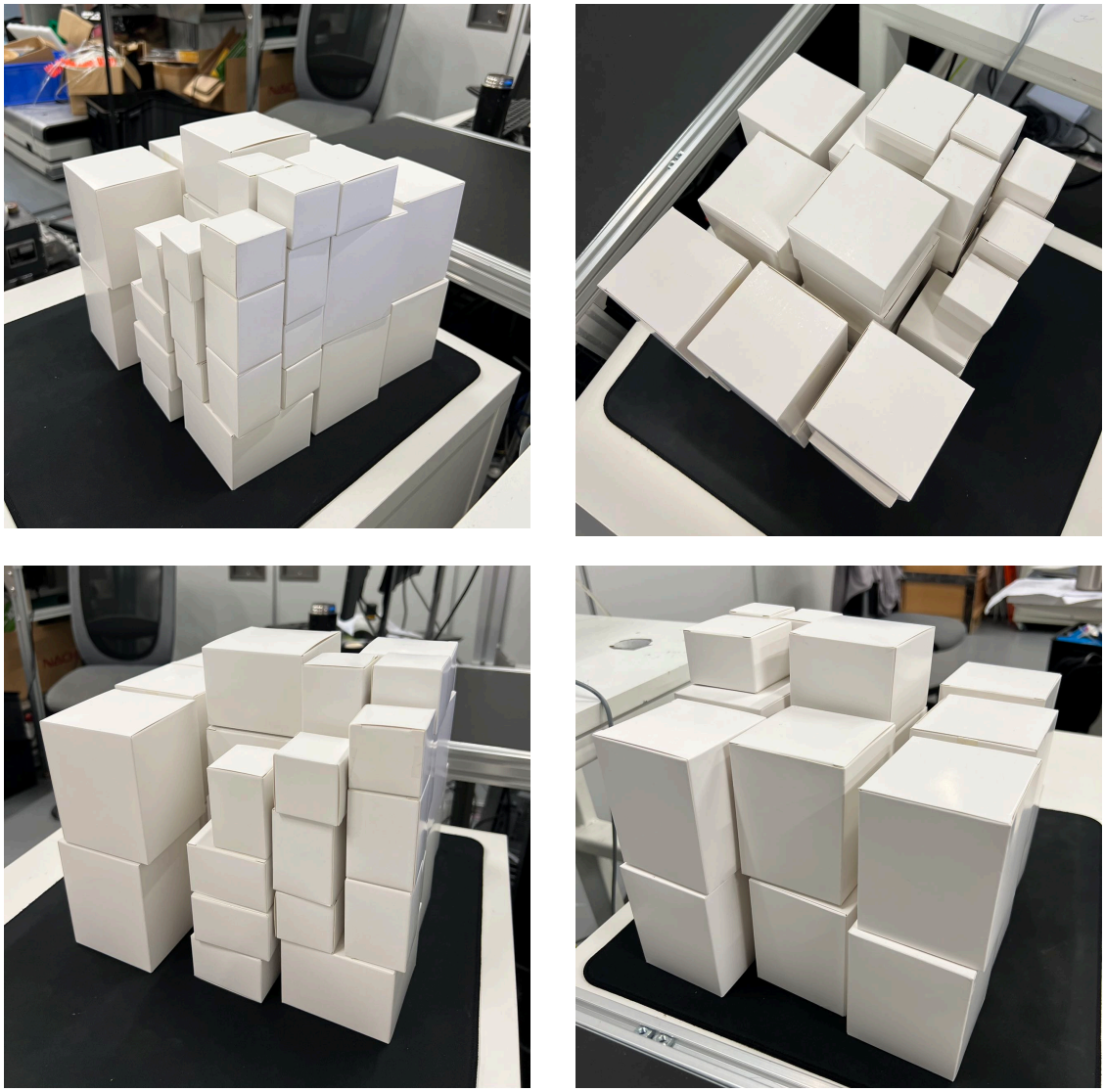


Figure 4.8: Final pallet configuration from different viewpoints. The resulting pallet is compact and stable, demonstrating the effectiveness and robustness of the learned task planner.

substantial improvements in both the efficiency and reliability of task planning for robotic palletization. The experimental validations, both in simulated environments and real-world deployments, have demonstrated the enhanced learning efficiency and operational performance of our proposed method.

A major limitation of our current study is the lack of integration between the task planning phase and the generation of collision-free trajectories for robotic execution. Specifically, our methodology does not account for the ease with which robots can execute the generated

plans without encountering physical obstacles. Merging the task planning process with trajectory planning to produce plans that are not only efficient but also physically feasible for robots to execute without collision presents a promising avenue for future research.

Chapter 5

A Framework to Learn and Adapt Primitive-Based Skills for Industrial Insertion Tasks

5.1 Introduction

Reinforcement learning (RL) has been widely used to acquire robotic manipulation skills in recent years [33, 48, 37]. However, training an agent to perform certain tasks using RL is typically data-hungry and hard to generalize to novel tasks. This data-inefficiency problem limits the adoption of RL on real robot systems, especially in real-world scenarios.

Motion primitives, due to their flexibility and reliability, serve as a popular skill representation in practical applications [45, 3, 113]. A motion primitive is characterized by a desired trajectory generator and an exit condition. It is often realized by hybrid motion/force controllers, such as a Cartesian space impedance controller. Taking moving until contact as an example primitive, the robot compliantly moves towards the surface until the sensed force exceeds a pre-defined threshold; a formal definition of motion primitives is deferred to Section 5.3. Despite the light-weighted representation and wide generalizability of the primitives, their parameters are often task-dependent, and the tuning requires domain expertise and significant trial-and-error efforts [67].

Our work aims to develop a data-efficient framework to learn and generalize insertion skills based on skill primitives. Recently, several research efforts have been devoted to learning primitives for manipulation tasks [64, 114]. However, most works treat primitives as a mid-level controller and use RL to learn the sequence of primitives. While these methods reduce the exploration space leveraging primitives, they still suffer from the inherent drawbacks of RL: data inefficiency and lack of generalizability. To overcome data inefficiency issue, [45] proposed to use a black-box optimizer to obtain the optimal primitive parameters in a pre-defined range by minimizing the task completion time. This approach achieves promising performance on the real robot and is shown to be more efficient than RL. However,

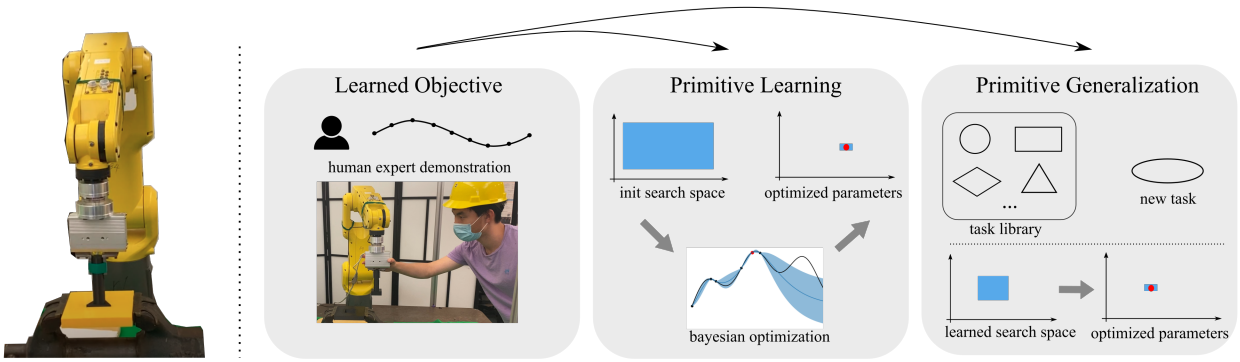


Figure 5.1: An overview of our proposed primitive learning and generalization framework. Our framework learns a dense objective function from human demonstrations. It applies black-box optimization (Bayesian Optimization (BO) in our experiments) to learn the primitive parameters w.r.t. the learned objective function. When generalizing to unseen tasks, we first select similar tasks from the task library based on the introduced similarity metric and then obtain a transferable search space for the new task for BO.

we empirically observe that the parameter range needs to be narrowly set; otherwise, the optimizer spends a long time exploring unpromising parameter choices. In addition, the objective (task completion time), though continuous, is a sparse reward, as it is only triggered when the task succeeds. This prevents the optimizer from extracting information from failed task execution trials, which requires a narrow parameter range to be carefully chosen.

Motivated from the above limitations, we propose a dense objective function that measures the likelihood of the induced execution trajectory sampled from the same distribution as the successful task demonstrations. This model-based objective function provides dense rewards even when a task execution trial fails, encouraging the optimizer to select parameters inducing execution trajectories more similar to the successful demonstrations, thus navigating the parameter space more efficiently. Furthermore, we propose a generalization method to adapt learned insertion skills to novel tasks via a task similarity metric, which alleviates the problem of requiring domain expertise to carefully set parameter ranges for novel tasks. In particular, socket (or hole) geometry information is extracted from the insertion tasks, and the L_1 distance between turning functions [7] of two hole shapes is used as the task similarity metric. An overview of our learning and generalization framework is shown in Figure 5.1. Extensive experiments on 8 different peg-hole and connector-socket insertion tasks are conducted to compare our proposed method with baselines. We experimentally demonstrate that our learning and generalization framework can effectively and efficiently i) learn to acquire insertion skills with about 40 iterations (less than an hour) training on a real robot and ii) generalize the learned insertion skills to unseen tasks with as few as 15 iterations (less than 15 minutes) on average.

The contributions of our work are summarized as follows:

1. We designed an insertion policy composed of learnable motion primitives leveraging impedance control. Rather than using sparse objective functions, we propose to learn a dense objective function from task demonstrations by modeling the execution trajectories.

2. We put forward a transfer learning method that retrieves similar tasks from a task library leveraging task meta information and adapts the previously learned parameters to the novel task with few interactions.

3. We collected 8 insertion tasks with diverse geometry shapes and clearance. Experimental results demonstrate that our method can acquire and adapt the insertion skills at practically acceptable time cost, 1 hour and 15 minutes respectively, while achieving a higher task success rate than baselines.

5.2 Related Work

Learning Robotic Assembly Skills

Robotic assembly tasks, e.g., peg insertion, have been studied for decades and are still one of the most popular and challenging topics in the robotics community [67]. Recently, many works have focused on developing learning algorithms for assembly tasks, among which deep reinforcement learning (RL) methods gained the most attention. For example, [60] proposes a self-supervised learning method to learn a neural representation from sensory input and used the learned representation as the input of deep RL. [92] combines a simple P-controller with a RL policy to reduce the exploration space of RL training. In [120], a reward learning approach is proposed to learn rewards from the high dimensional sensor input, and the reward is used to train a model-free RL policy. Nevertheless, these methods suffer from data inefficiency when facing novel tasks. There are also some works that combine RL and learning from demonstrations (LfD) to address the above issue [111, 18]. However, the impractical amount of robot-environment interactions required by deep RL algorithms and the domain expertise needed to adapt to different insertion tasks limit their adoption in real-world, particularly industrial scenarios.

By contrast, motion primitives, implemented with hybrid motion/force or impedance controllers, are often used for insertion tasks in practice. [45] makes the first attempt to learn the primitive parameters via black-box optimizers, minimizing the task completion time. This algorithm work effectively if primitive parameter ranges are narrowly set; otherwise, due to the sparse reward choice, it takes a large amount of robot-environment interactions for the optimizer to escape the parameter region leading to unsuccessful executions. In comparison, we propose a model-based dense objective function learned from demonstration, which guides the optimizer to explore more promising parameter regions earlier in training.

Generalizing Robotic Manipulation Skills to Unseen Tasks

Transferring the existing policies to novel tasks is extensively studied in the field of robotic manipulation recently. It is especially important for RL based approaches, as most RL algorithms consider to learn a task in simulation first and then transfer the policy to the real world. One common approach is domain randomization, in which a variety of tasks are trained in simulation in order to capture the task distribution [109, 81]. Meta RL has gained significant attraction in recent years [85, 93], where the experience within a family of tasks can be adapted to a new task in that family. However, for motion primitive learning methods, often optimized with gradient-free parameter search methods [45, 113], there haven't been efforts to transfer such prior experience to similar tasks. To the best of our knowledge, our work is the first attempt to encode the prior relevant experience and reduce the parameter exploration space during primitive learning on a novel insertion task.

5.3 Our Method

In this section, we detail our primitive learning and generalization framework for insertion tasks. Compared against traditional RL methods, our framework is more data-efficient and interpretable. We deploy the proposed method on a wide range of peg-hole and connector-socket insertion (peg-in-hole for short) tasks for illustration, and it can be generalized to other manipulation tasks.

Cartesian Impedance Control and the State-Action Space

Impedance control is used to render the robot as a mass-spring-damping system following the dynamics below,

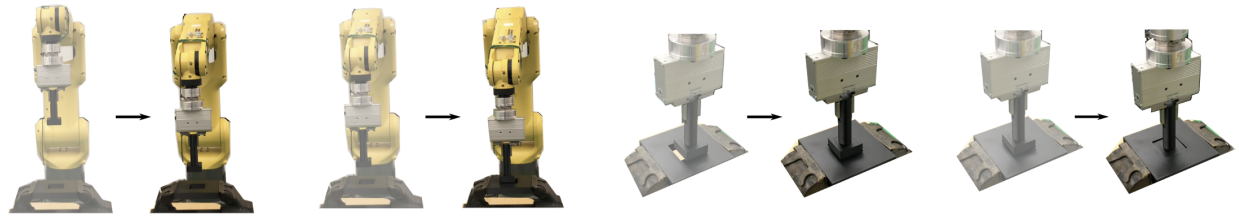
$$\mathbf{M}(\ddot{\mathbf{x}} - \ddot{\mathbf{x}}_d) + \mathbf{D}(\dot{\mathbf{x}} - \dot{\mathbf{x}}_d) + \mathbf{K}(\mathbf{x} - \mathbf{x_d}) = -\mathbf{F}_{ext}, \tag{5.1}$$

where $\mathbf{M}$, $\mathbf{D}$, $\mathbf{K}$ are the desired mass, damping, and stiffness matrices, and $\mathbf{F}_{ext}$ denotes the external wrench. $\ddot{\mathbf{x}}_d$, $\dot{\mathbf{x}}_d$, $\mathbf{x}_d$ are the desired Cartesian acceleration, velocity, and pose of the end-effector, and $\ddot{\mathbf{x}}$, $\dot{\mathbf{x}}$, $\mathbf{x}$ are the current values correspondingly. We assume a small velocity in our tasks and set $\ddot{\mathbf{x}}$, $\dot{\mathbf{x}}$ to 0, thus arriving at this control law,

$$\begin{aligned} \boldsymbol{\tau} &= \mathbf{J}(\mathbf{q})^T\mathbf{F}, \\ \mathbf{F} &= -\mathbf{K}(\mathbf{x} - \mathbf{x_d}) - \mathbf{D}\dot{\mathbf{x}} + \mathbf{g}(\mathbf{q}), \end{aligned} \tag{5.2}$$

where $\boldsymbol{\tau}$ is the control torque, $\mathbf{F}$ is the control wrench, $\mathbf{J}(\mathbf{q})$ is the Jacobian, and $\mathbf{g}(\mathbf{q})$ is the gravity compensation force.

Throughout an insertion task, we would like to design a desired trajectory and a variable impedance to guide the robot movement. In favor of stability and ease of learning, we use a diagonal stiffness matrix $\mathbf{K} = \mathbf{Diag}[K_x, K_y, K_z, K_{roll}, K_{pitch}, K_{yaw}]$, and, for simplicity, the damping matrix $\mathbf{D}$ is scaled such that the system is critically damped.



(a) Free space alignment (b) Move until contact (c) Search (d) Insertion

Figure 5.2: An illustrative figure of the motion primitives designed for peg-in-hole tasks. We show the start and the end states of the robot for each primitive.

In summary, our insertion policy output, $\mathbf{a}_t \in \mathcal{A}$, fed to the impedance controller defined above, is composed of a desired end-effector pose $\mathbf{x}_d$ and the diagonal elements of the stiffness matrix $\mathbf{k} = \{K_x, K_y, K_z, K_{roll}, K_{pitch}, K_{yaw}\}$. The input to the policy, $\mathbf{s}_t \in \mathcal{S}$, consists of end-effector pose $\mathbf{x}_t$ and the sensed wrench $\mathbf{f}_t$, and is extensible to more modalities such as RGB and depth images.

Manipulation Policy with Motion Primitives

In this section, we provide a detailed design on our insertion policy, which entails a state machine with state-dependent motion primitives. Each motion primitive $\mathcal{P}_m$ associated with the m-th state defines a desired trajectory $f_{\boldsymbol{\theta}_m}(\mathbf{x}_{enter}, \mathcal{T})$, an exit condition checker $h_{\boldsymbol{\theta}_m}(\cdot) : \mathcal{S} \to \{1, 0\}$, and a 6-dimensional stiffness vector $\mathbf{k}_m$. $\boldsymbol{\theta}_m$ contains all the learnable parameters in the primitive $\mathcal{P}_m$. $\mathbf{x}_{enter}$ denotes the end-effector pose upon entering the m-th state. $\mathcal{T}$ contains the task information such as the 6 DOF poses of the peg and the hole; often, the hole pose defines the task frame of the motion primitives.

In the following, we formally describe the 4 motion primitives used in the peg-in-hole tasks, as shown in Figure 5.2.

Free space alignment. The end-effector moves to an initial alignment pose.

$$\begin{aligned} f_{\boldsymbol{\theta}_1} &= u(\mathbf{x}_{enter}, \mathbf{x}_{target}) \\ h_{\boldsymbol{\theta}_1}(\mathbf{s}_t) &= \mathbb{I}\left[||\mathbf{x}_t - \mathbf{x}_{target}||_2 < \sigma\right] \\ \mathbf{k}_1 &= \mathbf{k}_{max}. \end{aligned} \tag{5.3}$$

where $\mathbb{I}[\cdot]$ is an indicator function mapping the evaluated condition to $\{0, 1\}$. $u(\cdot, \cdot)$ generates a linearly interpolated motion profile from the first to the second pose provided. The target end-effector pose $\mathbf{x}_{target}$ is extracted from the task information $\mathcal{T}$ as $\mathbf{x}_{target} = T^{base}_{hole} \cdot T^{hole}_{peg} \cdot T^{peg}_{ee}$, where T^{base}_{hole} and T^{peg}_{ee} denote the detected hole pose in robot base frame and the end-effector pose in peg frame. T^{hole}_{peg} is the desired peg pose in hole frame when the peg is above and coarsely aligned with the hole. $\mathbf{k}_{max}$ denotes a 6-dimensional vector composed of the highest

stiffness values along each axis. σ is a pre-defined threshold to determine if the robot arrives at the desired pose. No learnable parameters exist in this primtive. The parameters in this 1-st primitive involves $\boldsymbol{\theta}_1 = \{\emptyset\}$.

Move until contact. The end-effector moves towards the hole until the peg is in contact with the hole top surface.

$$\begin{aligned} f_{\boldsymbol{\theta}_2} &= u(\mathbf{x}_{enter}, \mathbf{x}_{enter} - [0\ 0\ \delta\ 0\ 0\ 0]^T) \\ h_{\boldsymbol{\theta}_2}(\mathbf{s}_t) &= \mathbb{I}[f_{t,z} > \eta] \\ \mathbf{k}_2 &= \mathbf{k}_{max}. \end{aligned} \tag{5.4}$$

δ is the desired displacement along z-axis in the task frame, $f_{t,z}$ is the sensed force along z-axis at time t, and η is the exit force threshold. Therefore the parameters defining this 2-nd primitive consists of $\boldsymbol{\theta}_2 = \{\delta, \eta\}$.

Search. The robot searches for the location of the hole while keeping contact with the hole until the peg and the hole are perfectly aligned. After empirical comparisons with alternatives, including the commonly used spiral search, we choose the Lissajous curve as the searching pattern, which gives the most reliable performance. While searching for the translation alignment, the peg simultaneously rotates along the z-axis to address the yaw orientation error. The roll and pitch orientation errors are expected to be corrected by the robot being compliant to the environment with the learned stiffness.

$$\begin{aligned} f_{\boldsymbol{\theta}_3}(t) &= \mathbf{x}_{enter} + \begin{bmatrix} Asin(2\pi a\frac{n_1}{T}t) \\ Bsin(2\pi b\frac{n_1}{T}t) \\ -\gamma \\ 0 \\ 0 \\ \varphi sin(2\pi\frac{n_2}{T}t) \end{bmatrix} \\ h_{\boldsymbol{\theta}_3}(\mathbf{s}_t) &= \mathbb{I}[x_{enter,z} - x_{t,z} > \zeta] \\ \mathbf{k}_3 &= \mathbf{k}_{search}, \end{aligned} \tag{5.5}$$

where $a = 7, b = 6$ are the Lissajous numbers selected and T is the cycle period in Lissajous search, φ is the maximum tolerated yaw error of the estimated hole pose, set as 6 degree in our experiments. The learnable parameters of this primitive are $\boldsymbol{\theta}_3 = \{A, B, \frac{n_1}{T}, \frac{n_2}{T}, \gamma, \zeta, \mathbf{k}_{search}\}$.

Insertion. The peg is inserted into the hole in a compliant manner.

$$\begin{aligned} f_{\boldsymbol{\theta}_4} &= u(\mathbf{x}_{enter}, \mathbf{x}_{enter} - [0\ 0\ \lambda\ 0\ 0\ 0]^T) \\ h_{\boldsymbol{\theta}_3} &= \mathbb{I}[\text{success condition}] \\ \mathbf{k}_4 &= \mathbf{k}_{insertion}, \end{aligned} \tag{5.6}$$

where the success condition is provided by the task information $\mathcal{T}$, e.g., $||\mathbf{x}_t - \mathbf{x}_{success}||_2 < \epsilon$. The primitive parameters to learn are $\boldsymbol{\theta}_4 = \{\lambda, \mathbf{k}_{insertion}\}$.

Learning Primitive Parameters

In this section, we illustrate how to learn the primitive parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4\}$. The core idea is using a black-box optimizer to optimize a task-relevant objective function $J(\cdot)$. While a similar idea has been explored in [45], the objective function used is simply the measured task execution time. The major drawback of this objective function is that the objective signal is sparse and can only be triggered when the task is successfully executed. This makes the optimizer challenging to find a feasible region initially, especially when the primitive parameter space is large. Motivated by this, we propose a dense objective function that measures the likelihood of the induced execution trajectory being sampled from the distribution of successful task demonstrations $\mathcal{E}_D = \{\boldsymbol{\xi_i}\}(i = 1, 2, ..., M)$. Assuming the trajectroies are Markovian, a trajectory rollout $\boldsymbol{\xi} = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}]$ is modeled as:

$$p(\boldsymbol{\xi}; \boldsymbol{\Theta}) = p(\boldsymbol{x}_0) \prod_{i=1}^{n-1} p(\boldsymbol{x}_i | \boldsymbol{x}_{i-1}). \tag{5.7}$$

In order to learn $p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1})$ from demonstrations, we first use a Guassian Mixture Model (GMM) to model the joint probability as $p(\begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{x}_{i-1} \end{bmatrix}) = \sum_{j=1}^{K} \phi_j \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, where $\sum_{j=1}^{K} \phi_j = 1$, and K is the number of GMM clusters.

We further represent the Gaussian mean $\boldsymbol{\mu}_j$ and variance $\boldsymbol{\Sigma}_j$ as: $\boldsymbol{\mu}_j = \begin{bmatrix} \boldsymbol{\mu}_j^1 \\ \boldsymbol{\mu}_j^2 \end{bmatrix}$, $\boldsymbol{\Sigma}_j = \begin{bmatrix} \boldsymbol{\Sigma}_j^{11} \boldsymbol{\Sigma}_j^{12} \\ \boldsymbol{\Sigma}_j^{21} \boldsymbol{\Sigma}_j^{22} \end{bmatrix}$. We can then derive the conditional probability $p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}) = \sum_{j=1}^{K} \phi_j \mathcal{N}(\overline{\boldsymbol{\mu}}_j, \overline{\boldsymbol{\Sigma}}_j)$, where

$$\begin{aligned} \overline{\boldsymbol{\mu}}_j &= \boldsymbol{\mu}_j^1 + \boldsymbol{\Sigma}_j^{12} (\boldsymbol{\Sigma}_j^{22})^{-1} (\boldsymbol{x}_{i-1} - \boldsymbol{\mu}_j^2) \\ \overline{\boldsymbol{\Sigma}}_j &= \boldsymbol{\Sigma}_j^{11} - \boldsymbol{\Sigma}_j^{12} (\boldsymbol{\Sigma}_j^{22})^{-1} \boldsymbol{\Sigma}_j^{21}. \end{aligned} \tag{5.8}$$

Then, the objective function is designed as $J(\boldsymbol{\xi}) = \log p(\boldsymbol{\xi}; \boldsymbol{\Theta}) + B$, where the first term encourages exploring parameters inducing similar trajectories to the successful demonstration traces, and the second term B denotes a sparse bonus reward if the task succeeds. We use black-box optimizers to solve $\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\text{argmax}} J(\boldsymbol{\Theta})$, and Bayesian Optimization (BO) is selected in our work. Expected Improvement (EI) is used as the acquisition function, and we run BO for N iterations. The learned parameter $\boldsymbol{\Theta}^*$ that achieves maximum $J(\boldsymbol{\Theta})$ during N training iterations is selected as the optimal primitive configuration. Note that BO can be seamlessly replaced by other black-box optimization methods and the optimizer choice is not the focus of this work.

Task Generalization

In this section, we detail our method on how to leverage prior experience when adapting to a novel insertion task, in particular, how to adapt previously learned peg-in-hole policies

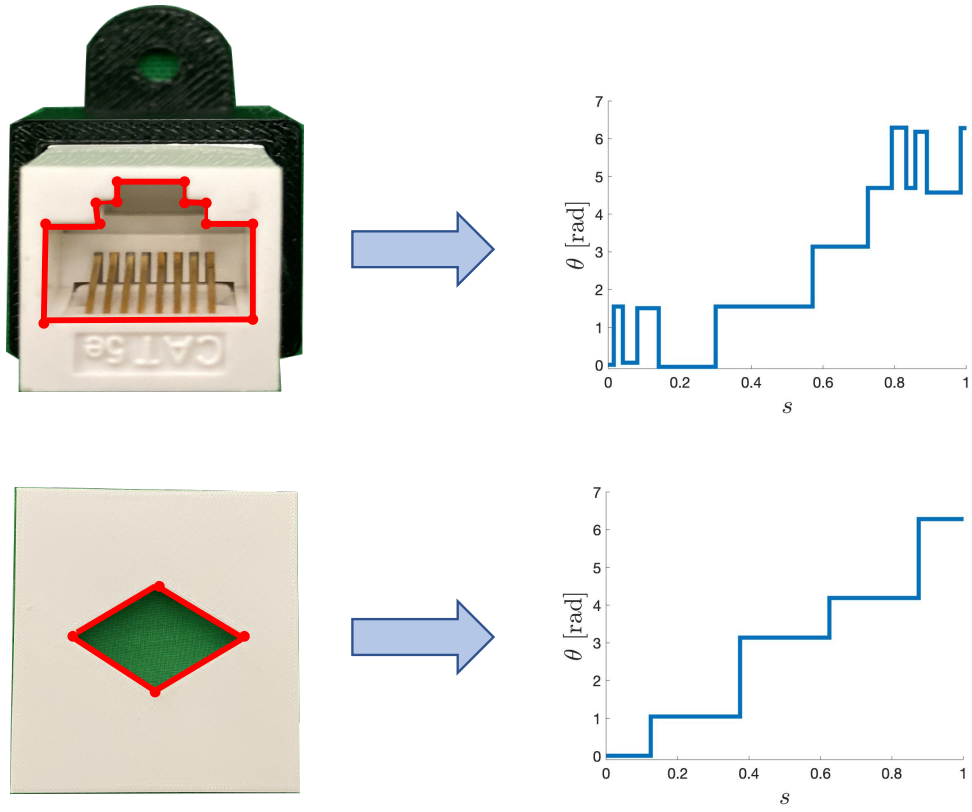


Figure 5.3: Examples of two hole's turning functions. Our task similarity metric is defined as the L_1 distance between the hole's turning functions.

to different tasks with unseen hole shapes. Our adaptation procedure is composed of two core steps: measuring task similarities and transferring similar task policies to the unseen shape.

Measuring task similarity

Given an insertion skill library, i.e., a set of learned peg insertion policies for different shapes, $\mathcal{M} = \{\pi_1(\mathbf{\Theta}_1), \pi_2(\mathbf{\Theta}_2), ..., \pi_n(\mathbf{\Theta}_n)\}$ and an unseen shape, our goal is to first identify which subset of the n tasks are most relevant to the new task. While there is a diverse range of auxiliary task information that can be used to measure task similarity, here, we define the task similarity as the similarity between the hole cross-section contours. This assumption is based on the intuition that similar hole shapes would induce similar policies for insertion. For example, the insertion policies for a square hole and a rectangle hole are likely to be similar, and the optimal policy for a round hole might still work for a hexadecagon hole. The similarity between a shape pair is measured by the L_1 distance between the two shapes' turning functions [7].

Turning functions are a commonly used representation in shape matching, which represents the angle between the counter-clockwise tangent and the x-axis as a function of the travel distance along a normalized polygonal contour. Two example turning functions are shown in Figure 5.3. After obtaining the shape distances of the unseen shape and each shape in the task library, we choose the top L shapes that are closest to the unseen shape as similar shapes. The policies of the similar shapes are then used as input for transfer learning detailed below in Section 5.3.

Adapting to unseen shapes

Given a novel task, our goal is to efficiently adapt the already learned insertion policies of the most similar shapes. We build upon BO with hyperparameter transfer [82]. Unlike many works framing BO transfer learning as a multi-task learning problem, we attempt to learn the **search space** of BO from similar task policies and apply it to learning for the new task.

Specifically, let $\mathcal{T} = \{T_1, T_2, ..., T_t\}$ denote the task set of different hole shapes we selected as described in Section 5.3, and $\mathcal{F} = \{J_1, J_2, ..., J_t\}$ denotes the corresponding objective functions for each task. All the objective functions are initially defined on a common search space $\mathcal{X} \subseteq \mathrm{R}^{|\mathbf{\Theta}|}$, and it's assumed that we already obtained the optimal policies for the t tasks $\{\pi_1(\mathbf{\Theta}_1^\star), \pi_2(\mathbf{\Theta}_2^\star), ..., \pi_t(\mathbf{\Theta}_t^\star)\}(\mathbf{\Theta}_i^\star \in \mathcal{X})$. Given an unseen task T_{t+1}, we aim to learn a new search space $\overline{\mathcal{X}} \subseteq \mathcal{X}$ from the previous tasks to expedite the new task learning process. We define the new search space as $\overline{\mathcal{X}} = \{\mathbf{\Theta} \in \mathrm{R}^{|\mathbf{\Theta}|} | \mathbf{l} \leq \mathbf{\Theta} \leq \mathbf{u}\}$, where $\mathbf{l}, \mathbf{u}$ are the lower and upper bounds. It was proved in [82] that the new search space can be obtained by solving the constrained optimization problem:

$$\min_{\mathbf{l},\mathbf{u}} \frac{1}{2}||\mathbf{u} - \mathbf{l}||_2^2 \text{ such that for } 1 \leq i \leq t, \mathbf{l} \leq \mathbf{\Theta}_i^\star \leq \mathbf{u}. \tag{5.9}$$

The optimization problem has a closed-form solution:

$$\mathbf{l}^\star = \min\{\mathbf{\Theta}_i^\star\}_{i=1}^t, \mathbf{u}^\star = \max\{\mathbf{\Theta}_i^\star\}_{i=1}^t. \tag{5.10}$$

This new search space is then utilized for policy training of this unseen shape task, following the procedure described in Section 5.3.

5.4 Experimental Results

We aim to investigate the effectiveness of our primitive learning and generalization framework by answering two questions: 1) whether the dense objective function proposed in Section 5.3 expedites the primitive learning process and improves policy performance, and 2) whether the generalization algorithm described in Section 5.3 is effective when transferring to an unseen shape. An insertion task library of 8 different peg-hole pairs is constructed, including 6 representative 3D-printed geometry shapes (round, triangle, parallelogram, rectangle,

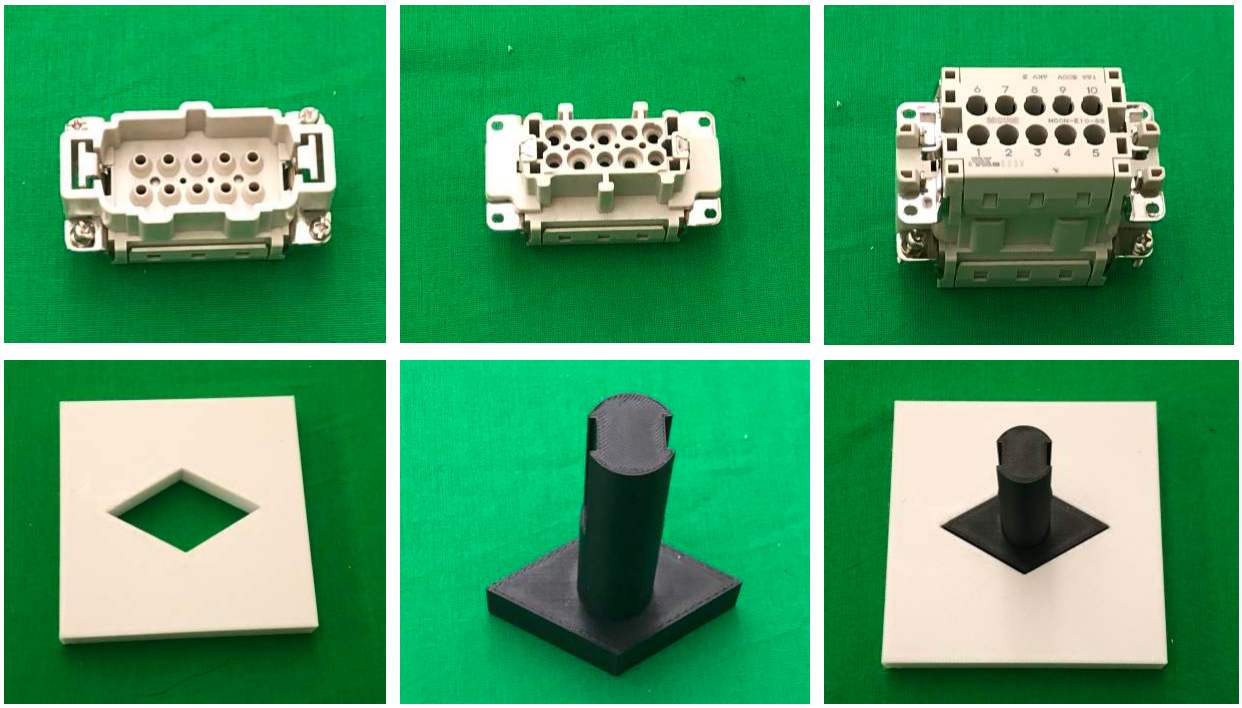


Figure 5.4: Two peg-hole pair instances (waterproof and parallelogram) used in our experiments.

hexadecagon, ellipse) and two common industrial connectors (RJ45, waterproof). Examples of the pegs and holes are shown in Figure 5.4.

Experimental Setup

As shown in Figure 5.1, our hardware setup consists of a 6-DoF FANUC Mate 200iD robot and an ATI Mini45 Force/Torque sensor. The clearances for all 3D printed peg-hole pairs are 1mm; the waterproof and RJ45 are unaltered off-the-shelf connectors. To mimic the pose estimation error during industrial deployments, a uniform perturbation error of +/- 5mm in translation and +/- 6 degree in orientation is applied along each dimension. The controller takes the policy output at 10 Hz and computes the torque command streamed to the robot at 1000Hz. All the learnable parameters of the policy and their initial range are listed in Table 5.1.

Two metrics are used to evaluate the effectiveness and efficiency of the approaches: 1) the number of iterations the robot takes to accomplish the first successful insertion during training (denoted as *number of iterations*), and 2) the success rate of the optimal policy after a fixed number of iterations (denoted as *success rate*).

Table 5.1: Learnable parameters and corresponding range in the motion primitives.

	Move until contact		Search						Insertion	
Parameters	$\delta(m)$	$\eta(N)$	$A(m)$	$B(m)$	$n_1/T_{(s^{-1})}$	$n_2/T_{(s^{-1})}$	$\gamma(m)$	$\zeta(m)$	$\mathbf{k}_{search}(N/m, Nm/rad)$	$\lambda(m)$	$\mathbf{k}_{insertion}(N/m, Nm/rad)$
min	0	1	0	0	0/60	0/60	0	0	$\mathbf{0}^{[6\times1]}$	0	$\mathbf{0}^{[6\times1]}$
max	0.1	10	0.02	0.02	2/10	20/10	0.02	0.02	$[\mathbf{600}^{[3\times1]}, \mathbf{40}^{3\times1}]$	0.05	$[\mathbf{600}^{[3\times1]}, \mathbf{40}^{3\times1}]$

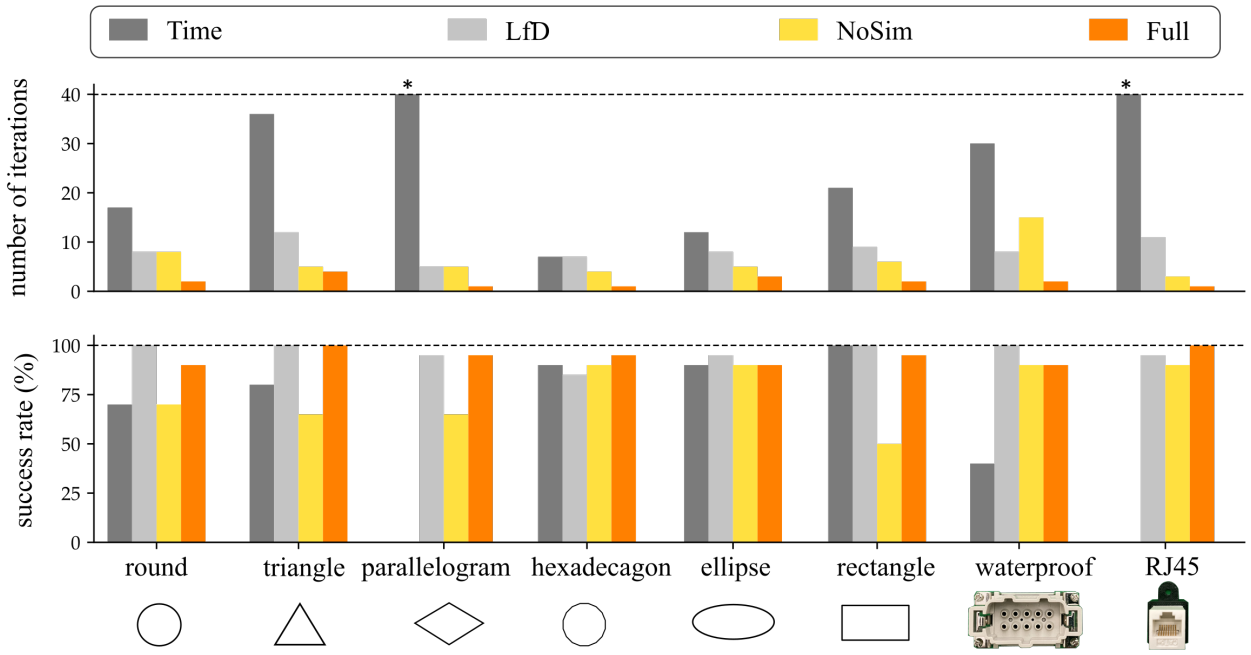


Figure 5.5: Experimental results for primitive learning and generalization. `Time`: primitive learning using task execution time as objective function, `LfD`: primitive learning using learned objective function as described in Section 5.3, `NoSim`: primitive generalization without measuring task similarities, `Full`: full generalization method as described in Section 5.3. $\star$ represents no successful trials is found during the learning process.

Learning Primitive Parameters

In this experiment, we would like to examine if the learned objective function expedites the primitive learning process. Specifically, we applied our learned objective function from demonstration (`LfD`) for primitive parameter optimization as described in Section 5.3, and compare the results against primitive optimization by minimizing the measured task execution time (`Time`) [45]. When learning the objective function, we collected 10 demonstrations for each insertion task. The demonstrations have 43.5 time steps on average across different insertion tasks. The number of GMM clusters is set as $K = 25$.

For each of the 8 peg-hole pairs, we run BO for 40 iterations. Within each iteration, the current policy is executed twice with independently sampled hole poses. The average objective of the two trials is used as the final objective value consumed by the BO step. The optimal policy is selected as the policy at the BO step achieving the optimal objective value, and evaluated by being executed 20 trials with independently sampled hole poses. As shown in Figure 5.5, `LfD` outperforms `Time` on almost all the insertion tasks in both *number of iterations* and *success rate*. In some tasks, e.g., parallelogram and RJ45, `Time` cannot find proper parameters to achieve a single successful trial within 40 iterations, while `LfD`

successfully navigates through the parameter space and accomplishes successful trials for all of the tasks. This validates the learned dense objective function, provides richer information for primitive learning than sparse signals alone like task success and completion time.

To better illustrate the advantage of our learned objective function from demonstration, we show a sample trajectory during training in Figure **??**. The search trace thoroughly covers the the hole area and the corresponding parameters should be a reasonable candidate for the desired primitive configuration. However, the insertion did not succeed due to stochastic noise, and the optimizer discourages exploring the search space around this candidate in `Time`. By contrast, this candidate gets assigned a high objective value in `LfD` since its induced trajectory has a likelihood evaluated by our learned model from demonstration.

Generalizing to Unseen Shapes

We now examine how our generalization method described in Section 5.3 performs when transferring to unseen shapes. We consider the leave-one-out cross-validation setting, i.e., when presented 1 of the 8 tasks as the task of interest, the learning algorithm has access to all interaction data during policy learning on the other 7 tasks. Two sets of experiments are conducted. First, we apply the full method described in Section 5.3 (`Full`) to learn a reduced search space using the $L = 3$ most similar tasks, within which the primitive parameters are optimized.

Compared with `LfD` where parameters are optimized over the full space, `Full` reached a comparable or better success rate, meanwhile achieved the first task completion with a lower number of iterations. Second, we consider learning the search space without measuring the task similarities (`NoSim`), i.e., the new search space is obtained using all the other tasks instead of only the similar ones. As seen in Figure 5.5, `Full` outperforms `NoSim` consistently on all insertion tasks, indicating the significance of finding similar tasks using the task geometry information before learning the new search space. The similarity distances between all insertion tasks are shown in Figure 5.6 to qualitatively demonstrate the effectiveness of using the task geometry information. For example, given an unseen waterproof insertion task, the triangle, rectangle, and parallelogram shapes are selected with the smallest similarity distances. With such knowledge encoded, in `Full`, a narrower search space is obtained compared to `NoSim`, thus leading to better performance by focusing the BO budgets in the parameter subspace with a higher chance of task success.

5.5 Conclusion

We propose a data-efficient framework for learning and generalizing insertion skills with motion primitives. Extensive experiments on 8 different peg-hole and connector-socket insertion tasks are conducted to demonstrate the advantages of our method. The results show that our framework enables a physical robot to learn peg-in-hole manipulation skills and to adapt the learned skills to unseen tasks at low time cost.

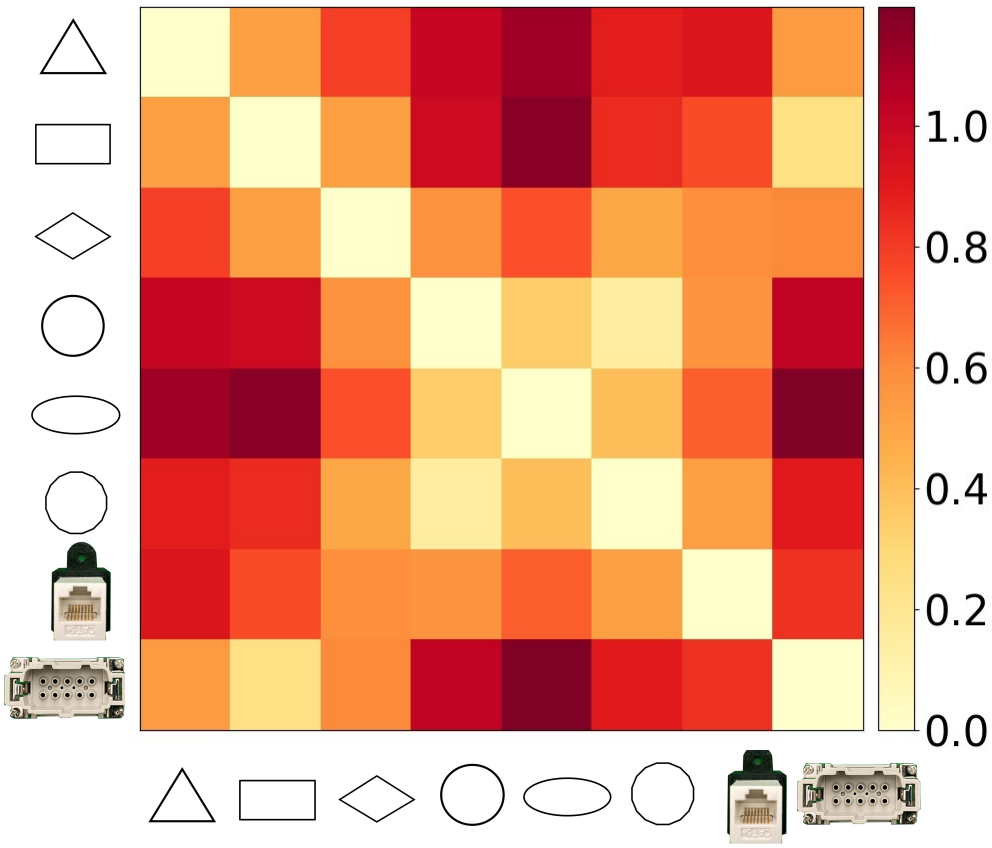


Figure 5.6: The similarity distances between all insertion tasks. Given a new task, we use the obtained similarity distances to select similar tasks from the existing task library and transfer the task knowledge, i.e., primitive parameters, to the new task.

One limitation of the proposed generalization method is that the skill transfer is only beneficial across a narrow task family, e.g., peg-in-hole with different shapes. One future direction is generalizing the skills across more diverse tasks [67, 101]. Besides, our current framework assumes an already designed primitive sequence composing the manipulation policy. Learning the primitive sequence from demonstration or self-supervisedly is another path to explore.

Part III

Data-Efficient Policy Generalization

Chapter 6

Zero-Shot Policy Transfer with Disentangled Task Representation

6.1 Introduction

Policy transfer has been a long-standing challenge in the robotics community. Reasoning over existing task experiences and transferring knowledge to novel combinatorial tasks with familiar elements is vital for human rapid learning and adaptation. For instance, the images in Figure 6.1(a) can be abstracted as combinations of (*digit*, *color*), and humans can effortlessly imagine other unseen combinatorial images. In this work, we aim to enable similar ability in the context of reinforcement learning (RL). Specifically, we consider the generalization scenarios where tasks of interest can be described by a set of degrees of variation (DoVs) and our goal is to achieve policy transfer across unseen compositional tasks, namely, tasks of unseen combinations of existing DoVs. Considering the example depicted in Figure 6.1(b), the robot policy is influenced not only by the *goal* that can be directly measured, but also by other *environment*-related properties of the system, such as robot model, friction, control gain, etc. When facing an unseen compositional task, *e.g.,* (*real*, *front*) in Figure 6.1(b), we aim to combine the knowledge obtained from other related tasks, *e.g.,* (*real*, *left*), (*sim1*, *front*), to directly get the policy for the new task.

The idea of policy transfer across novel task combinations of known DoVs is also explored in [20], where the authors proposed a modular policy architecture to decompose the policy into goal-specific and robot-specific modules and demonstrated zero-shot policy transfer to unseen robot-goal combinations using existing modules. Specifically, the authors decompose the observations into goal-related and robot-related observations, which are fed as input to the corresponding neural network modules to output desired actions. However, this method requires training an individual neural network based module for each possible value of DoV, making it computationally expensive to scale up when certain DoV have many possible values. For example, given 20 different possible goals and 10 different possible robots, this method needs to train 30 different modules in total. Additionally, one implicit assumption

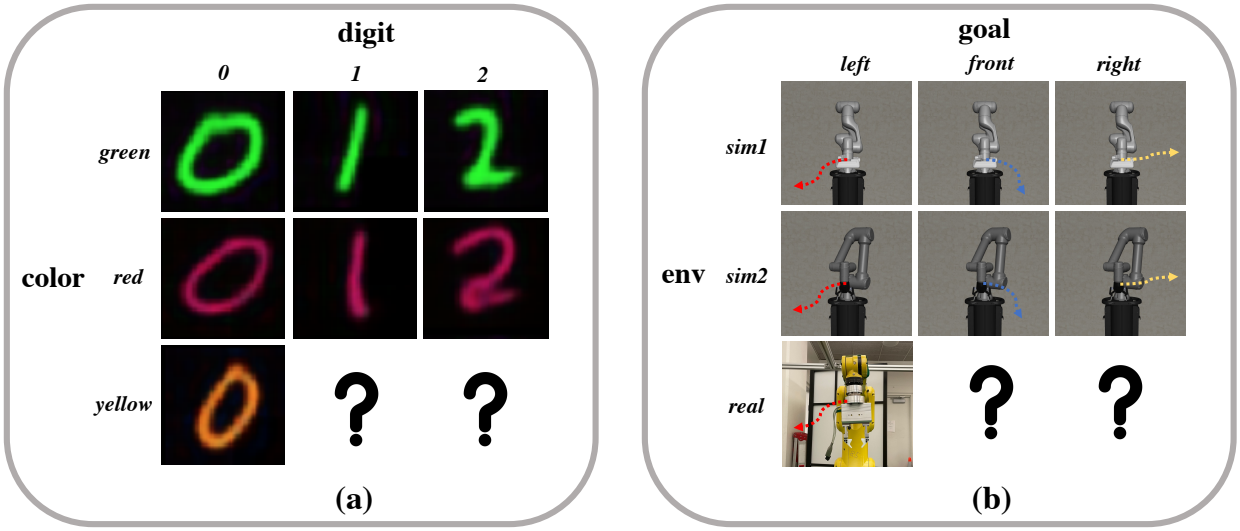


Figure 6.1: Illustration of unseen compositional tasks with existing degrees of variation. (a) While (*yellow, 1*) image is unseen, both *yellow* and *1* are present in other images. (b) While (*real, front*) is a novel task, both *real* and *front* are present in other tasks.

of [20] is that observation alone can capture the differences between tasks with different DoV combinations. However, there exist certain scenarios where the available observations fail to reflect the differences between tasks. For example, when learning an RL agent to push objects of different weights using image as observation, the dynamics variations between tasks are not captured by observation o_t, but instead by observed transitions (o_t, a_t, o_{t+1}, r_t). Solely performing task decomposition *w.r.t.* the observation space limits the potential to be applied in such scenarios.

In contrast, the key insight of our work is to achieve task decomposition from the perspective of task representation instead of policy architecture as in [20]. Specifically, we build upon the framework of context-based meta-RL framework [27, 85, 135, 43, 142] which jointly learns a latent task embedding from observed transitions and a policy conditioned on the inferred task representation. The task decomposition is achieved by disentangling latent task embedding as the concatenation of embeddings for each task DoV. For this purpose, we develop a task disentanglement regularization objective for meta-training (Section 6.3). The achieved disentanglement in the latent task space allows us to infer novel task representations without extra experiences by combining the existing representations of task DoVs in the unseen tasks, thus achieving zero-shot policy generalization (Section 6.3).

To evaluate our approach, we extend three traditional meta-RL tasks to compositional tasks composed of certain DoVs and test the generalization performance of our method on the unseen compositional tasks. Experimental results indicate that the generalized policies obtained from our method in a zero-shot manner outperform other baselines. We also demonstrate that our proposed approach can be seamlessly applied as a zero-shot sim-to-

real generalization method. We evaluate the sim-to-real algorithm on challenging real-world tilted peg-in-hole tasks. Results demonstrate the effectiveness of our approach.

6.2 Related Work

RL Policy Generalization across Similar Tasks

Policy generalization has been a long-standing challenge in robotics community. While RL [105] achieves promising results on robotic manipulation tasks [48, 60], the impedance of deploying RL algorithms lies at the numerous data required during training. To this end, many efforts have been devoted to generalizing the existing policies to similar tasks. A common application of policy generalization is sim-to-real [136], where a policy is learned in simulation and then transferred to the real world. This is usually achieved by domain randomization, which randomizes the tasks with a wide task distribution in simulation and assumes the real-world task is captured by this distribution [109, 81, 83]. Meta-RL approaches [85, 135, 142, 21, 25, 35, 141] offer another perspective of policy generalization by learning to learn from a distribution of tasks, enabling the agent to quickly adapt to novel tasks with limited explorations. Context-based meta-RL methods [85, 135], which view meta-RL as task inference and learn a hidden task variable from collected experience, have demonstrated the ability to adapt meta-policy to real-world manipulation tasks [135, 93]. However, those methods still require collecting online rollout data for task inference and are therefore not zero-shot. Zero-shot policy transfer is particularly useful in circumstances when online interactions are expensive; for example, the task objects are fragile such as the thru-hole components in PCB assembly. In this work, we improve context-based meta-RL methods for policy transfer in a zero-shot manner by considering the task decomposition in the latent task representation.

Task Decomposition in Robotics

Task decomposition is widely investigated in the robotics field to ease the learning process and reuse the knowledge from similar tasks. Most of the research efforts focus on decomposing long-horizon manipulation tasks temporally into several sub-tasks and learning a sub-policy for each sub-task [5, 37, 42, 44, 57, 137]. The learned sub-policies can then be re-arranged in a novel way to obtain the policy for unseen long-horizon tasks. There are also some works that consider task decomposition from other perspectives. For example, [78] performs task decomposition by leveraging the correspondence in the provided analogy and decompose a task as a (action, object) tuple. [130] decomposes the task in the feature space as principle and non-principle features. [19] reasons task decomposition from the object-centric way. The task decomposition considered in our work is most similar to [20], in which tasks are decomposed by a set of degrees of variation. By leveraging knowledge from different tasks with common degrees of variation, we aim to directly obtain policies for novel

combinatorial tasks with existing degrees of variation. However, unlike [20] that performs task decomposition through a modular policy architecture, we achieve task decomposition in the context of meta-RL by disentangling the latent task representation.

6.3 Our Proposed Approach

In this section, we first give a brief review of prior works on context-based meta-RL algorithms in Section 6.3. This provides the background to formalize our problem setup in Section 6.3. Afterwards, we describe our method in two parts: how to enable task decomposition in the latent task representation during training in Section 6.3, and zero-shot policy generalization leveraging the achieved decomposition in Section 6.3.

Preliminary: Context-based Meta-RL

Meta-RL typically assumes a distribution of tasks $p(\mathcal{T})$. Each task $\mathcal{T}$ is modeled as an independent Markov decision process (MDP), $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$, where $\mathcal{S}$ corresponds to the state space, $\mathcal{A}$ corresponds to the action space, $\mathcal{P}$ denotes the transition probability, and R represents the real-valued reward. Meta-RL algorithms learn the task-conditioning policy from training tasks sampled from $p(\mathcal{T})$, and adapt the learned policy to new tasks. Specifically, we focus on one perspective of meta-RL, *i.e.* context-based meta-RL, which views meta-learning as task inference [85, 135]. These methods, with an encoder q_ϕ, map each task $\mathcal{T}_i$ into a task embedding $\mathbf{z}_i$ based on the past experience on this task, *e.g.* context $\mathbf{c}_i = \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t, r_t)\}_{t=1\ldots N}$ [85]. The policy π_θ is conditioned on the task embedding so that it can be adapted to different tasks, *i.e.*,

$$\mathbf{a} \sim \pi_\theta(\cdot|\mathbf{s}, \mathbf{z}_i), \quad \text{where } \mathbf{z}_i \sim q_\phi(\cdot|\mathbf{c}_i). \tag{6.1}$$

While our method is agnostic to the specific context-based meta-RL algorithms, in our implementation, we build on the framework of probabilistic embeddings for actor-critic RL (PEARL) [85], which optimizes the objective,

$$\mathbb{E}_{\mathcal{T}_i}\left[\mathbb{E}_{\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|\mathbf{c}_i)}\left[R(\mathcal{T}_i, \mathbf{z}_i) + \beta D_{\mathrm{KL}}\left(q_\phi\left(\mathbf{z}_i \mid \mathbf{c}_i\right) \| p(\mathbf{z}_i)\right)\right]\right], \tag{6.2}$$

where $R(\mathcal{T}_i, \mathbf{z}_i)$ denotes the addition of actor and critic objectives as defined in [85]. We refer the readers [85] for more technical details.

Problem Formulation

Our goal is to achieve zero-shot policy transfer across unseen compositional tasks. We define compositional task as the task equipped with a predefined set of degrees of variation (DoVs) [20]. Loosely speaking, a task DoV describes one varied aspect of the task. For example, in Fig. 6.1(b), there are two DoVs describing the tasks, *i.e. environment* and *goal*,

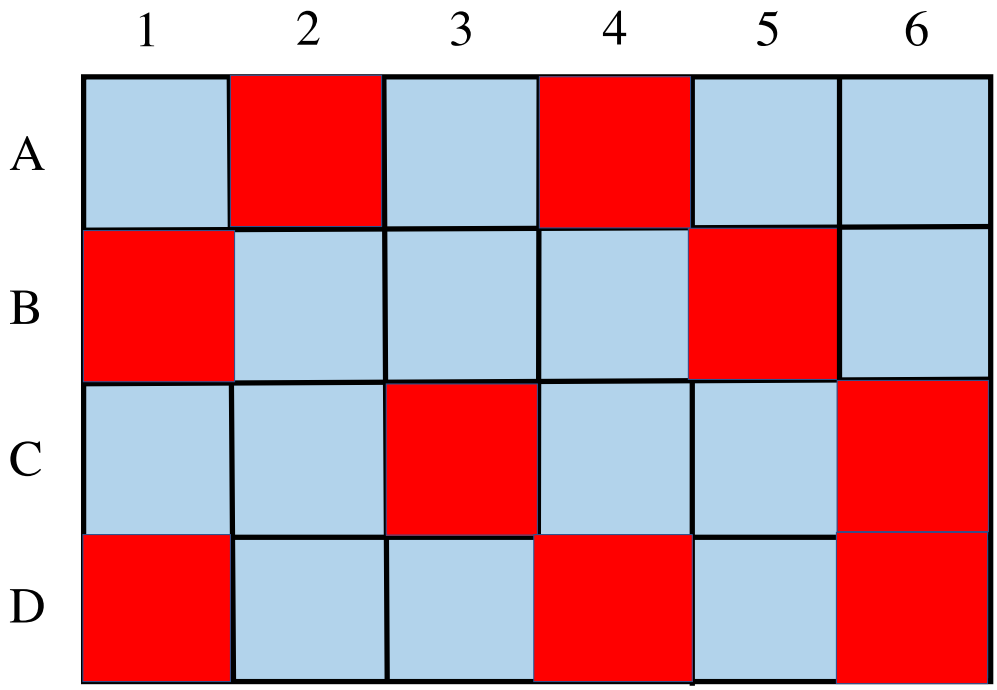


Figure 6.2: Illustration of the targeted policy generalization setting in our work. We assume the tasks of interest can be described as combinations of degrees of variation (DoV). Given experience from training task set $\mathbf{T}_{train}$ (blue), we aim to generalize across tasks (in red) whose DoV combinations are unseen but each DoV label is present in $\mathbf{T}_{train}$.

and the DoV *goal* has three unique values, *i.e. left* (red trajectory), *front* (blue trajectory), and *right* (yellow trajectory). Formally, considering the task distribution $p(\mathcal{T})$, each $\mathcal{T}_i$ is equipped with a unique combination $\mathcal{C}_i$ of M task DoVs as $\mathcal{C}_i = (y_1^i, \ldots, y_M^i)$, where y_j^i is the label of the j-th DoV of task $\mathcal{T}_i$. The M DoVs are problem-dependent and identical for all the tasks $\mathcal{T}_i$ in the task set $\mathbf{T}$, while each DoV can have different labels and the task DoV combination $\mathcal{C}_i$ for each task $\mathcal{T}_i$ is unique. We assume the DoV labels, $\mathcal{C}_i$, of each task $\mathcal{T}_i$ in $\mathbf{T}$ are given. Our problem setting aims at generalization across tasks whose DoV combinations are unseen but the DoV labels are present in training tasks $\mathbf{T}_{train}$, as illustrated in Figure 6.2.

Task Decomposition via Latent Task Embedding Disentanglement

In this section, we describe how we achieve task decomposition *w.r.t.* task DoV combinations in the context of meta-reinforcement learning (Meta-RL). Figure 6.3 illustrates our framework. The key insight is to achieve task decomposition by enforcing the disentanglement of the latent task representation in [85] during meta-training. While traditional meta-RL algorithms learn the task embedding $\mathbf{z}_i$ with a single encoder q_ϕ as Eq. 6.1, we replace it with M separate DoV encoders $q_{\phi_1}, q_{\phi_2}, \ldots, q_{\phi_M}$, where $q_{\phi_j}(\mathbf{z}_i^{y_j}|\mathbf{c}_i)$ learns the DoV

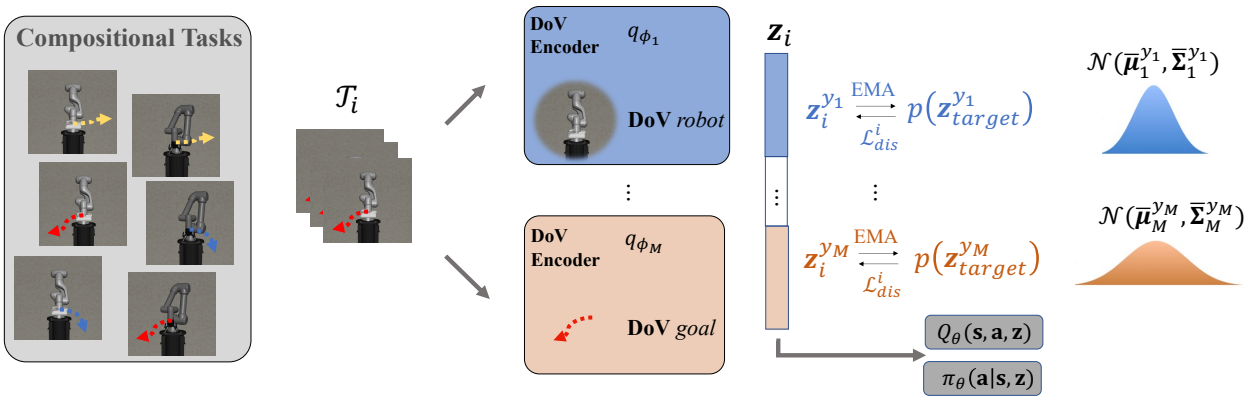


Figure 6.3: We achieve task decomposition via disentanglement in the latent task space. The disentanglement is achieved by encoding different task DoVs with individual DoV encoders and enforcing identical task DoVs across different tasks have similar embeddings.

embedding for the j-th DoV of task $\mathcal{T}_i$. The task embedding is obtained by concatenating all M DoV embeddings as $\mathbf{z}_i = [\mathbf{z}_i^{y_1}, \ldots, \mathbf{z}_i^{y_M}]$, and the policy π_θ is conditioned on the disentangled task embedding:

$$\begin{aligned} \mathbf{a} \sim \pi_\theta(\cdot|\mathbf{s}, \mathbf{z}_i) \quad & \text{where } \mathbf{z}_i = [\mathbf{z}_i^{y_1}, \ldots, \mathbf{z}_i^{y_M}] \\ & \mathbf{z}_i^{y_j} \sim q_{\phi_j}(\cdot|\mathbf{c}_i), j = 1, 2, \ldots, M. \end{aligned} \tag{6.3}$$

For different tasks $\mathcal{T}_u, \mathcal{T}_v$ sharing the same label in the j-th DoV, *i.e.*, $y_j^u = y_j^v = y_j$, their corresponding DoV embedding should be regularized to follow an identical probability distribution. One naive way to implement this principle is to bring close the corresponding DoV embedding of different tasks with shared DoV labels within the sampled data batch. However, we found empirically that this strategy is easily prone to the affection of noise, making the training process unstable. To this end, we introduce a target distribution $p(\mathbf{z}_{target}^{y_j})$ for the j-th task DoV with y_j label, and minimize the Kullback–Leibler (KL) divergence between $q_{\phi_j}(\mathbf{z}_i^{y_j}|\mathbf{c}_i)$ and $p(\mathbf{z}_{target}^{y_j})$. Inspired by [106, 32], Exponential Moving Average (EMA) is incorporated into our training scheme to update this target distribution during training, achieving robustness against outliers during training. For each DoV label y_j of the j-th DoV, we maintain an individual target distribution $p(\mathbf{z}_{target}^{y_j})$. Following PEARL [85], our DoV embedding is modeled as Gaussian distribution. So we maintain the EMA for both the mean $\bar{\boldsymbol{\mu}}_j^{y_j}$ and variance $\bar{\boldsymbol{\Sigma}}_j^{y_j}$ of $p(\mathbf{z}_{target}^{y_j})$. This task disentanglement regularization is written as follows for each task $\mathcal{T}_i$, where λ is a hyper-parameter to balance different loss items.

$$\begin{aligned} \mathcal{L}_{dis}^i &= \lambda \sum_{j=1}^{M} D_{KL}(q_{\phi_j}(\mathbf{z}_i^{y_j}|\mathbf{c}_i)||p(\mathbf{z}_{target}^{y_j})) \\ p(\mathbf{z}_{target}^{y_j}) &= \mathcal{N}(\bar{\boldsymbol{\mu}}_j^{y_j}, \bar{\boldsymbol{\Sigma}}_j^{y_j}) \end{aligned} \tag{6.4}$$

where $q_{\phi_j}(\mathbf{z}_i^{y_j}|\mathbf{c}_i) = \mathcal{N}(\boldsymbol{\mu}_i^{y_j}, \boldsymbol{\Sigma}_i^{y_j})$ is the current prediction for task $\mathcal{T}_i$. $\bar{\boldsymbol{\mu}}_j^{y_j}$ and $\bar{\Sigma}_j^{y_j}$ are updated in each training step t with EMA:

$$\begin{aligned}\bar{\boldsymbol{\mu}}_j^{y_j,t} &\leftarrow \tau \cdot \bar{\boldsymbol{\mu}}_j^{y_j,t-1} + (1-\tau)\cdot \boldsymbol{\mu}_i^{y_j} \\ \bar{\boldsymbol{\Sigma}}_j^{y_j,t} &\leftarrow \tau \cdot \bar{\boldsymbol{\Sigma}}_j^{y_j,t-1} + (1-\tau)\cdot \boldsymbol{\Sigma}_i^{y_j},\end{aligned} \tag{6.5}$$

where $\tau = 0.99$ in our implementation. The task disentanglement regularization is applied during meta-training, as summarized in Algorithm 3, where red lines highlight the differences compared with PEARL.

Algorithm 3 Our meta-training procedure. Differences with PEARL are highlighted in red.

Require: Batches of compositional training tasks $\{\mathcal{T}_i\}_{i=1...T}$ with corresponding task DoV combinations $\{\mathcal{C}_i\}_{i=1...T}$, learning rate α_1

Init. replay buffer $\mathcal{B}_i$ for each training task
while not done **do**
  for each task $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}_i = \{\}$
    for $k = 1, ..., K$ **do**
      Sample $\mathbf{z}_i^{y_j} \sim q_{\phi_j}(\cdot|\mathbf{c}_i), j = 1, \ldots, M$
      Concatenate task embedding $\mathbf{z}_i = [\mathbf{z}_i^{y_1}, \ldots, \mathbf{z}_i^{y_M}]$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z}_i)$ and add to $\mathcal{B}_i$ and update $\mathbf{c}_i = \{(\mathbf{s}_l, \mathbf{a}_l, \mathbf{s}'_l, \mathbf{r}_l)\}_{l:1...N} \sim \mathcal{B}_i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context batch $\mathbf{c}_i \sim \mathcal{S}_c(\mathcal{B}_i)$ and RL batch $b_i \sim \mathcal{B}_i$
      Sample $\mathbf{z}_i^{y_j} \sim q_{\phi_j}(\cdot|\mathbf{c}_i), j = 1, \ldots, M$
      Concatenate task embedding $\mathbf{z}_i = [\mathbf{z}_i^{y_1}, \ldots, \mathbf{z}_i^{y_M}]$
      Compute $\mathcal{L}^i_{actor}, \mathcal{L}^i_{\text{critic}}, \mathcal{L}^i_{KL}$ same as PEARL
      $\mathcal{L}^i_{dis} = \lambda \sum_{j=1}^{M} D_{KL}(q_\phi(\mathbf{z}_i^{y_j}|\mathbf{c}_i)||p(\mathbf{z}_{target}^{y_j}))$, where $p(\mathbf{z}_{target}^{y_j}) = \mathcal{N}(\bar{\boldsymbol{\mu}}_j^{y_j}, \bar{\boldsymbol{\Sigma}}_i^{y_j})$
      $\bar{\boldsymbol{\mu}}_j^{y_j,t} \leftarrow \tau \cdot \bar{\boldsymbol{\mu}}_j^{y_j,t-1} + (1-\tau)\cdot \boldsymbol{\mu}_i^{y_j}$
      $\bar{\boldsymbol{\Sigma}}_j^{y_j,t} \leftarrow \tau \cdot \bar{\boldsymbol{\Sigma}}_j^{y_j,t-1} + (1-\tau)\cdot \boldsymbol{\Sigma}_i^{y_j}$
    end for
    $\phi_j \leftarrow \phi_j - \alpha_1 \nabla_{\phi_j} \sum_i \left(\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL} + \mathcal{L}^i_{dis}\right), j = 1, \ldots, M$
    Update θ_π, θ_Q same as PEARL
  end for
end while

Zero-Shot Policy Generalization through Task Decomposition

When generalizing policies to novel tasks, traditional meta-RL algorithms usually requires adequate explorations on the new tasks. However, exploration can be costly in some scenarios, *e.g.*, the task objects are fragile such as the thru-hole components in PCB assembly. In contrast, we can achieve zero-shot policy generalization leveraging the task decomposition obtained from Section 6.3. We identify two scenarios of zero-shot policy generalization as follows.

S1 Test tasks are defined by combinations of DoV labels already seen in training tasks $\mathbf{T}_{train}$. In this case, we directly concatenate the EMA $\{\mathbf{z}_{target}^{y_j}\}_{j=1}^{M}$ of the corresponding DoV embeddings obtained from training as the test task representation. The meta-policy conditioned on the concatenated task representation is used as the generalized policy for the new task.

S2 Test tasks have DoV label that does not exist in the training tasks $\mathbf{T}_{train}$, denoted as, $y_j = y^*$. Similar to PEARL meta-test, our method first collects context in one single task $\mathcal{T}_{test}$ with the unseen DoV label and acquires the latent task representation $\mathbf{z}_{test}$. Empirically, this process takes around 10 episodes to obtain a descent $\mathbf{z}_{test}$. Then the embedding of the DoV label, $\mathbf{z}_{test}^{y_j}$, is inferred as the corresponding dimensions extracted from $\mathbf{z}_{test}$. After obtaining the embedding of this unseen DoV label, we repeat the steps in **S1** to achieve generalization across other test tasks with $y_j = y^*$. While our method does require collecting experiences from one single task in test task set, the generalization across other test tasks is still **zero-shot**. As demonstrated in Section 6.4, this can be readily used as a sim-to-real generalization method. For instance, we meta-train a policy to complete 10 similar manipulation tasks IN different simulation environments. When generalizing to the real world, our method only requires interacting with the real environment on one single task, and is able to generalize to the other 9 tasks in a zero-shot fashion.

6.4 Experimental Results

In the experiments, we aim to investigate the following questions. 1) Does our method described in Section 6.3 effectively disentangle the learned task representation? 2) How does the disentanglement affect the model training for compositional tasks? 3) Can our method achieve zero-shot policy generalization across novel tasks through task decomposition?

Simulation Experiments

Compositional Task Setup. To study the targeted policy transfer problem in this work, we build three set of simulated compositional tasks shown in Figure 6.4. Each set of compositional tasks has two task degrees of variation (DoVs), *i.e., goal* and *physics*. For (a)

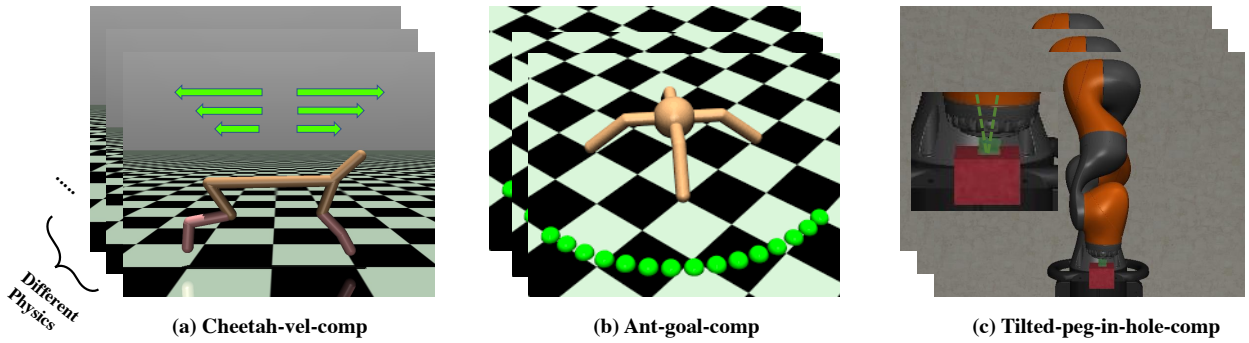
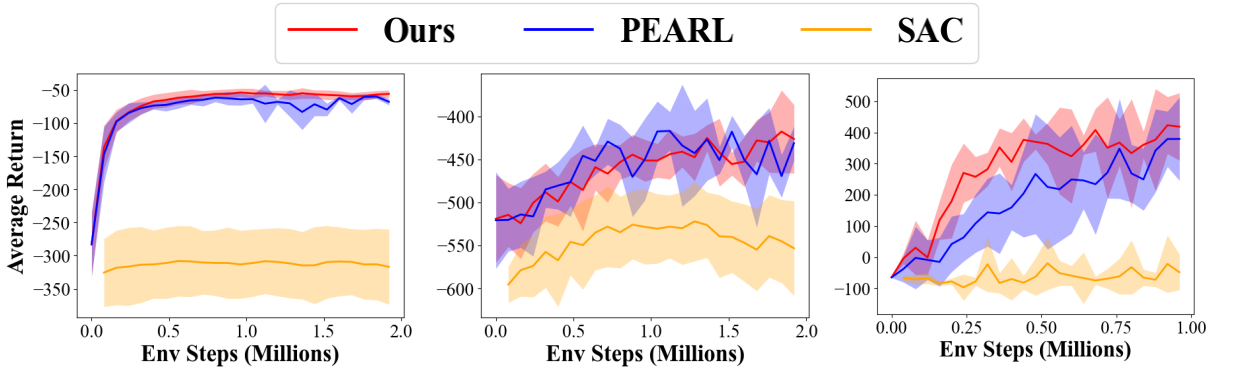


Figure 6.4: Three set of compositional tasks for algorithm evaluation. (a) Cheetah-vel-comp with varying *goal velocities* and *physics*. (b) Ant-goal-comp with varying *goal positions* and *physics*. (c) Tilted-peg-in-hole-comp with varying *hole tilting directions* and *physics*.

Cheetah-vel-comp and (b) Ant-goal-comp, we extend the traditional Cheetah-vel and Ant-goal from [25] by randomizing some physical parameters, *i.e.*, *body mass, body inertia, damping* and *friction*. Specifically, we randomly sample 20 different sets of physical parameters and 20 different goal velocities for Cheetah-vel-comp, leading to 400 compositional tasks in total. Thus each task has a unique *(physics, goal)* DoV combination. Similarly for Ant-goal-comp, 15 different groups of physical parameters and 15 different goal 2D locations are randomly sampled. In (c) Tilted-peg-in-hole-comp, we create challenging tilted peg-in-hole tasks based on robosuite [139]. The tasks are performed using a 7-Degree-of-Freedom (DoF) KUKA LBR IIWA robot with operation space control [54]. We adopt a square peg-hole task with $0.5mm$ clearance and tilted the hole surface plane 5° towards different directions. Tilted peg-in-hole tasks are prone to jamming issues during execution, thus posing more challenges to robust policy learning. The observation space is 6-dimensional end-effector pose and the actions are the desired end-effector poses. The compositional tasks are composed of 4 different hole tilting directions towards $\pm x$ and $\pm y$ axis separately as well as 20 different sets of physical parameters (*i.e. friction between peg and hole, damping* and *controller step size*).

Training Tasks Selection. The training task set $\mathbf{T}_{train}$ is a subset of the whole compositional task set $\mathbf{T}$. When selecting $\mathbf{T}_{train}$, we randomly select tasks in $\mathbf{T}$ with probability $\alpha \in (0, 1)$, while ensuring each task DoV label in $\mathbf{T}$ appear at least once in $\mathbf{T}_{train}$. For example, in Figure 6.2, the 15 tasks in blue are selected for training ($\alpha = 0.625$). In the implementation, we use $\alpha = 0.5$, which is discussed in Section 6.4. Other tasks in $\mathbf{T}$ are set aside as novel test tasks, *i.e.* $\mathbf{T}_{test} = \mathbf{T} \backslash \mathbf{T}_{train}$.

Baselines. We compare our method against two baselines, PEARL [85] and SAC [38]. PEARL directly learns the task embedding from training tasks without exploiting the inherent task combinatorial structure. During generalization, *i.e.*, meta-test, PEARL requires extra explorations on each test task, so it is not a zero-shot approach. The SAC agent is



(a) Cheetah-vel-comp (b) Ant-goal-comp (c) Tilted-peg-in-hole-comp

Figure 6.5: Average return on test tasks during the training period. Our method and PEARL apply the same meta-test strategy, where the first two trajectories are aggregated into the context for task inference.

trained with all the training tasks together where a training task is randomly sampled in each episode. This can be viewed as a variant of domain randomization. The learned SAC policy is directly applied to test tasks. It is worth noting that [20] can not be directly applied as a baseline because solely decomposing tasks *w.r.t.* observations in the three tasks fail to capture the dynamics differences, as discussed in Section 6.1.

Effect on Training. Figure 6.5 shows the performance of the learned policies on $\mathbf{T}_{test}$ as training proceeds, across 5 random seeds. Our method achieves comparable meta-test results compared to PEARL, indicating that the enforced regularization in latent task representation (Section 6.3) does not deteriorate the learning process of meta-RL. It is worth noting that the aim of our proposed method is to improve the performance of zero-shot policy transfer across test tasks, instead of meta-test where the experience of the test tasks is given.

Zero-Shot Evaluation. The evaluation is conducted for the two scenarios **S1** and **S2** (Section 6.3) respectively. In **S1**, $\mathbf{T}_{test} = \mathbf{T} \backslash \mathbf{T}_{train}$ is used as the test tasks. While in **S2**, we first sample task attributes that do not exist in $\mathbf{T}$ and the test tasks are set as the combinations of the unseen attributes and other seen attributes, e.g., (*unseen* physics, *seen* goal), (*unseen* goal, *seen* physics). For each unseen attribute, we allow the agent to collect transitions with 8 episodes on one single task with the unknown attribute to infer the unknown attribute representation, as described in Section 6.3. The experimental results across 5 random seeds are shown in Figure 6.6. We compare our zero-shot performance (*Zero-shot (ours)*) with multiple baselines: 1) *SAC:* A single model is trained on all training tasks with SAC [37]. 2) *Prior (PEARL):* The same $\mathcal{N}(0, 1)$ prior distribution is applied to all the task embeddings $\mathbf{z}_i$ of PEARL model [85]. 3) *Meta-test (PEARL):* We apply the

same meta-test for PEARL as in [85]. To ensure enough contexts on the test task, the first 8 trajectories are aggregated into the context. 4) *Meta-test (ours):* The same meta-test with PEARL is applied to the model trained with our method. It is worth mentioning that only 1) and 2) can achieve zero-shot generalization on novel tasks, while 3) and 4) require extra exploration. As shown in Figure 6.6, our zero-shot generalization significantly outperforms the two zero-shot baselines, *SAC* and *Prior (PEARL)*. Interestingly, it is also superior to the two meta-test baselines, although our generalization does not explore on the test task. We hypothesize that the reason is our method benefits from the EMA embedding $\mathbf{z}_{target}^{y_j}$ used to compose the embedding $\mathbf{z}_u$ of the novel task, serving as a temporal ensemble of information accumulated during training. In contrast, purely collecting context from a single test task may suffer from its inherent bias and easily get trapped in the local optimum.

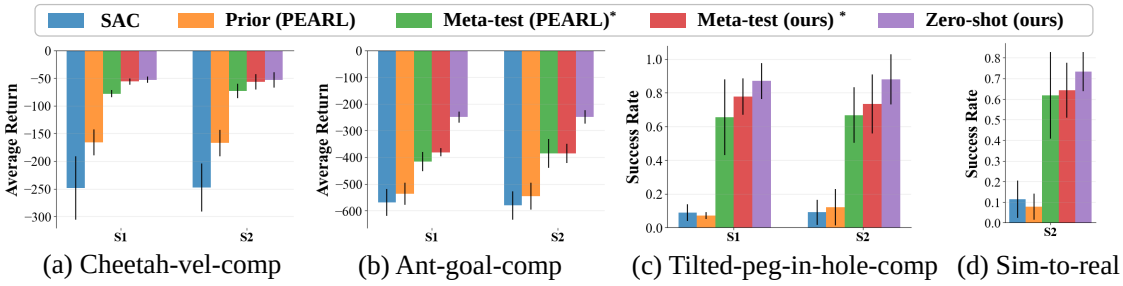


Figure 6.6: Evaluation results of policy generalization on test tasks (* not a zero-shot method). Our zero-shot generalization significantly outperforms the two toher zero-shot baselines, i.e., *SAC* and *Prior (PEARL)*, indicating the effectiveness of our proposed method.

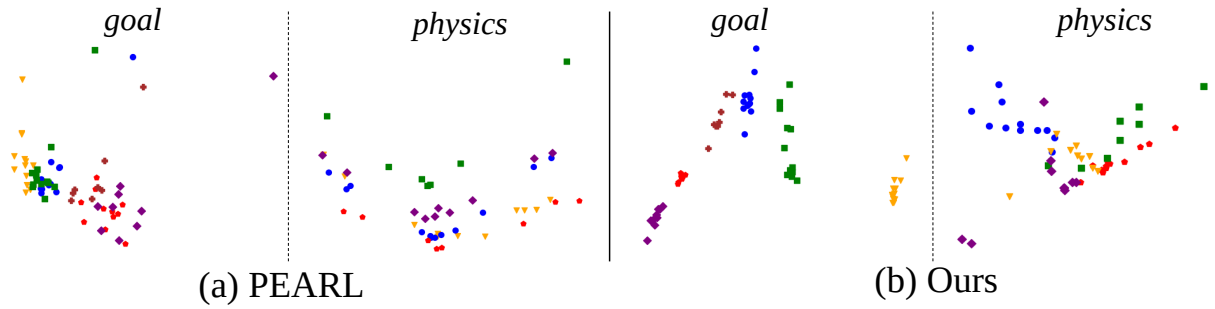


Figure 6.7: Visualization of latent task embedding of our method compared to PEARL. The embeddings of tasks with identical *goal* or *physics* labels are in the same color.

Visualization of Latent Embedding. To investigate if the proposed method in Section 6.3 effectively disentangle the encodings in latent space, Figure 6.7 visualizes sampled

test task representations output by our method and PEARL on Cheetah-vel-comp. The embeddings of test tasks with same *goal* or *physics* are in the same color. All the embeddings are reduced to 2-d with PCA [2] for visualization. It is observed that our method can distinguish different labels of each attribute better than PEARL. Different goals are distinctly separable by our method. For different physics, despite their effect being indirect, similar physics parameters still tend to induce closer embeddings.

Real Experiments

As described in Section 6.3, our method can be applied as a sim-to-real algorithm, by treating real-world tasks as tasks with unseen *physics* attribute. We evaluate the method on a real-world 6-DoF FANUC Mate 200iD robot with the similar setup as Tilted-peg-in-hole-comp in Section 6.4, as shown in Figure 6.8. The evaluation of real world is identical to the evaluation of **S2** in simulation (Section 6.4), where, after meta-training in simulation, the meta-policy collects context for one held-out task to infer the real-world physics embedding. The obtained physics embedding is combined with other goal embeddings learned from simulation to generalize across the other three tasks in the real world. Each task is used as the held-out task once for thorough evaluation. When evaluating the generalized policy, the evaluated task is executed for 30 trials with 3 different random seeds (10 trials for each random seed). The results are reported in Figure 6.6(d). Our generalized policy achieves a success rate of 73.3% over different unseen tasks.

Ablation Studies

In this section, we ablate our methods from two perspectives: the sparsity ratio (α) of training tasks (Section. 6.4) and the effect of Exponential Moving Average (EMA) (Section. 6.3).

Sparsity of Training Tasks. In our previous experiments, we sample 50% of compositional tasks (*i.e.* $\alpha = 0.5$, see Section 6.4 (b) for details) for training, so that each task DoV label would appear multiple times during training. To examine the limit of our method, we try to reduce the number of training tasks, *i.e.* $\alpha = 0.2$ or 0.1, on Cheetah-vel-comp. In this case, each task DoV label would appear only 4 or 2 times on average among training tasks. Our zero-shot generalization performance is visualized in Figure 6.9a. Despite the performance decrease with fewer training tasks, our approach still notably outperforms the *Prior* even with the fewest training tasks ($\alpha = 0.1$). This indicates our method learns meaningful disentangled DoV embeddings even with sparse training tasks.

Effect of EMA. To justify the role of Exponential Moving Average (EMA) in our training and zero-shot generalization, we consider the following alternative strategies: 1) *training w/o EMA:* Equivalently, we set $\tau = 0$ in Eq. 6.5. In this case, the task DoV embedding is

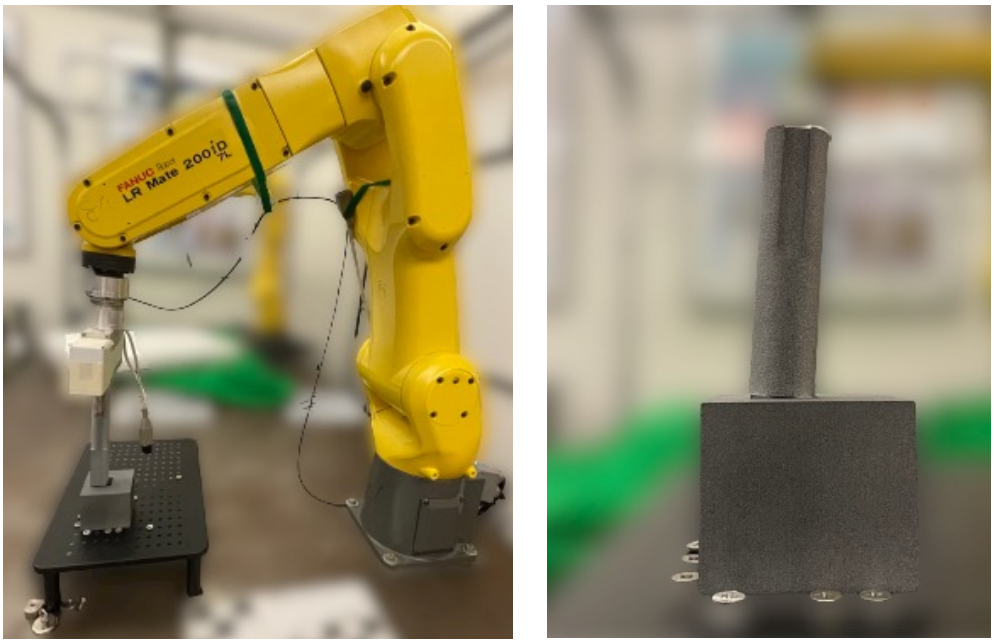
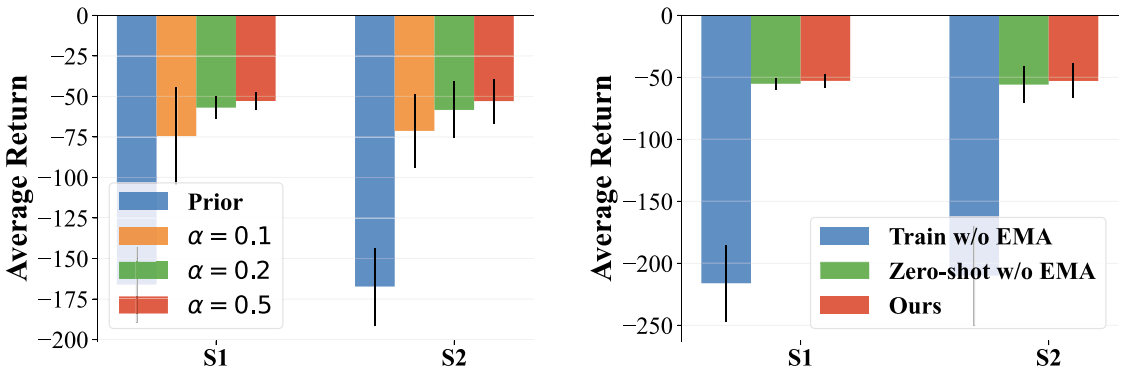


Figure 6.8: The Tiled-peg-in-hole-comp task setup in the real world. (a) Real-world setup with a FANUC LR Mate 200iD robot. (b) 5° tilted hole for experiment.



(a) Training Task Sparsity

(b) Effect of EMA

Figure 6.9: Results of ablation studies on (a) the sparsity ratio (α) of training tasks and (b) the effect of Exponential Moving Average (EMA).

regularized by the training task with the same DoV label sampled in the last training batch. 2) *zero-shot generalization w/o EMA:* We try another strategy to get the corresponding DoV embeddings for an unseen task $\mathcal{T}_u$ instead of using EMA in **S1** of Section 6.3. The average

DoV embeddings from training tasks with the same task DoV label with $\mathcal{T}_u$ is adopted instead, *i.e.* $\mathbf{z}_u^{y_j} = \mathbf{mean}\left(\{\mathbf{z}_v^{y_j} | \mathcal{T}_v \in \mathbf{T}_{train}, y_j^u = y_j^v\}\right)$. The results are shown in Figure 6.9b. Training without EMA fails because of the large noise in the regularization of the DoV embeddings. EMA "smoothes out" the noise, serving as a temporal ensemble of context accumulated during training. In addition, it provides unbiased DoV embeddings for zero-shot generalization, as indicated by the generalization performance gap with and without EMA.

6.5 Conclusion

In this chapter, we propose a meta-RL algorithm to achieve zero-shot policy generalization across unseen compositional tasks. Our method learns disentangled task representation by encoding each task DoV separately. The DoV embeddings obtained from training tasks enable zero-shot policy generalization on test tasks by composing the task embeddings of novel tasks. Our approach achieves significantly superior performance on simulation tasks and can be seamlessly applied to sim-to-real generalization.

One major limitation of the current work is the restriction to discrete set of task combinations. Achieving zero-shot generalization over continuous attribute values would be an interesting direction to explore.

Chapter 7

Concluding Remarks and Suggested Future Works

This dissertation focuses on improving data efficiency in learning algorithms for industrial robotic systems, facilitating their application in industrial settings. We explore methodologies on three important and interconnected components of robot learning systems: reward learning, policy learning, and policy generalization. Specifically in Part I, Chapter 2-3 discuss the algorithmic designs for data-efficient reward learning from two different aspects, i.e., inverse reinforcement learning (Chapter 2) and representation learning (Chapter 3). Chapter 2 builds on maximum entropy inverse reinforcement learning (IRL) and explicitly leverages the prior knowledge on efficiently generating feasible long-horizon driving trajectory samples to develop an algorithm that extracts reward function of human drivers. In Chapter 3, we learn to embed multi-modal sensory input to a structured latent space and derive reward signals from the latent embedding for contact-rich manipulation tasks.

Part II then explore data-efficient policy learning algorithms. Chapter 4 presents a novel data-efficient reinforcement learning (RL) approach for task planning in robotic palletization systems. It addresses the challenge of a vast action space, which hinders the application of standard RL methods, by using supervised learning to iteratively refine and manage the action space. This method speeds up the learning process and enhances the efficiency and reliability of task planning in robotic palletization. Chapter 5 instead focuses on a learning from demonstrations (LfD) approach that is based on motion primitives (MP). Unlike RL, this MP-based method leverages task-specific priors for increased sample efficiency. It is crafted to quickly learn and adjust MP-based insertion skills from demonstrations, using black-box function optimization to fine-tune parameters based on previous experiences. Human demonstrations provide a rich source of rewards, enabling precise parameter learning.

Finally, Part III investigates data-efficient policy generalization across different environments. Chapter 6 presents a novel zero-shot policy generalization algorithm for reinforcement learning (RL) agents, designed to apply learned policies to new, unseen tasks. Focusing on compositional tasks—new combinations of known task elements—the chapter introduces a meta-RL algorithm using disentangled task representations. This strategy enables agents to

generalize policies to novel tasks without the need for further exploration by inferring task characteristics from their disentangled representations.

Our studies in this dissertation covers multiple industrial robotics settings, including autonomous driving, robotic palleting, robotic assembly, etc. While we believe the contributions made in this dissertation further the development of data-efficient algorithms for robot learning, thereby enhancing the widespread deployment of industrial robotic systems, there are still numerous open questions that remain to be addressed. In the remainder of this chapter, we will outline potential avenues for future research.

Representation-based reward learning algorithm for non-monotonic tasks

In Chapter 3, we introduce a representation learning method to derive reward signals from multi-modal sensory input for contact-rich manipulation tasks. This method presupposes that tasks are monotonic, meaning task progression in the observation space is consistently forward. While this assumption is valid for various assembly tasks characterized by clear start and end states, it does not apply universally, such as in ping-pong, where similar states may indicate different stages of task progression. To broaden the applicability of our approach, further research could explore reward learning methods tailored specifically to non-monotonic tasks.

Integrating tasking planning and trajectory planning for better robotic palletization task planner

In Chapter 4, we present a reinforcement learning (RL)-based task planner specifically designed for robotic palletization tasks. While being effective, the current task planner only considers generating task plans that maximize the packing utility of the pallet and does not consider the ease with which a trajectory planner can generate collision-free trajectories. Given that robotic palletization systems often operate in cluttered environments, planning collision-free trajectories becomes a significant challenge. Integrating task planning with trajectory planning within the RL-based task planner's algorithmic framework presents a promising avenue for creating a more practical and efficient solution for industrial applications.

Developing a generic framework that segments, recognizes and learns motion primitives from human demonstrations

Chapter 5 introduces a framework designed to learn the parameters of motion primitives for industrial insertion tasks. For broader applicability across various industrial manipulation tasks, future research could focus on creating a comprehensive framework capable of: i) temporally segmenting any human demonstration into distinct segments, each representing a single primitive; ii) identifying the type of primitive within each segment; and iii) learning the specific parameters of each primitive demonstrated.

Bibliography

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

[2] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[3] Benjamin Alt et al. "Robot Program Parameter Inference via Differentiable Shadow Program Inversion". In: *arXiv preprint arXiv:2103.14452* (2021).

[4] Dario Amodei et al. "Concrete Problems in AI Safety". In: *CoRR* abs/1606.06565 (2016). arXiv: 1606.06565. URL: http://arxiv.org/abs/1606.06565.

[5] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular multitask reinforcement learning with policy sketches". In: (2017), pp. 166–175.

[6] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[7] Esther M Arkin et al. *An efficiently computable metric for comparing polygonal shapes*. Tech. rep. CORNELL UNIV ITHACA NY, 1991.

[8] Christopher G Atkeson and Stefan Schaal. "Robot learning from demonstration". In: *ICML*. Vol. 97. Citeseer. 1997, pp. 12–20.

[9] J Andrew Bagnell, Nathan Ratliff, and Martin Zinkevich. "Maximum margin planning". In: *Proceedings of the International Conference on Machine Learning (ICML)*. Citeseer. 2006.

[10] Christopher Berner et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv preprint arXiv:1912.06680* (2019).

[11] Andreea Bobu et al. "LESS is More: Rethinking Probabilistic Models of Human Behavior". In: *arXiv preprint arXiv:2001.04465* (2020).

[12] Wendelin Böhmer et al. "Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations". In: *KI-Künstliche Intelligenz* 29.4 (2015), pp. 353–362.

[13] Serkan Cabi et al. "Scaling data-driven robotics with reward sketching and batch reinforcement learning". In: *arXiv* (2019), arXiv–1909.

[14] Jianyu Chen, Wei Zhan, and Masayoshi Tomizuka. "Constrained iterative lqr for on-road autonomous driving motion planning". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–7.

[15] Liang-Chieh Chen et al. "Semantic image segmentation with deep convolutional nets and fully connected CRFs". In: *arXiv preprint arXiv:1412.7062* (2014).

[16] Sonia Chernova and Andrea L Thomaz. "Robot learning from human teachers". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 8.3 (2014), pp. 1–121.

[17] Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. "TS2PACK: A two-level tabu search for the three-dimensional bin packing problem". In: *European Journal of Operational Research* 195.3 (2009), pp. 744–760.

[18] Todor Davchev et al. "Residual Learning from Demonstration: Adapting Dynamic Movement Primitives for Contact-rich Insertion Tasks". In: *arXiv e-prints* (2020), arXiv–2008.

[19] Coline Devin et al. "Compositional plan vectors". In: *Advances in Neural Information Processing Systems* 32 (2019).

[20] Coline Devin et al. "Learning modular neural network policies for multi-task and multi-robot transfer". In: (2017), pp. 2169–2176.

[21] Yan Duan et al. "Rl2: Fast reinforcement learning via slow reinforcement learning". In: *arXiv preprint arXiv:1611.02779* (2016).

[22] Gabriel Dulac-Arnold et al. "Deep reinforcement learning in large discrete action spaces". In: *arXiv preprint arXiv:1512.07679* (2015).

[23] Oluf Faroe, David Pisinger, and Martin Zachariasen. "Guided local search for the three-dimensional bin-packing problem". In: *Informs journal on computing* 15.3 (2003), pp. 267–283.

[24] Nima Fazeli et al. "See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion". In: *Science Robotics* 4.26 (2019).

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: (2017), pp. 1126–1135.

[26] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". In: *International Conference on Machine Learning*. 2016, pp. 49–58.

[27] Chelsea Finn, Kelvin Xu, and Sergey Levine. "Probabilistic model-agnostic meta-learning". In: *Advances in neural information processing systems* 31 (2018).

[28] Carlos Florensa et al. "Reverse curriculum generation for reinforcement learning". In: *arXiv preprint arXiv:1707.05300* (2017).

[29] Justin Fu, Katie Luo, and Sergey Levine. "Learning robust rewards with adversarial inverse reinforcement learning". In: *arXiv preprint arXiv:1710.11248* (2017).

[30] Justin Fu et al. "Variational inverse control with events: A general framework for data-driven reward definition". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8538–8547.

[31] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[32] Jean-Bastien Grill et al. "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in Neural Information Processing Systems* 33 (2020).

[33] Shixiang Gu et al. "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 3389–3396.

[34] Tianyu Gu et al. "Tunable and stable real-time trajectory planning for urban autonomous driving". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 250–256.

[35] Abhishek Gupta et al. "Meta-reinforcement learning of structured exploration strategies". In: *Advances in neural information processing systems* 31 (2018).

[36] Chi Trung Ha et al. "An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the Physical Internet". In: *Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part II 20*. Springer. 2017, pp. 140–155.

[37] Tuomas Haarnoja et al. "Composable deep reinforcement learning for robotic manipulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6244–6251.

[38] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*. 2018, pp. 1861–1870.

[39] Dylan Hadfield-Menell et al. "Inverse reward design". In: *Advances in neural information processing systems*. 2017, pp. 6765–6774.

[40] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems*. 2016, pp. 4565–4573.

[41] Ruizhen Hu et al. "TAP-Net: transport-and-pack using reinforcement learning". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15.

[42] De-An Huang et al. "Neural task graphs: Generalizing to unseen tasks from a single video demonstration". In: (2019), pp. 8565–8574.

[43] Jan Humplik et al. "Meta reinforcement learning as task inference". In: *arXiv preprint arXiv:1905.06424* (2019).

[44] Yiding Jiang et al. "Language as an abstraction for hierarchical deep reinforcement learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[45] Lars Johannsmeier, Malkin Gerchow, and Sami Haddadin. "A framework for robot manipulation: Skill formalism, meta learning and adaptive control". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5844–5850.

[46] Mrinal Kalakrishnan et al. "Learning force control policies for compliant manipulation". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 4639–4644.

[47] Mrinal Kalakrishnan et al. "Learning objective functions for manipulation". In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 1331–1336.

[48] Dmitry Kalashnikov et al. "Scalable deep reinforcement learning for vision-based robotic manipulation". In: *Conference on Robot Learning*. 2018, pp. 651–673.

[49] Rudolf Emil Kalman. "When is a linear control system optimal?" In: *Journal of Basic Engineering* (Mar. 1964).

[50] Kyungdaw Kang, Ilkyeong Moon, and Hongfeng Wang. "A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem". In: *Applied Mathematics and Computation* 219.3 (2012), pp. 1287–1299.

[51] Daniel Kappler et al. "Data-Driven Online Decision Making for Autonomous Manipulation." In: *Robotics: Science and Systems*. 2015.

[52] Korhan Karabulut and Mustafa Murat İnceoğlu. "A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method". In: *International Conference on Advances in Information Systems*. Springer. 2004, pp. 441–450.

[53] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (June 2011), pp. 846–894.

[54] Oussama Khatib. "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1 (1987), pp. 43–53.

[55] Kenneth Kimble et al. "Benchmarking Protocols for Evaluating Small Parts Robotic Assembly Systems". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 883–889.

[56] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[57] Thomas Kipf et al. "Compile: Compositional imitation learning and execution". In: (2019), pp. 3418–3428.

[58] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[59] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. "Learning driving styles for autonomous vehicles from demonstration". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2641–2646.

[60] Michelle A Lee et al. "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8943–8950.

[61] Sergey Levine and Vladlen Koltun. "Continuous inverse optimal control with locally optimal examples". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. 2012, pp. 475–482.

[62] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[63] Kun Li, Mrinal Rath, and Joel W Burdick. "Inverse reinforcement learning via function approximation for clinical motion analysis". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 610–617.

[64] Tingguang Li et al. "Learning hierarchical control for robust in-hand manipulation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8855–8862.

[65] Weiwei Li and Emanuel Todorov. "Iterative linear quadratic regulator design for nonlinear biological movement systems." In: *ICINCO (1)*. 2004, pp. 222–229.

[66] Zhaoting Li et al. "Adaptive sampling-based motion planning with a non- conservatively defensive strategy for autonomous driving". In: *The 21st IFAC World Congress*. 2020.

[67] Wenzhao Lian et al. "Benchmarking Off-The-Shelf Solutions to Robotic Assembly Tasks". In: *arXiv preprint arXiv:2103.05140* (2021).

[68] Jacky Liang et al. "GPU-accelerated robotic simulation for distributed reinforcement learning". In: *arXiv preprint arXiv:1810.05762* (2018).

[69] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[70] Silvano Martello, David Pisinger, and Daniele Vigo. "The three-dimensional bin packing problem". In: *Operations research* 48.2 (2000), pp. 256–267.

[71] Roberto Martín-Martín et al. "Variable Impedance Control in End-Effector Space: An Action Space for Reinforcement Learning in Contact-Rich Tasks". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1010–1017.

[72] Oier Mees et al. *Adversarial Skill Networks: Unsupervised Robot Skill Learning from Video*. 2020. arXiv: `1910.09430 [cs.CV]`.

[73] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[74] Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[75] Katharina Muelling et al. "Learning strategies in table tennis using inverse reinforcement learning". In: *Biological cybernetics* 108.5 (2014), pp. 603–619.

[76] Maximilian Naumann* et al. "On the Suitability of Cost Functions for Explaining and Imitating Human Driving Behavior". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[77] Andrew Y Ng, Stuart J Russell, et al. "Algorithms for inverse reinforcement learning." In: *Icml*. Vol. 1. 2000, p. 2.

[78] Junhyuk Oh et al. "Zero-shot task generalization with multi-task deep reinforcement learning". In: (2017), pp. 2661–2670.

[79] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[80] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.

[81] Xue Bin Peng et al. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[82] Valerio Perrone et al. "Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning". In: *arXiv preprint arXiv:1909.12552* (2019).

[83] Lerrel Pinto et al. "Asymmetric actor critic for image-based robot learning". In: *arXiv preprint arXiv:1710.06542* (2017).

[84] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[85] Kate Rakelly et al. "Efficient off-policy meta-reinforcement learning via probabilistic context variables". In: *International conference on machine learning*. PMLR. 2019, pp. 5331–5340.

[86] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning." In: *IJCAI*. Vol. 7. 2007, pp. 2586–2591.

[87] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[88] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 627–635.

[89] Stéphane Ross et al. "Online planning algorithms for POMDPs". In: *Journal of Artificial Intelligence Research* 32 (2008), pp. 663–704.

[90] Dorsa Sadigh et al. "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state". In: *Autonomous Robots* 42.7 (2018), pp. 1405–1426.

[91] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems*. 2016, pp. 2234–2242.

[92] Gerrit Schoettler et al. "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards". In: *arXiv preprint arXiv:1906.05841* (2019).

[93] Gerrit Schoettler et al. "Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks". In: *arXiv preprint arXiv:2004.14404* (2020).

[94] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[95] Dexter RR Scobee and S Shankar Sastry. "Maximum Likelihood Constraint Inference for Inverse Reinforcement Learning". In: *International Conference on Learning Representations*. 2019.

[96] Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[97] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[98] Daniel Shapiro and Ross Shachter. "User-agent value alignment". In: *Proc. of The 18th Nat. Conf. on Artif. Intell. AAAI*. 2002.

[99] Avi Singh et al. "End-to-end robotic reinforcement learning without reward engineering". In: *arXiv preprint arXiv:1904.07854* (2019).

[100] Satinder Singh, Richard L Lewis, and Andrew G Barto. "Where do rewards come from". In: *Proceedings of the annual conference of the cognitive science society*. Cognitive Science Society. 2009, pp. 2601–2606.

[101] Francisco Suárez-Ruiz and Quang-Cuong Pham. "A framework for fine robotic assembly". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 421–426.

[102] Liting Sun et al. "Courteous autonomous cars". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 663–670.

[103] Liting. Sun, Wei. Zhan, and Masayoshi. Tomizuka. "Probabilistic Prediction of Interactive Driving Behavior via Hierarchical Inverse Reinforcement Learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2018, pp. 2111–2117.

[104] Jaeyong Sung, J Kenneth Salisbury, and Ashutosh Saxena. "Learning to represent haptic feedback for partially-observable tasks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2802–2809.

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

[106] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems* 30 (2017).

[107] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey." In: *Journal of Machine Learning Research* 10.7 (2009).

[108] Stephen Tian et al. "Manipulation by feel: Touch-based control with deep predictive models". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 818–824.

[109] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[110] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.

[111] Mel Vecerik et al. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *arXiv preprint arXiv:1707.08817* (2017).

[112] Oriol Vinyals et al. "Starcraft ii: A new challenge for reinforcement learning". In: *arXiv preprint arXiv:1708.04782* (2017).

[113] Florian Voigt, Lars Johannsmeier, and Sami Haddadin. "Multi-Level Structure vs. End-to-End-Learning in High-Performance Tactile Robotic Manipulation". In: *Conference on Robot Learning*. PMLR. 2020.

[114] Nghia Vuong, Hung Pham, and Quang-Cuong Pham. "Learning Sequences of Manipulation Primitives for Robotic Assembly". In: *arXiv preprint arXiv:2011.00778* (2020).

[115] Fan Wang and Kris Hauser. "Robot packing with known items and nondeterministic arrival order". In: *IEEE Transactions on Automation Science and Engineering* 18.4 (2020), pp. 1901–1915.

[116] Fan Wang and Kris Hauser. "Stable bin packing of non-convex 3D objects with a robot manipulator". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8698–8704.

[117] Ruohan Wang et al. "Random expert distillation: Imitation learning via expert policy support estimation". In: *arXiv preprint arXiv:1905.06750* (2019).

[118] Zheng Wu et al. "Efficient Reinforcement Learning of Task Planners for Robotic Palletization through Iterative Action Masking Learning". In: *CoRR* abs/2404.04772 (2024). arXiv: 2404.04772. URL: http://arxiv.org/abs/2404.04772.

[119] Zheng Wu et al. "Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5355–5362.

[120] Zheng Wu et al. "Learning Dense Rewards for Contact-Rich Manipulation Tasks". In: *arXiv preprint arXiv:2011.08458* (2020).

[121] Zheng Wu et al. "Learning dense rewards for contact-rich manipulation tasks". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6214–6221.

[122] Zheng Wu et al. "Prim-lafd: A framework to learn and adapt primitive-based skills from demonstrations for insertion tasks". In: *IFAC-PapersOnLine* 56.2 (2023).

[123] Zheng Wu et al. "Zero-shot policy transfer with disentangled task representation of meta-reinforcement learning". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 7169–7175.

[124] Deheng Ye et al. "Mastering complex control in moba games with deep reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6672–6679.

[125] Changxi You et al. "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning". In: *Robotics and Autonomous Systems* 114 (2019), pp. 1–18.

[126] Kevin Zakka et al. "Form2fit: Learning shape priors for generalizable assembly from disassembly". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9404–9410.

[127] Wei Zhan et al. "A non-conservatively defensive strategy for urban autonomous driving". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2016, pp. 459–464.

[128] Wei Zhan et al. "Constructing a highly interactive vehicle motion dataset". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 6415–6420.

[129] Wei Zhan et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: *arXiv preprint arXiv:1910.03088* (2019).

[130] Chi Zhang, Hao Zhang, and Lynne E Parker. "Feature space decomposition for effective robot adaptation". In: (2015), pp. 441–448.

[131] Jingwei Zhang, Bin Zi, and Xiaoyu Ge. "Attend2pack: Bin packing through deep reinforcement learning with attention". In: *arXiv preprint arXiv:2107.04333* (2021).

[132] Hang Zhao, Yang Yu, and Kai Xu. "Learning efficient online 3D bin packing on packing configuration trees". In: *International Conference on Learning Representations*. 2021.

[133] Hang Zhao et al. "Learning practically feasible policies for online 3D bin packing". In: *Science China Information Sciences* 65.1 (2022), p. 112105.

[134] Hang Zhao et al. "Online 3D bin packing with constrained deep reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 1. 2021, pp. 741–749.

[135] Tony Z Zhao et al. "Meld: Meta-reinforcement learning from images via latent state models". In: *arXiv preprint arXiv:2010.13957* (2020).

[136] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: (2020), pp. 737–744.

[137] Allan Zhou et al. "Policy Architectures for Compositional Generalization in Control". In: *arXiv preprint arXiv:2203.05960* (2022).

[138] Yuke Zhu et al. "Reinforcement and imitation learning for diverse visuomotor skills". In: *arXiv preprint arXiv:1802.09564* (2018).

[139] Yuke Zhu et al. "robosuite: A Modular Simulation Framework and Benchmark for Robot Learning". In: (2020).

[140] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[141] Luisa Zintgraf et al. "Fast context adaptation via meta-learning". In: (2019), pp. 7693–7702.

[142] Luisa Zintgraf et al. "Varibad: A very good method for bayes-adaptive deep rl via meta-learning". In: *arXiv preprint arXiv:1910.08348* (2019).