Reinforcement Learning for Robot Assembly

Vuong Quoc Nghia

School of Mechanical & Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15 August 2023

Date

Vuong Quoc Nghia

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15 August 2023

Date

Quang-Cuong Pham

Authorship Attribution Statement

This thesis contains material from 4 papers published / submitted in the following peer-reviewed journals / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Nghia Vuong, Hung Pham, and Quang-Cuong Pham. "Learning sequences of manipulation primitives for robotic assembly." In 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 4086-4092. IEEE, 2021. DOI: 10.1109/ICRA48506.2021.9561029.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Hung Pham and Prof Quang-Cuong Pham.
- I performed all experiments.
- Dr. Hung Pham assisted in the methodology development.

Chapter 4 is published as Lee Yee Sien, Nghia Vuong, Nicholas Adrian, and Quang-Cuong Pham. "Integrating Force-based Manipulation Primitives with Deep Learning-based Visual Servoing for Robotic Assembly." In ICRA 2022 Workshop: Reinforcement Learning for Contact-Rich Manipulation. 2022.

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction.
- I co-designed the methodology with Lee, Yee Sien.
- Lee, Yee Sien and I performed all experiments.
- Lee, Yee Sien prepared the manuscript drafts. I reviewed the manuscript.

Chapter 5 is submitted as Nghia Vuong, and Quang-Cuong Pham. "Controller Influence on Reinforcement Learning performance for Contact-rich tasks." in 2024 IEEE/SICE International Symposium on System Integration (SII2024)

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction.
- I prepared the manuscript drafts. The manuscript was revised by Prof Quang-Cuong Pham
- I performed all experiments.

Chapter 6 is published as Nghia Vuong and Quang-Cuong Pham. "Contact Reduction with Bounded Stiffness for Robust Sim-to-Real Transfer of Robot Assembly." DOI: https://doi.org/10.48550/arXiv.2306.06675

The contributions of the co-authors are as follows:

- Prof Quang-Cuong Pham provided the initial project direction.
- I prepared the manuscript drafts. The manuscript was revised by Prof Quang-Cuong Pham
- I performed all experiments.

15 August 2023

Date

Vuong Quoc Nghia

Acknowledgements

I would like to thank my supervisor, Quang-Cuong Pham. Cuong has taught me how to do good research and given me insightful advice and constructive criticism that are distilled into this thesis. I am grateful for his continuous support during my PhD, not only in the lab but also in life.

I am grateful to Thao Doan, Huy Nguyen, and Quang-Cuong Pham for introducing me to scientific research. This journey wouldn't have started without their support.

I've been fortunate enough to have amazing friends and colleagues in CRI group: Huy Nguyen, Hung Pham, Joyce Lim Xin Yan, Jianhui Lim, Shohei Fuji, Thach Do, Bing Song, Nicholas Adrian, and many others. Thank you for the discussions and all the fun experiences.

Finally, I thank my parents for their constant love and for supporting me in pursuing a PhD degree in Singapore. They have always been a source of comfort and motivation throughout this process.

Abstract

Robotic systems are traditionally employed in manufacturing to automate repetitive tasks such as welding, painting, and pick-and-place. Despite tremendous progress in robotics research, the classical assembly skill remains a challenge. In most cases, the difficult assembly skills still rely heavily on the engineer's expertise [1]. In addition, the skills are prone to failure in the face of new tasks or variations, such as the shape or size of objects. This is particularly important as customer demand for greater product variety has recently increased. Learning approaches will become prominent in this context since learning shifts the burden from humans to the robot. Instead of attempting to obtain an accurate model of the surrounding environments or to program the controller, the robot can acquire a dynamics model or directly learn optimal control policies from experience. Reinforcement Learning endows a robot with the ability to find optimal behavior autonomously by interacting with its surrounding environment. The integration of deep learning models into RL, known as deep reinforcement learning, has gained significant traction and demonstrated remarkable achievements across various domains. However, contemporary deep reinforcement learning algorithms still encounter numerous challenges when applied in real-world robot manipulation. First, samples on a robotics system are expensive and tedious to obtain. Adding to this problem, model-free deep reinforcement learning algorithms are known to be sample inefficient, i.e., they require a large number of samples. Second, real-world training raises safety concerns. The environment or the engineer might impose several constraints that the robot must satisfy at all times to ensure safety. These constraints are difficult to maintain during the exploration phase, which often involves random action sampling. The two mentioned challenges are among the fundamental issues that prevent integrating deep reinforcement learning into robotics control systems.

This thesis demonstrates how we can possibly improve sample efficiency and enable safe learning, making RL more practical for realistic robot tasks. Firstly, it demonstrates substantial improvement in sample efficiency by using manipulation

primitives as actions. Manipulation primitives are simple yet generic enough to generalize across various tasks. Secondly, incorporating low-level feedback controllers into RL provides prior knowledge, which can increase learning speed and improve policy performance. A key message in this work is that a robust and high-performance low-level controller can further improve the robustness and performance of policies. Finally, this thesis examines methods to narrow the reality gap - the fundamental problem in sim-to-real reinforcement learning. This work proposes a novel contact reduction method to improve simulation accuracy, facilitating sim-to-real transfer for complex assembly tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robotic systems are traditionally employed in manufacturing to automate repetitive tasks such as welding, painting, and pick-and-place. Despite tremendous progress in robotics research, the classical assembly skill remains a challenge. In most cases, the difficult assembly skills still rely heavily on the engineer's expertise [1]. In addition, the skills are prone to failure in the face of new tasks or variations, such as the shape or size of objects. This is particularly important as customer demand for greater product variety has recently increased . Learning approaches will become prominent in this context since learning shifts the burden from humans to the robot. Instead of attempting to obtain an accurate model of the surrounding environments or to program the controller, the robot can acquire a dynamics model or directly learn optimal control policies from experience.

Reinforcement Learning (RL) endows a robot with the ability to find optimal behavior autonomously by interacting with its surrounding environment. The robot selects *actions* (e.g. motor torque commands) to alter the environment according to a *policy*; the *state* of the system then evolves according to the system *dynamics* unknown to the robot. The state of the system includes the robot state (e.g. joint angles, joint velocities, links' position) and the environment state (e.g. position and velocity of all the objects in the environment). In practice, robots are equipped with sensors that provide *observations*. The observations might include noisy measurements of the states, or statistics that can be used to infer the states

(e.g. images from a camera). The designer defines the goal by providing a *reward function* that evaluates instantaneous performance of executing an action. The goal of RL is to find an optimal policy to maximize the accumulated rewards.

Recently, the integration of deep learning models into RL, known as Deep Reinforcement Learning (DRL), has gained significant traction and demonstrated remarkable achievements across various domains. Notable accomplishments include reaching superhuman performance in playing Atari games [2], achieving human-level performance in Go [3], and learning control policies for continuous control problems in simulation [4]. However, contemporary DRL algorithms still encounter numerous challenges when applied in real-world robot manipulation. First, samples on a robotics system are expensive and tedious to obtain. In addition, model-free DRL algorithms are known to be sample inefficient, i.e., they require a large number of samples. For instance, days or years of simulation corresponding to bilions of state transitions are necessary to train control polices for locomotion or dexterous manipulation [5, 6, 7]. Second, real-world training raises safety concerns. The environment or the engineer might impose several constraints that the robot must satisfy at all times to ensure safety. Examples of constraints include the separation between rigid bodies (natural constraint) or limits on contact forces (artificial constraint). These constraints are difficult to maintain during the exploration phase, which often involves random action sampling. The two mentioned challenges are among the fundamental issues that prevent the integration of DRL into robotics control systems.

A generic robot control paradigm with an RL agent in the loop is illustrated in Fig. 1.1. The framework includes an RL agent, an optional low-level controller, and an optional sensing module. On one extreme, the RL agent directly outputs motor commands based on raw sensor inputs [8]. Alternatively, the RL agent can reason on a high-level action representation and utilizes a low-level feedback controller to map actions to low-level motor commands [9, 10, 7, 11] . The low-level controller serves as prior knowledge, simplifying policy learning by solving some sub-problems, such as regulating the desired end-effector pose, regulating a desired end-effector force, or ensuring the robot's stability during interaction with the environment. Similarly, prior knowledge about sensor processing, such as feature extraction from images, can be incorporated into the sensing module.

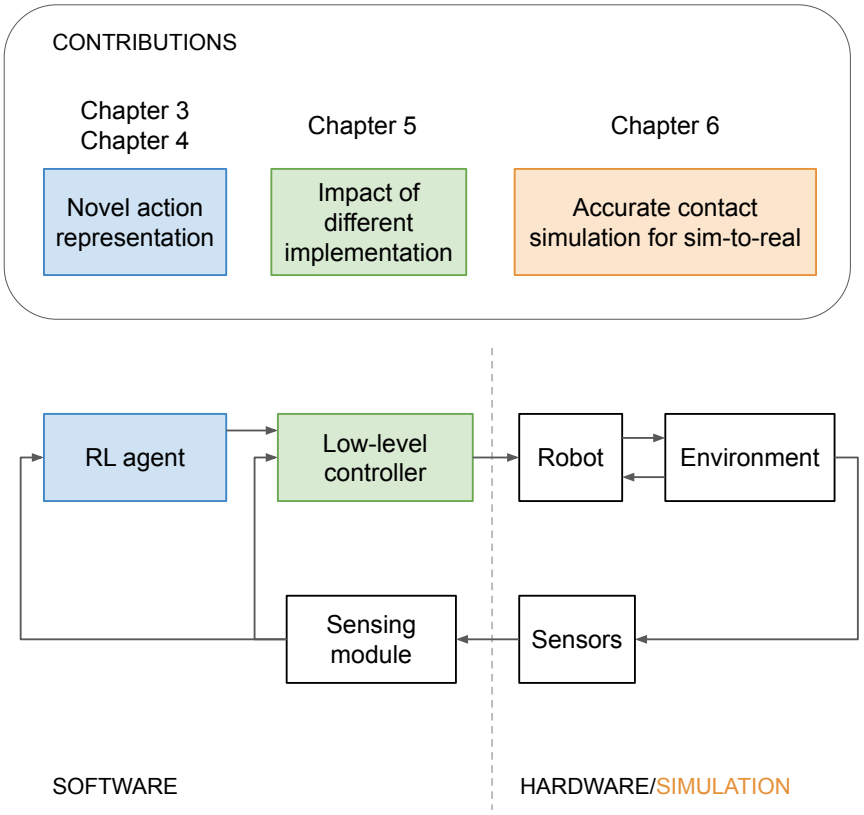


FIGURE 1.1: A general RL-integrated control paradigm for robot systems and overview of contributions

This thesis looks into three crucial components of the described framework: action representation, low-level controller, and simulation. The motivations are described as follows.

- The role of action representation: A fundamental problem in designing RL systems is the representation of action: What is the set of atomic actions that RL agents reason upon? Early works on robot RL have explored various approaches to simplify learning by choosing appropriate action representations, including action discretization or utilizing *temporal abstraction*, i.e., constructing more intelligent actions. As an example of the latter approach, Kalmár et al. [12] decompose a task into sub-tasks and design controllers to accomplish each sub-task. The RL agent's objective is to learn sequences of controllers to fulfill the primary task. The underlying idea is that sub-tasks are key steps toward accomplishing the main task and are often much easier

to solve. As only a few sub-tasks are necessary for a given task, this approach maintains a low-dimensional action space, thus enhancing learning efficiency.

On the other hand, DRL research in robotics has focused on learning in continuous action spaces [8, 7]. However, these methods often suffer from poor sample efficiency. Temporal abstraction potentially provides a solution to this issue, given its historical success in hierarchical and robot reinforcement learning studies. However, effectively integrating temporal abstraction presents two challenges (1) the choice of an appropriate high-level action representation and (2) the trade-off between the expressiveness and compactness of the action representation: only a few actions might limit the achievable behavior of the robot.

- The role of the low-level controller: In the generic control paradigm, the designer usually chooses the action representation first, then designs a low-level controller to map the action to low-level motor commands. A matter that is usually overlooked in the literature is that, given a control objective, there exists a variety of implementations for the low-level controller from the vast literature of control theory and robot control. For instance, Roy and Whitcomb [13] reported at least three implementations for direct force control of position-controlled robot manipulators. Nakanishi et al. [14] reported eight implementations for the task space control of redundant robot manipulators. Different implementations may have very different control performance and robustness. How different implementations influence RL and how the difference in the performance and robustness of the implementation projects to difference in RL performance are still open problems.

- The role of simulation: Experience on a robot is expensive and tedious to obtain. Safety is another concern when exploring manipulation tasks. In contrast, data can be generated in simulation cheaper, faster, and safer. This advantage motivates the *sim-to-real* methodology: the policy is first learned with data generated in simulation and then transferred to the physical robots. The main challenge of sim-to-real RL is overcoming the *reality gap* - the discrepancies between the real world and its simulated counterpart. The reality gap may originate from unmodelled physical phenomena, inaccurate parameter estimation, or the discretized numerical integration. For example, a challenging aspect to simulate accurately is friction. An underestimation

of frictional force can result in lacking control effort to get the robot moving in the real world. In the worst case, the learner might exploit the imperfect simulator, resulting in physically implausible motion [15]. For these reasons, bridging the reality gap is critical for successful sim-to-real transfer.

1.2 Contributions

1.2.1 Learning Sequences of Manipulation Primitives for Robot Assembly

This thesis hypothesizes that Manipulation Primitives (MPs) offer an appropriate action representation for learning robot assembly. Manipulation Primitives , such as "Move down until contact", "Slide along x while maintaining contact with the surface", have enough complexity to keep the search tree shallow (typically a sequence of 6 to 8 MPs is enough to achieve tight insertion), yet are generic enough to generalize across a wide range of assembly tasks (peg insertion with different peg shapes, large hole estimation errors, random initial positions...) Another key advantage of MPs is their additional *semantics*, which make them robust in sim-to-real and against model/environment variations and uncertainties: consider how "Move down until contact" is inherently more robust than a sequence of several short "Move down" actions.

1.2.2 Integrating Force-based Manipulation Primitives with Deep Learning-based Visual Servoing for Robotic Assembly

Manipulation primitives assume the existence of a task frame and that the task frame can be estimated with high accuracy. Such an assumption limits the practicality of the proposed RL framework in the previous section. This thesis proposes to integrate Deep Learning-based Visual Servoing (DLVS) to alleviate this assumption. In particular, DLVS is employed to achieve initial alignment between the two parts and attain an estimate of the task frame simultaneously.

1.2.3 Controller Influence on Reinforcement Learning Performance for Contact-rich Tasks

This thesis presents an experimental study on the influence of low-level controller implementation on training performance and policy performance. In particular, we focus on direct force control for contact-rich manipulation tasks with position-controlled robots. Three controllers are designed by two force control methods: Proportional-Integral controllers and Convex Controller Synthesis (CCS). Policies trained by these controllers are then compared in terms of task performance and robustness, both in simulation and the real world.

1.2.4 Contact Reduction with Bounded Stiffness for Robust Sim-to-Real Transfer of Robot Assembly

Generic geometric representations, such as convex decomposition, triangular mesh [16], signed distance field [17] may generate many contact points for geometrically-complex objects. Excessive contact points reduce simulation speed and potentially cause numerical instability. This thesis proposes a contact reduction method with bounded stiffness to improve the simulation accuracy. Our method is beneficial when the simulation consists of stiff rigid bodies, in which cases we argue that the number of contact points greatly influences simulation accuracy. Compared to previous works, our method includes an additional post-processing step, which relies on the concept of *contact stiffness*. We show that the proposed method enables training RL policy for a tight-clearance double pin insertion task and successfully deploying the policy on a rigid, position-controlled robot.

1.3 Outline of the Thesis

Chapter 2 reviews the related literature. The main contributions are presented in Chapter 3, 4, 5, 6. The summary of contributions and the future work are presented in Chapter 7.

Chapter 2

Literature Review

2.1 Conventional methods for robot assembly

The main challenge in robot assembly is the requirement for high precision between the mating parts. Consequently, past research has looked into methods to accomplish assembly tasks despite uncertainty in the robot's position relative to the environment. Two key ingredients to achieve this goal are (1) compliant motion and (2) sensor-based programming. Compliant motion refers to a robot's movement while in contact with the environment, involving the fulfillment of specific constraints on robot motion. This section will review approaches for achieving compliant motion and high-level sensor-based strategies.

A robotic assembly control system may comprise a Low-Level Controller (LLC), a High-Level Controller (HLC), and a sensing module. Their interplay is illustrated in Fig. 2.1. The LLC aims to address sub-problems, such as regulating the desired end-effector pose or force or ensuring the system's stability during contact. This simplifies high-level control strategy design. For instance, in pick-and-place tasks, reasoning and planning can be performed in the task space, while an LLC is used to achieve the task-space motion plan. The HLC serves as the interface between task specifications and robot execution. Examples of the HLC include a reference trajectory generated by a motion planner, a sequence of manipulation primitives, or Dynamics Movement Primitives learned from demonstrations. The sensing module processes raw sensory signals by, for instance, filtering, extracting features, or

state estimation; the processed signals are then used as feedback to the low-level controller or the high-level controller.

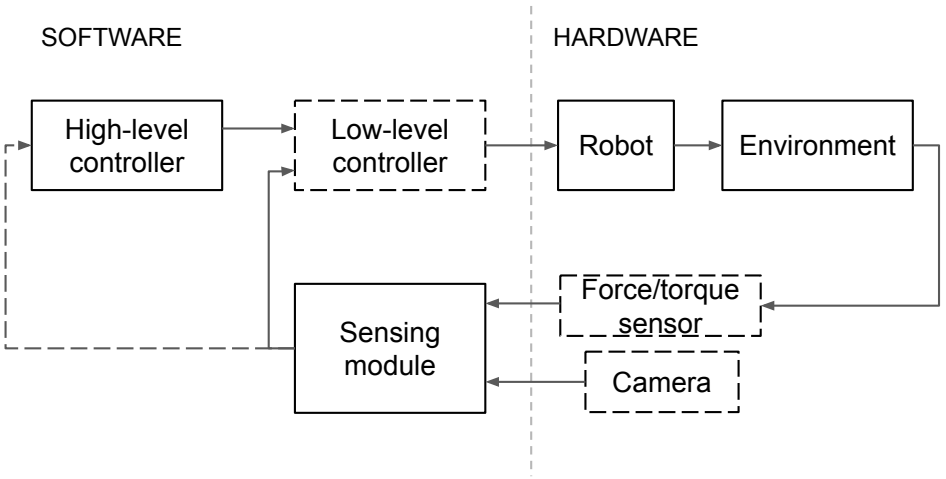


FIGURE 2.1: Components and signal flows in a robotic assembly system. The dashed lines and dashed boxes indicate optional components.

We categorize approaches to the LLC into position control and force control. Position control aims to bring the robot to a specific configuration as accurately as possible. This type of controller is usually employed in industrial robots to achieve high control precision. However, using position control for manipulation is prone to failure. Indeed, successful manipulation via position control hinges on collision-free motion planning. This would, in turn, require an accurate model of the environment (geometry of the contact surface, relative position of the objects with respect to the robot), which is difficult to obtain in practice. Model uncertainties result in planning errors, causing the robot to collide with the environment and deviate from the desired trajectory. The control system reacts to reduce such deviation, adding up the contact force until the breakage of the parts in contact occurs. Compliant motion with position control can be achieved through specialized hardware. In contrast, force control is a method to attain programmable compliant motion.

HLC approaches are divided into sensorless and sensor-based categories based on whether these approaches utilize sensors, such as visual sensors or force/torque sensors, to extract environment-related information. A sensorless HLC can be thought of as an open-loop motion. As such, it can fail due to disturbances or unconsidered uncertainties during motion generation. In contrast, sensor-based

TABLE 2.1: Taxonomy of research on robotic assembly

LLC / HLC	Position control (mechanical compliance)	Force control (programmable compliance)
sensorless	Simunovic, 1975 [18], Whitney, 1982 [19], Hollis, 1991 [20], Joo and Miyazaki, 1998 [21], Sturges and Laowattana, 1995 [22]	Peshkin, 1990 [23], Schimmels, 1997 [24], Hirai et al., 1996 [25], Asada, 1993 [26], Lozano-Perez et al., 1983 [27], Erdmann, 1986 [28], Laugier, 1989 [29], McCarragher and Asada, 1993 [30], Hirukawa et al., 1994 [31], Xiao and Ji, 2001 [32], Ji and Xiao, 2001 [33]
sensor-based		Finkemeyer et al., 2005 [34], Chhatpar and Branicky, 2001 [35], Johannsmeier et al., 2019 [36], Suarez-Ruiz and Pham, 2016 [1], Dakin and Popplestone, 1992 [37], Dakin and Popplestone, 1993 [38], Rosell et al., 1999 [39],

HLC may exploit sensor information to guide robot motion and correct errors (e.g., deviations from a reference trajectory) during execution. Yet, designing sensor-based HLC is challenging due to limited theory.

2.1.1 Robot assembly with position control

Compliant behavior through position control can be achieved using a mechanical device known as *remote center compliance* (RCC). The RCC comprises a mechanical spring structure attached to the robot manipulator's end-effector to hold one part. The RCC is designed to have high stiffness along the insertion direction but high lateral and angular compliance. This setup prevents scenarios where the peg becomes stuck due to opposing contact forces (wedging) or imbalanced forces/moments (jamming) [19]. Moreover, it projects the compliance center near the tip of the part - hence the name "remote center compliance" - to correct minor lateral and angular misalignments.

The RCC is originally designed for the cylindrical peg-in-hole task with chamfers on the hole entrance [18, 19]. These studies assume partial peg insertion, preferably contacting a chamfer. Following these works, several improvements have been proposed to expand the applicability of RCC. Whitney and Rourke [40] introduce a simplified mechanical structure of the RCC for real assembly lines. Variable compliance or variable center of compliance is proposed in [20, 21, 41]. The extension to square peg-in-hole assembly is achieved in [22]. Other works incorporate vibratory motion into RCC for faster assembly [42, 43].

While the RCC is low-cost and reliable for fast insertions, its design complexity rises quickly for parts with complex shapes, resulting in more possible contact states. For example, consider the square peg-in-hole task with hundreds of feasible contact states. Designing an RCC that guides the held part to the goal state from all possible contact states is extremely challenging. This limitation hinders the RCC's application to diverse assembly tasks involving various parts.

2.1.2 Robot assembly with force control

2.1.2.1 Force control methods

As previously mentioned, force control enables programmable compliant motion, also known as active compliant motion [44]. There are two classes of force control: indirect force control and direct force control. Indirect force control, such as impedance control or admittance control [45] aims to achieve a static or dynamic relation between end-effector force and motion. This relationship is an impedance if the robot responds to the motion deviation by generating forces and corresponds to an admittance if the robot reacts to interaction forces by issuing a deviation from the desired motion. A robot manipulator under impedance or admittance control behaves as a mass-spring-damper system with programmable parameters.

In contrast, direct force control explicitly regulates the contact force and moment with a feedback control law. A prominent approach within this category is hybrid force/motion control, which simultaneously controls the motion and force/moment in the unconstrained and constrained subspace, respectively. The user must specify the desired motion and the desired contact force and moment in a consistent way with respect to the constraints imposed by the environment. This can be achieved

by specifying a task frame [46], accompanied by a selection matrix in simple cases [47]. In a general contact task, one must define appropriate projection matrices. These projection matrices can be derived from explicit constraint equations [48, 49, 50]. Various implementations of the force control loop exist, such as passivity-based [51], outer force control loop closed around an inner motion control loop [46], or general controllers obtained through Convex Controller Synthesis [52].

2.1.2.2 Sensorless high-level controller

Damping control [53], a special case of impedance control, is a force control method that can work with a simple sensorless HLC. In damping control, the commanded velocity of the held part is modified based on the sensed contact force.

The manipulator equations in joint space are given by:

$$A(\mathbf{q})\ddot{\mathbf{q}} + b(\mathbf{q}, \dot{\mathbf{q}}) + g(\mathbf{q}) = \boldsymbol{\tau} \tag{2.1}$$

where $b(\mathbf{q}, \dot{\mathbf{q}}), g(\mathbf{q}), \boldsymbol{\tau}$ represent the Coriolis and centrifugal forces, gravitational forces, and joint torques, respectively. $A(\mathbf{q})$ is the joint space inertia matrix. The generalized forces relates to the operational forces by:

$$\boldsymbol{\tau} = J^T \mathbf{f} \tag{2.2}$$

The manipulator can be controlled in the operational space by the following control law:

$$\mathbf{f} = \ddot{\mathbf{x}}_d + k_v(\dot{\mathbf{x}}_d - \dot{\mathbf{x}}) - k_p(\mathbf{x}_d - \mathbf{x}) \tag{2.3}$$

where $\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d$ are the desired position, velocity, and acceleration, respectively, and $\mathbf{x}, \dot{\mathbf{x}}$ are the actual position and velocity. k_p, k_v are the position and velocity gain matrices, respectively. The damping control law can be obtained by choosing the desired end-effector velocity as follows:

$$\dot{\mathbf{x}}_d = \dot{\mathbf{x}}_0 + \mathbf{D}\mathbf{f}_e \tag{2.4}$$

where $\dot{\mathbf{x}}_0$ is the six-dimensional reference velocity given by the HLC (e.g., a constant velocity along the insertion direction), $\mathbf{D}$ is the 6×6 damping matrix, and $\mathbf{f}_e$ is the six-dimensional sensed contact force. By appropriately designing the damping

matrix $\mathbf{D}$, the robot can achieve compliance with the environment while being able to correct small position or orientation errors of the held part.

The effectiveness of the damping control law depends on designing a proper admittance matrix D. Past research has concentrated on finding a single D that ensures successful assembly operations irrespective of what contact states the held peg may encounter throughout the process [23, 24, 25, 26]. However, these works are limited to the round peg-in-hole insertion. Moreover, this approach encounters the same challenge as the RCC method: an explosion of possible contact states for parts with complex shapes.

Lozano-Perez [54] propose an approach to planning motion strategies that would not fall in the presence of uncertainties based on the concept of preimages in configuration space. A preimage of the goal state is a function of a commanded velocity that returns configurations from which the velocity will guarantee that the held part reaches the goal state despite location and velocity uncertainties. Given the initial position and the goal state, the preimage approach generates a motion plan in backward chaining by finding the preimage of the goal state associated with a commanded velocity and then the preimage of the preimage, repeating until a preimage that includes the initial configuration of the held part is found. However, the main drawback of this method is the expensive preimage computation.

Another class of sensorless HLC performs planning on a predetermined graph of topological contact states [29], [30]. McCarragher and Asada [30] models an assembly task of polygonal parts as a discrete event system via Petri nets. However, the manual generation of contact states and transitions is tedious even for objects with simple geometry and infeasible for complex tasks due to the enormous number of contact states. Thus, an automated method for creating a contact state graph is preferable. Hirukawa et al. [31] initiated this by exhaustively enumerating all potential contact states and transitions. Xiao and Ji [32] later proposed an efficient divide-and-merge method to autonomously construct a contact state graph for arbitrary polyhedra, being able to create hundreds or thousands of nodes and links within seconds. The contact state graph enables the decomposition of compliant motion planning into two sub-problems: (1) high-level graph search for state transitions in the contact state graph and (2) low-level motion planning within one contact state's configurations. Motion planning techniques like probabilistic

roadmap motion planning [55] can be adapted to address low-level motion planning [33].

2.1.2.3 Sensor-based high-level controller

A prevalent approach to sensor-based HLC is sequencing simpler sensor-based motion primitives, such as "move until contact" and "slide along the x-axis with a constant velocity" [34, 35, 36, 1]. Several works have proposed general frameworks to ease the task specification and motion strategy design [56, 57, 58]. This approach is appealing for its simplicity yet effective in many tasks. However, the design of the motion strategy is tedious and requires considerable engineering effort and expertise.

Another notable class of sensor-based HLC is error-correction modules that detect failure and adjust the original motion plan accordingly. For instance, the two-phase approach [38], [37], [39] performs motion planning to find the reference motion in the first phase; the second phase implements a local and online replanning to deal with unintended contacts due to uncertainties. The success of such an approach depends on the accurate online identification of contact states.

Contact state identification typically exploits sensory information about position and force/torque data. This problem is nontrivial due to sensing uncertainties and complex mapping between sensory signals and contact states. A major direction involves the analytical model of contact states. A method along this line predetermines a set of features (configurations, force/torque limits) for each contact state and matches them against sensed data for real-time identification [59], [60]. Learning is another prevalent direction. Past research has explored various models for learning, including hidden Markov models [61], [62], neural network structures [63], [64], and fuzzy classifiers [65]. However, these solutions are task-dependent: new tasks or objects require re-training.

2.2 Reinforcement Learning for Robot Manipulation

2.2.1 Reinforcement learning overview

A Reinforcement Learning (RL) problem involves an agent interacting with an environment in discrete timesteps. At each timestep t, the agent receives the state s_t, takes an action a_t defining how the agent should act, and receives a scalar reward r_t, indicating the performance of the action. The action a_t alters the state of the agent and the environment according to the transition probabilities $p(s_{t+1}|s_t, a_t)$. The agent selects action $a \in \mathcal{A}$ according to a policy π. A policy can be either deterministic or stochastic, mapping a state to an action $a_t = \pi(s_t)$ or a probability distribution over actions $\pi(a_t|s_t)$ respectively. Altogether, the agent-environment interaction can be modeled as a Markov Decision Process with a state space $\mathcal{S}$, action space $\mathcal{A}$, an initial state distribution $p(s_0)$, transition dynamics $p(s_{t+1}|s_t, a_t)$, and reward function $r(s_t, a_t)$ or $r_T(s_T)$ for the final reward of episodic problems.

The performance of the agent can be evaluated by the long-term sum of reward $R(\tau)$ where $\tau = (s_0, a_0, s_1, a_1, \dots)$ is called a *trajectory*. In an *episodic* setting, where the agent is restarted to the initial state after reaching a final state, the accumulated reward for an episode is the sum of rewards:

$$R(\tau) = r_T(s_T) + \sum_{t=0}^{T-1} r(s_t, a_t) \tag{2.5}$$

In constrast, *continuing* tasks, such as on-going process control tasks, go on continually. For these tasks, a discount factor $\gamma \in (0, 1)$ can be used to discount future rewards:

$$R(\tau) = \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t) \tag{2.6}$$

The goal of RL is to find an optimal policy π^* that maximizes the expected long-term rewards

$$J_\pi = \mathbb{E}[R(\tau)|\pi] = \int R(\tau) p_\pi(\tau) d\tau \tag{2.7}$$

where $p_\pi(\tau)$ is the distribution over trajectories obtained by following policy π. For a stochastic policy $\pi(a|s)$, the trajectory distribution is

$$p_\pi(\tau) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{2.8}$$

For a deterministic policy, the trajectory distribution is

$$p_\pi(\tau) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, \pi(s_t)) \tag{2.9}$$

Many tasks in robotics can be naturally formulated as RL problems. For instance, consider controlling a robot manipulator equipped with a parallel gripper to grasp an object in clutter and moves it to the target position. The state consists of the robot's states (e.g., joint angles, joint velocities, end-effector pose, and gripper position) and the environment states (e.g., poses of target object, pose of surrounding objects). The action available to the robot might be the motor torque or the desired joint angles sent to a joint position controller. Finally, the reward function could comprise a primary term based on the success of grasping and moving the object to the target location and secondary criteria such as avoid of forceful impacts between the robot and surrounding objects or smoothness of the motion. In general, robotics problems pose three fundamental challenges for RL algorithms:

(1) Robotics problems are often characterized by high-dimensional, continuous state and action space.

(2) The system's state is partially observable in practice. The RL agent is typically provided with the estimated state obtained by sensors or observations such as image pixels or contact force/torque.

(3) Collecting data on a real physical system is costly and might be unsafe.

RL methods can be broadly categorized into three approaches: value-function approaches, policy search, or actor-critic methods. *Value-function approaches*, such as Q-learning [66, 67], or SARSA [68], rely on the value function $V^\pi(s)$ or the

action-value function $Q^{\pi}(s,a)$ defined as follows

$$V^{\pi}(s) = \mathbb{E}[R(\tau)|s_0 = s, \pi] \tag{2.10}$$

$$Q^{\pi}(s,a) = \mathbb{E}[R(\tau)|s_0 = s, a_0 = a, \pi] \tag{2.11}$$

A wide variety of value-function-based algorithms attempt to estimate the optimal value function $V^*(s)$ or the optimal action-value function $Q^*(s,a)$ corresponds to the optimal policy. These methods make use of the Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \mathbb{E}[r(s,a) + \gamma V^*(s')|s,a] \tag{2.12}$$

$$Q^*(s,a) = \mathbb{E}[r(s,a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a')|s,a] \tag{2.13}$$

Given the optimal value function (or the optimal action-value function), the optimal policy can be deduced:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \mathbb{E}[r(s,a) + \gamma V^*(s')|s,a] \tag{2.14}$$

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a) \tag{2.15}$$

In contrast, *policy search* find the optimal policy in a class of parameterized policy π_θ, where θ is the set of policy parameters. The policy parameters are updated iteratively:

$$\theta_{i+1} = \theta_i + \Delta\theta_i \tag{2.16}$$

The key step is the update rule $\Delta\theta_i$. One of the most popular approaches is based on gradient ascent:

$$\Delta\theta_i = \alpha \nabla_\theta J(\theta_i) \tag{2.17}$$

where α is the step size. The gradient $\nabla_\theta J(\theta_i)$ can be estimated based on finite difference [69], policy gradients [70]. Other than gradient ascent, other methods have also been proposed, including approaches based on expectation maximization [71, 72], information-theoretic methods [73, 74], and path integral methods [75].

A value function can also be learned along with the parameterized policy, resulting in a class of methods called *actor-critic* methods. Actor-critic methods combine the strong points of value-function methods and policy search. For instance, the value function can be used as a "baseline" in the policy gradient approach [73] to reduce the variance of policy updates.

Policy search approaches are more suitable for robotics in many aspects. In high-dimensional problems, value-function approaches require function approximation since attaining total coverage in the high-dimensional state space is impractical. Most theoretical guarantees no longer hold for this approximation, and finding optimal action can be a complex optimization problem itself. Another issue is that learning processes might be unstable under approximation [76] since a slight change in the policy may cause a large change in the value function, which again causes a large change in the policy.

In contrast, using parameterized policies reduces the search space of possible policies, allowing policy search to scale into high-dimensional, continuous action space. The local optimization nature of policy search often leads to local optima, but it is the local coverage that results in improved scalability of policy search. Furthermore, local optima are often sufficient in many robotics tasks. In addition, policy search enables the incorporation of pre-structured policies, such as movement primitives [77], model-based controllers [11, 78], that can significantly speed up the learning process. For these reasons, early successful demonstrations of RL on robotics problems are mostly policy search methods [79, 73, 80, 81].

The integration of deep learning models into reinforcement learning, resulting in deep reinforcement learning (DRL), has enabled RL to scale to complex, high-dimensional problems. In DRL, the policy, value function, or both are approximated with deep neural networks. The rise of DRL starts from the Deep Q-Network (DQN) algorithm [2], which can achieve human-level performance in Atari games. The DQN algorithm approximates the action-value function with a deep convolutional network and performs batch updates by sampling from an experience replay. Following the publication of DQN, numerous extensions and improvements have been proposed, such as DDQN (Double DQN) [82], Prioritized experience replay [83], Dueling network [84], Rainbow [85].

DRL also finds success in policy search and actor-critic approaches. In this approach, the policy is represented by neural networks, enabling the agent to learn complex mappings from high-dimensional states to actions. Most algorithms are based on the policy gradient theorem [70]. These algorithms, such as Trust Region Policy Optimization (TRPO) [86], Proximal Policy Optimization (PPO) [87], Deep Deterministic Policy Gradient (DDPG) [4], Twin Delayed Deep Deterministic (TD3) [88], Soft Actor-Critic (SAC) [89], have shown impressive performance in

various domains playing video games from pixels, including simulated continuous control tasks, robot manipulation, and locomotion.

2.2.2 Action representation

In robotics applications, actions are most naturally represented as torque commands to the robot actuators or joint position commands sent to the embedded controller on industrial robots. However, this is extremely challenging to achieve with classical reinforcement learning methods since most methods assume discrete state and action spaces, and require full or significant coverage of all possible state-action pairs. One approach to address this issue is discretization, which involves approximating continuous spaces using discrete sets by subdividing each dimension into multiple regions, but this approach quickly becomes infeasible with higher dimensions. While policy search methods can accommodate continuous action spaces, they come with their own limitations. They do not guarantee the satisfaction of constraints imposed by physical systems or the users, such as acceleration and velocity limits or stable interactions. Consequently, these methods can potentially lead to unsafe, catastrophic behavior during the exploration phase.

Early robot RL aims to simplify learning by seeking action representations that reduce the dimensionality of action spaces. Discretization has proved effective in low-dimensional tasks like 1-D ball balancing [90] or 2-D crawling motion [91]. Further efficiency comes from selecting relevant actions, such as when a mobile robot maneuvers left-forward but not backward-forward. Similarly, Inoue et al. [92] devises a set of four or five force-controlled actions for the search and insertion phases of high-precision peg-in-hole tasks. In the same task, Gullapalli et al. [93] has successfully applied policy search to learn hybrid motion/force control policies. Here, hybrid motion/force control actions require a low-level force controller, which maps actions to low-level joint position commands and ensures stable interactions between the robot manipulator and its environment.

Another approach is to develop more intelligent actions. These actions comprise sequences of low-level robot commands that directly achieve specific tasks. This approach is closely related to the concept of *temporal abstraction*, which refers to the creation of high-level abstractions to aid reasoning and planning in robotics. For example, Kalmár et al. [12] propose decomposing a task into sub-tasks and

designing modules to accomplish these sub-tasks. The modules include a closed-loop controller and its operating condition. In a vision-based pick-and-place task, sub-tasks may include "finding the object," where the robot moves until the object is in the camera's field of view, or "grasping the object." The RL agent's objective is to learn sequences of modules to fulfill the primary task. The underlying idea is that sub-tasks are key steps toward accomplishing the main task and are often much easier to solve. As only a few sub-tasks are necessary for a given task, this approach maintains a low-dimensional action space, thus enhancing learning efficiency. With a low-dimensional action space, classical RL algorithms like Q-learning or dynamic programming can be employed to address the vision-based pick-and-place task. Building on a similar principle, [94] successfully uses Q-learning to teach turning gaits for a quadruped. Taking a slightly different perspective, [95] represent a task as a "fixed" sequence of dynamic movement primitives (DMPs), using a policy search variation to learn DMP parameters for a pouring task with a humanoid robot. Other applications of temporal abstraction in robot RL include navigation [96], quadrupedal locomotion [97], and mobile robot manipulation [98, 99].

Following the advent of Deep Reinforcement Learning (DRL), research has shifted towards learning within high-dimensional, continuous action spaces. Initial DRL studies showcased its capability to map high-dimensional pixel observations to low-level actuation commands for robot manipulation [8]. Subsequent research successfully employed DRL for robot manipulation across various action spaces, encompassing joint angles [100, 101, 102, 7], joint velocities [10, 102, 103], end-effector pose [104, 11, 105], end-effector force/torque [106, 107], or even joint impedance or task impedance [108, 109].

Several studies have evaluated various action spaces in the context of locomotion [9] and robot manipulation [110, 109]. A common finding is that action spaces integrating local feedback control can enhance the efficiency and policy performance of RL algorithms. This phenomenon can be attributed to the fact that representing actions as low-level commands might hinder RL from directly optimizing primary objectives [111]. RL might need to implicitly re-learn relationships between low-level commands and high-level task-space behaviors, like gravity compensation and forward kinematics. Furthermore, RL with low-level robot commands lacks the interface to integrate numerous successful model-based controllers, such as whole-body control techniques [112] in locomotion or operational space control [113],

impedance control [114], or direct force control [13] for robot manipulation.

Similar reasoning applies to task-space actions and joint-space actions. As numerous task objectives can be effectively represented in the task space, joint space policies might necessitate re-learning forward and inverse kinematics, even though solutions are readily accessible. n contrast, exploring the task space directly affects primary control objectives and agent behaviors, making it more efficient. Empirical evidence from various studies supports the hypothesis that task-space actions are a favorable choice for robot manipulation [109], [110].

Recent studies have shifted their attention towards integrating temporal abstraction into DRL to improve its sample efficiency. Chitnis et al. [115] suggest learning task schemas comprising skills like grasping, goal-reaching, and lifting to tackle more complex bimanual manipulation tasks. However, their state-independent task schema encounters difficulties with complex manipulation tasks like nut assembly and peg insertion [116]. Sharma et al. [117] define multiple controllers operating on the task frame's axes. The RL agent learns to combine these controllers concurrently to achieve a broader range of behaviors. However, this approach does not consider temporally-extended actions since the controllers are executed for a fixed number of steps. Dalal et al. [118] propose temporally-extended action primitives by associating each action primitive with a set of goal states. This method is comprehensively evaluated across various simulated robot manipulation tasks. However, it remains uncertain whether this approach extends to real-world scenarios. In contrast to the aforementioned works, this thesis postulates that manipulation primitives offer an appropriate level of abstraction for robot manipulation and focuses on. In addition, this thesis focuses on solving high-precision assembly tasks within real-world settings.

2.2.3 Low-level control

As mentioned in the previous section, incorporating low-level feedback in RL can improve sample efficiency and policy performance. These low-level controllers possess a set of adjustable parameters, commonly fine-tuned to meet specific performance requirements such as stability and robustness. The significance of parameter tuning for low-level controllers has been briefly mentioned by Peng and van de Panne [9] and Beltran-Hernandez et al. [107]. In light of this, several studies have

proposed learning state-dependent controller parameters by treating them as RL actions [108, 109, 119, 107]. However, learning state-dependent controller parameters may potentially compromise the stability or performance of these controllers.

Numerous implementations of low-level controllers are available within the vast literature on control theory and robot control. These implementations exhibit diverse characteristics concerning stability, robustness, and performance. For instance, Roy and Whitcomb [13] outline four approaches to force control for position-controlled robots. Nakanishi et al. [14] document eight implementations for the task space control of redundant robot manipulators. How the choice of controller implementation impacts RL is still an open question.

Somewhat related to the above matter, Beltran-Hernandez et al. [107] conducted a comparison of two force control methods for learning assembly tasks: a parallel position/force controller with a Proportional-Integral force control loop and an admittance controller. Their findings indicate that the admittance controller performs slightly better and leads to fewer collisions. However, while both the parallel position/force controller and admittance control aim to stabilize the robot-environment interaction, they differ in control objectives: the former directly regulates the external force on the robot end-effector, while the latter seeks to govern the relationship between external force and end-effector position. Hence, a direct comparison between these two controllers is not feasible. In contrast to Beltran-Hernandez et al. [107], this thesis contributes to this problem through an experimental study. The study examines two implementations of direct force control methods and compares the performance of the policies associated with each of these implementations.

2.2.4 Sim-to-real methods

Collecting the necessary data for deep reinforcement learning (DRL) algorithms from a robot can be resource-intensive, hindering DRL from scaling up. Alternatively, data can be generated faster, cheaper, and safer in simulation. This advantage motivates the sim-to-real methodology: the policy is first learned with data generated in simulation and then transferred to the physical robot. The main challenge of sim-to-real transfer is overcoming the *reality gap*, arising from several

factors. These include the unmodeled physical phenomena, errors in parameter estimation, or the inherent limitations of discretized numerical integration methods utilized in solvers.

Approaches to bridge the reality gap can be divided into two categories: modeling methods focused on enhancing simulation accuracy and algorithms designed to learn robust policies that perform well in real-world applications despite the reality gap. This section focuses on the latter approach, while the former will be reviewed in Section 2.2.5.

2.2.4.1 Learning robust policy through domain randomization

Domain randomization is a technique characterized by the introduction of perturbations into the learning process. These perturbations can impact the simulator's parameters, observations of the system's state, or the actions executed by RL policies. The simplest approach to domain randomization involves sampling a set of domain parameters from pre-defined distribution at the start of each RL episode. Uniform and normal distributions are two commonly used distributions. The distribution's mean corresponds to default parameters, which can be provided by the manufacturers (e.g., link masses in a robot), measured directly (e.g., mass and size of objects), or derived from system identification (e.g., joint damping, friction coefficients, control delay). On the other hand, the standard deviation of the domain parameter distribution is an important design decision.

Lowrey et al. [120] demonstrate this approach on a manipulation task whose goal is to control three Phantom robots to push an object to various desired positions. They show that the policy learned by varying the object's mass robustly transfers to the physical robot despite parametric errors. In contrast, a policy learned under the presence of parametric error fails to perform the task. Peng et al. [121] performs domain randomization on a larger scale with 95 randomized parameters influencing the system dynamics. Similar to Lowrey et al. [120], they show that the policy learned with domain randomization outperforms the policy learned with the nominal parameters on a pushing task with a robot manipulator. Other studies have demonstrated the efficacy of domain randomization on various tasks. These include quadrupedal locomotion [5], [122], bipedal locomotion [123], control of unmanned aerial vehicles [124], [125], [126], deformable object manipulation [127],

in-hand manipulation [7]. By randomizing visual-related features, such as camera position, lighting condition, or textures, domain randomization also enables the sim-to-real transfer of vision-based policies [125], [128], [127], [7], [126]. Despite impressive results, this approach necessitates the manual design of the domain parameter distribution. A narrow distribution could lead to policy failure in the target domain. In contrast, an overly broad distribution might yield a conservative policy, resulting in inferior performance compared to a policy directly trained on the target domain.

An approach to address this issue is adaptive domain randomization (ADR), which introduces domain parameter distribution adjustments between policy updates. The adaptation of domain parameter distribution is usually formulated as an optimization problem. Methods for ADR differ in the objective function and solution algorithm for this optimization problem. Chebotar et al. [129] propose to minimize the discrepancy between the real-world and simulated observations obtained by the current policy. This problem is treated as an RL problem and solved by a policy search algorithm. Mehta et al. [130] also employs a policy search algorithm to update the distribution toward "hard" environments where the current policy behaves differently from a reference environment. Leveraging Bayesian Optimization, Muratore et al. [131] update the randomized parameter distribution to maximize policy performance in the target domain. Mozifian et al. [132] takes a different perspective by updating the parameter distribution to balance conservativeness and robustness. However, the real-world applicability of this method is uncertain, as it doesn't consider real-world performance explicitly.

Another viewpoint treats adaptive domain randomization as an inference problem. This inference problem aims to find the posterior distribution over the domain parameters that best explain real-world observations. Ramos et al. [133] propose a method called BayesSim which approximates this posterior distribution with Mixture Density Network learned with Likelihood-Free Inference. They show that the posterior can serve as the domain parameter distribution in domain randomization. Possas et al. [134] extend this work with Online BayesSim. They show that Online BayesSim can be integrated into Adaptive Domain Randomization by iteratively optimizing the policy based on the current posterior distribution and updating the posterior using data collected by the current policy.

Another approach to address the issue of conservative performance is to learn domain-parameter-conditioned policies. This process can be viewed as acquiring multiple strategies in simulation and selecting the optimal one for real-world performance. Optimization is often achieved using gradient-free methods like CMA-ES [135], Bayesian Optimization [136], or reinforcement learning [137]. Alternatively, Yu et al. [138] train an NN to estimate domain parameters from historical observations and actions. During deployment, this NN provides the parameter estimations as inputs to the policy.

2.2.4.2 Multi-task learning and meta reinforcement learning

In a multi-task setting, simulated and real tasks can be seen as variations or related tasks. Different tasks may share state space, action space, and reward function but have different dynamics. A policy trained in simulation may be a good starting point for learning in reality, avoiding training from scratch. Therefore, the policy can be fine-tuned by additional training with real data. Alternatively, Rusu et al. [139] propose using the Progressive Neural Network (PNN) [140], which learns sequential tasks without forgetting prior knowledge. The authors demonstrated that PNN outperforms traditional fine-tuning in a robot manipulator's reaching task.

A more principled approach in the multi-task setting is meta reinforcement learning which aims to learn a policy that can quickly adapt to a new task. This is the main difference between meta RL and domain randomization, which focuses on learning a policy that performs well within the considered domains. Consequently, domain-randomized policies exhibit better initial performance, while meta-RL policies adapt over time, ultimately achieving higher performance [141]. Meta-RL policies also demonstrate better generalization capability. Examples include adapting to missing legs or novel terrain in quadrupedal locomotion [142] or solving new assembly tasks given a few human demonstrations [143].

2.2.4.3 Aligning state trajectories

This approach attempts to align the source and target domains as closely as possible. The medium for alignment is often the dynamics model: state transitions

resulting from state-action pairs should be similar in both domains. A direct consequence is that a policy should yield the same state trajectories given an initial state. Based on this idea, [144] propose learning an inverse dynamics network that predicts an action resulting in a state transition. During real robot deployment, policy actions are adjusted to yield matching state transitions as in simulation. Conversely, [145] integrate action transformation during training in simulation. State transformation has also been considered [146]. Unlike the above studies, Wulfmeier et al. [147] achieve alignment of state trajectories by adding an auxiliary reward. Since most of the above methods involve training an additional dynamics model, discrepancies between testing and training distribution may cause policy failure. Additionally, these methods overlook the hybrid nature of contact dynamics, which may result in large prediction errors.

2.2.5 Contact simulation methods

A simulator must be computationally efficient for applying Reinforcement Learning and be physically accurate to transfer the RL policy to the real world. Efficient simulators minimize the computational cost of RL, as samples are generated using the dynamics model implemented in the simulator. Accurate simulations decrease the reality gap for transferring RL policies. Despite advancements in physics simulation, challenges persist in modeling and simulating dynamics contact between rigid or soft bodies, often necessitating approximations and costly computation.

A typical rigid body contact simulation pipeline consists of three main steps

- Collision detection assesses if two bodies are overlapping.
- Contact generation finds the contact region, commonly represented as a finite set of contact points with position, contact normal, and penetration depth.
- Contact response finds the motion of rigid bodies. The key component in this step is a contact model, which computes contact forces to prevent interpenetration between contacting bodies.

The contact response phase has two main approaches: event-driven and time-stepping methods [148]. Event-driven approaches detect the collision/breaking

time of each contact point. When a collision/breaking is detected, event-driven pipelines solve a complementarity problem and use its solution as the initial condition for the next event. In contrast, time-stepping methods treat all constraints within a time interval Δt as if they happen simultaneously. Event-driven methods are more accurate but also more computationally expensive. The latter problem typically accounts for more weights in robotics which can experience a high frequency of contact breaking/making. For this reason, time-stepping methods are the choice in most established robotics simulators.

Contact modeling relies on three fundamental principles: the Signorini condition (or normal complementarity), Coulomb's law of friction, and the maximum dissipation principle. The Signorini condition imposes unilateral constraints on contact forces and motion along the normal direction at each contact point. The other two principles enforce constraints on friction forces to prevent slip. Mathematically, these principles define a Nonlinear Complementarity Problem (NCP).

The solution technique for this NCP is a distinguished feature of modern robotics simulators. Most techniques involve various approximations or relaxations to the NCP to enhance tractability or efficiency. However, these approximations may adversely impact the physical accuracy. Notable approximations include (1) linearizing the friction constraints to obtain a Linear Complementarity Problem and (2) transforming the NCP to a convex optimization problem. The first category includes Bullet [149], PhysX [150], ODE [151], and DART [152], while the second category comprises Mujoco [153] and RaiSim [154]. Approaches to directly solve the NCP have also been proposed [155]. Detailed analysis and performance evaluations of these approaches can be found in the literature [156, 157]. A common conclusion is that no single approach fully satisfies all criteria, as each contact model sacrifices either accuracy, robustness, or efficiency.

Besides contact modeling, object geometry is often simplified using convex hulls or basic shapes. As a resolution, generic geometric representations (e.g., convex decomposition, triangular meshes, signed distance fields) can offer fine approximations for complex contact surfaces. However, they can lead to excessive contact points, slowing simulations and potentially causing instability. To tackle this issue, contact reduction methods have been proposed to limit the number of contact points [16], [158], [17]. These methods cluster similar contacts based on some metrics and select representative contacts for each group. Common clustering

algorithms are k-means clustering and hierarchical clustering. Various distance metrics have been proposed, including the distance between contact points and normal similarity. Narang et al. [17] iteratively form contact patches and assign the remaining contact points to each patch based on normal similarity. These methods boost simulation speed and stability, but their impact on simulation accuracy is often overlooked. On the contrary, this thesis proposes a novel contact reduction method considering physical accuracy. The proposed method enables successful policy transfer on high-precision assembly tasks, while the clustering approach fails to do so. Kim et al. [159] address the same problem using a data-driven approach that minimizes the difference between real and simulated contact forces. However, this method requires a data collection and learning phase for new scenarios. In addition, while Kim et al. [159] validate their method in surface-surface and edge-surface contacts, this thesis focuses on more complex industrial assembly tasks.

Chapter 3

Learning Sequences of Manipulation Primitives for Robot Assembly

3.1 Introduction

This chapter explores the idea that skillful assembly is best represented as dynamic sequences of Manipulation Primitives, and that such sequences can be automatically discovered by Reinforcement Learning.

In recent years, increasingly complex assembly tasks have been demonstrated on robot systems [160]. However, in most cases, the difficult assembly skills, such as tight pin insertion or part mating, are still accomplished by hand-designed, hard-coded, strategies (e.g. spiral search followed by force-controlled insertion) [1]. Designing and fine-tuning such strategies require considerable engineering expertise and time, thus putting a brake on the deployment of intelligent robotic manipulation in the factories and in the homes. This chapter investigates how to automatically discover *in silico* assembly strategies that robustly transfer to physical robots.

The first, crucial, question is the representation of the assembly skills: what is the set of atomic actions to be reasoned upon? In [92], the authors consider very

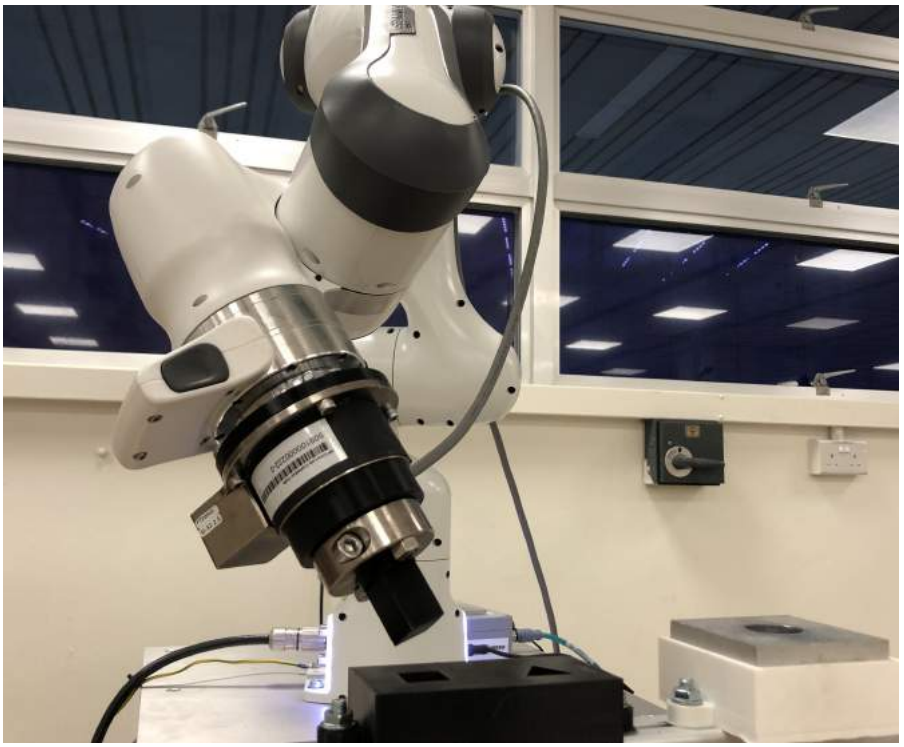


FIGURE 3.1: Robotic assembly setup. The video of the experiments is available at https://youtu.be/P0NNjjQNOVo

simple atomic actions such as pure force-controlled translations or pure position-controlled rotations. This results in extremely long sequences of atomic actions to achieve a given task, making the search complexity overwhelming.

Consider how a robot would learn to play chess. One option is to learn directly the sequences of robot commands to physically move the pieces throughout the full game. Alternatively, it would be much more efficient to learn the sequences of piece moves (e.g. 1. e4, 2. Nf3, 3. Bb5...), and then rely on grasp planning, inverse kinematics, inverse dynamics, etc. to physically realize the moves.

Here, we propose, by analogy, to consider Manipulation Primitives (MP) [36] as the atomic actions. Manipulation Primitives, such as "Move down until contact", "Slide along x while maintaining contact with the surface", have enough complexity to keep the search tree shallow (typically a sequence of 6 to 8 MPs is enough to achieve tight insertion), yet are generic enough to generalize across a wide range of assembly tasks (peg insertion with different peg shapes, large hole estimation errors, random initial positions...). Another key advantage of MPs is their additional *semantics*, which make them robust in sim2real and against model/environment

variations and uncertainties: consider how "Move down until contact" is inherently more robust than a sequence of several short "Move down" actions.

Contribution: learning dynamic sequences of Manipulation Primitives

In [36], the authors consider a set of MPs with tunable parameters, the parameters being optimized through task execution on the physical platform. However, the *temporal sequence* of MPs to accomplish a given task is manually designed and fixed, which re-raises the initial concern about expertise and time required to address new tasks.

By contrast, we propose here to *automatically discover dynamic sequences* of MPs by Reinforcement Learning (RL). Particularly, an MP and possibly its parameters are decided by the RL agent; the MP then executes until the stopping condition is met. This approach allows to easily incorporate domain knowledge by constructing a set of only relevant MPs. We consider two approaches to improve the versatility of the MP sets: discretization over the MP parameters or learning MP parameters by RL. The former approach splits the dimensions of the MP parameters into a number of regions. However, this might greatly increase the dimensionality of the action space. Alternatively, the MP parameters can be learned by RL. In this *hybrid* approach, the action space involves a mixture of a discrete space over possible MPs and continuous spaces of MPs' parameters. We show that both approaches significantly improve sample efficiency of RL on three high-precision peg-in-hole tasks with different peg profiles. Direct sim2real transfer (without retraining in real) achieves 100% and 95% success rate on round peg insertion with respectively 0.1 mm and 0.04 mm clearance, and despite 1 mm and 1 deg errors in hole position/orientation estimation. On harder tasks with rectangular and triangular profile, direct sim2real still achieves promising results.

The rest of the chapter is organized as follows. In Section 3.2, we formally define the concept of Manipulation Primitives. In Section 3.3, we introduce in detail the proposed Reinforcement Learning formulation. Section 3.4 presents the experimental setup and quantitative results. Finally, in Section 3.5, we discuss the advantages and limitations of the presented approach, as well as some directions for future work.

3.2 Manipulation Primitives

3.2.1 Definition

We follow [36] to define manipulation primitives (MPs). An MP represents a desired motion of the robot end-effector (E) in the task frame (T). More precisely, it consists of:

- a desired six-dimensional velocity command ${}^{T}\mathbf{v}_{E}$ (in short $\mathbf{v}_{\text{des}}$);
- a desired six-dimensional force command ${}^{T}\mathbf{f}_{E}$ (in short $\mathbf{f}_{\text{des}}$);
- a stopping condition λ.

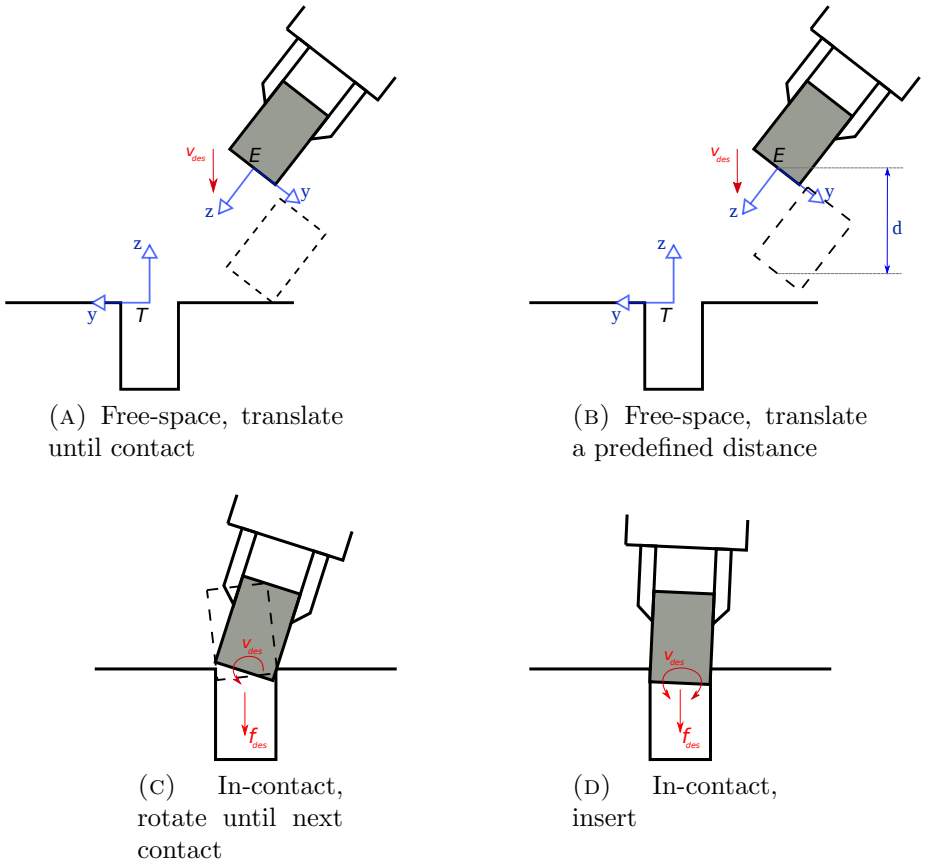
The desired velocity and force commands are defined as

$$\begin{aligned} \mathbf{v}_{\text{des}}(t) &:= g_v(t, \boldsymbol{\Omega}_t; \boldsymbol{\theta}_v), \\ \mathbf{f}_{\text{des}}(t) &:= g_f(t, \boldsymbol{\Omega}_t; \boldsymbol{\theta}_f), \end{aligned} \tag{3.1}$$

where g_v and g_f are any functions parameterized respectively by $\boldsymbol{\theta}_v$ and $\boldsymbol{\theta}_f$, and $\boldsymbol{\Omega}_t$ is the vector of all sensor signals at time t (e.g., force/torque reading, joint position). The stopping condition is defined as $\lambda : (t, \boldsymbol{\Omega}_t) \mapsto \{\texttt{SUCCESS}, \texttt{FAILURE}, \texttt{CONTINUE}\}$. An MP terminates when λ returns either `SUCCESS` or `FAILURE`. In summary, an MP is fully defined by the functions g_v, g_f, λ, and the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}_v, \boldsymbol{\theta}_f, \boldsymbol{\theta}_s]$.

Executing an MP requires the control of desired end-effector velocity/position and force, which can be realized by the hybrid motion/force control scheme. In this scheme, the desired velocity and desired force is achieved by a position control loop and a force control loop, respectively. In this chapter, we use the inverse dynamics control in the operational space approach [113] for the position control loop and a simple feedforward force controller for the force control loop.

The next section instantiates our definition in the context of peg-in-hole insertion tasks and clarifies the motivations.



(A) Free-space, translate until contact

(B) Free-space, translate a predefined distance

(C) In-contact, rotate until next contact

(D) In-contact, insert

FIGURE 3.2: Examples of Manipulation Primitives for insertion task. See text for details.

3.2.2 MPs for peg-in-hole insertion tasks

For insertion tasks, we consider two families of MPs: free-space MPs and in-contact MPs:

- Free-space MPs are to be executed when the robot is not in contact with the environment, i.e., when all external forces/torques are zero. MPs in this family are then associated with zero desired force/torque command.
- In-contact MPs are to be executed when the robot is in contact with the environment, i.e. when some external force/torque components are non-zero. In addition to other objectives, MPs in this family have some components of their desired force/torque command to be non-zero in order to maintain the same contact state during the execution.

Each family of MPs are subdivided into several types: (i) move until (next) contact, (ii) move a predefined amount, (iii) insert. Figure 3.2 illustrates some examples of MPs, which are further detailed as follows

Free-space, move until contact. Translate the end-effector along a direction, or rotate the end-effector about a direction, until contact is detected. The example of (Fig. 3.2a) translates the end-effector in the $-z$ direction with speed v until the measured force is larger than f_{thr} (`SUCCESS`), or $t > 2s$ (`FAILURE`), which is formally defined by

$$\begin{aligned} \mathbf{v}_{\text{des}}(t) &= [0, 0, -v, 0, 0, 0] \\ \mathbf{f}_{\text{des}}(t) &= \mathbf{0} \\ \lambda(t) &= \begin{cases} \texttt{SUCCESS} & \text{if } \mathbf{f}_{\text{ext}}^T \mathbf{u}_v > f_{\text{thr}}, \\ \texttt{FAILURE} & \text{if } t > 2, \\ \texttt{CONTINUE} & \text{otherwise.} \end{cases} \end{aligned} \tag{3.2}$$

where $\mathbf{f}_{\text{ext}}$ is the measured external force, $\mathbf{u}_v = \mathbf{v}_{\text{des}}/||\mathbf{v}_{\text{des}}||$ is the moving direction.

Free-space, move a predefined amount. Translate the end-effector along a direction over a predefined distance d, or rotate about a direction over a predefined angle α. The example of (Fig. 3.2b) translates the end-effector in the $-z$ direction with speed v, until the distance d is reached (`SUCCESS`), or a large contact force is detected (`FAILURE`), which is formally defined by

$$\begin{aligned} \mathbf{v}_{\text{des}}(t) &= [0, 0, -v, 0, 0, 0] \\ \mathbf{f}_{\text{des}}(t) &= \mathbf{0} \\ \lambda(t) &= \begin{cases} \texttt{SUCCESS} & \text{if } \Delta\mathbf{p}^T \mathbf{u}_v > d \\ \texttt{FAILURE} & \text{if } \mathbf{f}_s^T \mathbf{u}_v > f_{\text{thr}} \\ \texttt{CONTINUE} & \text{otherwise} \end{cases} \end{aligned} \tag{3.3}$$

where $\Delta\mathbf{p}$ is the distance between the current pose $\mathbf{p}$ and the start pose $\mathbf{p}_0$.

In-contact, move until next contact. Track a non-zero force in a some directions, and translate the end-effector along a direction, or rotate the end-effector about a direction until next contact is detected. The example of Fig 3.2c control a force f_d in the $-z$ direction and rotate the peg around the x direction with speed v,

until the measured force is larger than f_{thr} (`SUCCESS`), or $t > 2s$ (`FAILURE`), which is formally defined as

$$\begin{aligned}
\mathbf{v}_{\text{des}}(t) &= [0, 0, 0, 0, v, 0] \\
\mathbf{f}_{\text{des}}(t) &= [0, 0, -f_d, 0, 0, 0] \\
\lambda(t) &= \begin{cases} \texttt{SUCCESS} & \text{if } \mathbf{f}_{\text{ext}}^T \mathbf{u}_v > f_{\text{thr}}, \\ \texttt{FAILURE} & \text{if } t > 2, \\ \texttt{CONTINUE} & \text{otherwise.} \end{cases}
\end{aligned} \tag{3.4}$$

In-contact, insert. Track a non-zero force in the direction of insertion and regulate the forces and torques to zero in all the other directions. The example of (Fig. 3.2d) performs the insertion in the z direction, as formally defined by

$$\begin{aligned}
\mathbf{v}_{\text{des}}(t) &= -K_d \mathbf{f}_{\text{ext}} \\
\mathbf{f}_{\text{des}}(t) &= [0, 0, f_d, 0, 0, 0] \\
\lambda(t) &= \begin{cases} \texttt{SUCCESS} & \text{if } d(\mathbf{p}, \mathbf{p}_t) < \epsilon \\ \texttt{FAILURE} & \text{if } t > 2 \\ \texttt{CONTINUE} & \text{otherwise} \end{cases}
\end{aligned} \tag{3.5}$$

where K_d is a compliant 6×6 diagonal matrix, $d(.)$ is a metric measuring distance between two poses, $\mathbf{p}_g$ is the goal pose. For simplicity, we consider a diagonal compliant matrix of the form $K_d = \begin{bmatrix} k_{dt}\mathbb{I}^{3\times3} & \mathbf{0} \\ \mathbf{0} & k_{dr}\mathbb{I}^{3\times3} \end{bmatrix}$.

3.3 Learning Dynamic Sequences of Manipulation Primitives by RL

3.3.1 Reinforcement learning with parameterized action space

We consider here the discounted episodic RL problem. In this setting, the problem is described as a Markov Decision Process (MDP) [161]. At each time step t, the agent observes current state $\mathbf{s}_t \in \mathcal{S}$, executes an action $\mathbf{a}_t \in \mathcal{A}$, and receives

an immediate reward r_t. The environment evolves through the state transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The goal in RL is to learn a policy $\mathbf{a}_t = \pi(\mathbf{s}_t)$ that maximizes the expected discounted return $R = \sum_{t=1}^{T} \gamma^t r_t$, where γ is the discount factor that tends to emphasize the importance of most recent rewards.

A parameterized action space consists of a finite set $\mathcal{A}_d = \{a_1, a_2 \dots a_n\}$ with cardinality n and n sets $\mathcal{X}_i,\ i = \overline{1, n}$. An action is a tuple $(a_i, \mathbf{x})$, where $a_i \in \mathcal{A}_d$ and $\mathbf{x} \in \mathcal{X}_i$. Problems with parameterized action space can be formally defined as parameterized action MDP (PAMDP) [162]. In a PAMDP, the RL agent makes a decision in two steps: it first selects a_i and then choose the continuous variable $\mathbf{x}$. One can think of the variable $\mathbf{x}$ as some parameters of the discrete action a_i. For example, consider the manipulation primitive defined in 3.2.2, $\mathbf{x}$ can be the parameters of an MP. Finally, the policy can be written as

$$\pi(a_{i,t}, \mathbf{x}_t|\mathbf{s}_t) = \pi^d(a_{i,t}|\mathbf{s}_t)\pi_i^c(\mathbf{x}_t|\mathbf{s}_t) \tag{3.6}$$

where $\pi^d(a_i|\mathbf{s}_t)$ is denoted as the discrete-action policy and $\pi_i^c(\mathbf{x}|\mathbf{s}_t)$ is denoted as the action-parameter policy

3.3.2 Manipulation primitives as atomic actions

The problem of finding a sequence of MPs can be formulated as an RL problem by considering MPs as atomic actions, i.e. at each time step, the RL agent selects an MP $a_i \in \mathcal{A}$, where $\mathcal{A}$ is a predefined set of MPs. The MPs' parameters are also crucial for the success of the RL agent and for increasing the versatility of the method. We consider two approaches to incorporate the MPs' parameters: discretization over the parameters and hybrid approach. Discretization splits each component of the MPs' parameters into regions and include corresponding MP. This approach is simple but can greatly increase the dimensionality of the action space.

Alternatively, in the hybrid approach, the MPs' parameters can be learned by RL. At each time step, after selecting an MP a_i, the agent also decides on the MP parameters $\mathbf{x}_i$. This problem can be formulated as a PAMDP as described in Section 3.3.1, in which an MP is chosen according to the discrete-action policy and the MP parameters are then chosen according to the action-parameter policy.

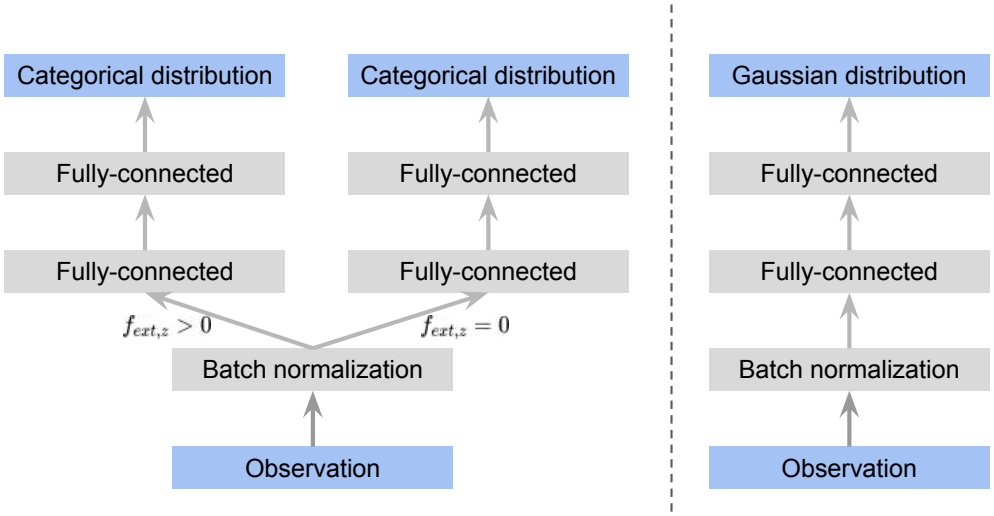


FIGURE 3.3: Discrete-action policy network (left of dash line) and action-parameter policy network (right). The networks share a layer of batch normalization. The discrete-action policy contains two separate networks, one for each action subspace.

3.3.3 Learning dynamic sequence of manipulation primitives for robot assembly

State and action. The state is defined by $\mathbf{s}_t = [\mathbf{p}_t, \mathbf{f}_{ext,t}]$, where $\mathbf{p}_t$ is the pose of the peg relative to the hole (position and orientation of frame E with respect to frame T in Fig. 3.2a); $\mathbf{f}_{ext,t}$ is the external force and torque acting on the end-effector. The action space is described in Section 3.3.2. Additionally, note that at a particular state, not all MPs are feasible. For instance, in-contact MPs shouldn't be executed while the robot is free space. Therefore, we propose state-dependent action space as follows. Denote $\mathcal{A}_{\text{free}}$ the set of free-space MPs and $\mathcal{A}_{\text{con}}$ the set of in-contact MPs, the set of feasible actions at each state is either $\mathcal{A}_{\text{free}}$ if $f_{\text{ext,z}} = \mathbf{0}$, or $\mathcal{A}_{\text{con}}$ if $f_{\text{ext,z}} \neq \mathbf{0}$. The episode terminates when the number of MPs exceed 15 or the task is successful. In the latter case, the RL agent receives an additional termination reward of 5. The task is considered successful if the end-effector reach the goal position within 2 mm.

Reward function. We define the reward function as

$$r(\mathbf{o}_t, \mathbf{o}_{t+1}, \mathbf{a}_t) := c_1 \left(e^{\frac{-||\mathbf{p}_{t+1} - \mathbf{p}_{goal}||_2^2}{k_1}} - 1 \right) - c_2 t(\mathbf{a}_t) + c_3 s(\mathbf{a}_t) \tag{3.7}$$

The first term rewards for moving closer to the goal, the second term is the execution time of the MP, to bias the algorithm to find solution with short execution time. The third term $s(\mathbf{a}_t) = 0$ if a `SUCCESS` status is returned, $s(\mathbf{a}_t) = -1$ if a `FAILURE` status is returned.

Algorithm and policy parameterization. The policy representation is shown in Fig 3.3. The discrete-action policy is represented by two Neural Networks whose outputs are logits of a categorical distribution over the free space MPs and in-contact MPs respectively. The action-parameter policy is a multi-head neural network; each head outputs the mean of a Gaussian distribution over the parameters of an MP. The policy is trained with Proximal Policy Optimization (PPO) [87]. In addition to the policy network, PPO requires a value network to approximate the value function. The value network has similar network architecture to the action-parameter networks and has independent weights.

3.4 Experiments

We conduct experiments to (1) validate the proposed method in simulation on various peg insertion with different shape (2) evaluate the sim-to-real performance of the learned policy.

3.4.1 Experimental setups

Task description. We evaluate the proposed method on tight-clearance peg-in-hole tasks with three types of peg profiles: round shape, square shape, and triangular shape. The properties of the pegs are shown in Table 3.1. The following assumptions are made for the assembly task:

- The peg is firmly grasped or rigidly attached to the end-effector, the hole is rigidly mounted in the environment.
- The hole position (defined as the point T in Fig. 3.2) and axis of insertion can be estimated with position error less than 1 mm and orientation error less than 1 degree. This can be achieved by, for example, using visual serving technique [163].

TABLE 3.1: Dimensions and material of pegs and holes. Size is diameter for round profile and side length for square and triangle ones

Profile	Hole size (mm)	Peg size (mm)	Material
Round	30.03	29.96	Aluminum
Round	30.03	29.9	Aluminum
Square	19.98	19.96	Aluminum
Square	19.98	19.72	Plastic
Triangle	25	24.9	Aluminum
Triangle	25	24.2	Plastic

- The task frame is chosen such that its origin coincides with the estimated hole position and the direction of insertion is along the -z axis. This assumption simplifies the design of MPs without the loss of generality.

Robot system setup. A 7-DOF Franka Emika Panda robot is used in our experiment. We additionally attach a Gamma IP60 force torque sensor to the flange of the robot to measure the external force and torque acting on the end-effector. The measurement from FT sensor is needed to implement the insert primitive, as we observe that the external torque estimation provided by `libfranka` is not precise enough to perform this motion. Controller is implemented based on Robot Operating System and runs at 1000Hz. Execution of manipulation primitives is implemented based on ROS service-client framework.

Simulation environment We use the Mujoco physics engine [153] and adapt an open-source Panda robot model [1]. The controller is simulated with a control frequency of 500Hz, similar to the simulation step. We use a lower frequency in simulation than that in real world to increase computational speed, while still maintaining a stable simulation. Three simulation environments are created for each of the peg shape shown in Fig. 3.1. The *perception error* is simulated by adding positional and rotational noise to the actual hole pose, this "estimated" hole pose is then used to compute the task frame for manipulation primitives, observation and reward for RL.

RL implementation details. We use `gym` [164] to design the RL environment and `garage` [165], an RL framework based on PyTorch for the implementation of

[1] available online at franka_sim

PPO algorithm. For the policy architecture, each fully-connected layer has 128 nodes in the discrete-action policy network and 24 nodes in the action-parameter policy networks. The hyperparameters for PPO are listed as follows: clip ratio is 0.1, minibatch size is 64, discount factor is 0.99, learning rate is 0.0005, and policy entropy coefficient is 0.001, 2048 samples are collected for each policy update.

At the start of each episode, the robot's end-effector is reset to a random position inside a box in the task space with its center located 10 mm from the task frame's origin. The perception error is varied at the beginning of each episode by sampling the positional uncertainty in $[-1, 1]$ mm uniformly for all axes. To generate random rotational uncertainty, we sample a random unit vector and a random angle in $[-1, 1]$ degree uniformly, which together define an axis-angle representation of the uncertainty rotation.

3.4.2 Learning sequence of MPs with parameters discretization

We use the set of 91 MPs in this experiment as shown in Table 3.2. The method is first validated in Mujoco simulation. The trained policies are then executed directly on the real robot. For each peg-in-hole task, policies are trained on two different Training Conditions:

(TC1) $\Delta\mathbf{p}_{\text{init}}$ is uniformly sampled in $(-1, 1)$ mm for position and in $(-1, 1)$ deg for orientation, $\Delta\mathbf{p}_{\text{hole}} = \mathbf{0}$;

(TC2) $\Delta\mathbf{p}_{\text{init}}$ is uniformly sampled in $(-2, 2)$ mm for position and $(-2, 2)$ deg for orientation, $\Delta\mathbf{p}_{\text{hole}}$ is uniformly sampled in $(-1, 1)$ mm for position and in $(-1, 1)$ deg for orientation,

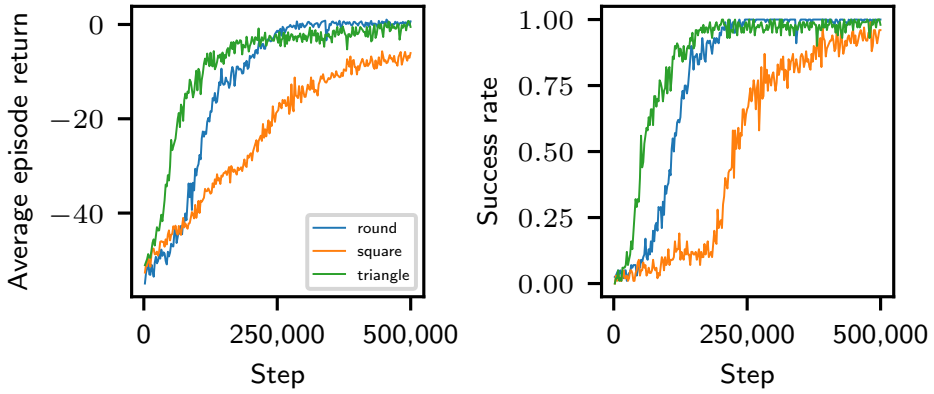
The weights of the policy trained for (TC1) is used to initialize the policy trained with (TC2). We do not train directly on (TC2) due to its difficulty. We also compare our method with a baseline in which the RL policy outputs the desired end-effector pose displacement to a task impedance controller.

TABLE 3.2: The set of 91 Manipulation Primitives used in our experiments.

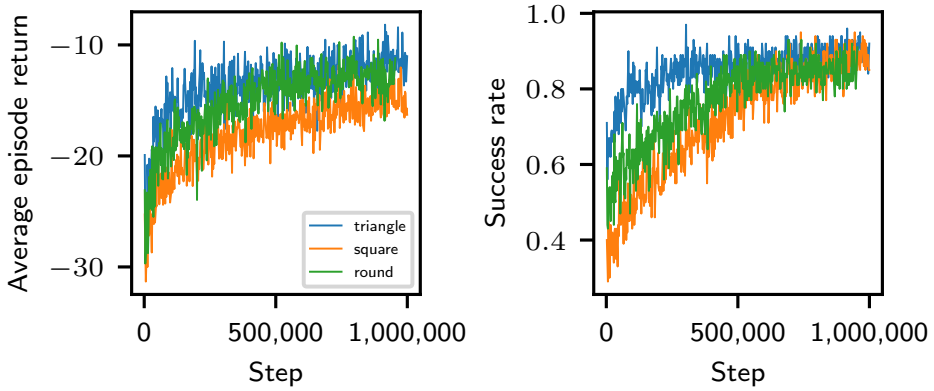
Family	Type	Axis	Parameters	Values	N
Free space	Translate until contact (T^c)	$-z$	v f_{thr}	10 mm/s 8 N	1
	Translate (T)	$\pm x, \pm y, \pm z$	v f_{thr} d	10 mm/s 15 N 2 or 4 mm	12
	Rotate (R)	$\pm x, \pm y, \pm z$	v f_{thr} d	9 deg/s 1 Nm 2 or 4 deg	12
In contact	Translate until next contact (T^c)	$\pm x, \pm y$	v f_{thr} f_d	4 or 7.5 mm/s 8 or 15 N -3 N	16
	Rotate until next contact (R^c)	$\pm x, \pm y, \pm z$	v f_{thr} f_d	4 or 7 deg/s 0.1 or 0.5 Nm -3 N	24
	Translate (T)	$\pm x, \pm y$	v f_{thr} d f_d	10 mm/s 15 N 2 or 4 mm -3 N	8
	Rotate (R)	$\pm x, \pm y, \pm z$	v f_{thr} d f_d	4.6 deg/s 1 Nm 2 or 4 deg -3 N	12
	Insert (I)	$-z$	ϵ k_{dt} k_{dr} f_d	2mm 0.01 0.05 or 0.1 -5 or -12 N	4

3.4.2.1 Simulation results

The training curves of the proposed method in three peg-in-hole tasks are reported in Fig. 3.4 and the training curves of the proposed method and the baseline are contrasted in Fig. 3.5. As can be seen from the figure, the baseline learns significantly slower than the proposed method. This suggests that using MPs improves the exploration at the initial stage of learning, thanks to the more shallow search tree. This advantage is also mentioned in the *option framework* [166].



(A)



(B)

FIGURE 3.4: Training curve (average episode reward and success rate) for (a) Training Condition TC1 and (b) Training Condition TC2.

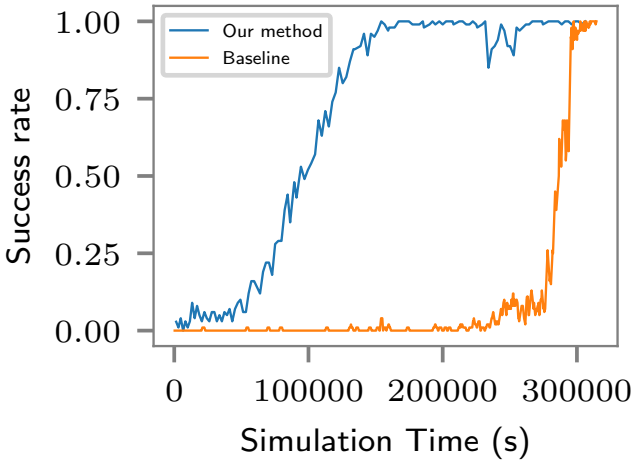
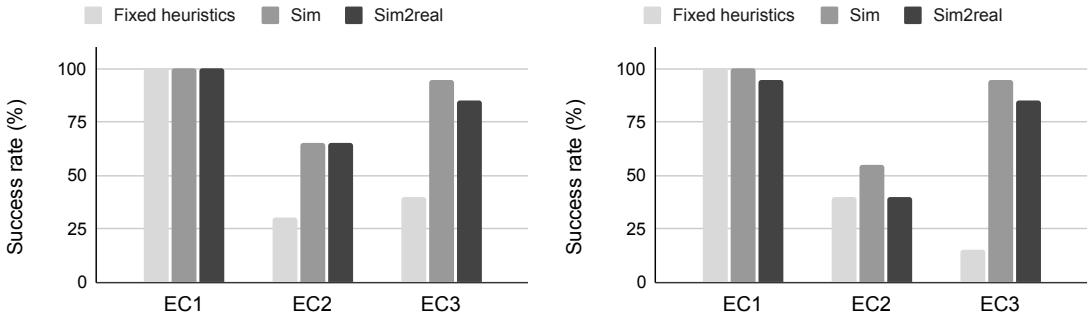
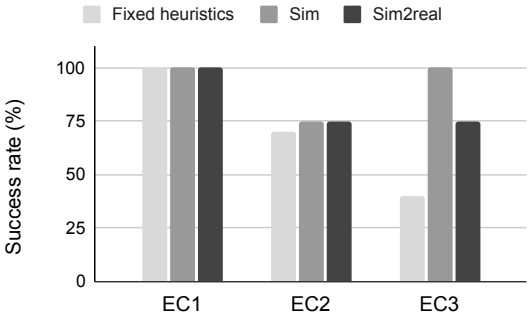


FIGURE 3.5: Comparison of learning performance between proposed method and the baseline.



(A) Round, 0.1 mm clearance

(B) Round, 0.04 mm clearance



(C) Square

FIGURE 3.6: Evaluation on the round and square peg-in-hole task on three Evaluation Conditions.

3.4.2.2 Sim2real policy transfer on physical robot

We evaluate the learned policies directly on the real robot without any further fine-tuning. Three evaluation conditions are considered:

(EC1) Nominal performance: $\Delta\mathbf{p}_{\text{init}} = \mathbf{0}$, $\Delta\mathbf{p}_{\text{hole}}$ is sampled in $(-0.5, 0.5)$ mm for position and in $(-0.5, 0.5)$ deg for orientation;

(EC2) Generalizability: $\Delta\mathbf{p}_{\text{init}} = \mathbf{0}$, $\Delta\mathbf{p}_{\text{hole}}$ is sampled in $(-1.5, 1.5)$ mm for position and in $(-1.5, 1.5)$ deg for orientation;

(EC3) Robustness: $\Delta\mathbf{p}_{\text{init}}$ is sampled in $(-1, 1)$ mm for position and $(-1, 1)$ deg for orientation, $\Delta\mathbf{p}_{\text{hole}}$ is sampled in $(-0.5, 0.5)$ mm for position an in $(-0.5, 0.5)$ deg for orientation

Different from the training phase, the hole estimation error and the initial pose displacement are sampled on the *boundary* of the box around the nominal values.

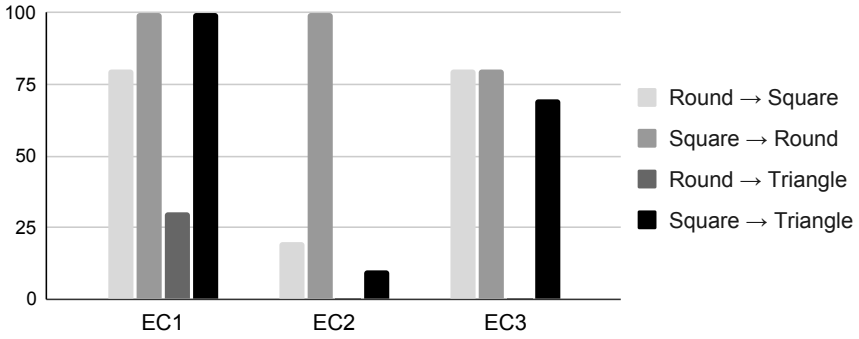


FIGURE 3.7: Results for the policy transfer experiments. $A \rightarrow B$: the policy trained for shape A is evaluated on shape B.

We also compare the transferred RL policy with a manually-defined sequence of MPs and report the result in Fig 3.6. This sequence is tuned for the round peg-in-hole task with an estimation error of 1 mm in translation and 1 deg in orientation. More specifically, the sequence is (1) rotate about $-y$ 5 deg; (2) translate along $-z$ until next contact; (3) translate along x until next contact; (4) rotate about y until next contact; (5) insert. One can see that the manually-defined solution does not generalize well: for both round and square tasks, the success rates for Evaluation Condition (EC2) are significantly lower than our proposed method.

We also run the trained policy on tasks with different shapes from the one the policy was trained for, on $N = 10$ trials. The results are shown in Fig 3.7. The result demonstrates the generalization capability of the trained policy across different geometries of the parts. For instance, the policy trained for square peg-in-hole confidently solves the round peg-in-hole task, even with large estimation error.

3.4.2.3 Dynamic character of the learned policies

We investigate next the emergent behaviors exhibited by the trained policies. All strategies tend to find a correct pose, such that the insert MP could complete the task afterward. To achieve such pose, the most commonly used MP is of rotation type (see Fig. 3.8 and video at https://youtu.be/P0NNjjQNOVo). Rotating motion induces a tilted peg posture. This posture effectively broadens the state spaces in which parts of the peg are inside the hole. Interestingly, for the square peg-in-hole task, the policy follows this strategy by rotating the peg in both x and y directions. After reaching such states, a "translate until contact" often comes next to achieves the locally "optimal" position, where the peg is at the lowest position (refer to two

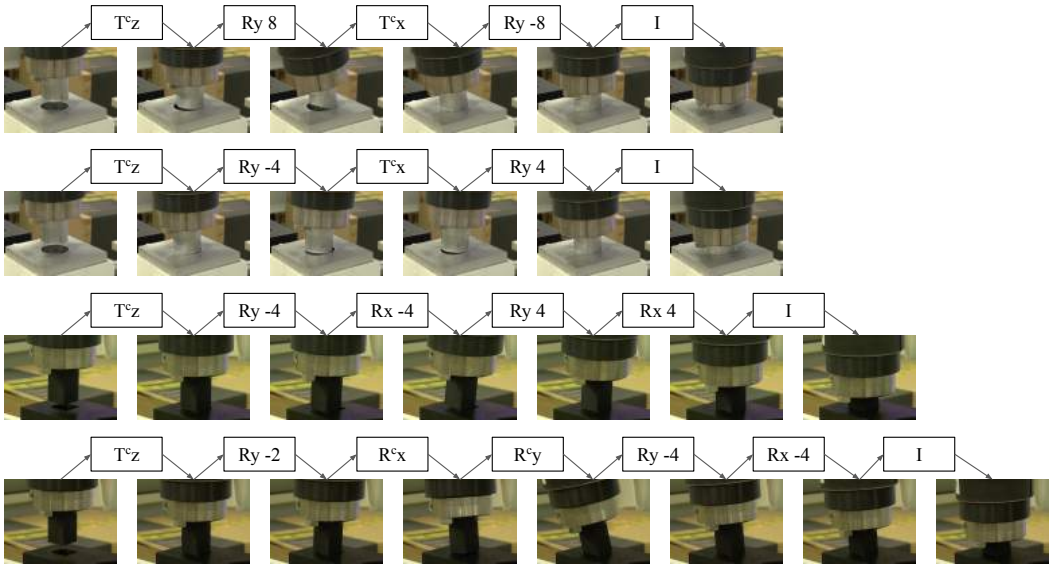


FIGURE 3.8: Snapshots of four runs on the round and square peg-in-hole insertion tasks. "Ry 8" means rotation of 8 deg around y, which is the concatenation of two MPs that rotate 4 deg each. Note the different sequences of MPs for the same task, which illustrates the *dynamic* character of the learned policies. See the full video of these sequences at https://youtu.be/P0NNjjQNOVo

top rows of Fig. 3.8). After that, a rotating motion is regulated to cancel the one in previous steps, before the insertion takes place.

3.4.3 Learning sequence of MPs with hybrid approach

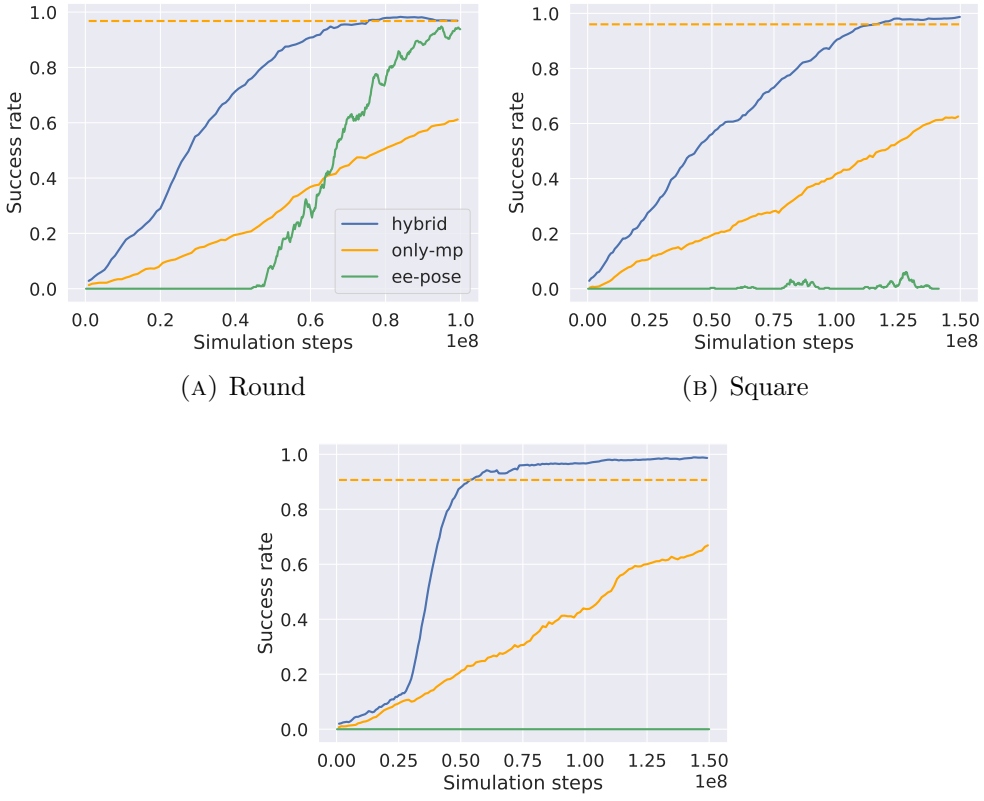
The set of 13 MPs used in the hybrid approach is shown if Fig. 3.3. The number of parameters learned by RL is 14.

Baselines We compare the proposed hybrid approach with three baselines. The first baseline learns in a purely continuous action space. Specifically, the control policy outputs the desired end-effector displacement at a rate of 40 Hz. Since the OSC runs at a higher frequency, the input to the OSC is interpolated between 0 and the desired end-effector displacement during one policy step. We refer to the first baseline `ee-pose`. The second baseline is the proposed discretization approach. We manually pick two values within the range of parameters shown in Fig. 3.3 to form a set of 81 MPs. We refer to this baseline `only-mp` and the hybrid approach `hybrid`.

TABLE 3.3: The set of 13 Manipulation Primitives used in hybrid approach. Ranges correspond to learnable parameters

Family	Type	Axis	Parameters	Value/Range	N
Free space	Translate until contact (T^c)	$-z$	v f_{thr} T	10 mm/s 5 N 2 s	1
	Translate (T)	x, y	v f_{thr} d	10 mm/s 20 N [-10, 10] mm	2
	Rotate (R)	x, y	v f_{thr} d	0.1 rad/s 2 Nm [-0.1, 0.1] rad	2
In contact	Translate until next contact (T^c)	x, y	v f_{thr} f_d T	[-10, 10] mm/s [5, 12] N -8 N [0.1, 2] s	2
	Translate (T)	x, y	v f_{thr} d f_d	[5, 10] mm/s 20 N [-10, 10] mm -8 N	2
	Rotate (R)	x, y, z	v f_{thr} d f_d	0.1 rad/s 2 Nm [-0.1, 0.1] rad -8 N	3
	Insert (I)	$-z$	ϵ k f_d T	2mm [0.01, 0.2] [6, 15] N [0.1, 2] s	1

The third baseline is based on [36], where a fix sequence of MPs is manually defined and MPs parameters are optimized through gradient-free optimization techniques. Following [36], the strategy has five steps (1) approach, (2) contact, (3) fit, (4) align, (5) insertion except for some slight differences. We use OSC as the low-level controller similar to our method and other baselines and use our insert MP in step 5 instead of sinusoidal motion. These modifications reduce the number of parameters included for optimization to four. The parameters are optimized by the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) algorithm [167], which shows best performance in [36]. We refer to this baseline `fix-seq`.



(A) Round

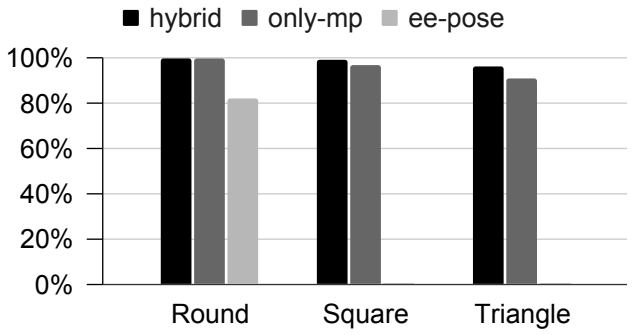
(B) Square

(C) Triangle

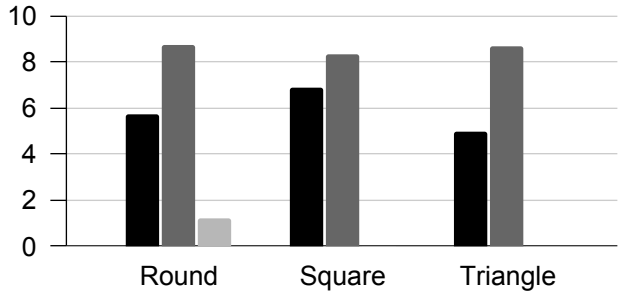
FIGURE 3.9: Training curve showing success rate over cumulative simulation steps. One simulation step corresponding to 2 ms. The dashed line shows the final performance of `only-mp` baseline

3.4.3.1 Simulation result

We train multiple policies, one for our method and one for each baseline (exclude `fix-seq`) on each of the peg insertion task (either with round, square, or triangle shape). The training curves are shown in Fig 3.9. In all three tasks, our method is much more sample efficient and achieve higher success rate than the other baselines. The number of samples required to reaches 60% success rate for `hybrid` is more than 3 times less than that of `only-mp`. We also found that `ee-pose` has trouble in exploration at start of learning. This could be explained by the fact that in high-precision assembly task, the region in state space that leads to the goal is very small. Furthermore, during insertion phase, random exploration noise also causes large interaction force between peg and hole along x and y axes, which prevents the peg to move along the z axis.



(A) Success rate



(B) Execution time (s)

FIGURE 3.10: Quantitative evaluation in simulation. The final success rate and execution time is averaged over 100 trials

After training, we obtain the quantitative performance of the trained policies by executing each policy on the corresponding tasks for 100 trials. The average success rate and average execution time is shown in Fig. 3.10. Overall, `hybrid` achieves best performance in terms of success rate, which is consistent with the training curve. Our method also achieves better execution time than `only-mp` as the manually chosen parameters in `only-mp` could be suboptimal. Another observation is that the average execution time for `ee-pose` is only 1.23 s, much less than `hybrid` and `ee-pose`. There are two reasons for this result. First, the strategy learned by `hybrid` and `only-mp` has a clear "search" phase whose purpose is to align the peg with the hole. The time required for this search phase depends on the environmental uncertainty (the hole pose in this case) causing larger execution time. Second, in `ee-pose`, robot motion is not constrained as in the other two baselines, i.e. the robot is free to move in any direction, thus the method can explore more diverse behavior. This is a limitation of our method which could be addressed by adding a dummy primitive that moves the robot end-effector to a desired position [118].

3.4.3.2 Sim2real policy transfer on physical robot

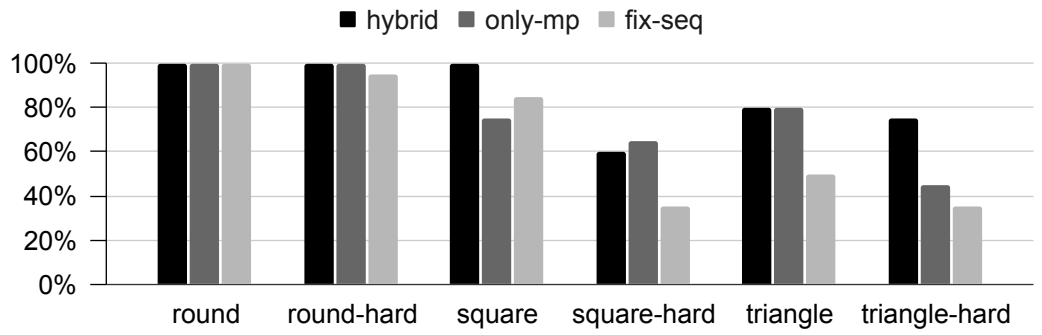
We evaluate the trained policies in simulation directly on six tasks corresponding to six peg-hole pairs shown in Table 3.1 without any further fine-tuning. In this experiment, we also report result for `fix-seq` baseline. For each task, an optimization is carried out to optimize the parameters of MPs. We then execute the final sequence for 20 trials and report the average success rate and execution time together with other methods. The results are shown in Fig. 3.11. We do not report the result for `ee-pose` because it always fails due to large contact force.

In general, we observe the same pattern as in the simulation: `hybrid` achieves higher success rate and shorter execution time than `only-mp`. An exception is for the triangle-hard task, `hybrid` has longer execution time than `only-mp`, but the success rate almost doubles that of `only-mp`. The `fix-seq` baseline achieves the best performance in terms of execution time. The reason is that the fix sequence always execute four MPs per trial, while RL policies learned by other methods attempt a longer sequence to search for the hole.

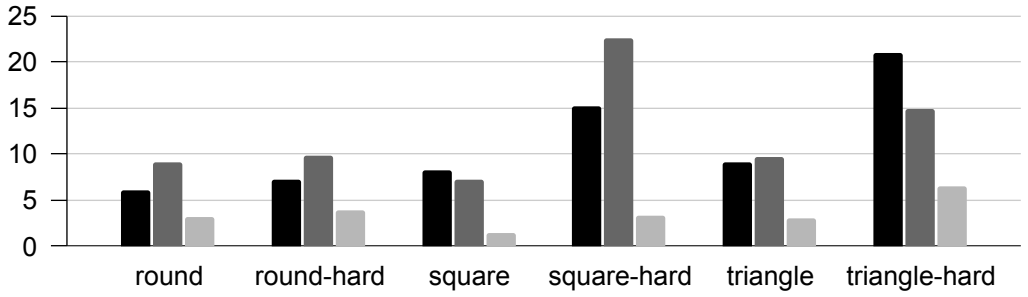
We observe two common failure modes during evaluation of the policy learned by our method. First the policy repeatedly choose the "move until contact" MP even after the peg has already made contact, causing the robot to be locked in the current contact state. The second failure mode is when the policy fails to fit and align with the hole. We hypothesize that this is because the policy makes decision based solely on the relative pose between peg and hole, which is amenable to sim2real gap in contact modeling (i.e. overlap between two bodies in contact) and difference in object's size. We believe that the first problem could be addressed by incorporating contact force to RL observation, as contact force is more natural signal to infer contact state than pose. However, this remains to be investigated in the future.

3.5 Conclusions

In this chapter, we have proposed a method to find dynamics sequence of manipulation primitives through Reinforcement Learning. Leveraging parameterized manipulation primitives, the proposed method was shown to greatly improve both



(A) Success rate



(B) Execution time

FIGURE 3.11: Quantitative evaluation on the physical robot across six different pin insertion tasks. The "hard" suffix denotes tasks with smaller clearance. The final success rate/execution time is averaged over 20 trials

assembly performance and sample efficiency of Reinforcement Learning. The experimental results showed that policies learned purely in simulation were able to consistently solve peg insertion tasks with different geometry and very small clearance.

A limitation of the approach is that the policy depends only on the kinematics information, while interaction force is only used to separate the free and in-contact action subspace. We hypothesize that this is the main causes for sim-to-real transfer failure. Integration of tactile information is thus an interesting future works.

In tasks that have complex dynamics, where instability is a key consideration, the choice of the low-level control law in each MP define the upper limit of the overall system performance. Hence, incorporating advanced robust control laws [52] is a promising direction.

Chapter 4

Integrating Force-based Manipulation Primitives with Deep Learning-based Visual Servoing for Robotic Assembly

4.1 Introduction

Robots have drastically increased industrial productivity by assisting humans to undertake high-volume and repetitive tasks such as lifting, assembly, and picking and placing of manufacturing parts. Specifically, robotic assembly has become progressively more common in the modern workspace, with increasingly complex and autonomous assembly tasks having been conducted in recent years [160]. However, robotic peg-in-hole assemblies require extremely high success rate and generalization to different contexts which are well beyond today's industrial robots' autonomous capability [1, 36]. Manual designing and fine-tuning are still required to achieve such tasks. Therefore, to achieve autonomous dexterous robotic peg-in-hole assembly, Vuong et al. proposed the idea of automatically discovering the dynamic sequence of Manipulation Primitives (MPs) via Reinforcement Learning (RL) [168].

The research from [168] utilized a force torque sensor to gauge the external force exerted on the end-effector. Besides maintaining a high accuracy, their method also

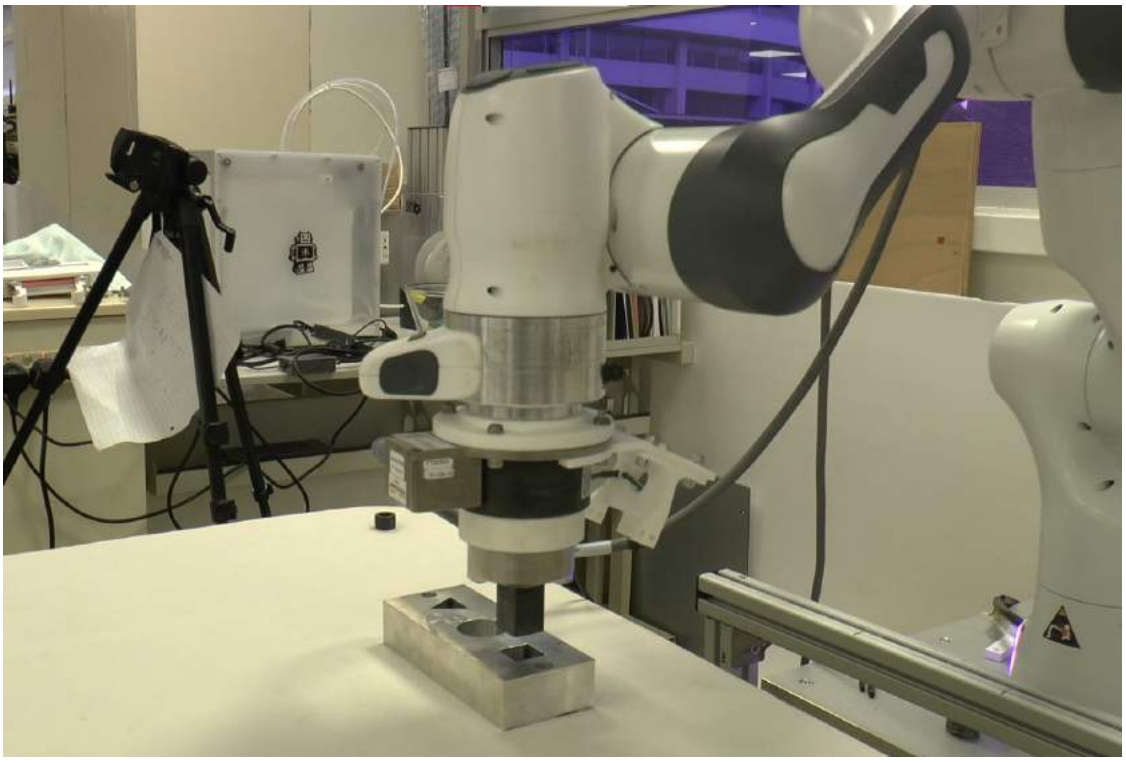


FIGURE 4.1: Robotic assembly setup. A square peg was used in this study.

showed promising generalization capability across different geometries. Nonetheless, the absence of a visual device limited the effectiveness of assembly as the peg had to be readily aligned within a small deviation range before insertion. This study aims to improve the solely force-based solution of [168] in terms of practicality in real-world settings by implementing Deep Learning-based Visual Servoing (DLVS) in the alignment phase. In this project, the DLVS work by Yu et al. [163] was chosen to complement the solely force-based solution. With DLVS capability, the hole pose can be estimated automatically in the alignment phase.

The scope of this study includes: (1) achieving high accuracy (1.5 mm in translation and 1.5 deg in rotation) autonomous estimation of hole pose in the alignment phase, and (2) enhancing the generalization capabilities across workspace with the newly integrated DLVS feature.

Related works

A study focusing on fast robust peg-in-hole insertion with continuous visual servoing was conducted in [169]. In the alignment phase, the peg was aligned to

the hole based on heatmaps generated from a Deep Neural Network (DNN). After alignment, peg insertion was attempted via compliance using force-feedback. This approach was able to achieve high accuracy (peg-hole clearance of 0.015 mm). However, there were two downsides: (1) DNN in alignment phase could only align position, but not orientation; (2) In the insertion phase, a simple compliant force insertion which was unable to account for large rotational errors was applied.

Triyonoputro et al. [170] focused on achieving peg-in-hole assembly using multi-view images and DLVS trained on synthetic data. There were two steps in the alignment phase: (1) DLVS quickly moved the peg closer to the hole; (2) spiral search then precisely aligned the peg to the hole. The process would then proceed to the insertion phase where impedance control was used to perform the insertion. The clearance of the hole in this experiment was 0.4 mm. However, this approach could not align orientation errors as well due to the limitations of the DNN. Another downside was the long execution time. The approach needed more than 40 seconds to complete peg insertion from the start of the search phase.

Deep learning-based visual servoing (DLVS) estimates the camera pose repeatedly while the robot is moving towards the target pose to achieve high final accuracy [163].

Bateux et al. explored an efficient method of generating dataset to train robust neural network for DLVS which considers changing lighting conditions and the addition of random occlusions [171]. The network achieves sub-millimeter accuracy but can only estimate a camera pose with respect to a fixed reference pose. The neural network has to be retrained every time a new reference pose is introduced, which is impractical in real-life usages. Thus, the authors proposed in the same paper another neural network which accepts a pair of images taken at random poses as input. Nevertheless, this extension could only achieve centimeter accuracy.

Yu et al. proposed a new neural network based on Siamese architecture that can output the relative pose between any pair of images taken at arbitrary poses with sub-millimeter accuracy [163]. The network is also effective under varying lighting conditions and with the inclusion of random occlusions, and can even generalize to objects with similar physical appearances. During actual insertion experiments, the model achieved sub-millimeter accuracy in camera pose estimation in one shot

from initial deviations of: (-5, 5) mm for x and y, (0, 10) mm for z, (-5, 5) deg for roll and pitch, (-10, 10) deg for yaw.

4.2 Methodology

4.2.1 Task Description

The peg-in-hole insertion task was split into two phases, namely (1) alignment phase and (2) insertion phase. In the alignment phase, the expected outcome was the improved alignment between the peg and hole through the DLVS algorithm from [163]. The peg was then manipulated to move down until contact with the hole block. After rotating the resulting peg pose by 180 deg against x-axis and translating it down along z-axis by the hole depth, the estimated hole pose was recorded (Fig. 4.2). Before proceeding to the insertion phase, to check whether the alignment phase had achieved complete insertion, the peg was manipulated to move in x, y, z - axes under two criteria, maximum movement duration and threshold of the force sensed. In unsuccessful attempts, the process subsequently proceeded to the insertion phase.

In the insertion phase, a dynamic sequence of MPs based on the RL policy trained in [168] were generated for final insertion. After each MP step, the policy would inspect the insertion status by finding the distance between the latest achieved pose and the estimated goal pose. If the x and y errors between the two poses were less than 5 mm and the z errors were smaller than 4 mm simultaneously, the insertion would be deemed successful.

4.2.2 Deep Learning-based Visual Servoing Neural Network

The neural network developed in [163] was designed to estimate the relative transformation between any two random camera poses. In training, the neural network took a pair of samples as input each time. Each sample in the pair comprised: (1) an image taken at a random pose and (2) the transformation matrix of the pose.

The output of the network was the relative pose between the input pair of camera poses in the form of translation (x, y, z) and quaternion (a, b, c, d).

Dataset Generation. Firstly, the peg was guided manually to the insertion pose. The peg was then lifted vertically for 12 cm so that a full view of the target hole could be captured. This end-effector pose was then recorded as the default pose T_d (Fig. 4.3). Samples were generated at random poses revolving around T_d. The origins of the new arbitrary end-effector poses were randomly sampled within a vertical cylinder of 10 mm radius and 20 mm height ($Cyl_{r=10,h=20}$), with the origin of T_d at the bottom center of the cylinder (Fig. 4.3). The rotation was randomly sampled within the range of -10 deg to 10 deg for roll and pitch, and -20 deg to 20 deg for yaw. At each random pose, an image was captured and the transformation matrix T_{de} of the pose was recorded. T_{de} is the transformation matrix which transforms the end-effector's coordinate frame to the default pose's coordinate frame. This image and T_{de} formed a complete sample which would later be input to the neural network as part of an input pair. The two input images in a pair were identified as I_A and I_B. Before training, the true label which was the relative transformation T_{BA} could be calculated as follows:

$$T_{BA} = [T_{dB}^{-1}][T_{dA}] \tag{4.1}$$

As the robotic arm's shadows could affect the network's performance, the samples were generated with the hole being placed at different positions and orientations to ensure the shadows did not always appear at the same position. The hole was placed at five points on the base, where four points would form the vertices of a 5-cm square and one point would be at the center of the square. At each point, the hole block was rotated clockwise at 0 deg, 30 deg, 60 deg, 90 deg. At each orientation, 200 samples were collected. This would amount to 4000 samples (5 points $\times$ 4 orientations $\times$ 200 samples).

4.2.3 Dynamic Sequences of Manipulation Primitives

Dynamic sequences of MPs could be discovered automatically through RL [168]. The RL policies were trained entirely in Mujoco simulation and transferred directly to physical execution.

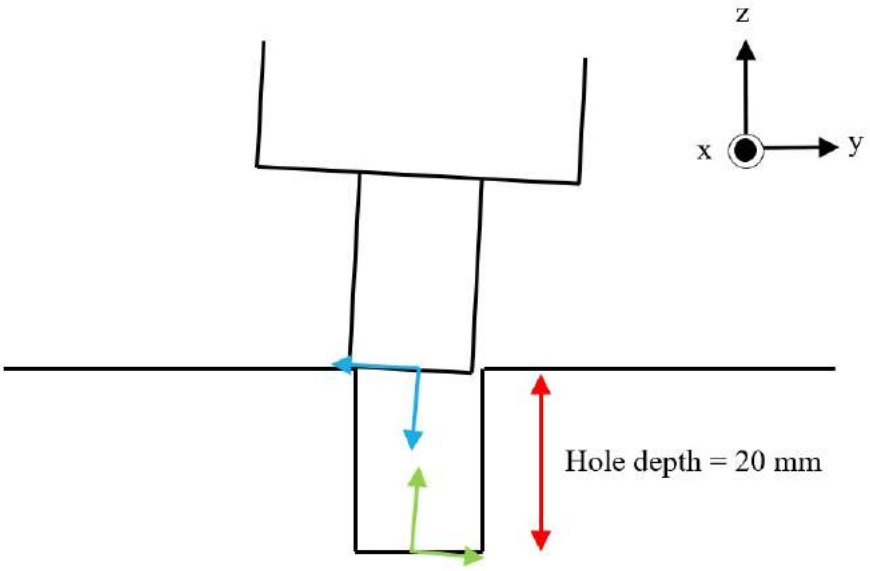


FIGURE 4.2: Hole pose (green) estimation deduced from after-contact peg pose (blue) in alignment phase.

Manipulation Primitives in the Insertion Phase. The MPs were defined as the appropriate motions of the end-effector in a task space. The motions were controlled by three types of instructions: (1) velocity command, (2) force command, and (3) stopping condition. The MPs were categorized into two families: free-space MPs and in-contact MPs. Free-space MPs were executed when the peg was not in contact with the hole block while in-contact MPs were executed when the peg was touching the hole block.

Using Reinforcement Learning to automatically generate dynamic sequences of Manipulation Primitives. The learning of dynamic sequences of MPs was regarded as a discounted episodic RL problem which could be addressed by a Markov Decision Process (MDP) [161]. An MDP is a function of state vector set S, action set A, state-transition probability P, reward R, and discount factor γ.

In this chapter, an action a way one of the MPs. The state vector s was defined as the position of the peg relative to the estimated hole frame. After an MP had been executed at time $t-1$ and the stopping condition had been reached, the new state at time t was measured. The reward function rewarded three terms: (1) MPs moving the peg closer to goal pose, (2) MPs with short execution time, and (3) MPs that had achieved SUCCESS stopping condition. With an initial pose

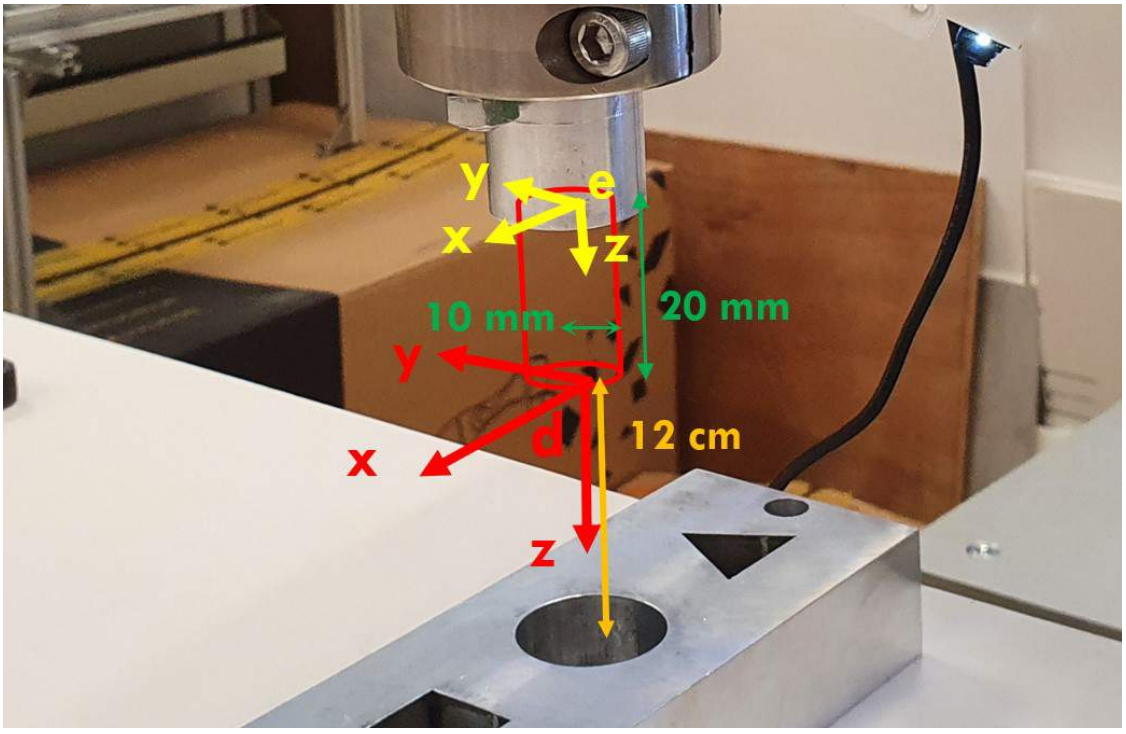


FIGURE 4.3: The red coordinate frame defines the default pose, T_d. The origin of the new random end-effector pose, T_{Oe} can be anywhere in the red cylinder.

deviation of $(-1.5, 1.5)$ mm and $(-1.5, 1.5)$ deg, this RL policy could achieve 94% success rate out of 50 insertion attempts with only one episode run. Thus, the peg's pose displacement errors needed to be within this range at the end of the alignment phase.

4.3 Experiments and Results

The performance of the model was first evaluated on the test set. After that, the model was tested on actual insertion tasks. To prove the usefulness of the pre-insertion alignment, two baseline experiments were conducted. Lastly, the model was appraised for its generalization capability over workspace.

4.3.1 Experimental setup

All experiments were conducted with a plastic square peg and a square hole which has 19.98-mm sides and 20-mm depth. The clearance between the mating parts

TABLE 4.1: Test set errors (e_ϕ: roll error, e_θ: pitch error, e_ψ: yaw error)

e_x/mm	e_y/mm	e_z/mm	e_ϕ/deg	e_θ/deg	e_ψ/deg
0.2441	0.2875	0.2044	0.1792	0.1856	0.2148

was 0.26 mm.

The robot used in this project was the 7-DOF Franka Emika Panda cobot. An in-hand camera was mounted on the end-effector (Fig 4.1). The camera was short-range with a field of view of 70 deg and a resolution of 640×480.

An additional force torque sensor, Gamma IP60 was used to measure the external force exerting on the peg as the force estimation in `libfranka` was too imprecise for the execution of force-based MPs.

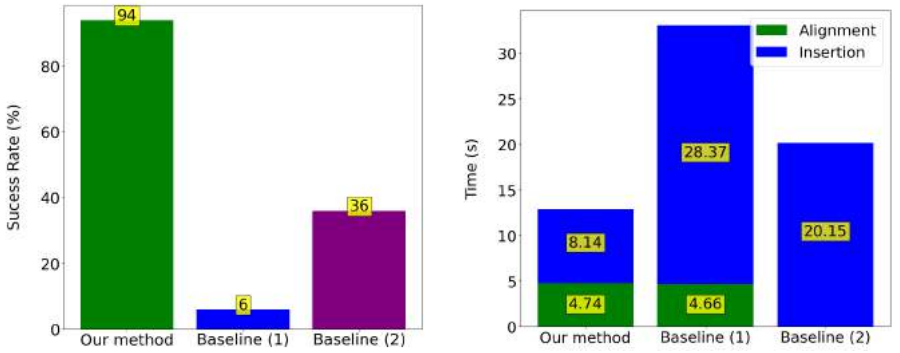
4.3.2 Training and model evaluation

Both samples in each input pair to the network had to be taken at random poses which were generated with respect to the same T_d. In total, there were $2002 \times 20 = 800000$ pairs of samples taken from 20 sets (4 orientations at each of the 5 points). 80% of the samples were used for training and the rest were used as the test set. A model was trained for 10 epochs. The learning rate was 10^{-4} at the beginning and halved after the 4^{th}, 6^{th}, 8^{th} epoch. The batch size of the training set was 256. The entire training was run on 4 GTX-1080Ti's.

The performance of the model on the test set is recorded in Table 4.1. Since the RL policy used in the insertion phase could accept errors up to 1.5 mm in translation and 1.5 deg in rotation, the test errors were low enough to proceed to physical execution.

4.3.3 Actual insertion task

The peg was manually guided to the goal pose at the beginning. 50 random poses within the sampling range defined by the $Cyl_{r=5,h=10}$ were generated around the goal pose. At each attempt, I_A and one of the 50 images taken at the arbitrary poses, I_B were input to the model. $\hat{T}_{BA}$ between the two poses was estimated through DLVS. The peg would move to $\hat{T}_{0A}$ at the end of the alignment phase.



(A) Insertion success rate (B) Average time per attempt

FIGURE 4.4: (a) Insertion success rates and (b) Average time taken per attempt for alignment and insertion of our method compared to the two baseline methods out of 50 attempts. In (b), there was no alignment phase in baseline (2).

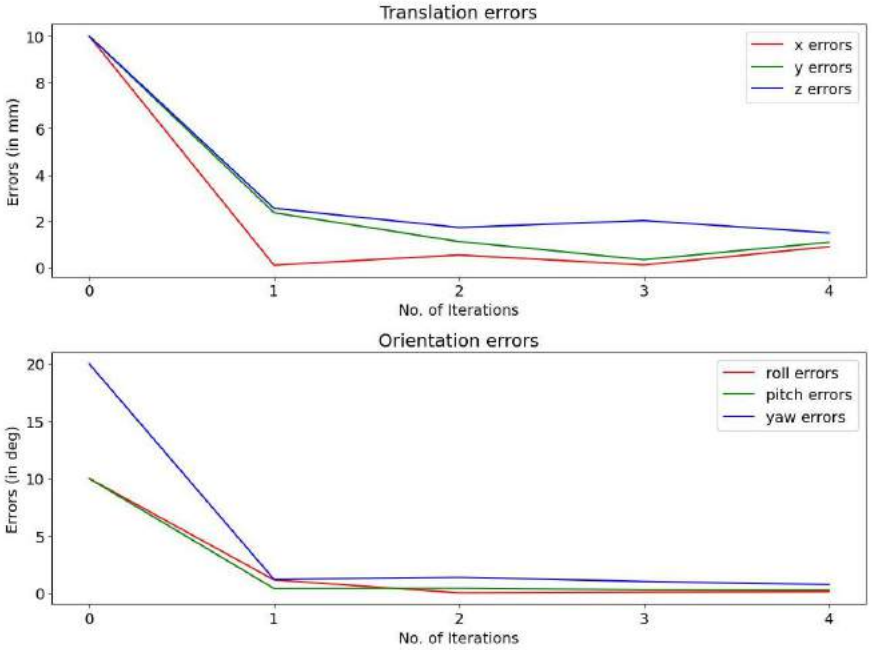
In the insertion phase, the true hole pose was not given explicitly to the RL policy. The estimated hole pose deduced from $\hat{T}_{0A}$ in the alignment phase was input to the policy. The peg was subsequently guided into the hole by a sequence of force-based MPs generated within one episode. The success rate and time taken are shown in Fig 4.4.

4.3.4 Comparing our method to baseline methods

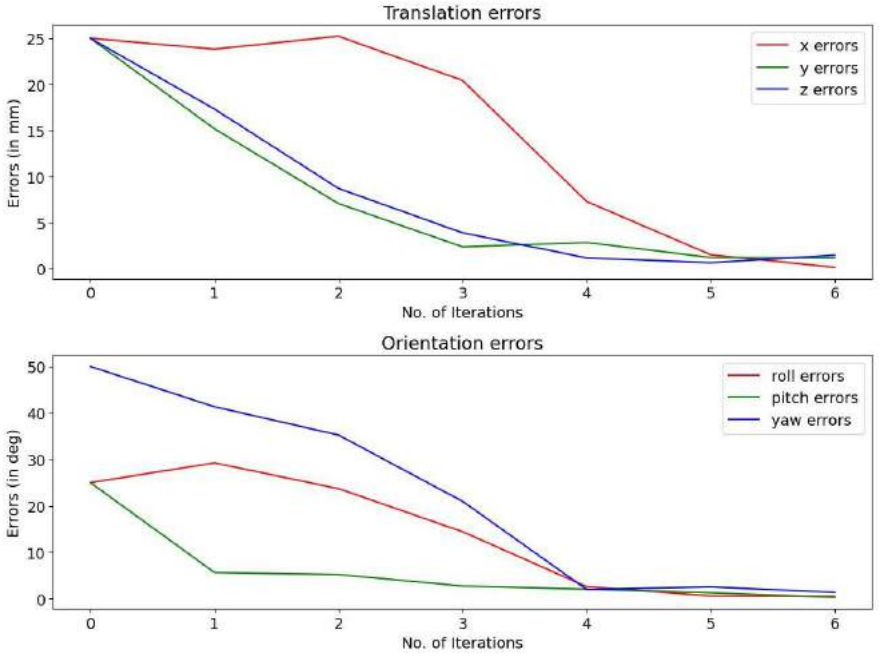
Two baseline experiments were conducted to prove the usefulness of the proposed approach: (1) aligned peg with the same DLVS algorithm followed by pure compliance insertion and (2) attempted insertion with RL-generated MPs without alignment from the same sampling range defined by $Cyl_{r=5,h=10}$. As shown in Fig 4.4a, both baseline methods' insertion success rates were much lower than that of our method whereas the time taken per attempt for alignment and insertion were much longer.

4.3.5 Generalization over workspace

Iterative estimations with the same DLVS algorithm managed to align the peg to be within acceptable deviation threshold from initial pose differences that were larger than the sampling range $Cyl_{r=5,h=10}$. Two test cases (1 easy, 1 hard) were



(A) Easy test case. Initial pose errors: (x, y, z) = $(10, 10, 10)$ mm, (roll, pitch, yaw) = $(10, 10, 20)$ deg. Converged to acceptable error thresholds after 4 iterations.



(B) Hard test case. Initial pose errors: (x, y, z) = $(25, 25, 25)$ mm, (roll, pitch, yaw) = $(25, 25, 50)$ deg. Converged to acceptable error thresholds after 6 iterations.

FIGURE 4.5: Initial pose differences that were larger than the sampling range $Cyl_{r=5,h=10}$ converged to within 1.5 mm and 1.5 deg in both easy and hard test cases.

executed. In both cases, all pose errors converged to within 1.5 mm and 1.5° after a number of iterations (Fig 4.5).

4.4 Conclusions

The addition of DLVS has improved the practicality of the force-based peg insertion solution proposed by Vuong et al. [168]. With visual capabilities at the alignment phase, the peg's starting pose error thresholds in both translation and orientation were increased. Even initial pose differences that were larger than the normal sampling range could be handled if iterative visual servoing was applied.

Furthermore, the true hole pose is no longer required in this new approach. The estimated hole pose can be deduced from DLVS and input to the RL policy. This improvement is significant as in real-world robotic assembly tasks, the pose of the part to be mated is normally unknown.

In future work, the DLVS model's generalization capability to different shapes can be evaluated without retraining. Optimal numbers of visual servoing iterations can also be found for different magnitudes of initial pose errors to boost the proposed approach's usability in real-life assembly tasks.

Chapter 5

Controller Influence on Reinforcement Learning performance for Contact-rich tasks

5.1 Introduction

The integration of deep learning neural networks into Reinforcement Learning (RL), termed Deep Reinforcement Learning (DRL), has enabled learning complex decision-making problems, such as playing Atari games from pixels [2]. This motivates the application of DRL to robotics control systems that often require reasoning from high-dimensional, noisy sensor measurements. Many works have demonstrated the impressive capabilities of DRL in learning control policies for complex robotics tasks, such as in-hand manipulation [7], locomotion [6].

The choice of action space is crucial to the sample efficiency of RL algorithms and the performance of the learned policies. Early works have demonstrated that DRL is capable of learning control policies that directly output low-level motor commands [8, 10]. Subsequent works have shown that choosing action as input to a low-level feedback controller can improve sample efficiency and policy performance for locomotion [9] and manipulation [110, 109]. Common choices of low-level controllers include joint position/velocity controller [100, 101, 10, 102, 7], task-space

position controller [104, 11, 105], or impedance controller [108, 109]. The low-level controller serves as prior knowledge to bootstrap control policy learning. For instance, a task-space position controller would implement gravity compensation and forward kinematics to compute position feedback so that RL policy can focus on learning task-related behaviors.

A matter that is usually overlooked in the literature is that, given a control objective, there exists a variety of implementations for the low-level controller from the vast literature of control theory and robot control. For instance, three implementations for direct force control of position-controlled robot manipulators are reported by Roy and Whitcomb [13], eight implementations for the task space control of redundant robot manipulators are reported by Nakanishi et al. [14]. Different implementations might have very different control performance and robustness. Therefore, it is natural to ask whether a better implementation of the low-level controller results in a better RL policy. To the best of our knowledge, there is no studies on this problem in the literature.

This chapter focuses on direct force control for contact-rich manipulation tasks with position-controlled robots. The high-precision peg insertion task is considered since it represents common problems seen in contact manipulation tasks. Three implementations of the force controller are designed by two methods: Proportional-Integral controller and Convex Controller Synthesis (CCS) [52]. Policies learned with these controllers are compared in terms of task performance and robustness. Our experiments, performed in both simulation and in the real world, suggest that a better low-level controller obtained with Convex Controller Synthesis can improve policy robustness and policy performance.

Related works

Previous works have benchmarked different action spaces for locomotion [9, 172] and robot manipulation [110, 109, 107]. A common conclusion is that incorporating low-level feedback control can improve learning speed and policy performance. However, only a few, if any, research have studied how the choice of low-level controller influences RL performance. In most of the above works, a representative controller is chosen for each action space, and the design of the controller is usually ignored.

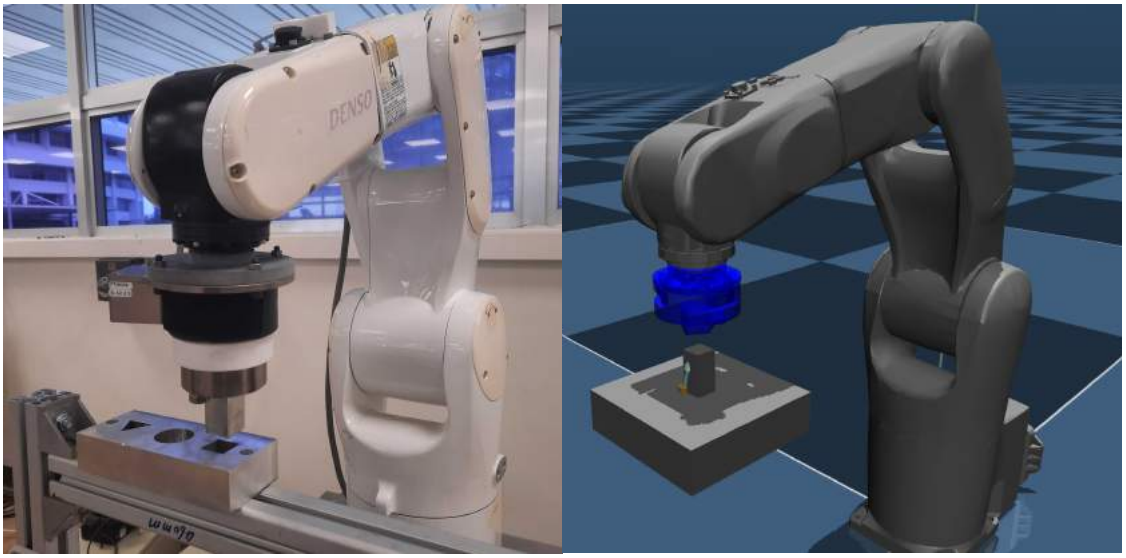


FIGURE 5.1: Simulated and physical task setup.

The importance of tuning the hyperparameters of the low-level controller has been recognized in several works [9, 107]. To circumvent this issue, many works have proposed to learn state-dependent controller parameters considering them as actions [108, 109, 119, 107]. In certain cases, changing the parameters of the low-level controller can be seen as regularizing change in compliance. However, in general, learning state-dependent controller parameters might compromise the stability or performance of the controllers.

Several works have evaluated and compared the performance of RL policies learned with different low-level controllers [110, 109]. However, the control objectives of the low-level controllers are different, and only one implementation for each control objective is considered. For instance, Varin et al. [110] compare RL policies learned with a joint position controller, an end-effector position controller, or an impedance controller. These controllers aim to regulate the joint position, end-effector position, and robot impedance, respectively. These controllers are not comparable since their control objectives are different.

Most related to our works, Beltran-Hernandez et al. [107] compared two methods of force control, a parallel position/force controller with a Proportional-Integral force control loop and an admittance controller for learning assembly tasks. The results show that the admittance controller achieves slightly better performance for a pin insertion task and produces fewer collisions. However, they study state-dependent controller parameters, while in our case, the controller parameters are fixed in the

training phase. Furthermore, while the parallel position/force controller and the admittance control both aim to stabilize the interaction between the robot and the environment, their control objectives are different: the former directly regulates the external force acting on the robot end-effector, while the latter aims to regulate the relation between external force and position of the end-effector. In contrast, this work considers two direct force control methods.

5.2 Methodology

5.2.1 Overview of control system

Fig. 5.2 depicts the overview of the control system used in this chapter. The control system features a force controller that regulates robot motion in contact and an RL policy that generates desired robot motion as inputs to the force controller. An alternative approach to integrate RL into the control system is to learn an RL policy to directly outputs the low-level robot command, such as motor torque or joint position for position-controlled robots [8, 10]. However, in this approach, the RL agent might need to first learn to stably interact with the environment before learning to solve the main task. In contrast, in our control system, the stability is ensured by the low-level force controller, thus allowing the RL agent to focus on learning the main manipulation objective. Furthermore, our approach allows the integration of successful model-based controllers from the extensive literature on control theory and robot control.

There exists pervasive literature on the topic of force control. Among the force control schemes, hybrid force/position control [47] and impedance control [45] are the two of the most popular methods. Impedance control aims to maintain a static or dynamic relation between the robot end-effector force and position. Hybrid control aims to explicitly control the end-effector force in the constrained directions and the end-effector position in the remaining directions. Both schemes can be incorporated into the control system. In this work, we focus on the hybrid control scheme. Particularly, we employ the parallel force/position implementation [173], which can cope with the uncertainties in the environment geometry. The original hybrid force/position control is not compatible with RL since it requires either an accurate model of the environment, which is one of the reasons RL is employed.

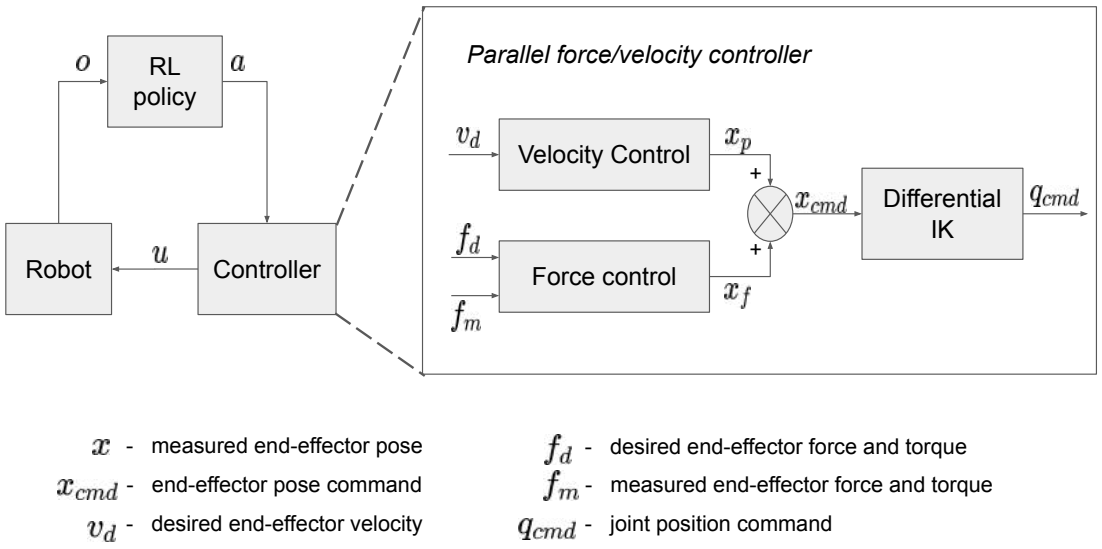


FIGURE 5.2: Reinforcement learning framework. RL policy outputs desired end-effector velocity and desired end-effector force to a hybrid velocity/force controller

The parallel force/position controller has three inputs: desired end-effector velocity, desired end-effector force/torque, and measured end-effector force and torque. The controller outputs a Cartesian position and orientation command. This command is fed to a Differential Inverse Kinematics module, which computes the joint position command for the robot. The parallel force/position controller includes two control loops: a velocity control loop and a force control loop. The velocity control loop is simply a feedforward velocity control without feedback. The force control loop needs to be carefully designed to ensure a stable response when the robot is in contact. We also employ decoupled control scheme in which each degree of freedom is controlled separately.

The RL policy outputs desired end-effector velocity and force trajectories as inputs to the parallel force/position controller. In some cases, we only need to learn a subset of the controller inputs while setting the remaining controller inputs according to some well-known heuristics. For instance, the yaw angles can be fixed in a cylindrical pin insertion task. To address this issue, we formulate the RL policy based on the Residual Policy Learning approach [11] as follows

$$\pi(s) = \pi_l(s) + \pi_f(s) \tag{5.1}$$

Where π_l is the policy learned by RL and $\pi_f(s)$ is the hand-designed policy. The above examples can be formulated in this form by choosing $\pi_l(s)$ such that $\pi_l(s) = [v_{dx}, v_{dy}, v_{dz}, \omega_{dx}, \omega_{dy}, 0]$, where $v_d = [v_{dx}, v_{dy}, v_{dz}, \omega_{dx}, \omega_{dy}, \omega_{dz}]$.

5.2.2 Direct force control methods

The two force control methods considered in this study are the Proportional-Integral (PI) controller and Convex Controller Synthesis [52] (CCS). The PI controller is characterized by the following transfer function with two parameters K_p and K_i:

$$K(s) = K_p + \frac{K_i}{s} \tag{5.2}$$

In contrast, CCS synthesize a generic class of controller represented as follows

$$K(s) = (I + Q(s)T(s))Q(s) \tag{5.3}$$

where $T(s)$ is a fixed transfer function deduced from the plant and $Q(s)$ is any stable transfer matrices. To perform numerical optimization in practice, $Q(s)$ can be represented as a linear combination of basis stable transfer matrices $Q(s) = \sum_{i=1}^{n} \theta_i Q_i(s)$. The parameters θ_i are obtained via a convex optimization problem, where the performance specifications are formulated as either convex cost functions or convex constraints.

The PI controller is widely used in practice owing to its simplicity and fine performance in many situations. However, the PI controller is limited by its fixed structure with only two tunable parameters, while CCS can synthesize a generic class of controller. Therefore, CCS controllers can achieve a significantly higher level of performance. In this way, we can easily design controllers with varying levels of performance.

5.2.3 Modeling of position-controlled robot in simulation

We aim to conduct experiments both in the simulation and in the real world. Instead of training RL policies from scratch in the real world, the trained policy in simulation can be directly executed on the physical robot. The policy might

achieve worse performance in the real world due to the inevitable discrepancies between the system and its simulated model. Therefore, we propose several modeling approaches in the Mujoco physics simulator to improve simulation accuracy.

Modeling of position-controlled robot Position-controlled robots come with a joint position controller. User access to this controller is typically limited or unavailable [13]. Following the work by Hung et al. [52], we assume decoupled robot dynamics, and the dynamics of each joint are modeled as a first-order linear time-invariant system with time delay. An implicit assumption of this model is that the effect of contact force on the inner joint position controller is negligible, which is a common assumption in the literature [13]. As a direct consequence, the robot is assumed to be much stiffer than the environment. In other words, a very small position error may result in a huge interaction force. Motivated by this model, we model the joint position controller of the robot with a *computed torque controller*

$$\tau = M(\ddot{q}_d + D(\dot{q}_d - q) + K(q_d - q)) + \tau_{ext} + C(q, \dot{q}) + g(q) \tag{5.4}$$

where M is the joint inertia matrix, $q, \dot{q}$ is the joint position, joint velocity respectively, $q_d, \dot{q}_d, \ddot{q}_d$ is the desired joint position, desired joint velocity, and desired joint acceleration respectively, τ_{ext} is the external torque, C is Coriolis and centrifugal torque, and g is the gravitational torque. K and D are diagonal gain matrices. Compared to [52], controller (5.4) also results in decoupled dynamics, but the dynamics of each joint is a second-order system instead of first-order.

5.3 Experiment

The main objective of the experiments is to test whether a better controller results in better RL performance. For this purpose, we design three controllers with different levels of performance and compare the performance of policies learned with these controllers both in simulation and in the real world.

5.3.1 Experimental setup

5.3.1.1 Task description

We consider the classical high-precision peg-in-hole task. The setup for the peg-in-hole task is shown in Fig 5.1. The following assumptions are made for the assembly task:

- The peg is firmly grasped or rigidly attached to the end-effector, and the hole is rigidly mounted in the environment.
- The hole position and axis of insertion can be estimated with a position error less than 2 mm and an orientation error less than 2 degree. This can be achieved by, for example, using a visual servoing technique [163].
- The task frame is chosen such that its origin coincides with the estimated hole position and the direction of insertion is along the negative z-axis. This assumption simplifies the design of MPs without the loss of generality.

The peg and hole are made of aluminum with a rectangular shape. The size of the peg is 19.98 mm and the clearance between the peg and hole is measured to be 20 μm clearance.

5.3.1.2 Robot system setup

Real experiments were carried out on a 6-axis Denso VS-060 robot manipulator. An ATI Gamma force/torque sensor was rigidly attached to the end-effector to provide contact force measurements. A personal computer running Ubuntu 16.04 was used to send joint position commands to the robot at 125 Hz and train RL policy. For both tasks, the pegs were rigidly attached to the force torque sensor. The holes were rigidly mounted within the robot's workspace. The hardware setup and corresponding simulated environment for the double pin insertion task are shown in Fig 5.1. The environment stiffness is measured to be 100 N/mm.

5.3.1.3 RL environment implementation

The state is defined as $s = [f_m, x]$ where f_m is the measured end-effector force, x is the measured end-effector pose. The subset of control inputs learned by RL is defined by $\pi_l(s) = [v_x, v_y, v_z, f_z, s_z]$, where $v_x, v_y, s_z v_z$ is the desired translational velocity along the x, y, z direction, respectively, $(1 - s_z) f_z$ is the desired force along the z direction. The variable s_z is a discrete variable that is either 0 or 1. It is used to incorporate the natural constraint: the robot can be either force-controlled or position-controlled along any axes. The other components of the desired velocity and desired force are set to 0.

Before starting an episode, the robot first moves to a random initial pose such that the peg is located on top of the hole, the initial positional error is within 2 mm, and the initial orientation error is within 2 deg with respect to the target pose. The robot is then commanded to move along the insertion direction until the measured force is larger than 5 N, after which an episode starts. The episode terminates when either (1) the number of environment steps exceeds 1250 steps, (2) the measured force/torque exceeds a predefined maximum allowable force/torque in any direction, or (3) a success condition is met. The maximum allowable force is set to 50 N along the insertion direction and 30 N along the other direction. The episode is considered a success if the peg reaches 80% of the insertion depth and the measured force along the insertion direction is larger than 15 N. The current inserted depth is estimated as an offset to the initial contact position along the insertion direction.

The reward function is

$$r = \frac{z - z_{env}}{D} - 40||v_d|| \tag{5.5}$$

The RL policy runs at the same rate as the controller (125 Hz) and is parameterized as two Multilayer Perceptrons (MLPs). The first MLP contains two layers with 16 neurons each and outputs the logits of a Bernoulli distribution over the two possible values of s_z. The second MLP contains two layers with 64 neurons each and outputs the mean of the Gaussian distribution over the continuous control inputs $[v_x, v_y, v_z, f_z]$. The policy is trained with Proximal Policy Optimization (PPO) [87] algorithm. In addition to the policy network, PPO requires a value network to approximate the value function. The value network is also an MLP with two layers, 64 neurons in each layer.

5.3.1.4 Controllers design

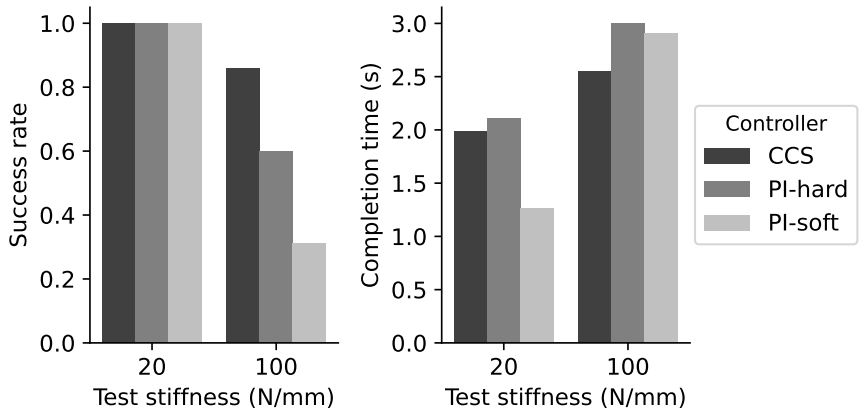
CCS requires the model of the position-controlled robot, which we acquire by system identification. Following [52], the robot is modeled as a first-order system with a time delay. The model is identified by applying a step input to the robot. The model is also utilized to tune PI controllers with MATLAB to ensure a fair comparison.

A practical case is that the environment stiffness is uncertain. A good controller in this situation must be achieved robust performance and robust stability over a range of environmental stiffness. Similarly, a good policy must also achieve robust task performance over a range of stiffness. We designed three controllers for direct force control

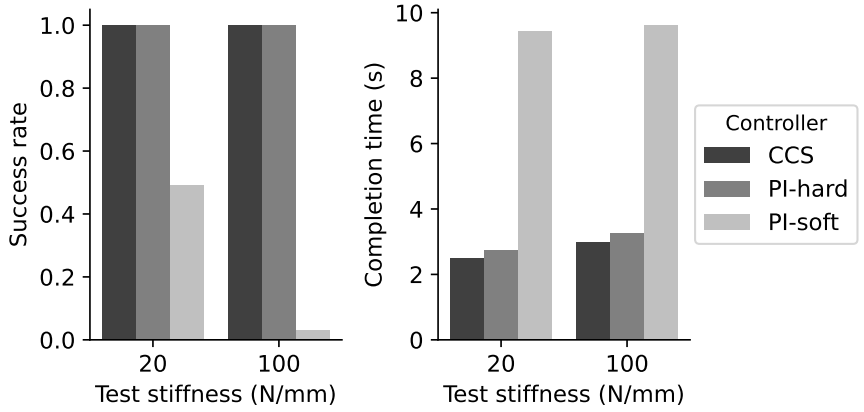
- `CCS` controller is designed based on two main performance specifications. First, the time-domain response to step input should be that of a first-order system with a time constant of 0.05 s at the nominal stiffness $K = 50\,\mathrm{N/mm}$. Second, the CCS controller should be robust to environment stiffness K from $20\,\mathrm{N/mm}$ to $100\,\mathrm{N/mm}$.
- `PI-soft` controller is a Proportional-Integral controller and is tuned to have a similar step response to the `CCS` controller with the assumption that the environment stiffness is $K = 20\,\mathrm{N/mm}$.
- `PI-hard` controller is a Proportional-Integral controller and is tuned to have a similar step response to the `CCS` controller with the assumption that the environment stiffness is $K = 100\,\mathrm{N/mm}$

The performance of the three controllers is as follows

- for any stiffness K in $[20, 100]\,\mathrm{N/mm}$, the `CCS` controller is better than `PI-hard` controller in terms of the bandwidth of the closed-loop system.
- If the environment stiffness $K = 20\,\mathrm{N/mm}$, the `PI-soft` controller is better than `CCS` controller and `PI-hard` controller in terms of the bandwidth of the closed-loop system. However, the `PI-soft` controller is unstable when the environment stiffness is $K = 100\,\mathrm{N/mm}$.



(A) Train stiffness: 20 N/mm



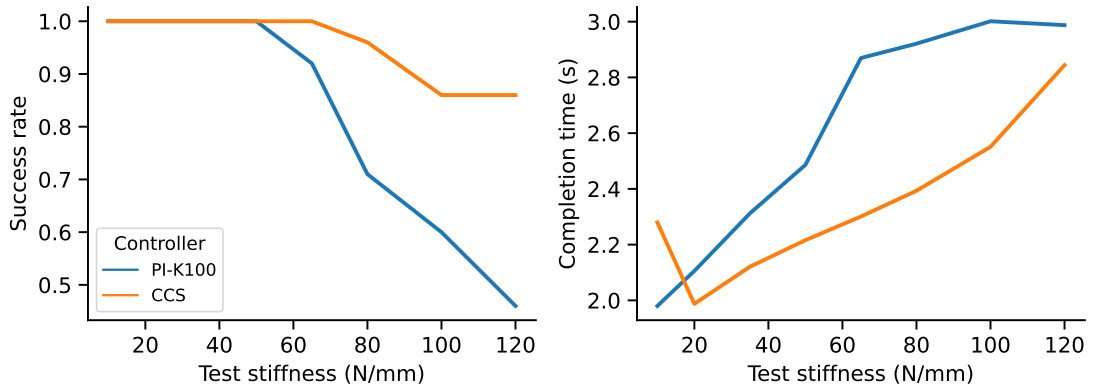
(B) Train stiffness: 100 N/mm

FIGURE 5.3: Performance of three controllers over different train and test stiffnesses

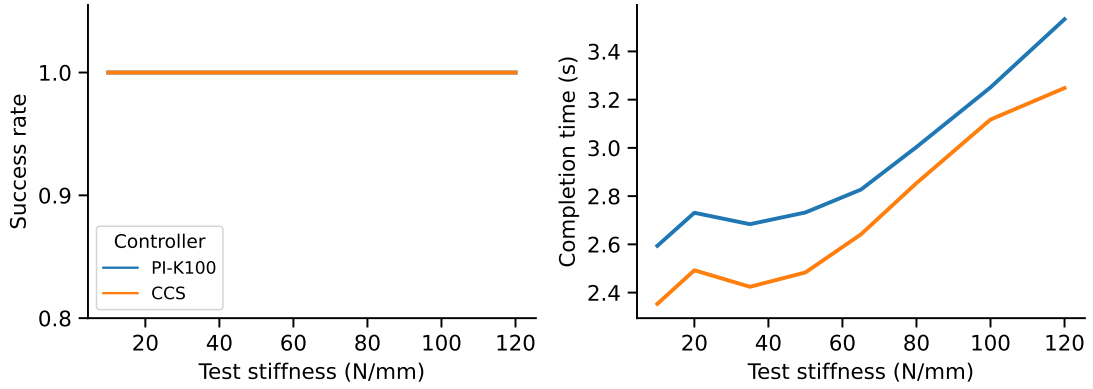
5.3.2 Simulation experiments

For each of the three controllers, we train policies in two cases: environment stiffness $K = 20\,\mathrm{N/mm}$ and $K = 100\,\mathrm{N/mm}$. After training, the policies are evaluated on environment stiffness $K = 20\,\mathrm{N/mm}$ and $K = 100\,\mathrm{N/mm}$ for 100 trials each. The success rate and average completion time are shown in Fig. 5.3.

When the train stiffness and test stiffness is $K = 20\,\mathrm{N/mm}$, all policies trained on stiffness $K = 20\,\mathrm{N/mm}$ achieve 100% success rate. The policy trained with `PI-soft` has the shortest average completion time. The reason could be that the closed-loop system realized by the `PI-soft` controller has the highest bandwidth, thus making the robot respond faster to changes in the measured force. We conclude that, in this case, better controller results in a better RL policy since three



(A) Train stiffness: 20 N/mm



(B) Train stiffness: 100 N/mm

FIGURE 5.4: Evaluation on different stiffness

policies are trained under the same training configuration except for the low-level controller.

A similar conclusion can be drawn in the case train, and test stiffness is $K = 100\,\mathrm{N/mm}$. The policy trained with the `PI-soft` controller is unable to complete the task since the controller is unstable, while the policy trained with the `CCS` controller has a faster task completion time than the one trained with the `PI-hard` controller.

We next evaluate the robustness of the policies by testing them on different environment stiffnesses than the train stiffness. The results are summarized as follows

- Train on $K = 20\,\mathrm{N/mm}$, test on $K = 100\,\mathrm{N/mm}$: The success rate and task completion time of policies learned with `PI-soft` and `PI-hard` controller are greatly reduced. Since the `PI-soft` controller is unstable under

$K = 100\,\mathrm{N/mm}$, vibration occurs at the contact interface (the robot keeps making and breaking contact), the policy sees new observations that are not seen during training. The policy learned with `CCS` controller achieves a better success rate and completion time than `PI-hard` owing to its higher bandwidth.

- Train on $K = 100\,\mathrm{N/mm}$, test on $K = 20$,N/mm: The policy trained with the `PI-soft` controller has the lowest success rate and completion time. This may be caused by the unstable response of the `PI-soft` controller, which affect the training phase.

- Test on stiffness K in $[10, 120]\,\mathrm{N/mm}$ (Fig. 5.4): The policy trained with `CCS` controller consistently achieves better success rate and completion time than the policy trained with `PI-hard` controller due to its higher bandwidth.

Based on the above results, we can conclude that a robust controller is able to improve the robustness of the learned policy. This is particularly useful since learning robust RL policy is important since the RL policy might work in a different environment than the training environment or in case the RL policy is transferred from simulation to the real robot.

5.3.3 Physical robot experiment

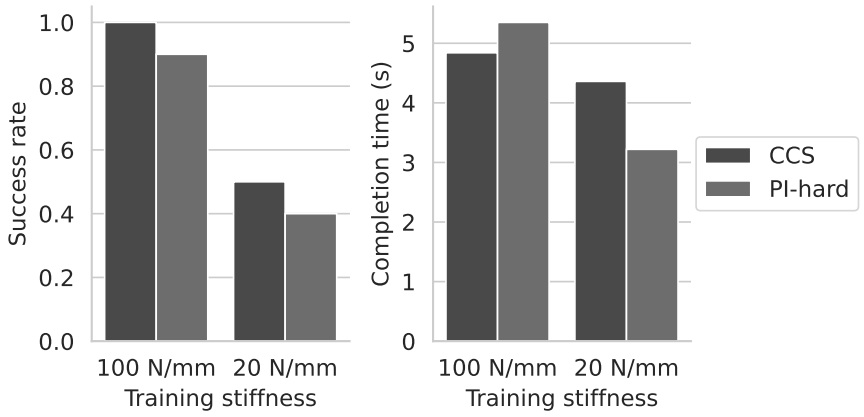


FIGURE 5.5: Zero-shot sim-to-real performance of two controllers trained on a simulated square peg-in-hole task and evaluate on the real square peg-in-hole task. The environment stiffness in the real task is measured to be $100\,\mathrm{N/mm}$

In this section, we carry out experiments to validate the hypotheses in the previous section on the physical robot. Specifically, we evaluate policies trained with the `PI-hard` controller and the `CCS` controller for 30 trials on the cubical peg-in-hole task described in Section 5.3.1. We didn't evaluate the policy learned with the `PI-hard` controller since the controller is unstable.

The results are shown in Fig. 5.5. The policy performance has dropped slightly compared to that in the simulation. A similar trend can be observed: the policy trained with `CCS` has a better success rate and task completion time than `PI-K100`.

5.4 Conclusion

We have presented an experimental study on the influence of the low-level controller on RL policy performance. Our results suggest high-bandwidth controllers improve policy performance in a tight-clearance industrial peg-in-hole task. Additionally, a robust controller can also improve the robustness of the learned policy. A sim-to-real experiment further validates the above hypotheses on the physical system.

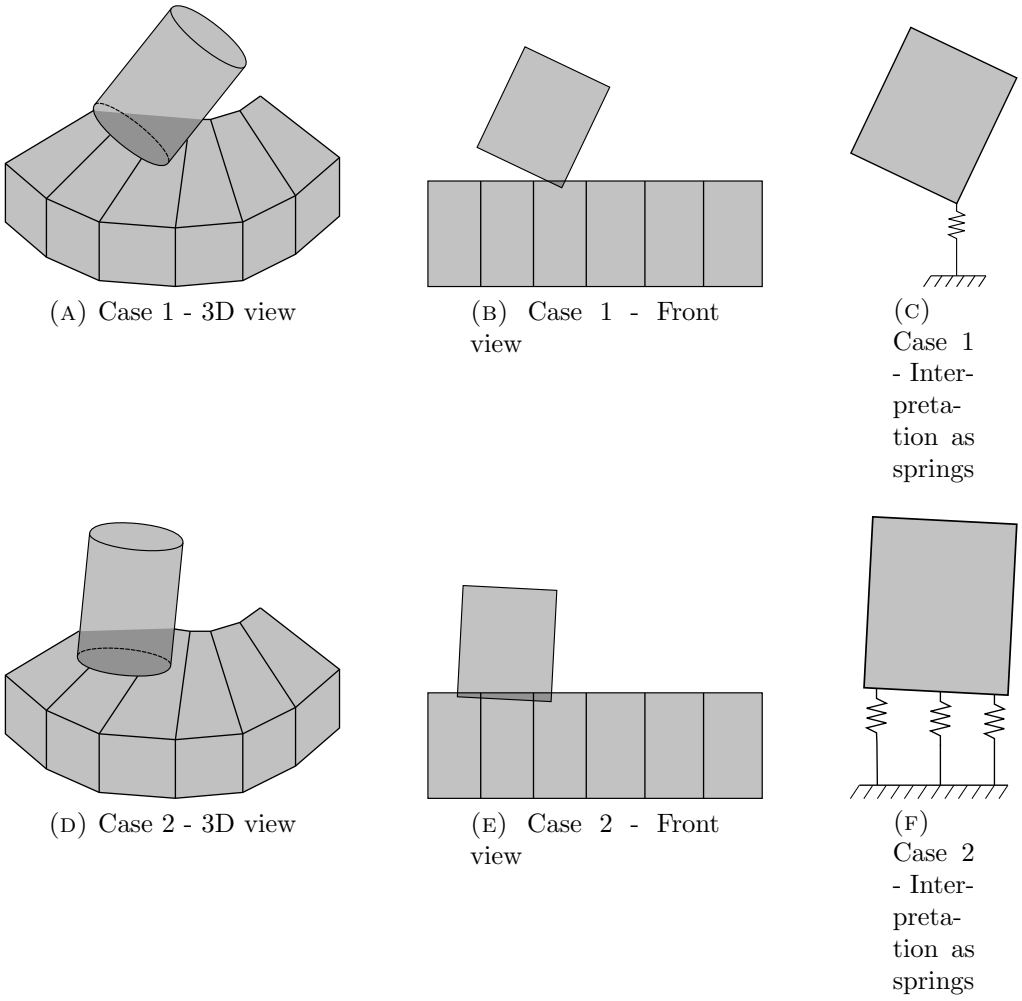
These results might give a new perspective for the integration of RL into robot control systems: more efforts should be put into the design of the low-level controller since it can improve learned policy performance and robustness. However, the limitation of our work is that the study is conducted in a specific case of industrial pin insertion. One future direction is thus to perform extensive experiments with different robot platforms (e.g., torque-controlled robot), controllers (e.g., impedance controller in joint space or Cartesian space), and different tasks.

Chapter 6

Contact Reduction with Bounded Stiffness for Robust Sim-to-Real Transfer of Robot Assembly

6.1 Introduction

Learning robot manipulation skills through Reinforcement Learning (RL) is challenging. Modern RL algorithms typically have high sample complexities, resulting in lengthy robot execution time. Moreover, model-free RL algorithms perform random action sampling during the exploration phase, raising safety concerns when the robot interacts with its environment. One way to address these problems entails learning an RL policy in a virtual environment and deploying it on the physical system. Numerous studies have demonstrated the ability of this sim-to-real methodology to teach sophisticated robotics tasks such as dexterous in-hand manipulation [7], locomotion on unknown terrain [174]. The main challenge of sim-to-real RL is overcoming the *reality gap* - the discrepancies between the real world and its simulated counterpart. Simulated environments should be created to best represent the real world to reduce the reality gap. However, the majority of works build simulated environments using collision geometries that approximate the true geometry with primitive shapes (cylinders, boxes, spheres). For objects with concave surfaces, such a simple approximation may fail to realize the actual contact region.



(A) Case 1 - 3D view

(B) Case 1 - Front view

(C) Case 1 - Interpretation as springs

(D) Case 2 - 3D view

(E) Case 2 - Front view

(F) Case 2 - Interpretation as springs

FIGURE 6.1: Collision detection between a cylinder and a convex-decomposed shape generates one contact point in case 1 ((a), (b) and (c)) and three contact points in case 2 ((d), (e), and (f)) with a slight change in the cylinder's pose. With finer decomposition, the number of contacts may vary significantly. Each contact point can be interpreted as a spring connecting the two bodies to prevent interpenetration. Since each spring adds up to the total stiffness, the system may become overstiff if care is not taken.

A possible solution is to use generic geometric representations such as convex decomposition, triangular mesh [16], signed distance field [17]. Common to these approaches is that many contact points can be generated for geometrically-complex objects. Too many contact points reduce simulation speed and potentially cause numerical instability. To address this problem, several works have proposed contact reduction methods to limit the number of contact points [158], [16], [17]. However, none of these works explore how contact reduction methods influence simulation accuracy, which is crucial for sim-to-real reinforcement learning.

In this chapter, we present a contact reduction method with bounded stiffness to improve the simulation accuracy. Our method is beneficial when the simulation consists of stiff rigid bodies, in which cases we argue that the number of contact points greatly influences simulation accuracy. Compared to previous works, our method includes an additional post-processing step, which relies on the concept of *contact stiffness*. We show that the proposed method enables training RL policy for a tight-clearance double pin insertion task and successfully deploying the policy on a rigid, position-controlled robot.

6.2 Background

6.2.1 Contact simulation pipeline and contact clustering

A typical rigid body contact simulation pipeline consists of three main steps

- Collision detection checks whether two bodies overlap.
- Contact generation determines a representation for the contact region, commonly in the form of a finite set of contact points. A contact point is defined by its position, normal direction, and possibly penetration depth.
- Contact response finds motion of rigid bodies to prevent interpenetration. Solution methods include complementarity-based approaches [175] and complementarity-free approaches [153].

If two bodies are represented by mesh, collision detection algorithms often work by decomposing meshes into primitive shapes (e.g. cylinder, box, sphere), triangles, or convex parts. Contact points are then generated independently for each of these elements. In this way, many contact points can be generated. The number of contact points greatly influences the accuracy, stability, and speed of the simulation. A reasonable number of contact points are needed to accurately realize the contact region, while too many contact points lead to expensive simulation and may cause numerical instability [158], [17].

To avoid expensive simulations and improve simulation stability, contact clustering reduces the number of contact points obtained by the contact generation step. This

method works by putting similar contacts into groups using some heuristics, then choosing/recomputing from each group one or several representative contact points.

6.2.2 Reinforcement learning

In RL, an agent learns to maximize the total reward received through interacting with its environment. A discounted episodic RL problem can be formalized as a Markov Decision Process (MDP) [161]. In MDP, an agent interacts with its environment in discrete time steps. At each time step t, the agent observes current state $\mathbf{s}_t \in \mathcal{S}$, executes an action $\mathbf{a}_t \in \mathcal{A}$, and receives an immediate reward r_t. The environment evolves through the state transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The goal in RL is to learn a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ that maximizes the expected discounted return $R = \sum_{t=1}^{T} \gamma^t r_t$, where γ is the discount factor.

6.3 Contact Reduction with Bounded Stiffness

Our proposed contact reduction method consists of two steps. In the first step, k-means clustering is used to reduce the number of contact points to a user-defined number k. In the second step, the contact stiffnesses of all the contact points are determined by a quadratic program, such that the net stiffness is upper bounded.

6.3.1 Contact clustering

Each contact point is represented by a 6D vector $[\mathbf{n}, \mathbf{x}]$ where $\mathbf{n}$ is the contact normal, and $\mathbf{x}$ is the contact position. The axis-weighted distance metric [16] is used. Specifically the distance between two points $[n_1, p_1]$ and $[n_2, p_2]$ is computed by

$$d = ||n_2 - n_1||_2^2 + c||p_2 - p_1||_2^2 \tag{6.1}$$

The centers of cluster are initialized with a deterministic version of k-means++ algorithm [176]. This algorithm helps avoid suboptimal clustering by spreading out the initial clusters. Although the initialization takes extra time, the main k-means algorithm quickly converge and thus the computation time is actually faster.

6.3.2 Example: Direct force control of a position-controlled manipulator

To see why the net stiffness should be bounded, consider the direct force control of a position-controlled manipulator. Model-based design methods require a model of the environment. A common model considers the robot as a single point or a small region and represents the environment as an n-dimensional spring ($n \leq 6$), or a spring and a damper [46], [177], [52]. In the former case, the model has the following form

$$\mathbf{F}_e = \mathbf{K}_e(\mathbf{x}_e - \mathbf{x}) \tag{6.2}$$

where $\mathbf{K}_e$ is n-dimensional matrix representing the stiffness of the environment, $\mathbf{x}_e$ is the robot position just before contact. It is usually assumed that the environment stiffness in different directions are uncoupled. In this case, $\mathbf{K}_e$ is a diagonal matrix, and equation (6.2) is replaced by n scalar equations

$$F_e = K_e(x_e - x) \tag{6.3}$$

Controllers designed using this simple model has proven to be effective even in applications involving complex contact scenarios such as hand guiding [52], assembly [177]. When the stiffness K_e is unknown, it can be estimated online or offline [178].

While the above contact model is useful for controller design and analysis, it is too simple for simulation purpose. In this context, the robot-environment interaction is typically represented by a finite number of contact points. Assuming frictionless contact, the contact force at each contact point is

$$\mathbf{F}_i = F_{ni}\mathbf{n}_i \tag{6.4}$$

The contact force between robot and environment can then be computed

$$\mathbf{F}_e = \sum \mathbf{F}_i \tag{6.5}$$

Many approaches have been proposed for the computation of F_n. In this example, we focus on the spring model which defines the normal component of contact force as follows

$$F_{ni} = K_i\delta_i \tag{6.6}$$

where K_i is the *contact stiffness*, δ_i is the penetration depth calculated by the contact generation step. It's common to use a single value for all contact points.

The penetration depth relates to the robot position $\mathbf{x}$ through the equation

$$\delta_i = \mathbf{n}_i^T(\mathbf{x} - \mathbf{x}_0) \tag{6.7}$$

where $\mathbf{x}_0$ is the robot position before contact. From (6.4)-(6.7), the contact force can be written as

$$\mathbf{F}_e = K \sum \mathbf{n}_i \mathbf{n}_i^T (\mathbf{x} - \mathbf{x}_0) \tag{6.8}$$

which has the form of 6.2 with the equivalent environment stiffness

$$\mathbf{K}_e = K \sum \mathbf{n}_i \mathbf{n}_i^T \tag{6.9}$$

From equation (6.9), it can be inferred that the net stiffness may vary significantly during the course of simulation when the contact points change position and normal direction. Specifically, the stiffness along one direction can be any value in the range $[0, NK]$ where N is the number of contact points. Therefore, the environment may appear stiffer or softer depending on the simulation state. For example, consider the collision between a cylinder and a convex-decomposed object as shown Fig 6.1, convex decomposition may cause "redundant" contact points which add stiffness to the system.

6.3.3 Scaling contact stiffness

Our main idea is to limit the net stiffness of the rigid bodies. We propose to scale contact stiffness such that the net stiffness of the system in (6.9) is upper bounded by K_{max}. The scaling coefficients s_i are computed by solving the following quadratic program

$$\begin{aligned} \min_{s_i} \quad & \sum (s_i - 1)^2 \\ s.t. \quad & k \sum s_i \mathbf{c}_{ij} \leq K_{max}, \quad j = 1, 2, 3 \end{aligned} \tag{6.10}$$

where $\mathbf{c}_i = diag(\mathbf{n}_i \mathbf{n}_i^T)$ is the stiffness induced by contact i (ignore the coupling stiffness between different directions), $\mathbf{c}_{ij}$ is the j component of $\mathbf{c}_i$. The scaling coefficient is optimized such that the change in contact stiffness is minimized. In

theory, setting K_{max} to the contact stiffness K is a good choice. Using a larger value of K_{max} can also be beneficial for, as an example, learning robust RL policy.

6.4 Learning contact-rich tasks with Position-controlled robots

6.4.1 Modeling of position-controlled robot

Position-controlled robots come with a joint position controller. User access to this controller is typically limited or unavailable [13]. Following the work by Hung et al. [52], we assume decoupled robot dynamics and the dynamics of each joint is modeled as a first-order linear time-invariant system with time delay. An implicit assumption of this model is that the effect of contact force on the inner joint position controller is negligible, which is a common assumption in the literature [13]. As a direct consequence, the robot is assumed to be much stiffer than the environment. In other words, a very small position error may result in a huge interaction force. Motivated by this model, we model the joint position controller of the robot with a *computed torque controller*

$$\tau = M(\ddot{q}_d + D(\dot{q}_d - q) + K(q_d - q)) + \tau_{ext} + C(q, \dot{q}) + g(q) \tag{6.11}$$

where M is the joint inertia matrix, $q, \dot{q}$ is the joint position, joint velocity respectively, $q_d, \dot{q}_d, \ddot{q}_d$ is the desired joint position, desired joint velocity, and desired joint acceleration respectively, τ_{ext} is the external torque, C is Coriolis and centrifugal torque, and g is the gravitational torque. K and D are diagonal gain matrices. Compared to [52], controller (6.11) also results in a decoupled dynamics, but the dynamics of each joint is a second-order system instead of first-order.

6.4.2 Reinforcement learning framework

The reinforcement learning framework includes two main components: a parallel position/force controller and an RL policy. The parallel position/force controller comprises a velocity control loop and a force control loop. The velocity control

loop is simply a feedforward controller $u_v = \int v_d dt$, while the force control loop is a Proportional-Integral controller $u_f = k_p(f_d - f) + k_i \int f_d - f dt$. The outputs of two control loops are added to obtain the commanded Cartesian pose. Finally, the commanded joint position is obtained through Differential Inverse Kinematics.

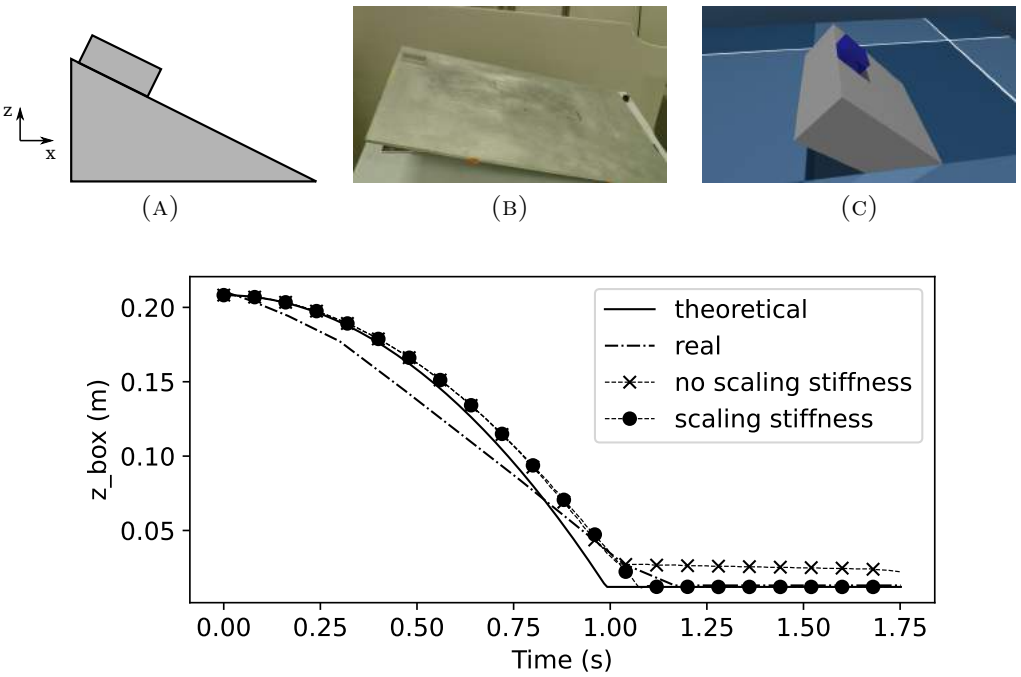
The action of RL policy is the desired velocity and desired force, the two inputs of the position/force controller. Note that it is a common practice that RL action are chosen to be the input of a high-level controller [107], [179], [109]. The state is the end-effector pose (with axis angle as the representation of orientation) and the external force acting on the end-effector. End-effector poses can be measured from the joint encoders and the known kinematics model, while the external force can be measured using a six-axis force/torque sensor attached between the robot flange and the end-effector.

The policy is parameterized by a neural network and trained with the Proximal Policy Optimization (PPO) algorithm [87]. PPO also trains an additional value function, which maps observation to the value of the current observation. The value function is also parameterized by a neural network

Since the value function is only used during training, we use an Asymmetric Actor-Critic [128] approach. Asymmetric Actor-Critic exploits the fact that the value function can have access to information that is not available on the real robot system (for instance, error-free poses of objects in the scene). The additional input potentially accelerates the learning of good value estimates since less information needs to be inferred.

6.5 Experiments

The aims of the experiments are to (1) validate the performance and advantages of the proposed contact reduction method in terms of speed and stability (2) demonstrate sim-to-real transfer for a tight-clearance cylindrical pin insertion task and a double pin insertion task.



(A) (B) (C)

(D)

FIGURE 6.2: (a) Illustration of a simple example: a box slides down an inclined plane, (b) Real experiment, (c) The example is simulated in Mujoco, (d) The evolution of the position of the box center along the z axis over time in three cases. With scaling stiffness, the trajectory of the box position closely matches theoretical solution. Without scaling stiffness, the box stuck on the plane.

6.5.1 Contact reduction performance

The proposed contact reduction method was implemented into Mujoco physics engine [153] and evaluated in several scenarios. First we show that the proposed method improve the accuracy of multiple-contacts simulation in a simple, yet ubiquitous case. The scenario includes a box sliding on a inclined plane as shown in Fig 6.2a. In Mujoco, this scenario can be simulated using the box shapes (see Fig 6.2c). Using this primitive shape generates at most four contact point when the box slide on the surface. To simulate multiple contacts, the inclined plane was modeled with a mesh instead of the primitive box shape. The mesh was decomposed into smaller parts, each of which can generate one contact point with the box. The decomposition was done in such a way that the number of contact points increase as the box slides down the plane and at most 512 contact points could be generated at a time. Note that the decomposition was done for the purpose

of evaluating the method in multiple-contacts simulation. Although the decomposition is unnecessary to model a plane, it is required to model objects containing concave features such as holes.

We run simulation in two settings: without scaling stiffness, and with scaling stiffness. The evolution of the box center's position is recorded in Fig 6.2d. The theoretical solution can be easily obtained by Newton's second law and is used as the reference. We also carried out a real experiment for this problem. The progress was recorded by a Panasonic HC-X920M camera. The box position were then estimated roughly from the video. With scaling stiffness, the trajectory of the box center closely follow theoretical solution. The box eventually reaches the ground, which comply with the theoretical and real results. On the other hand, the box stuck on the plane without scaling stiffness. The reason is that the contact force between the cylinder and the plane became too large, causing a large friction force. On the other hand, our method maintains the contact force regardless of the number of contacts, thus make the simulator more realistic.

Next we evaluate how the proposed method affect simulation speed in several assembly scenes. Each scene includes a 6-axis Denso VS-060 robot and two mating parts. The joint position controller and the hybrid motion/force controller were implemented as described in Section 6.4.1 and Section 6.4.2, respectively. The collision geometries of the mating parts were modeled with primitive shapes (cylinders, boxes, spheres) if possible; otherwise, they were decomposed into multiple convex hulls manually or by V-HACD [180]. One part was attached rigidly to the robot end-effector (grasping was not simulated), and the other part was rigidly placed in the environment. The scenes are described as follows

USB insertion The USB female and male meshes were sourced from the manufacturer. Both parts were decomposed into 200 pieces using V-HACD.

Round peg insertion The round peg was modeled using the cylinder shape in Mujoco. The round hole was manually decomposed into 100 convex hulls.

Nut and bolt assembly Both the nut and bolt were decomposed into 1500 pieces using V-HACD

The number of clusters was set to 10 in all scenes. We manually designed motion scripts to simulate the insertion phase and record the average number of contacts,

	Number of contact points	Collision detection time (ms)		Contact response time (ms)	
		Baseline	Proposed	Baseline	Proposed
USB insertion	16	1.071	1.072	0.74	0.22
Round pin insertion	21	0.23	0.232	0.17	0.034
Nut and bolt assembly	204	16.4	16.402	0.45	0.24

TABLE 6.1: Influence of the the proposed method on simulation speed

average collision detection time, and average contact response time over all simulation steps in Table 6.1. We observe that the proposed contact reduction algorithm adds a very small extra time (about 0.002 ms in the three scenes) to the collision detection time. Owing to the reduced number of contacts, our method reduces the contact response time by 0.52 ms, 0.136 ms, and 0.21 ms in the USB insertion, round peg insertion, and the nut and bolt assembly, respectively. Since the gain in contact response performance far exceeds the loss in the collision detection, the proposed method significantly improve simulation speed.

6.5.2 Reinforcement learning and sim-to-real tranfer result

In this section, we show successful sim-to-real transfer results for two tasks: the round pin insertion and the double round pin insertion task. The round pin insertion task is similar to the scene used in the previous section. The double pin insertion task is more challenging due to the yaw error. The number of clusters was set to four for the round pin insertion task and ten for the double pin insertion tasks. The maximum net stiffness K_{max} was set to twofold the unscaled contact stiffness.

Real experiments were carried out on a 6-axis Denso VS-060 robot. An ATI Gamma force/torque sensor was rigidly attached to the end-effector to provide contact force measurements. A personal computer running Ubuntu 16.04 was used to send commands to the robot and train RL policy. For both tasks, the pegs were rigidly attached to the force torque sensor. The holes were rigidly mounted within the robot's workspace. For the round pin insertion task, the peg and hole were made of aluminum with 0.05 mm clearance. For the double round pin insertion, the parts were 3D printed and the clearance was measured to be 0.2 mm. The hardware

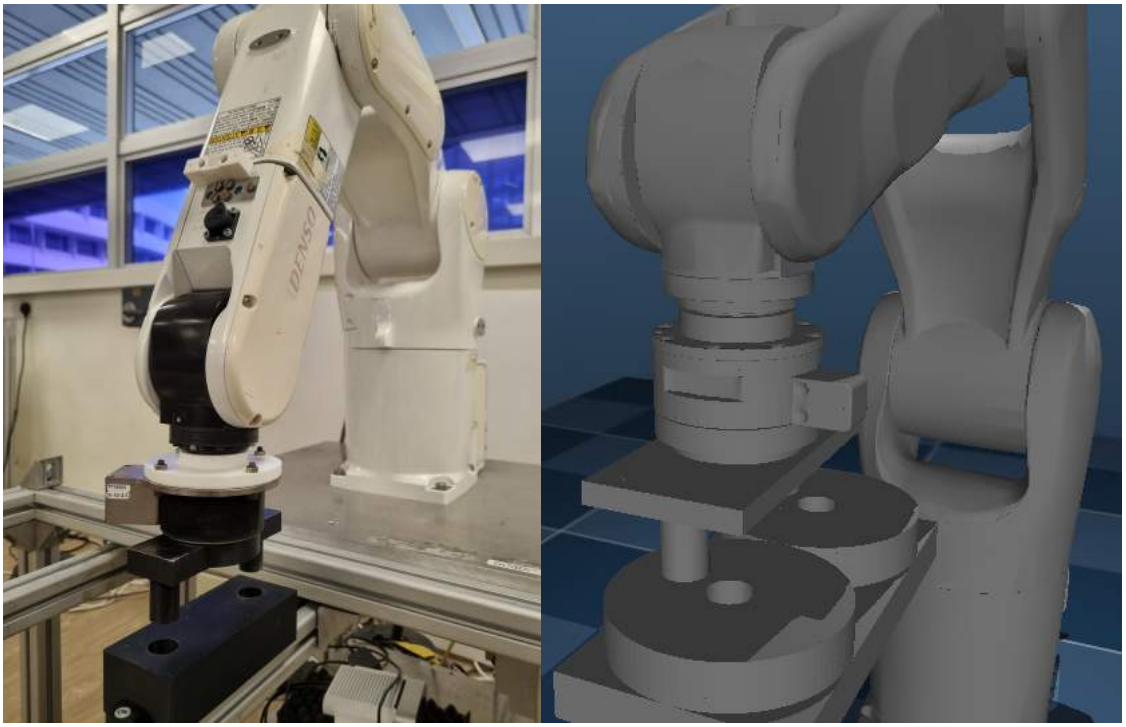


FIGURE 6.3: Hardware setup (left) and the corresponding simulated environment of the double pin insertion task (right)

setup and corresponding simulated environment for the double pin insertion task are shown in Fig 6.3

For both tasks, the following reward function was used

$$r = \frac{z - z_0}{D} - 1 + r_f \tag{6.12}$$

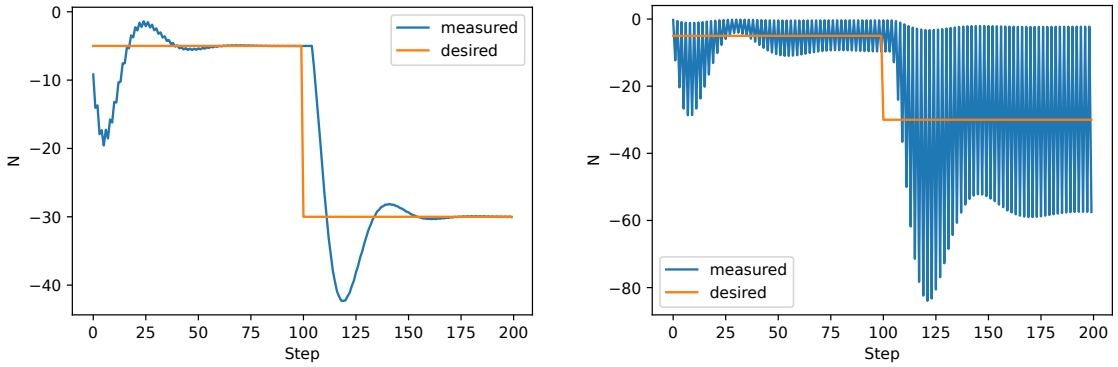
$$r_f = \begin{cases} -2 & \text{if } \max(\mathbf{f} - \mathbf{f}_u) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.13}$$

where z is the end-effector position along the $z-$axis (assumed to be the insertion axis), z_0 is the end-effector position when it first touches the hole (i.e. when a force along the $z-$axis is sensed), D is the insertion depth, $\mathbf{f}$ and $\mathbf{f}_u$ is the measured contact force and the upper limit force.

Each policy was trained for each task for 500 epochs. At the beginning of each training episode, the robot is reset to a random configuration such that the relative position between the peg and the hole is less than 2 mm (except for the insertion direction) and the relative orientation error is less than 1 deg.

TABLE 6.2: Sim-to-real result

Task	Success rate	Average completion time (s)
Round pin insertion	1	2.31 ± 0.47
Double pin insertion	0.95	2.87 ± 0.33



(A) Four contact points

(B) Six contact points

FIGURE 6.4: Without scaling stiffness, the force controller becomes instable when the number of contact increases. The robot is commanded such that the peg comes into contact with the hole surface along the surface's normal. Input desired forces of 5 N and 30 N are then sequentially sent to the force controller.

After training, each policy was evaluated 20 times on the real robot. The success rate and average completion time are reported in Table 6.2. The only failure in the double pin insertion task is because the peg drifts away from the initial pose and fails to recover. The learned strategy is also quite intuitive. The peg is tilted to easier align with the hole rim, followed by an oscillating motion to insert the peg.

6.5.3 Effect of scaling contact stiffness

First, we experimentally show that the environment may appear stiffer due to the increase in the number of contacts as described in Section 6.3.2. The round pin insertion scene was used for this purpose. The robot was reset to a configuration such that the angle between the peg's surface and the hole's surface is one degree, then commanded to come into contact with the hole surface along the surface's normal. A constant desired force of 5 N and 30 N is successively sent to the force controller. The number of contact points depends on the number of convex shapes composing the hole.

(A) Success rate

(B) Return

FIGURE 6.5: Comparison of training performance for different number of clusters k in the set $2, 4, 6, 8, 10$ without scaling stiffness. We also show training performance for $k = 10$ with scaling stiffness (the proposed method).

As shown if Fig 6.4, the force controller becomes instable with six contact points. To prevent instability, the number of clusters should be four or less. However, this may compromise simulation accuracy. For instance, consider the double pin insertion task, at least six contact points are required to account for all possible contact configurations.

We also explore how the number of contact points influences RL training performance. Several policies were trained for different numbers of clusters in the set

TABLE 6.3: Sim-to-real result for different number of clusters, with or without scaling stiffness

	Success rate	Average completion time
2 clusters	0	-
4 clusters	1	3.63 ± 0.72
10 clusters, scaling stiffness	0.95	2.87 ± 0.33

$2, 4, 6, 8, 10$ without scaling stiffness. The training curve is shown in Fig 6.5. As the number of clusters increases, RL training converges slower and the final success rate and return also decline. With $k \geq 6$ clusters, the final success rate can't reach 100 %. The worse RL performance is possibly due to simulation instability, causing the force/torque reading to be noisy and unstable.

Three policies reaching 100 % success rate are tested on the real robot with the same procedure in Section 6.5.2. The results are reported in Table 6.3. Most noteworthy is that the policy trained our proposed method reduces 20% average task completion time as compared to the policy trained with $k = 4$ clusters, while achieves comparable success rate.

The policy trained with $k = 2$ clusters can only partially insert the peg but is unable to fully insert it. A possible reason is that during the insertion phase, the most dominant contact configuration is four-point contact, so two clusters are not enough to accurately represent the actual contact configuration. This manifests into inaccurate contact force/torque reading, causing the policy to fail to transfer.

Suprisingly, the policy trained with $k = 4$ clusters achieves 100% success rate. This may be because the five- or six-point contact configurations only happen when the peg is tilted, which rarely occurs. As the six-point contacts are indistinguishable from four-point contacts, the learned policy mostly selects translational motion during the search phase, in contrast to the tilting strategy learned by the policy trained with $k = 10$ clusters. The translational motion during the search phase seems to be random, which explains the worse completion time than the tilting strategy.

6.6 Conclusions

In this chapter, we have presented a contact reduction method with bounded stiffness to improve the simulation accuracy. Our method is particularly beneficial when simulating interaction between stiff objects. Our experimental results have shown that the proposed method improves simulation speed and that it enables training RL policy in simulation and deploying the trained policy on a challenging double pin insertion task using a position-controlled robot. In future works, we plan to extensively evaluate this method in different simulation scenarios and apply this method for learning different assembly tasks, such as cable insertion.

Chapter 7

Conclusion

7.1 Summary

The invention of Deep Reinforcement Learning has opened new opportunities into the field of robotics, along with new challenges. With the goal of tackling these challenges, this thesis has presented three contributions: a novel action representation, an experimental study on the role of low-level controller in a control system with RL in the loop, and a novel contact reduction method for multi-contact simulation.

A novel action representation for robot assembly

The choice of action representation is crucial for the success of RL. This thesis has introduced manipulation primitives as a novel action representation for learning high-precision robot assembly. Manipulation Primitives have enough complexity to keep the search tree shallow (typically a sequence of 6 to 8 MPs is enough to achieve tight insertion), yet are generic enough to generalize across a wide range of assembly tasks (peg insertion with different peg shapes, large hole estimation errors, random initial positions...). Another key advantage of MPs is their additional semantics, which make them robust in sim-to-real and against model/environment variations and uncertainties. Leveraging parameterized manipulation primitives, the proposed method was shown to significantly improve both assembly performance and sample efficiency of Reinforcement Learning. The experimental results

showed that policies learned purely in the simulation could consistently solve peg insertion tasks with different geometry and very small clearance.

Manipulation primitives assume the existence of a task frame and that the task frame can be estimated with high accuracy. Such an assumption limits the practicality of the proposed method. The thesis has shown that incorporating Deep Learning-based Visual Servoing can alleviate the assumption. The experimental result has demonstrated that DLVS can consistently estimate the task frame with high accuracy with a mild assumption that the object is in the field of view of an in-hand camera.

The role of low-level controller in a control system with RL in the loop

It is well known that choosing action as input to a low-level feedback controller can improve both sample efficiency and policy performance for locomotion and robot manipulation. While several studies have compared different classes of low-level controllers (e.g., joint position controller, end-effector controller), these works overlooked the fact that many implementations exist for each class of low-level controller.

This thesis has presented an experimental study on the influence of low-level controller implementation on training performance and policy performance. Particularly, we focus on direct force control for contact-rich manipulation tasks with position-controlled robots. Three controllers were designed by two force control methods: Proportional-Integral controller and Convex Controller Synthesis (CCS). Our experiments, both in simulation and in the real world, suggest that high-bandwidth controllers can improve policy performance in a tight-clearance industrial peg-in-hole task. In addition, a robust controller can improve the robustness of the learned policy. A sim-to-real experiment further validates the above hypotheses on the physical system. This results might give a new perspective for the integration of RL into robot control systems: we can improve the task performance and robustness of learned policies by designing a better low-level controller.

A novel contact reduction method for multi-contact simulation

The main challenge of sim-to-real transfer is bridging the reality gap. Representing contact surfaces with generic geometric representation is one step towards this goal. However, the simulator might become slow and inaccurate due to many contact points being generated. To tackle this issue, this thesis has presented a contact reduction method with bounded stiffness. Our approach is particularly beneficial when simulating the interaction between stiff objects. The experimental results have shown that the proposed method improves simulation speed and enables training RL policy in simulation and deploying the trained policy on a challenging double-pin insertion task using a position-controlled robot.

7.2 Future works

This section discusses some future directions that we think are exciting and most impactful towards the application of RL in real-world robot assembly and, more general, robot manipulation.

Semi-autonomous sim-to-real pipeline

A recurring theme in this thesis is the use of sim-to-real methods. An exciting future direction is to build a semi-autonomous sim-to-real pipeline that allows a user to construct the simulated environment quickly, design RL problem efficiently, train transferrable RL policies, and finally deploy it on the real robot. This would require breakthroughs in the sim-to-real method, the development of open-source software, and advances in simulation modeling.

Using generic geometric representation is a crucial step in improving modeling accuracy. We are convinced that SDF-based representation is the most promising approach owing to its unmatched efficiency in distance calculation while still providing a fine approximation [17]. SDFs require a time-consuming and memory-demanding pre-computation phase, but this can be alleviated by exploiting powerful computers, which are becoming more and more accessible. Our next probable

step would be integrating SDFs with the contact reduction methods described in Chapter 6.

Aside from ongoing research in sim-to-real methods for bridging sim-to-real gaps, open-source software development is also critical. Open-source software, such as `gymnasium` [181], `dm_control` [182], or `mujoco_menagerie` [183], significantly increases the efficiency of constructing simulation instances and designing RL problems.

New tasks and variations

The experiments in this thesis mainly involve high-precision peg-in-hole assembly with different shapes, sizes, and tolerances. However, these tasks constitute only a small portion of the assembly variations seen in the real world. Some variations of objects that are not considered in this work include gears, industrial pistons in automobile engines (the clearance can be as low as $2.5 - 7.5\,\mu$m), industrial cables, electrical components in a PCB board,... We are optimistic that the proposed methods in this thesis are applicable to these tasks, but more experiments are needed to validate this hypothesis.

New fronts in RL

Meta Reinforcement Learning [184, 185, 143] and Multi-task Reinforcement Learning [186, 187] are two promising techniques to improve the efficiency of RL in the face of task variations. Both methods require training samples from multiple tasks. The difference is that the former performs a meta-training phase such that subsequent RL training are more sample efficiency for tasks that are similar to the ones during training. In contrast, the latter aims to learn a single policy that works for all the tasks and possibly generalizes to similar ones. One downside of both methods is their sample inefficiency. In addition, collecting data for multiple tasks are extremely tedious. For these reasons, learning in simulation and sim-to-real transfer will become central to these approaches.

Bibliography

[1] F. Suárez-Ruiz and Q.-C. Pham, "A framework for fine robotic assembly," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 421–426. xi, 1, 9, 13, 29, 51

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. 2, 17, 63

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017. 2

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971 [cs, stat]*, Jul. 2019. 2, 17

[5] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," *arXiv:1804.10332 [cs]*, May 2018. 2, 22

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, Jan. 2019. 2, 63

[7] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, Jan. 2020. 2, 4, 19, 23, 63, 77

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016. 2, 4, 19, 63, 66

[9] X. B. Peng and M. van de Panne, "Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter?" *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–13, Jul. 2017. 2, 19, 20, 63, 64, 65

[10] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3389–3396. 2, 19, 63, 66

[11] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual Reinforcement Learning for Robot Control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6023–6029. 2, 17, 19, 64, 67

[12] Z. Kalmár, C. Szepesvári, and A. Lorincz, "Modular Reinforcement Learning: An Application to a Real Robot Task," in *Learning Robots*, ser. Lecture Notes in Computer Science, A. Birk and J. Demiris, Eds. Berlin, Heidelberg: Springer, 1998, pp. 29–45. 3, 18

[13] J. Roy and L. Whitcomb, "Adaptive force control of position/velocity controlled robots: Theory and experiment," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 121–137, Apr. 2002. 4, 20, 21, 64, 69, 83

[14] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and empirical comparison," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008. 4, 21, 64

[15] J. Lehman, J. Clune, D. Misevic, C. Adami, L. Altenberg, J. Beaulieu, P. J. Bentley, S. Bernard, G. Beslon, D. M. Bryson, N. Cheney, P. Chrabaszcz, A. Cully, S. Doncieux, F. C. Dyer, K. O. Ellefsen, R. Feldt, S. Fischer, S. Forrest, A. Fŕenoy, C. Gagńe, L. Le Goff, L. M. Grabowski, B. Hodjat, F. Hutter, L. Keller, C. Knibbe, P. Krcah, R. E. Lenski, H. Lipson, R. MacCurdy, C. Maestre, R. Miikkulainen, S. Mitri, D. E. Moriarty, J.-B. Mouret, A. Nguyen, C. Ofria, M. Parizeau, D. Parsons, R. T. Pennock, W. F. Punch, T. S. Ray, M. Schoenauer, E. Schulte, K. Sims, K. O. Stanley, F. Taddei, D. Tarapore, S. Thibault, R. Watson, W. Weimer, and J. Yosinski, "The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities," *Artificial Life*, vol. 26, no. 2, pp. 274–306, May 2020. 5

[16] K. Hauser, "Robust Contact Generation for Robot Simulation with Unstructured Meshes," in *Robotics Research: The 16th International Symposium ISRR*, ser. Springer Tracts in Advanced Robotics, M. Inaba and P. Corke, Eds. Cham: Springer International Publishing, 2016, pp. 357–373. 6, 26, 78, 80

[17] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory:

Fast Contact for Robotic Assembly," in *Proceedings of Robotics: Science and Systems*, Jun. 2022. 6, 26, 27, 78, 79, 95

[18] S. N. Simunovic, "Force information in assembly processes," in *Proc. 5th Int. Symp. Industrial Robots*. IFS Publications Bedford, UK, 1975, pp. 415–431. 9, 10

[19] D. E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 1, pp. 65–77, 1982. 9, 10

[20] R. Hollis, S. Salcudean, and A. Allan, "A six-degree-of-freedom magnetically levitated variable compliance fine-motion wrist: Design, modeling, and control," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 320–332, Jun. 1991. 9, 10

[21] S. Joo and F. Miyazaki, "Development of variable RCC and its application," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, vol. 2, Oct. 1998, pp. 1326–1332 vol.2. 9, 10

[22] R. Sturges and S. Laowattana, "Fine motion planning through constraint network analysis," in *Proceedings. IEEE International Symposium on Assembly and Task Planning*, Aug. 1995, pp. 160–170. 9, 10

[23] M. Peshkin, "Programmed compliance for error corrective assembly," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 473–482, Aug. 1990. 9, 12

[24] J. Schimmels, "A linear space of admittance control laws that guarantees force-assembly with friction," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 656–667, Oct. 1997. 9, 12

[25] S. Hirai, T. Inatsugi, and K. Iwata, "Learning of admittance matrix elements for manipulative operations," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 2, Nov. 1996, pp. 763–768 vol.2. 9, 12

[26] H. Asada, "Representation and learning of nonlinear compliance using neural nets," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 6, pp. 863–867, Dec. 1993. 9, 12

[27] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots," Dec. 1983. 9

[28] M. Erdmann, "Using Backprojections for Fine Motion Planning with Uncertainty," p. 27. 9

[29] C. Laugier, "Planning fine motion strategies by reasoning in the contact space," in *1989 International Conference on Robotics and Automation Proceedings*, May 1989, pp. 653–659 vol.2. 9, 12

[30] B. McCarragher and H. Asada, "A discrete event approach to the control of robotic assembly tasks," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 331–336 vol.1. 9, 12

[31] H. Hirukawa, Y. Papegay, and T. Matsui, "A motion planning algorithm for convex polyhedra in contact under translation and rotation," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 3020–3027 vol.4. 9, 12

[32] J. Xiao and X. Ji, "Automatic Generation of High-Level Contact State Space," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 584–606, Jul. 2001. 9, 12

[33] X. Ji and J. Xiao, "Planning Motions Compliant to Complex Contact States," *The International Journal of Robotics Research*, vol. 20, no. 6, pp. 446–465, Jun. 2001. 9, 13

[34] B. Finkemeyer, T. Kröger, and F. M. Wahl, "Executing assembly tasks specified by manipulation primitive nets," *Advanced Robotics*, vol. 19, no. 5, pp. 591–611, Jan. 2005. 9, 13

[35] S. Chhatpar and M. Branicky, "Search strategies for peg-in-hole assemblies with position uncertainty," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 3. Maui, HI, USA: IEEE, 2001, pp. 1465–1470. 9, 13

[36] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A Framework for Robot Manipulation: Skill Formalism, Meta Learning and Adaptive Control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5844–5850. 9, 13, 30, 31, 32, 46, 51

[37] G. Dakin and R. Popplestone, "Simplified fine-motion planning in generalized contact space," in *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, Aug. 1992, pp. 281–286. 9, 13

[38] ——, "Contact space analysis for narrow-clearance assemblies," in *Proceedings of 8th IEEE International Symposium on Intelligent Control*, Aug. 1993, pp. 542–547. 9, 13

[39] J. Rosell, L. Basaniz, and R. Suarez, "Compliant-motion planning and execution for robotic assembly," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, May 1999, pp. 2774–2779 vol.4. 9, 13

[40] D. E. Whitney and J. M. Rourke, "Mechanical Behavior and Design Equations for Elastomer Shear Pad Remote Center Compliances," *Journal of Dynamic Systems, Measurement, and Control*, vol. 108, no. 3, pp. 223–232, Sep. 1986. 10

[41] S. Lee, "Development of a New Variable Remote Center Compliance (VRCC) With Modified Elastomer Shear Pad (ESP) for Robot Assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 2, pp. 193–197, Apr. 2005. 10

[42] S. Kilikevicius and B. Bakšys, "Analysis of insertion process for robotic assembly," *Journal of Vibroengineering*, vol. 9, no. 4, pp. 35–40, Dec. 2007. 10

[43] S. Kilikevicius and B. Baksys, "Dynamic analysis of vibratory insertion process," *Assembly Automation*, vol. 31, no. 3, pp. 275–283, Jan. 2011. 10

[44] M. T. Mason, "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981. 10

[45] N. Hogan, "Impedance Control: An Approach to Manipulation: Part I—Theory," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 1–7, Mar. 1985. 10, 66

[46] J. De Schutter and H. Van Brussel, "Compliant Robot Motion II. A Control Approach Based on External Control Loops," *The International Journal of Robotics Research*, vol. 7, no. 4, pp. 18–33, Aug. 1988. 11, 81

[47] M. H. Raibert and J. J. Craig, "Hybrid Position/Force Control of Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 126–133, Jun. 1981. 11, 66

[48] T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators–Description of hand constraints and calculation of joint driving force," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 386–392, Oct. 1987. 11

[49] N. McClamroch and D. Wang, "Feedback stabilization and tracking of constrained robots," *IEEE Transactions on Automatic Control*, vol. 33, no. 5, pp. 419–426, May 1988. 11

[50] J. Mills and A. Goldenberg, "Force and position control of manipulators during constrained motion tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 30–46, Feb. 1989. 11

[51] L. Villani, C. De Wit, and B. Brogliato, "An exponentially stable adaptive control for force and position tracking of robot manipulators," *IEEE Transactions on Automatic Control*, vol. 44, no. 4, pp. 798–802, Apr. 1999. 11

[52] H. Pham and Q.-C. Pham, "Convex controller synthesis for robot contact," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3330–3337, 2020. 11, 50, 64, 68, 69, 72, 81, 83

[53] D. E. Whitney, "Force Feedback Control of Manipulator Fine Motions," *Journal of Dynamic Systems, Measurement, and Control*, vol. 99, no. 2, pp. 91–97, Jun. 1977. 11

[54] Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers*, vol. C-32, no. 2, pp. 108–120, Feb. 1983. 12

[55] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996. 13

[56] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the "task frame formalism"-a synthesis," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 581–589, Aug. 1996. 13

[57] T. Kroger, B. Finkemeyer, and F. Wahl, "A task frame formalism for practical implementations," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 5, Apr. 2004, pp. 5218–5223. 13

[58] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, May 2007. 13

[59] M. Skubic and R. Volz, "Identifying single-ended contact formations from force sensor patterns," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 597–603, Oct. 2000. 13

[60] J. Xiao and L. Zhang, "Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives," *Robotics and Automation, IEEE Transactions on*, vol. 13, pp. 456–466, Jul. 1997. 13

[61] B. Hannaford and P. Lee, "Hidden Markov Model Analysis of Force/Torque Information in Telemanipulation," *The International Journal of Robotics Research*, vol. 10, no. 5, pp. 528–539, Oct. 1991. 13

[62] G. E. Hovland and B. J. McCarragher, "Hidden Markov Models as a Process Monitor in Robotic Assembly," *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 153–168, Feb. 1998. 13

[63] E. Cervera, A. P. Del Pobil, E. Marta, and M. A. Serna, "Perception-based learning for motion in contact in task planning," *Journal of Intelligent and Robotic Systems*, vol. 17, no. 3, pp. 283–308, Nov. 1996. 13

[64] M. Nuttin and H. Van Brussel, "Learning the peg-into-hole assembly operation with a connectionist reinforcement technique," *Computers in Industry*, vol. 33, no. 1, pp. 101–109, Aug. 1997. 13

[65] L. Brignone and M. Howarth, "A geometrically validated approach to autonomous robotic assembly," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Sep. 2002, pp. 1626–1631 vol.2. 13

[66] W. C. J. C. H, "Learning from Delayed Rewards," *PhD thesis, Cambridge University, Cambridge, England*, 1989. 15

[67] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. 15

[68] R. G. A, "On-line Q-learning Using Connectionist Systems," *Technical Report*, 1994. 15

[69] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992. 16

[70] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation." in *NIPs*, vol. 99. Citeseer, 1999, pp. 1057–1063. 16, 17

[71] M. Toussaint, A. J. Storkey, and S. Harmeling, "Expectation-Maximization Methods for Solving (PO)MDPs and Optimal Control Problems." in *Bayesian Time Series Models*. Cambridge University Press, 2011, pp. 388–413. 16

[72] G. Neumann, "Variational inference for policy search in changing situations," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, Jun. 2011, pp. 817–824. 16

[73] J. Peters and S. Schaal, "Policy Gradient Methods for Robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China: IEEE, Oct. 2006, pp. 2219–2225. 16, 17

[74] T. Rückstieß, M. Felder, and J. Schmidhuber, "State-Dependent Exploration for Policy Gradient Methods," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Heidelberg: Springer, 2008, pp. 234–249. 16

[75] H. J. Kappen, "Path integrals and symmetry breaking for optimal control theory," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 11, p. P11011, Nov. 2005. 16

[76] S. Kakade and J. Langford, "Approximately Optimal Approximate Reinforcement Learning," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 2002, pp. 267–274. 17

[77] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning Movement Primitives," in *Robotics Research. The Eleventh International Symposium*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Berlin, Heidelberg: Springer, 2005, pp. 561–572. 17

[78] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual Learning from Demonstration: Adapting DMPs for Contact-rich Manipulation," *arXiv:2008.07682 [cs]*, Sep. 2021. 17

[79] H. Kim, M. Jordan, S. Sastry, and A. Ng, "Autonomous Helicopter Flight via Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 16. MIT Press, 2003. 17

[80] J. Kober and J. Peters, "Policy Search for Motor Primitives in Robotics," in *Advances in Neural Information Processing Systems*, vol. 21. Curran Associates, Inc., 2008. 17

[81] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a Monte Carlo EM algorithm," *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, Aug. 2009. 17

[82] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv:1509.06461 [cs]*, Dec. 2015. 17

[83] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *arXiv:1511.05952 [cs]*, Feb. 2016. 17

[84] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1995–2003. 17

[85] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. 17

[86] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *arXiv:1502.05477 [cs]*, Apr. 2017. 17

[87] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Aug. 2017. 17, 38, 71, 84

[88] S. Fujimoto, H. Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul. 2018, pp. 1587–1596. 17

[89] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications," *arXiv:1812.05905 [cs, stat]*, Jan. 2019. 17

[90] H. Benbrahim, J. Doleac, J. Franklin, and O. Selfridge, "Real-time learning: A ball on a beam," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 1, Jun. 1992, pp. 98–103 vol.1. 18

[91] M. Tokic, W. Ertel, and J. Fessler, "The Crawler, A Class Room Demonstrator for Reinforcement Learning." Jan. 2009. 18

[92] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, BC: IEEE, Sep. 2017, pp. 819–825. 18, 29

[93] V. Gullapalli, J. Franklin, and H. Benbrahim, "Acquiring robot skills via reinforcement learning," *IEEE Control Systems Magazine*, vol. 14, no. 1, pp. 13–24, Feb. 1994. 18

[94] M. Huber and R. A. Grupen, "A feedback control structure for on-line learning tasks," *Robotics and Autonomous Systems*, vol. 22, no. 3, pp. 303–315, Dec. 1997. 19

[95] B. Nemec, M. Tamošiūnaitė, F. Wörgötter, and A. Ude, "Task adaptation through exploration and action sequencing," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, Dec. 2009, pp. 610–616. 19

[96] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning," *Machine Learning*, vol. 23, no. 2, pp. 279–303, May 1996. 19

[97] P. Fidelman, "Learning Ball Acquisition on a Physical Robot," 2004. 19

[98] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Autonomous Skill Acquisition on a Mobile Manipulator," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 1468–1473, Aug. 2011. 19

[99] ——, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, Mar. 2012. 19

[100] J. Peters and S. Schaal, "Learning to control in operational space," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008. 19, 63

[101] M. Deisenroth and C. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," in *ICML*, 2011. 19, 63

[102] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," *arXiv:1707.08817 [cs]*, Oct. 2018. 19, 63

[103] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and Imitation Learning for Diverse Visuomotor Skills," in *Robotics: Science and Systems XIV*, vol. 14, Jun. 2018. 19

[104] V. Gullapalli, R. Grupen, and A. Barto, "Learning reactive admittance control," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, May 1992, pp. 1475–1480 vol.2. 19, 64

[105] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang, "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality," May 2023. 19, 64

[106] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 4639–4644. 19

[107] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning Force Control for Contact-rich Manipulation Tasks with Rigid Position-controlled Robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, Oct. 2020. 19, 20, 21, 64, 65, 84

[108] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, Jun. 2011. 19, 21, 64, 65

[109] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," *arXiv preprint arXiv:1906.08880*, 2019. 19, 20, 21, 63, 64, 65, 84

[110] P. Varin, L. Grossman, and S. Kuindersma, "A Comparison of Action Spaces for Learning Manipulation Tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 6015–6021. 19, 20, 63, 64, 65

[111] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning Task Space Actions for Bipedal Locomotion," May 2021. 19

[112] J. Luo, Y. Zhao, D. Kim, O. Khatib, and L. Sentis, "Locomotion control of three dimensional passive-foot biped robot based on whole body operational space framework," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2017, pp. 1577–1582. 19

[113] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987. 19, 32

[114] C. Ott, "Cartesian Impedance Control: The Rigid Body Case," in *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, ser. Springer Tracts in Advanced Robotics, C. Ott, Ed. Berlin, Heidelberg: Springer, 2008, pp. 29–44. 20

[115] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta, "Efficient bimanual manipulation using learned task schemas," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1149–1155. 20

[116] S. Nasiriany, H. Liu, and Y. Zhu, "Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks," *arXiv:2110.03655 [cs]*, Oct. 2021. 20

[117] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer, "Learning to Compose Hierarchical Object-Centric Controllers for Robotic Manipulation," *arXiv:2011.04627 [cs]*, Nov. 2020. 20

[118] M. Dalal, D. Pathak, and R. Salakhutdinov, "Accelerating Robotic Reinforcement Learning via Parameterized Action Primitives," *arXiv:2110.15360 [cs]*, Oct. 2021. 20, 48

[119] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 3080–3087. 21, 65

[120] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov, "Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system," in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, May 2018, pp. 35–42. 22

[121] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "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. 22

[122] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid Motor Adaptation for Legged Robots," *arXiv:2107.04034 [cs]*, Jul. 2021. 22

[123] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Robotics: Science and Systems XVII*, vol. 17, Jul. 2021. 22

[124] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 59–66. 22

[125] F. Sadeghi and S. Levine, "CAD2RL: Real Single-Image Flight Without a Single Real Image," in *Robotics: Science and Systems XIII*, vol. 13, Jul. 2017. 22, 23

[126] R. Polvara, M. Patacchiola, M. Hanheide, and G. Neumann, "Sim-to-Real Quadrotor Landing via Sequential Deep Q-Networks and Domain Randomization," *Robotics*, vol. 9, no. 1, p. 8, Mar. 2020. 22, 23

[127] J. Matas, S. James, and A. J. Davison, "Sim-to-Real Reinforcement Learning for Deformable Object Manipulation," in *Proceedings of The 2nd Conference on Robot Learning*. PMLR, Oct. 2018, pp. 734–743. 22, 23

[128] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric Actor Critic for Image-Based Robot Learning," *Robotics*, 2018. 23, 84

[129] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," *arXiv:1810.05687 [cs]*, Mar. 2019. 23

[130] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active Domain Randomization," *arXiv:1904.04762 [cs]*, Jul. 2019. 23

[131] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient Domain Randomization with Bayesian Optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 911–918, Apr. 2021. 23

[132] M. Mozifian, J. C. G. Higuera, D. Meger, and G. Dudek, "Learning Domain Randomization Distributions for Training Robust Locomotion Policies," *arXiv:1906.00410 [cs, stat]*, Sep. 2019. 23

[133] F. Ramos, R. Carvalhaes Possas, and D. Fox, "BayesSim: Adaptive domain randomization via probabilistic inference for robotics simulators," Jun. 2019. 23

[134] R. Possas, L. Barcelos, R. Oliveira, D. Fox, and F. Ramos, "Online BayesSim for Combined Simulator Parameter Inference and Policy Improvement," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5445–5452. 23

[135] W. Yu, C. K. Liu, and G. Turk, "Policy Transfer with Strategy Optimization," in *International Conference on Learning Representations*, Sep. 2018. 24

[136] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-Real Transfer for Biped Locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3503–3510. 24

[137] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning Agile Robotic Locomotion Skills by Imitating Animals," Apr. 2020. 24

[138] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the Unknown: Learning a Universal Policy with Online System Identification," *arXiv:1702.02453 [cs]*, May 2017. 24

[139] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-Real Robot Learning from Pixels with Progressive Nets," *arXiv:1610.04286 [cs]*, May 2018. 24

[140] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," Oct. 2022. 24

[141] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta Reinforcement Learning for Sim-to-real Domain Adaptation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 2725–2731. 24

[142] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning," *arXiv:1803.11347 [cs, stat]*, Feb. 2019. 24

[143] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine, "Offline Meta-Reinforcement Learning for Industrial Insertion," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 6386–6393. 24, 96

[144] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model," *arXiv:1610.03518 [cs]*, Oct. 2016. 25

[145] J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone, "Grounded action transformation for sim-to-real reinforcement learning," *Machine Learning*, vol. 110, no. 9, pp. 2469–2499, Sep. 2021. 25

[146] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, "Sim-to-Real Transfer with Neural-Augmented Robot Simulation," in *Proceedings of The 2nd Conference on Robot Learning*. PMLR, Oct. 2018, pp. 817–828. 25

[147] M. Wulfmeier, I. Posner, and P. Abbeel, "Mutual Alignment Transfer Learning," *arXiv:1707.07907 [cs]*, Sep. 2017. 25

[148] E. Drumwright and J. J. Trinkle, "Contact Simulation," Jan. 2017, pp. 1–55. 25

[149] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," Mar. 2022. 26

[150] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller, "Small steps in physics simulation," in *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 1–7. 26

[151] "Open Dynamics Engine," http://www.ode.org/. 26

[152] "DART: Dynamic Animation and Robotics Toolkit," https://dartsim.github.io/. 26

[153] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 5026–5033. 26, 39, 79, 85

[154] J. Hwangbo, J. Lee, and M. Hutter, "Per-Contact Iteration Method for Solving Contact Dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, Apr. 2018. 26

[155] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable Simulation for Physical System Identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, Apr. 2021. 26

[156] P. C. Horak and J. C. Trinkle, "On the Similarities and Differences Among Contact Models in Robot Simulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 493–499, Apr. 2019. 26

[157] Q. L. Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, "Contact Models in Robotics: A Comparative Analysis," Apr. 2023. 26

[158] M. Otaduy and M. Lin, "A modular haptic rendering algorithm for stable and transparent 6-DOF manipulation," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 751–762, Aug. 2006. 26, 78, 79

[159] M. Kim, J. Yoon, D. Son, and D. Lee, "Data-Driven Contact Clustering for Robot Simulation," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8278–8284. 27

[160] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham, "Can robots assemble an IKEA chair?" *Science Robotics*, vol. 3, no. 17, p. eaat6385, Apr. 2018. 29, 51

[161] R. S. Sutton and A. G. Barto, *Reinforcement Learning, Second Edition: An Introduction*. MIT Press, Nov. 2018. 35, 56, 80

[162] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016. 36

[163] C. Yu, Z. Cai, H. Pham, and Q.-C. Pham, "Siamese Convolutional Neural Network for Sub-millimeter-accurate Camera Pose Estimation and Visual Servoing," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 935–941. 38, 52, 53, 54, 70

[164] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540 [cs]*, Jun. 2016. 39

[165] "Garage: A toolkit for reproducible reinforcement learning research," 2019. 39

[166] A. G. Barto and S. Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning," *Discrete Event Dynamic Systems*, vol. 13, no. 1, pp. 41–77, Jan. 2003. 41

[167] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, Jun. 2001. 46

[168] N. Vuong, H. Pham, and Q.-C. Pham, "Learning Sequences of Manipulation Primitives for Robotic Assembly," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 4086–4092. 51, 52, 54, 55, 61

[169] R. Haugaard, J. Langaa, C. Sloth, and A. Buch, "Fast robust peg-in-hole insertion with continuous visual servoing," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, Oct. 2021, pp. 1696–1705. 52

[170] J. C. Triyonoputro, W. Wan, and K. Harada, "Quickly Inserting Pegs into Uncertain Holes using Multi-view Images and Deep Network Trained on Synthetic Data," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 5792–5799. 53

[171] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, "Training Deep Neural Networks for Visual Servoing," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 3307–3314. 53

[172] K. Zhang, J. Lee, Z. Hou, C. W. de Silva, C. Fu, and N. Hogan, "How does the structure embedded in learning policy affect learning quadruped locomotion?" Aug. 2020. 64

[173] S. Chiaverini and L. Sciavicco, "The parallel approach to force/position control of robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 361–373, Aug. 1993. 66

[174] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, Oct. 2020. 77

[175] D. Stewart and J. C. Trinkle, "An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction," *International Journal of Numerical Methods in Engineering*, vol. 39, pp. 2673–2691, 1996. 79

[176] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," http://ilpubs.stanford.edu:8090/778/?ref=https://githubhelp.com, Jun. 2006. 80

[177] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, "Adaptation of Force Control Parameters in Robotic Assembly," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 561–566, Jan. 2012. 81

[178] D. Erickson, M. Weber, and I. Sharf, "Contact Stiffness and Damping Estimation for Robotic Systems," *The International Journal of Robotics Research*, vol. 22, no. 1, pp. 41–57, Jan. 2003. 81

[179] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning Variable Impedance Control for Contact Sensitive Tasks," *arXiv:1907.07500 [cs]*, Jul. 2020. 84

[180] E. Lengyel, "Volumetric Hierarchical Approximate Convex Decomposition," in *Volumetric Hierarchical Approximate Convex Decomposition*. A K Peters/CRC Press, 2016. 86

[181] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023. [Online]. Available: https://zenodo.org/record/8127025 96

[182] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, P. Trochim, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess, "Dm_control: Software and Tasks for Continuous Control," *Software Impacts*, vol. 6, p. 100022, Nov. 2020. 96

[183] M. M. Contributors, "MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo," 2022. [Online]. Available: http://github.com/deepmind/mujoco_menagerie 96

[184] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL$ˆ2$: Fast Reinforcement Learning via Slow Reinforcement Learning," *arXiv:1611.02779 [cs, stat]*, Nov. 2016. 96

[185] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, May 2019, pp. 5331–5340. 96

[186] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," *arXiv:1806.10293 [cs, stat]*, Nov. 2018. 96

[187] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale," *arXiv:2104.08212 [cs]*, Apr. 2021. 96