Reinforcement Learning for Contact-Rich Assembly Tasks

CRISTIAN CAMILO BELTRAN-HERNANDEZ

SEPTEMBER, 2022

Reinforcement Learning for Contact-Rich Assembly Tasks

A dissertation submitted to

THE GRADUATE SCHOOL OF ENGINEERING SCIENCE

OSAKA UNIVERSITY

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN ENGINEERING

BY

CRISTIAN CAMILO BELTRAN-HERNANDEZ

SEPTEMBER, 2022

Abstract

General-purpose industrial robot manipulators play a more significant role in modern manufacturing industries, particularly in high-mix low-volume (HMLV) production industries. Part mating and insertion, also called Peg-In-Hole (PIH) tasks, are very common in the manufacturing industry. Though PIH tasks have been extensively researched, safely solving high-precision assembly in an ever-changing environment remains an open problem. Traditional methods require the design of manual engineered controllers and expertise to finetune the controller's parameters for a particular task. In HMLV production, where the task specifications change frequently, traditional methods are unfeasible due to the high cost required to redesign and finetune controllers for each new task.

Reinforcement Learning (RL) is a promising solution to the automation problem. RL methods have been proven successful at solving manipulation tasks autonomously. However, RL is still not widely adopted on real robotic systems because working with real hardware entails additional challenges, especially when using rigid position-controlled manipulators. These challenges include needing a robust controller to avoid undesired behavior that risks damaging the robot and its environment and constant supervision from a human operator. The main contributions of this dissertation are; first, we proposed a learning-based force control framework combining RL methods with traditional force control to enable learning on industrial position-controlled robotic manipulators on real-world hardware. Second, a simulation-to-real (sim2real) method is proposed to reduce the burden of learning RL policies directly on real hardware. A physics simulator is used to emulate the robot dynamics and to provide the RL agent with a rich and diverse set of task conditions, after which the learned policies are transferred and finetune on the real robot. The proposed method is designed to enable the robotic agent to tackle assembly tasks even in the presence of uncertainty of the task's goal. Finally, we present a study for accelerating robot learning of contact-rich manipulation tasks based on Curriculum Learning (CL) combined with Domain Randomization (DR). The main idea is to guide the learning process by presenting the RL agent with tasks in increasing order of difficulty. The proposed methods have been empirically evaluated with various

challenging industrial assembly scenarios in simulation and a real-world experimental setup. The results of such evaluations show the effectiveness of our proposed methods even when tackling high-precision contact-rich insertion tasks.

Contents

List of Figures

List of Tables

Abbreviations

CL Curriculum Learning

CNN Convolutional Neural Network

DOF Degrees of freedom

DR Domain Randomization

DRL Deep Reinforcement Learning

F/T Force-Torque

ROS Robot Operating System

RL Reinforcement Learning

SAC Soft Actor Critic

TCN Time Convolutional Network

sim2real Simulation to Real World

PID Proportional Integral Derivative

PD Proportional Derivative

PI Proportional Integral

UR3e Universal Robots 3 e-series

USB Universal Serial Bus

LAN Local Area Network

WRS World Robot Summit

HMLV High-Mix Low-Volume

DDPG Deep Deterministic Policy Gradient

PIH Peg-In-Hole

MDP Markov Decision Process

IK Inverse Kinematics

CPU Central Processing Unit

GPU Graphics Processing Unit

ROS Robot Operating System

GDR Domain Randomization with Gaussian probability distribution

UDR Domain Randomization with Uniform probability distribution

List of Symbols

$\ddot{\mathbf{x}}$	Robot's end-effector acceleration
$\ddot{\mathbf{x}}$	Robot's end-effector velocity
$\dot{q}_{max}$	Joint velocity limit
η	Scalar part of a Quaternion
γ	Learning discount factor
$\mathbb{R}$	Real numbers
$\mathcal{A}$	Action space
$\mathcal{O}$	Observation space
$\mathcal{P}$	Transition probability distribution
$\mathcal{R}$	Reward function
ω	Natural frequency, admittance controller
ϕ	Cartesian orientation vector
π	Policy
Ψ	Randomization space
ψ_i	Randomization parameter i^{th}
ψ_i^{high}	Upper-bound defined for randomized value of $i^{th} parameter$

ψ_i^{low}	Lower-bound defined for randomized value of $i^{th} parameter$
$\mathbf{p}$	Cartesian position vector
θ	Policy parametrization weights
ε	Vector part of a Quaternion
ζ	Damping ratio, admittance controller
a	Action
a_p	Policy's action for force controller parameters
a_x	Policy's action in translation/rotation of robot's end-effector
a_{max}	Maximum value of an action
b_d	Desired damping, admittance controller
ep	Episode number
ep_{max}	Maximum number of episodes
F_g	Desired insertion force
F_{ext}	Wrench measured at tip of the robot's end-effector, from Force/Torque sensor
F_{max}	Contact force limit
k_d	Desired stiffness, admittance controller
K_d^x	Derivative gain of position PD
K_i^f	Integral gain of force PI
K_p^f	Proportional gain of force PI
K_p^x	Proportional gain of position PD
L	Curriculum's level difficulty
L_{ep}	Curriculum's level difficulty for current episode

L_m Linear mapping, reward function

L_{step} Step size for curriculum's level difficulty increment or decrement

L_{thld_down} Threshold to decrease curriculum's level difficulty

L_{thld_up} Threshold to increase curriculum's level difficulty

m_d Desired inertial, admittance controller

o Observation

p_0 Probability distribution over initial states

q_c Joint configuration

q_c Target joint configuration

r Reward

r^d Dynamic reward based on curriculum's level of difficulty

S Parallel position-force controller's Selection matrix

s State

T Time horizon

t Time step

w_i Reward weights

x Cartesian pose of the robot's end-effector

x'_g Goal Pose

x_0 Initial state

x_c Cartesian pose command

x_e Relative pose error with respect to goal pose

x_g Goal Pose

x_{max} Maximum distance from goal

S State space

Chapter 1

Introduction

1.1 Background and motivation

In recent decades, a trend of declining and aging population has countries around the world facing the problem of labor shortage [1, 2]. Many industries are affected, particularly labor-intensive industrial production processes. A promising approach to address this problem is through automation [3].

In the past, manufacturing industries were driven by mass production of products which motivated the development of assembly lines to tackle the production demands. While such assembly line required many workers, typically, each worker needed to focus on a single repetitive task, which would rarely change. Therefore, automation of such tasks could be achieved either through the development of specialized machinery [4] or through the use of general purpose robotic manipulators programmed to repeat a sequential series of instructions [5]. However, modern market conditions have pushed manufacturing industries towards the production of high-mix low-volume (HMLV) products [6]. HMLV production is difficult to solve with assembly lines, as the task specification changes frequently. In consequence, assembly line automation methods are ineffective. In contrast to assembly lines, assembly cells have been proposed to tackle HMLV production [7]. In an assembly cell a worker is in charge of the partial or full assembly of a

product, i.e., a worker needs to be able to solve many tasks. Thus, in order to automate assembly cells, a robotic system needs to be able to solve several open challenges such as, high precision and careful control of the interaction forces between the robot and the manipulated objects (to avoid damaging the product or the robot), efficient set up to new working environments and the ability to adapt to changes in the task specifications.

This dissertation is aimed towards the industrial automation through general-purpose robotic manipulators. As most industrial robots are only joint position controlled, this work is focused on these type of robots. In joint position-control robots, the lowest-level control available the user allows to define target angles and speed of each joint. These type of robots tend to exert high joint torques to achieve precise motion trajectories. Solving contact-rich manipulation tasks with position-control robots is particularly challenging as small errors in a motion trajectory, such as contacting a surface, can result in large contact forces (i. e., a collision). Therefore, a safe and precise control of the interaction forces between the robot is essential to solve assembly tasks with industrial robots.

Reinforcement learning (RL) methods are a promising approach to address the challenge of industrial automating, where an *agent* is encourage to achieve a desired behavior through rewards and/or punishments [8]. RL has been proven to be effective at learning complex behaviors to solve challenging tasks from playing Atari video games at super-human level [9] to solving dexterous manipulation tasks with a multi-fingered robotic hand [10]. However, RL methods typically require a large amount of data (interaction) to learn a given task. Furthermore, RL is still not widely adopted on real applications (real hardware) as it requires tackling additional challenges, such as the need of robust controllers, the risk of tear and wear of training directly on real hardware, and in many cases the need of constant human supervision during training.

The work presented in this dissertation address the following challenges; learning to solve high-precision contact-rich assembly tasks with industrial position-control robotic manipulators though RL. Improving learning efficiency by exploiting the benefits of physics simulators and domain transfer (sim2real) techniques such as Domain

Randomization (DR) and Curriculum Learning (CL).

1.2 Objectives

The general objectives of this dissertation are as follows;

1. The integration of RL with rigid position-control robots to solve assembly tasks. In particular, we focus on the tasks of part mating and part insertion, which are very common in the manufacturing industry. The goal is to develop a safe learning framework to train and/or deploy RL policies directly on real-world hardware.

2. The development of RL-based methods to learn force control policies that can deal with the dynamic interaction of the robot with its working environment when solving high-precision contact-rich tasks such that neither the robot nor the manipulated parts are damage during the execution of the tasks. Additionally, the aim is to enable the robotic agent to handle task uncertainty, and to generalization to similar novel tasks, not seen during the training phase.

1.3 Dissertation Outline

This dissertation is organized as follows.

In Chapter 3, towards enabling RL methods to be applied to industrial position-controlled robots, a RL framework is introduced. The design of the framework is discussed and evaluated with a study of the ideal Action Spaces of the robotic agent for contact-rich manipulation tasks. Finally, the implementation of the RL framework for real-world applications is discuss; a fail-safe mechanism is designed and developed to enable industrial robot manipulator to tackle contact-rich assembly tasks.

In Chapter 4, a simulation-to-real (sim2real) method is proposed to reduce the burden of training RL policies on real hardware. The proposed method is designed to enable the robotic agent to tackle assembly tasks even in the presence of uncertainty of

the task's goal. A more robust policy representation is also presented in this chapter. Finally, the robustness of the proposed method and its generalization capability are evaluated on challenging real-world tasks not presented during training.

Chapter 5 is focused on addressing the problem of sample efficiency of RL methods. To this end, a Curriculum Learning (CL) study is presented. The overall idea is to guide the training of the RL agent in order to reduce the time required to learn an optimal control policy. In this chapter, a CL-based method is proposed and evaluated. The proposed method, compared to the methods proposed in previous chapters, is shown to be more sample efficient and to generalized better to novel task, not seen during the training phase.

Finally, in Chapter 6, the achievements and limitations of the proposed methods presented in this dissertation are discussed, as well as, open challenges and ideas for future work.

Chapter 2

Literature Review

2.1 Force Control

For industrial assembly tasks, where the robot end effector has to manipulate an object and interact with a complementary object or surface, controlling the interaction forces between the robot manipulator and its environment is critical for success. In this dissertation, these interaction tasks with the environment are referred as *contact-rich manipulation* tasks.

General purpose industrial robot arms are typically position controlled. A position-controlled robot refers to a robot where the lowest-level controller available to the user allows only the control of the angle (position) of each joint. A contact-rich manipulation task could be successfully executed with position control alone if the task were accurately planned. This would require an accurate model of both the robot manipulator and the environment geometry and dynamics. The robot manipulator can be modeled with enough precision, but a detailed description of the environment is difficult to obtain.

To solve a manipulation task, for example, a part mating task, the motion trajectory plan should have an accuracy much greater than the tolerances of the task. For *high-precision* manipulation tasks, a position control approach is not feasible in practice. Errors in the modeling of the robot or its environment would lead to errors in the motion

trajectory planned. In turn, the motion plan errors may result in unexpected contact with the environment causing the end effector to deviate from the desired trajectory. Consequently, the position control system attempts to reduce the deviation, by applying higher torque to the robot joints. Ultimately, the contact forces increase until either the robot or the parts are damaged. This issue can be overcome if a compliant behavior is ensured during the interaction [11].

A compliant behavior can be achieved either in a passive way by placing a compliant mechanical device between the robot end effector and the environment, or in an active way by devising a suitable interaction control strategy. A typical passive compliance method consists of a mechanical device called remote center compliance (RCC) [12] which is placed between the robot's wrist and end effector. For a part mating task (peg insertion), the passive compliance provided by the RCC lets the end effector move perpendicularly to the peg's axis and rotate freely so as to reduce resistance. However, the passive method does not work well with high-precision assembly [13]. On the other hand, active compliant methods correct position errors by controlling the interaction forces between the robot end effector and the environment. The contact force is the quantity describing the state of the interaction. To actively control the contact force, it is necessary to measure it. A typical approach is to mount a Force-Torque (F/T) sensor on the robot manipulator, between the wrist and the end effector. In general, these methods use the measured external forces and moments, and design control strategies on the basis of dynamic models of the task to minimize contact force. These active compliance methods are referred as *force control*.

Force control methods address the problem of interaction between a robot manipulator and its environment, even in the presence of some uncertainty (geometric and dynamic constraints) on contact-rich tasks. In this dissertation, we consider two common force control methods, first, a parallel force-position controller [14] and an admittance controller [15]. These methods provide direct control of the interaction through contact force feedback and a set of parameters, which describe the dynamic interaction between the manipulator and the environment. Solutions for high-precision contact-rich manipulation tasks have been proposed based on these force control approaches; Force control

based approaches has been proposed to solve assembly task, mainly considering the Peg-In-Hole (PIH) task [16, 17, 18, 19, 20, 21, 22]. Nevertheless, most of these assembly methods are not practical to use in real applications. Model parameters need to be identified, and controller gains need to be tuned. In both cases, the process is manually engineered for specific tasks, which requires a lot of time, effort, and expertise. These approaches are also not robust to uncertainties and do not generalize well to variations in the task specifications. Other approaches attempt to address this problem by either scheduling variable gains [23, 24], using adaptive methods for setting the gains [25, 26], or learning the gains from human demonstrations [27, 28]. This dissertation address the challenge of solving high-precision contact-rich manipulation task by combining traditional force control methods with Reinforcement Learning (RL) methods. Through RL the motion trajectory and parameters of the force controller are learned. Details of such approach are presented in Chapter 3.

2.2 Reinforcement learning

Reinforcement learning (RL) is a machine learning with the primary goal of producing autonomous agents that learn optimal behaviors through interaction with their environment. The core concepts of the RL problem are the agent, the policy, the reward, and the environment. The agent observe its environment and takes actions according to some rules, the policy [8]. The actions taken by the agent change the state of the environment, and then, the agent is rewarded, or punished, with a numerical value. The goal in RL is for the agent to explore the space of possible sequence of actions and from it find a *good*, if possible optimal, policy. An optimal policy is one that maximizes the cumulative reward obtain by the agent during its life-time. In an episodic setting, where the task is repeated at the end of each episode, the optimal policy maximizes the total reward per episode.

In the past, although RL had some successes [29, 30], such approaches lacked scalability, i. e., they were limited to low-dimensional problems. The limitations were due to issues such as, memory complexity, computational complexity, and sample complexity

[31]. In the last decade, the rise of deep learning have enable us to tackle such complexity issues by relying on the powerful function approximation and representation learning properties of deep neural networks [32]. With these new tools, RL has proven to be effective a learning complex behaviors to tackle challenging tasks. Notable examples of this are, first, the development of an algorithm that could learn to play a range of Atari 2600 video games at a superhuman level, directly from image pixels [9, 33]. Convincingly demonstrating that RL agents could be train on high-dimensional observations, solely based on a reward signal. Second, AlphaGo [34], a hybrid Deep Reinforcement Learning (DRL) system used to master the game of Go, a task considerably difficult due to the high number of possible states (board configurations) and similarly high number of possible actions for each state. In this case too, the RL based system was proven to achieve superhuman level of expertise by defeating the human world champion. Third, RL was used to master the game of StarCraft II, a multi-agent video game with complex decision-making mechanics [35]. Nevertheless, the RL was able to achieve a *Grandmaster* level of expertise.

In robotics, RL enables robots to autonomously discover optimal behaviors through trial-and-error interactions with its environment. Traditional control methods required the detailed design of solutions to a task, carefully crafting the sequence of actions the robot need to take to achieve a certain goal. With RL, instead, the designer of a control task provides feedback to the robot through a numerical reward given for each action taken. The reward serves as a measure of the performance of robot towards a given task [36]. In the context of robotics, RL methods have also been proven successful at autonomously learning complex behaviors in a variety of tasks ranging from playing a game of table tennis [37, 38, 39, 40], controlling Unmanned Aerial Vehicles [41, 42], locomotion [43, 44] to solving robotic manipulation tasks such as grasping [45, 46, 47, 48, 49], pick-and-place (e. g., block staking) [50, 51, 52, 53], and assembly [54, 55, 56, 57, 58, 59]. In particular, the task of interest of this dissertation are the subset of assembly task comprising part coupling and part insertion also referred as Peg-In-Hole tasks. Plenty of methods have been proposed to accelerate automation of robotic assembly tasks, such as PIH tasks. From search strategies to align the peg with the hole

[60, 61, 62], to learning-based methods [63, 64, 65].

The following sections discuss in more detail the most related work to the works presented in this dissertation.

2.2.1 Reinforcement learning and force control

Previous research has studied the use of RL methods to learn force control parameters (gains). Buchli et al. [66] uses policy improvements with path integrals (PI2) [67] to refine initial motion trajectories and learn variable scheduling for the joint impedance parameters. Similarly, Bogdanovic et al. [68], proposed a variable impedance control in joint-space, where the gains are learned with Deep Deterministic Policy Gradient (DDPG) [69]. Likewise, Martín-Martín et al [70], proposed a variable impedance control in end-effector space (VICES). However, in all these cases, access to the robot manipulator's low-level control of joint torques is assumed, which is not available for most industrial manipulators. Instead, we focus on position-controlled robot manipulators and provide a method to learn manipulation tasks using force feedback control where the controller gains are learned through RL methods. Luo et al. [57] propose a method for achieving peg-in-hole tasks on a deformable surface using RL and validated their approach on a position-controlled robot. They propose learning the motion trajectory based on the contact force information. However, the tuning of the compliant controller's parameters is not taken into account. In Chapter 3, we are proposing a method for learning not only the motion trajectory based on force feedback but simultaneously fine-tuning the compliant controller's parameters.

The representation of the action space is fundamental for the RL problem. Both Bogdanovic [68] and Martín-Martín [70] study the importance of different action representation in RL for contact-rich robot manipulation tasks. In a similar way, in Chapter 3, an empirical study is presented, comparing different choices of action space based on force feedback control methods for rigid robots on contact-rich manipulation tasks.

2.2.2 Reinforcement Learning for high-precision assembly tasks

RL can be applied to robotic agents to learn high-precision assembly skills instead of only transferring human skills to the robot program [71]. Recent studies showed the importance of RL for robotic manipulation tasks [72, 73, 51], but none of these methods can be applied directly to high-precision industrial applications due to the lack of fine motion control.

In [74], an RL technique was used to learn a simple peg-in-hole insertion operation. Similarly, Inuo et al. [54] proposed a robot skill-acquisition approach by training a recurrent neural network to learn a peg-in-hole assembly policy. However, these approaches used a finite number of actions by discretizing the action space, which has many limitations in continuous-action control tasks [69], as is the case for robot control, which is continuous and high-dimensional.

Xu et al. [75] proposed learning dual peg insertion by using the deep deterministic policy gradient [76] (DDPG) algorithm with a fuzzy reward system. Similarly, Fan et al. [59] used DDPG combined with guided policy search (GPS) [77] to learn high-precision assembly tasks. Luo et al. [57] also used GPS to learn a peg-in-hole tasks on a deformable surface. Nevertheless, these methods learn policies that control the motion trajectory only while they require the manual tuning of force control gains; therefore, they do not scale well to variations of the environment.

Ren et al. [78] proposed the use of DDPG to simultaneously control position and force control gains, but they assumed the geometric knowledge of the insertion task, which made the learned policies inflexible to be applied to different insertion tasks. To solve high-precision assembly tasks, our approach focused on learning policies that simultaneously control the robot's motion trajectory and actively tune a compliant controller to unknown geometric constraints.

Buchli et al. [66] accomplished variable stiffness skill learning on robot manipulators by using an RL algorithm call-policy improvement with path integrals (PI2). However, the method was formulated for torque-control robots. Another similar approach was to

use a flexible robot so as to focus only on the motion trajectory, as in [79]; however, rigid position-controlled robots are still more widely used. Therefore, we focus on industrial robot manipulators, which are mainly position-based-controlled.

Abu- Dakka et al. [80] proposed a learning method based on iterative learning control (ILC). Their method is focus on transferring manipulation skills from demonstrations that provide a reference trajectory and force profile. In this work, we present a method that can learn manipulation skills without prior knowledge of a reference trajectory or force profile. However, our method supports the use of such prior knowledge to speed up the learning phase.

In Chapter 4, a robust RL-based framework is proposed to solve high-precision assembly tasks with position-controlled robots even in the presence of uncertainty of the goal pose.

2.2.3 Learning with real-world robot manipulators

Some research projects have explored the capabilities of RL methods on real robots by testing them on a large scale. On the one hand, Levine et al. [46] and Pinto et al. [73], both in which a massive amount of data was collected for learning robotic grasping tasks. However, in both works, a high-level objective, the grasp posture, is learned from the experience obtained. In contrast, for contact-rich tasks it is require to learn direct low-level controller to, for example, reduce the contact force between the robot and the environment for safety reasons. On the other hand, Mahmood et al. [81] propose a benchmark for learning policies on real-world robots, so different RL algorithms can be evaluated on a variety of tasks. Nevertheless, the tasks available in [81] are either locomotion tasks with a mobile robot or contact-free tasks with a robot manipulator. In this work, we propose a framework for learning contact-rich manipulation tasks with real-world robot manipulators based on force control methods.

Alternative research approach have explicitly focused on the domain transfer of assembly tasks from simulation to real-world environments (sim2real). In [82], a meta-

RL technique is applied to transfer experiences and generalize better to the real world. In [83], system identification of the real robot (KUKA LBR iiwa) with its simulated counterpart is performed to improve sim2real transferability. While RL-based policies have been proposed and proven to have the potential to solve assembly tasks in the real world, there is still a lack of adoption of such methods in real industrial assembly tasks. One reason for this gap between research and industry is the sample efficiency of such learning methods; a large amount of interaction of the agent with its environment is still necessary to learn robust policies. In this dissertation, we aim to contribute to this area by proposing a more sample-efficient approach based on Curriculum Learning (CL) and Domain Randomization (DR), i.e., less time is required to train a successful policy without decreasing its transferability capabilities.

2.2.4 Domain Randomization

In the context of machine learning, DR has been proposed as a technique to improve domain transfer, such as going from one task to a harder one or moving from a simulated environment to a real-world environment, in particular for training vision-based models [84] or sim2real models [10]. In [85] an empirical study is presented to examine the effects of DR on agent generalization. Their results show that DR may lead to suboptimal, high-variance policies, which [85] attributes to the uniform sampling of environment parameters. Following those results, in Chapter 5, a method is proposed combining DR with CL and the improved performance of such approach is empirically studied.

2.2.5 Curriculum Learning

The concept of CL in the context of machine learning was first proposed by Bengio et al. [86]. CL can be understood as learning from easier to harder tasks, i.e., the order in which information is presented affects the policy's ability to learn. A comprehensive survey on Curriculum Learning applied to Reinforcement Learning has been presented in [87]. Most CL approaches have been validated mainly on simulated environments, such as toy examples (e.g., grid worlds, cart-pole, and low-dimensional environments),

video games, and simulated robotic environments. Few research works have focused on real-world robotic environments, such as in [88] where a robot is trained to shoot a ball into a goal, [89, 90] where reaching tasks with a robot arm are tackled, and [91] which focused on two tasks moving a cube to a target pose and cube stacking. Most recently, [92] presented a CL method focused on a specific automotive production task, trained on simulation, and transferred to its real-world equivalent. Similarly, [93] proposes a method to enable a robot to conduct anchor-bolt insertion, a peg-in-hole task for holes in concrete. On the other hand, the study presented in Chapter 5 is focuses on tackling various real-world complex industrial assembly insertion tasks, trained only on toy peg-in-hole simulated environments. Furthermore, we make an exhaustive study on the performance of several approaches to combine DR and CL. As a result, we propose a method to accelerate learning and domain transfer to real-world environments by an adaptive curriculum that affects how DR and the reward signal are considered during training.

Chapter 3

Learning force control for contact-rich manipulation tasks with rigid position-controlled robots

This thesis chapter originally appeared in the literature as

Beltran-Hernandez, C. C., Petit, D., Ramirez-Alpizar, I. G., Nishi, T., Kikuchi, S., Matsubara, T., & Harada, K. (2020). Learning force control for contact-rich manipulation tasks with rigid position-controlled robots. *IEEE Robotics and Automation Letters*, 5(4), 5709-5716.[1]

3.1 Introduction

In the age of the 4th industrial revolution, there is much interest in applying artificial intelligence to automate industrial manufacturing processes. Robotics, in particular, holds the promise of helping to automate processes by performing complex manipula-

[1] © 2020 IEEE Reprinted, with permission, from all authors.

tion tasks. Nevertheless, safely solving complex manipulation tasks in an unstructured environment using robots is still an open problem[94].

Reinforcement learning (RL) methods have been proven successful in solving manipulation tasks by learning complex behaviors autonomously in a variety of tasks such as grasping [45, 46], pick-and-place [51], and assembly [55]. While there are some instances of RL research validated on real robotic systems, most works are still confined to simulated environments due to the additional challenges presented by working on real hardware, especially when using rigid position-controlled robots. These challenges include the need for a robust controller to avoid undesired behavior that risk collision with the environment, and constant supervision from a human operator.

So far, when using real robotic systems with RL, there are two common approaches. The first approach consists of learning high-level control policies of the manipulator. Said approach assumes the existence of a low-level controller that can solve the RL agent's high-level commands. Some examples include agents that learn to grasp [45, 46] or to throw objects [95]. In said cases, the agent learns high-level policies, e.g., learns the position of the target object and the grasping pose, while a low-level controller, such as a motion planner, directly controls the manipulator's joints or end-effector position. Nevertheless, the low-level controller is not always available or easy to manually engineer for each task, especially for achieving contact-rich manipulation tasks with a position-controlled robot. The second approach is to learn low-level control policies using soft robots [96, 97, 98], manipulators with joint torque control or flexible joints, which are considerably safer to work with due to their compliant nature, particularly in the case of allowing an RL agent to explore its surroundings where collisions with the environment may be unavoidable. Our main concern with this approach is that most industrial robot manipulators are, by contrast, rigid robots (position-controlled manipulators). Rigid robots usually run on position control, which works well for contact-free tasks, such as robotic welding, or spray-painting [99]. However, they are inherently unsuitable for contact-rich manipulation tasks since any contact with the environment would be considered as a disturbance by the controller, which would generate a collision with a large contact force. Force control methods [11] can be used to enable the rigid manipulator to

perform tasks that require contact with the environment, though the controller's parameters need to be properly tuned, which is still a challenging task. Therefore, we propose a method to safely learn low-level force control policies with RL on a position-controlled robot manipulator.

This chapter presents three main contributions. First, a control framework for learning low-level force control policies combining RL techniques with traditional force control. Within said control scheme, we implemented two different conventional force control approaches with position-controlled robots; one is a modified parallel position/-force control, and the other is an admittance control. Secondly, we empirically study both control schemes when used as the action space of the RL agent. Thirdly, we developed a fail-safe mechanism for safely training an RL agent on manipulation tasks using a real rigid robot manipulator. The proposed methods are validated on simulation and real hardware using a UR3 e-series robotic arm.

This chapter is structured as follows. Section 3.2 presents in detail each component of the proposed method. The experimental setup and the evaluation of the proposed method is described in Section 3.3. Finally, in Section 3.4 a discussion of the work presented in this chapter is presented.

3.2 Methodology

The study presented in this chapter deals with high precision assembly tasks with a position-controlled industrial robot. Due to the difficulty of obtaining a precise model of the physical interaction between the robot and its environment, RL is used to learn both the motion trajectory and the optimal parameters of a compliant controller. The RL problem is described in Section 3.2.1. The architecture of the system and the interaction control methods considered are explained in Section 3.2.2, Section 3.2.2, and Section 3.2.3. Finally, our safety mechanism that allows the robot to learn unsupervised is described in Section 3.2.4.

3.2.1 Reinforcement Learning

Reinforcement Learning is an area of machine learning concerning sequential decision making to maximize a numerical reward signal. The main idea is that an agent can learn from experience, from interacting with its environment. As described in [8], the learner (agent) is not told how to behave, i. e., which actions to take, but instead must discover which actions yield the highest reward by trying them. We consider the problem of RL modeled as a Markov Decision Process (MDP) defined by $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, p_0, T\}$. $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition probability distribution defining $p(\mathbf{s}(t+1) \,|\, \mathbf{s}(t), \mathbf{a}(t))$, $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ the reward function, p_0 the probability distribution over initial states, $\gamma \in [0, 1]$ a discount factor, and T the time horizon (maximum number of time steps per episode). In other words, the environment is described by a state $\mathbf{s} \in \mathcal{S}$. The agent can perform actions $\mathbf{a} \in A$, and perceives the environment through observations $\mathbf{o} \in O$, which may or not be equal to $\mathbf{s}$. We consider an episodic interaction of finite time steps with a limit of T time steps per episode, after which the environment is reset to a previous initial state or some variation of it described by p_0.

During an episode, at each time step t, the agent find itself in a state $s(t)$, observes the environment $o(t)$, then takes actions $a(t)$ according to some rules (a policy) $\pi_\theta(\mathbf{a}(t) \,|\, \mathbf{o}(t))$, which is parameterized by weights θ. In turn, the agent receives a numerical reward $r(t)$ for taking such action and transitioning to a new state $s(t+1)$. This process repeats until a termination criteria is met, e. g., the fixed time horizon T is reached. Given an stochastic dynamics $p(\mathbf{s}(t+1) \,|\, \mathbf{s}(t), \mathbf{a}(t))$ and a reward function $r(\mathbf{s}, \mathbf{a})$, the aim is to find an optimal policy π_{θ^*} that maximizes the expected sum of future rewards given by

$$\theta^* := \arg\min_\theta \mathbb{E}_{\text{traj}} \sum_{t=0}^{T} \gamma^t r(s_t, \pi_\theta(s_t)),$$

where the expectation is taken from the possible trajectories
$\text{traj} = (s_0, \pi_\theta(s_0), ...s_T, \pi_\theta(s_T))$ due to the random nature of MDPs [100].

In this dissertation, we consider the agent as a robotic manipulator. The actions available to this robotic agent are Cartesian position commands and Force control pa-

rameters, as described in Section 3.2.3. The main observations are the position of the robot's end-effector, its velocity and the wrench measured at the tip of the end-effector. Additional observations are also considered in Chapter 4 and Chapter 5.

Soft Actor Critic

In this dissertation, the state-of-the-art model-free RL method called Soft Actor Critic (SAC) [101] is used. SAC is an off-policy actor-critic DRL algorithm based on the maximum entropy reinforcement learning framework. SAC aims to maximize the expected reward while optimizing a maximum entropy. The SAC agent optimizes a maximum entropy objective, which encourages exploration according to a temperature parameter α. The core idea of this method is to succeed at the task while acting as randomly as possible. Since SAC is an off-policy algorithm, it can use a replay buffer to reuse information from recent rollouts for sample-efficient training. We use the SAC implementation from TF2RL[2].

3.2.2 System overview

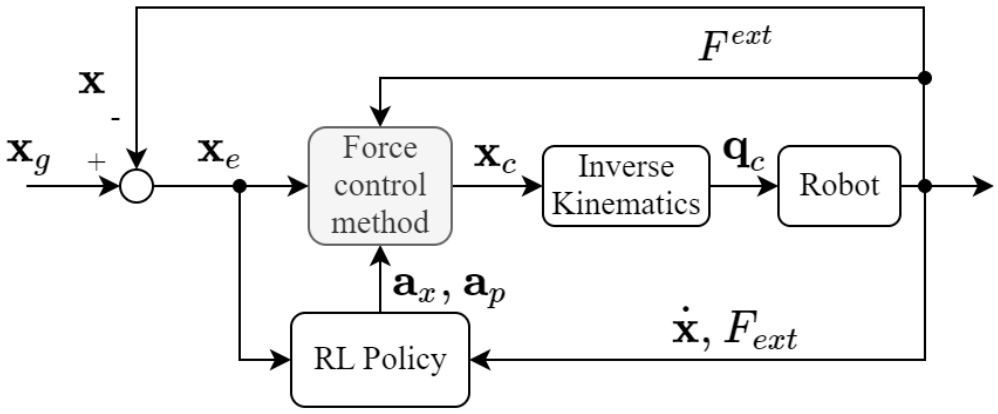


Figure 3.1: Proposed learning force control scheme. The input to the system is a goal end-effector pose, $\mathbf{x}_g$. The policy actions are trajectory commands, $\mathbf{a}_x$, and parameters, $\mathbf{a}_p$, of a force controller.

[2]TF2RL: RL library using TensorFlow 2.0. https://github.com/keiohta/tf2rl

Our proposed method aims to combine a force control with RL to learn contact-rich tasks when using position-controlled robots. Figure 3.1 describes the proposed control scheme combining an RL policy and a force control method. We assume knowledge of the goal pose of the robot's end-effector, $\mathbf{x}_g$. Both the policy and the force controller receive as feedback the pose error, $\mathbf{x}_e = \mathbf{x}_g - \mathbf{x}$, and the contact force F_{ext}. The velocity of the end-effector, $\dot{\mathbf{x}}$, is also included in the policy's observations. The F/T sensor signal is filtered using a simple low-pass filter.

The force control method has two internal controllers. First, a PD controller that generates part of the motion trajectory based on the pose error, $\mathbf{x}_e$. Second, a force feedback controller that alters the motion trajectory according to the perceived contact force, F_{ext}.

The RL policy has two objectives. First, to generate a motion trajectory, $\mathbf{a}_x$. Figure 3.2, shows how a simple P-controller (from the force control method) would not be enough to solve the task without producing a collision with the environment. For most cases, the P-controller trajectory would just attempt to penetrate the environment, since knowledge of the environment's geometry is not assumed. Nevertheless, the P-controller trajectory is good enough to speed up the agent's learning since it is already driven towards the goal pose. Therefore, to achieve the desired behavior, the nominal trajectory of the robot is the combination of the P-controller trajectory with the policy's trajectory. The second objective of the policy is to fine-tune the force control methods parameters, $\mathbf{a}_p$, to minimize the contact force when it occurs. We defined a collision as exceeding a maximum contact force in any direction. Therefore, contact with the environment is acceptable, but the policy's second goal is to avoid collisions. The policy also controls the P-controller's gains; thus, the policy decides how much to rely on the P-controller trajectory.

Pose Control Representation

The pose of the robot's end-effector is given by $\mathbf{x} = [\mathbf{p}, \phi]$, where $\mathbf{p} \in \mathbb{R}^3$ is the position vector and $\phi \in \mathbb{R}^4$ is the orientation vector. The orientation vector is described using

Euler parameters (unit quaternions) denoted as $\phi = \{\eta, \varepsilon\}$; where $\eta \in \mathbb{R}$ is the scalar part of the quaternion and $\varepsilon \in \mathbb{R}^3$ the vector part. Using unit quaternions allows the definition of a proper orientation error for control purposes with a fast computation compared to using rotation matrices [102].

The position command from the force controller is $\mathbf{x}_c = [p_t, \phi_t]$, where p_t is the commanded translation, and ϕ_t is the commanded orientation for the time step t. The desired joint configuration for the current time step, $\mathbf{q}_c$, is obtain from an Inverse Kinematics (IK) solver based on $\mathbf{x}_c$.

Learning force control

Two of the most common force control schemes are considered in these work, parallel position/force control [14] and admittance control [15]. The main drawback of said control schemes is the requirement to tune the parameters for each specific task properly. Changes in the environment (e.g., surface stiffness) may require a new set of parameters. Thus, we propose a self-tuning process using RL method.

The policy actions are $\mathbf{a} = [\mathbf{a}_x, \mathbf{a}_p]$, where $\mathbf{a}_x = [\mathbf{p}, \phi]$ are position/orientation commands, and $\mathbf{a}_p$ are controller's parameters. $\mathbf{a}_p$ is different and specific for each type of controller, see Section 3.2.3 and Section 3.2.3 for details. The policy has a control frequency of 20 Hz while the force controller has a control frequency of 500 Hz.

3.2.3 Force control implementation

PID parallel Position/Force Control

Based on [14], we implemented a PID parallel position/force control with the addition of a selection matrix to define the degree of control of position and force over each direction, as shown in Figure 3.3. The control law consists of a PD action on position,

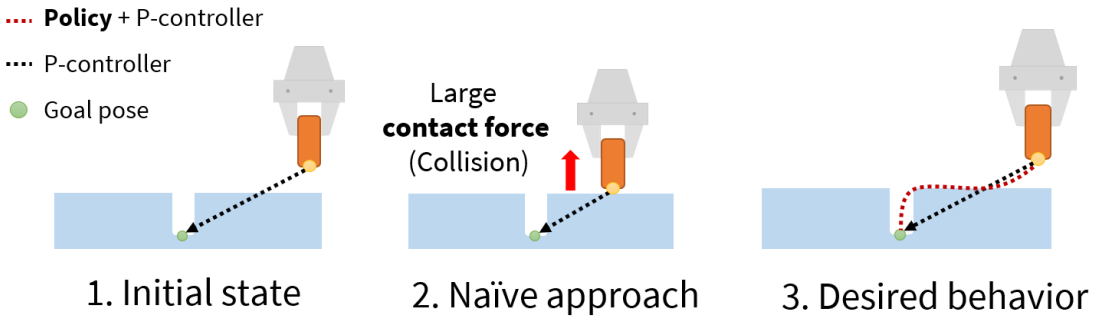


Figure 3.2: Proposed approach to solve contact-rich tasks. Assuming knowledge of the goal pose of the robot's end-effector, a simple P-controller can be designed. Our approach aims to combine this knowledge with the policy to generate the motion trajectory.

a PI action on force, a selection matrix and the policy position action, $\mathbf{a}_x$,

$$\begin{aligned} u \ = S(K_p^x \mathbf{x}_e + K_d^x \dot{\mathbf{x}}_e) + \mathbf{a}_x + \\ (I - S)(K_p^f F_{ext} + K_i^f \int F_{ext} dt) \end{aligned} \tag{3.1}$$

where u is the vector of driving generalized forces. The selection matrix is

$$S = diag(s_1, ..., s_6), \quad s_j \in [0, 1]$$

where the values correspond to the degree of control that each controller has over a given direction.

Our parallel control scheme has a total of 30 parameters, 12 from the position PD controller's gains, 12 from the force PI controller's (PI) gains, and 6 from the selection matrix S. We reduced the number of controllable parameters to prevent unstable behavior and to reduce the system's complexity. For the PD controller, only the proportional gain, K_p^x, is controllable while the derivative gain, K_d^x, is computed based on the K_p^x. K_d^x is set to have a critically damped relationship as

$$K_d^x = 2\sqrt{K_p^x}$$

Similarly, for the PI controller, only the proportional gain, K_p^f, is controllable, the integral gain K_i^f is computed with respect to K_p^f. In our experiments, K_i^f was set empirically to be 1% of K_p^f. In total, 18 parameters are controllable. In summary, the policy actions regarding the parallel controller's parameters are $\mathbf{a}_p = [K_p^x, K_p^f, S]$.

To narrow the agents choices for the force control parameters, we follow a similar strategy as in [68]. Assuming we have access to some baseline gain values, P_{base}. We then define a range of potential values for each parameter as $[P_{\text{base}} - P_{\text{range}}, P_{\text{base}} + P_{\text{range}}]$ with the constant P_{range} defining the size of the range. We map the agent's actions $\mathbf{a}_p$ from the range $[-1, 1]$ to each parameter's range. P_{base} and P_{range} are hyperparameters of both controllers.

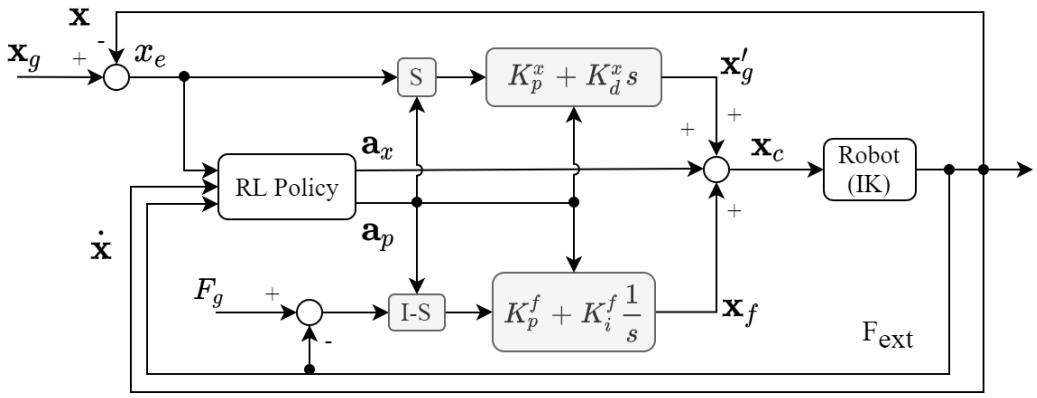


Figure 3.3: Proposed scheme for learning PID parallel position/force control. The RL agent controls the controller parameters PD gains, PI gains, and the selection matrix, S.

Admittance Control

is used to achieve a desired dynamic interaction between the manipulator and its environment. The admittance controller for position-controlled robots implemented is based on [103]. The admittance control is implemented on task-space instead of the robot joint-space. It follows the conventional control law

$$F_{ext} = m_d\ddot{x} + b_d\dot{x} + k_d x \tag{3.2}$$

where m_d, b_d, and k_d represent the desired inertia, damping, and stiffness matrices respectively. F_{ext} is the actual contact force vector. x, $\dot{x}$, $\ddot{x}$ are the displacement of the manipulator's end-effector, its velocity and acceleration respectively.

The admittance relationship can be expressed in Laplace-domain, adopting conventional expression of a second-order system as

$$\frac{X}{F}(s) = \frac{1/m_d}{s^2 + 2\zeta\omega_n s + \omega_n} \tag{3.3}$$

where ζ is the damping ratio and ω_n is the natural frequency, and they can be expressed by the admittance parameters as

$$\zeta = \frac{b_d}{2\sqrt{k_d\, m_d}} \qquad \omega_n = \sqrt{\frac{k_d}{m_d}} \tag{3.4}$$

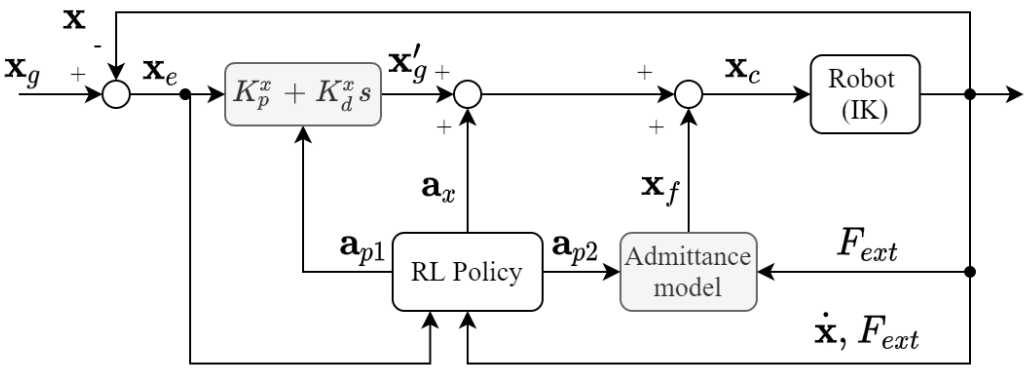


Figure 3.4: Proposed scheme for learning admittance control. A PD controller is included to regulate the input reference motion trajectory. The RL agent controls the PD gains, as well as, the admittance model parameters (inertia, damping and stiffness).

We are proposing a variable admittance controller, where the inertia, damping, and stiffness parameters are learned by the RL agent. Additionally, a PD controller is included in our admittance control. The PD controller with the policy action, $\mathbf{a}_x$, generates the nominal trajectory as explain in Section 3.2.2. The complete admittance control scheme is depicted in Figure 3.4. The PD gains are also controlled by the policy at each time step.

For the admittance control scheme, there are a total of 30 parameters; 12 from the position PD controller's gains and 18 from the inertia, damping, and stiffness parameters. Similarly, as mentioned in Section 3.2.3, we reduced the number of controllable parameters to prevent unstable behavior of the robot and reduce the system's complexity. Following the same strategy described in Section 3.2.3, of the PD controller, only

the proportional gain, K_p^x, is controllable. Additionally, we considered the inertia parameter for each direction as a constant, 0.1 kg·m^2 in all our experiments as a similar payload is used across tasks. Furthermore, we compute the damping with respect to the inertia parameter and the stiffness parameter by defining a constant damping ratio. From (3.4) we have that

$$b_d = 2\,\zeta\sqrt{k_d * m_d}$$

Therefore, only the stiffness parameters are controllable. In total, the controllable parameters of the admittance control are reduced to 12 parameters; 6 PD gains and 6 stiffness parameters. In summary, the policy actions regarding the admittance controller's parameters are $\mathbf{a}_p = [K_p^x, k_d]$

3.2.4 Fail-safe mechanism

Most modern robot manipulators already include a layer of safety in the form of an emergency stop. Nonetheless, the emergency stop exists at the extreme ends of the robot limits and completely interrupts the entire training session if triggered. To reactivate the robot, a human operator is required. To alleviate this inconvenience, we propose a mechanism that allows the robot to operate within less extreme limits. Thus, training of an RL agent can be done directly on the position-controlled manipulator with minimal human supervision.

Our system controls the robot as if teleoperating it by providing a real-time stream of task-space motion commands for the robot to follow. Therefore, we added our safety layer between the streamed motion command and the robot's actual actuation. The fail-safe mechanism validates that the intended action is within a defined set of safety constraints. As shown in Algorithm 1, for each action we check whether an IK solution exists for the desired position command, $\mathbf{x}_c$, if so, whether the joint velocity required to achieve the IK solution, $\mathbf{q}_c$, is within the speed limit.

If any of these validations are not satisfied, the intended action is not executed on

the robot, and the robot remains in its current state for the present time step. Finally, we check if the contact force at the robot's end-effector is within a defined range limit. If not, the episode ends immediately.

The first two validations are proactive and prevent unstable behaviors of the manipulator before they occur. In contrast, the third validation is reactive, i.e., only after a collision has occurred (the force limit has been violated), the robot is prevented from further actions.

Algorithm 1 Safe Manipulation Learning

Define joint velocity limit $\dot{\mathbf{q}}_{max}$
Define contact force limit F_{max}
Define initial state $\mathbf{x}_0$
Define goal state $\mathbf{x}_g$
for $n = 0, \cdots, N-1$ episodes **do**
    for $t = 0, \cdots, T-1$ steps **do**
        Get current contact force: F_{ext}
        $\mathbf{x}_e = \mathbf{x}_g - \mathbf{x}$
        Get Observation: $\mathbf{o} = [\mathbf{x}_e, \dot{\mathbf{x}}, F_{ext}]$
        Compute policy actions: $\pi_\theta(\mathbf{a}_x, \mathbf{a}_p | \mathbf{o})$
        $\mathbf{x}_c = control\_method(\mathbf{x}_e, \mathbf{a}_x, \mathbf{a}_p, F_{ext})$
        $\mathbf{q}_c = IK\_solver(\mathbf{x}_c)$
        if $\mathbf{q}_c$ not exists **then continue**
        if $|(\mathbf{q}_t - \mathbf{q}_c)/dt| > \dot{\mathbf{q}}_{max}$ **then continue**
        if $F_{ext} > F_{max}$ **then break**
        Actuate $\mathbf{q}_c$ on robot
    Reset to $\mathbf{x}_0$

3.2.5 Task's reward function

For all the manipulation tasks considered, the same reward function was used:

$$
\begin{aligned}
r(\mathbf{s}, \mathbf{a}) =& w_1 L_m(\|\mathbf{x}_e / \mathbf{x}_{max}\|_{1,2}) + w_2 L_m(\|\mathbf{a} / \mathbf{a}_{max}\|_2) + \\
& w_3 L_m(\|F_{ext} / F_{max}\|_2) + w_4 \rho + w_5 \kappa
\end{aligned}
\tag{3.5}
$$

where $\mathbf{x}_{max}$, $\mathbf{a}_{max}$, and F_{max} are defined maximum values. $L_m(y) = y \mapsto x, x \in [1, 0]$ is a linear mapping to the range 1 to 0, thus, the closer to the goal and the lower the contact force, the higher the reward obtained. $||\cdot||_{1,2}$ is L1,2 norm based on [96]. The $\mathbf{x}_e$ is the distance between the manipulator's end-effector and the target goal at time step t. $\mathbf{a}$ is the action taken by the agent. F_{ext} is the contact force. ρ is a penalty given at each time step to encourage a fast completion of the task. κ is a reward defined as follows

$$
\kappa = \begin{cases} 200, & \text{Task completed} \\ -10, & \text{Safety violation} \\ 0, & \text{Otherwise} \end{cases}
\tag{3.6}
$$

Finally, each component is weighted via w, all w's are hyperparameters.

3.3 Experiments

We propose a framework for safely learning manipulation tasks with position-controlled manipulators using RL. Two control schemes were implemented. With the following experiments, we seek to answer the following questions: Can a high-dimensional force controller be learned by the agent? Which action space, based on the number of adjustable controller's parameters provides the best learning performance?

A description of the materials used for the experiments is given in Section 3.3.1. An insertion task was used for evaluating the learning performance of the RL agents with the proposed method on a simulated environment, described in Section 3.3.2. Finally, the proposed method is validated on a real robot manipulator with high-precision assembly tasks.

3.3.1 Technical details

Experimental validation was performed both in a simulated environment using the Gazebo simulator [104] version 9 and on real hardware using the Universal Robot 3 e-series, with a control frequency of up to 500 Hz. The robotic arm has a Force/-Torque sensor mounted at its end-effector and a Robotiq Hand-e gripper. Training was performed on a computer with CPU Intel i9-9900k, GPU Nvidia RTX-2800 Super.

3.3.2 Action spaces for learning force control

Each control scheme proposed in Section 3.2 has a number of controllable parameters. The curse of dimensionality is a well known problem in RL [8]. Controlling few dimensions, number of parameters, makes the task easier to learn at the cost of losing dexterity.

In the following experiment, several policy models were evaluated. Each model has a different action space, i.e., a different number of controllable parameters. We evaluate the learning performance of the models described in Table 3.1, four models per control scheme. Each policy model has the same six parameters to control the position and orientation of the manipulator, $\mathbf{a}_x$, but a different number of parameters to tune the controller's gains, $\mathbf{a}_p$. From now on, we refer to each model by the name given in Table 3.1.

For a fair comparison, the action spaces were evaluated on a simulated peg-insertion environment so that we could guarantee the exact same initial conditions for each training session. The task is to insert a cube-shaped peg into a task board, where the hole has a clearance of 1 mm.

Each policy model was trained for 50.000 (50k) steps with a maximum of 150 steps per episode. The complete training session was repeated three times per model. Since the policy control frequency was set at 20 Hz, each episode lasts a maximum of 7.5 seconds. The episode ends if 1) the maximum number of time steps is reached, 2) a minimum distance error from the target pose is achieved, 3) or if a collision occurs. In

Table 3.1: Policy models with different action spaces.

Control Scheme	**Name**	**Pose**	**Gains**		
			PD	**PI / Stiffness**	**Selection Matrix S**
		$\mathbf{a}_x$	$\mathbf{a}_p$		
Parallel	P-9	6	1	1	1
	P-14	6	1	1	6
	P-19	6	6	6	1
	P-24	6	6	6	6
Admittance	A-8	6	1	1	-
	A-13	6	1	6	-
	A-13pd	6	6	1	-
	A-18	6	6	6	-

general, a complete training session takes about 50 minutes, including reset times.

Results

The comparison of learning curves for each policy model evaluated is shown in Figure 3.5. In the figure, the average cumulative reward per episode across the training sessions (bold line) is displayed along with the standard deviation error (shaded colored area). The results have been smoothed out using the exponential moving averages, with a 0.6 weight, to show the tendency of the learning curves.

From Figure 3.5, the overall best performance is achieved with the policy models combined with the parallel control scheme. By the end of the training session, these families of policies can yield higher rewards than the policy models combined with the admittance control scheme.

For the parallel control scheme, the model with the worst performance is P-9; it can be seen that there is not enough control of the controller's parameters to learn a

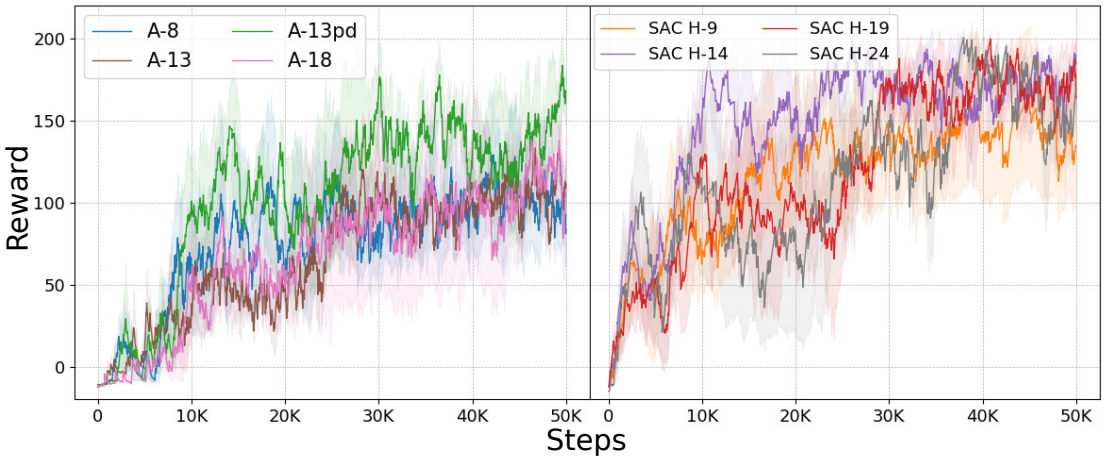


Figure 3.5: Learning curve of training session with active penalization of violation of the safety constraints. Peg-insertion scenario on simulation.

good policy consistently. On the other hand, the model P-24 has the slowest learning rate, but by the end of the training session, it can consistently learn a good policy. The policy model P-14 has the fastest learning rate and overall best performance.

For the admittance control scheme, the models A-13pd and A-18 have the best overall performance, with A-13pd yielding a cumulative reward as high as P-14 by the end of the training session. The model A-8, similar to P-9, has one of the worst performance; again, the lack of controllable parameters seems to have a big impact on learning a successful policy.

It is worth noting that for both control schemes, the models P-14 and A-13pd have the best overall performance. They provide the best trade-off between system complexity and learn-ability. On the other hand, the models with the largest number of parameters P-24 and A-18 can learn successful policies, but they require a longer training time to achieve it.

The parallel models' learning curve has larger standard deviation. One factor that contributes to these results is the selection matrix S, which highly affects the performance of the controller. Small changes of this parameter can make the behavior completely different. The agent's random exploration of this parameter can result in

very different results during the learning phase.

3.3.3 Safe learning

The developed fail-safe mechanism was not only evaluated as a mechanical safety that enables the real robot to explore random action without human supervision. We validate the usefulness of providing information to the robot about the safety constraints violations. Thus, we compare the proposed reward function Equation (3.5) with a variant that does not provide any punishment when a safety constraint is violated, i.e., κ gives a reward if the task is completed or zero otherwise, see Equation (3.6). We trained all policy models with this modified reward function.

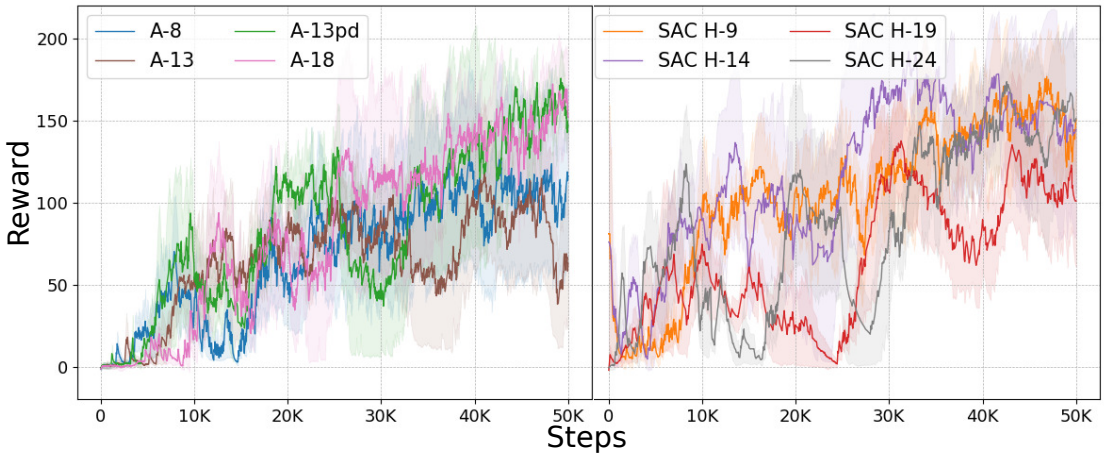


Figure 3.6: Learning curve of training without penalizing violation of safety constraints on the reward function. Peg-insertion scenario on simulation.

Results

Figure 3.6 shows the comparison of the learning curves of all models with a reward function that does not penalize violation of safety constraints. The results clearly show that the overall performance considerably decreases. The learning speed also decreases, as can be noted by comparing the performance of, for example, the model A-13pd. Learning with active penalization helps the agent learn policies that yield rewards of

Table 3.2: Collision detected during training session.

Model	**avg. # of collisions across training sessions**		
	Penalization	**No penalization**	**Difference**
A-8	326	455	-39%
A-13	350	408	-16%
A-13pd	**300**	**462**	**-54%**
A-18	451	457	-1%
P-9	187	369	-98%
P-14	**121**	**206**	**-70%**
P-19	183	392	-115%
P-24	219	337	-43%

+100 by 12,000 steps while it takes as much as 20,000 steps without penalization to achieve similar performance. Parallel control models show similar results. Moreover, the learning curves are nosier, meaning that the models can not reliably find a successful policy.

Additionally, we counted the average number of collisions detected during training sessions for each policy model. Table 3.2 shows the training session results using the proposed reward function with active penalization of the safety constraints and the reward function without penalization. In all cases, we see a high decrease in the number of collisions when actively penalizing collisions. In other words, the training session can be considered safer when the robot gets feedback on the undesired outcomes, i.e. when safety constraints are violated. Particularly, in the case of the parallel control scheme, the models have difficulty understanding that collisions are a poor behavior; thus, those models keep getting stuck on episodes that finish too soon due to collision. These results also highlight that the models A-13pd and P-14 do not only learn faster than other models but also produce the lowest number of collisions within their family of policies. On the other hand, the policy models with the highest number of parameters, A-18 and P-24, are able to learn successful policies at the cost of producing the highest number of collisions.

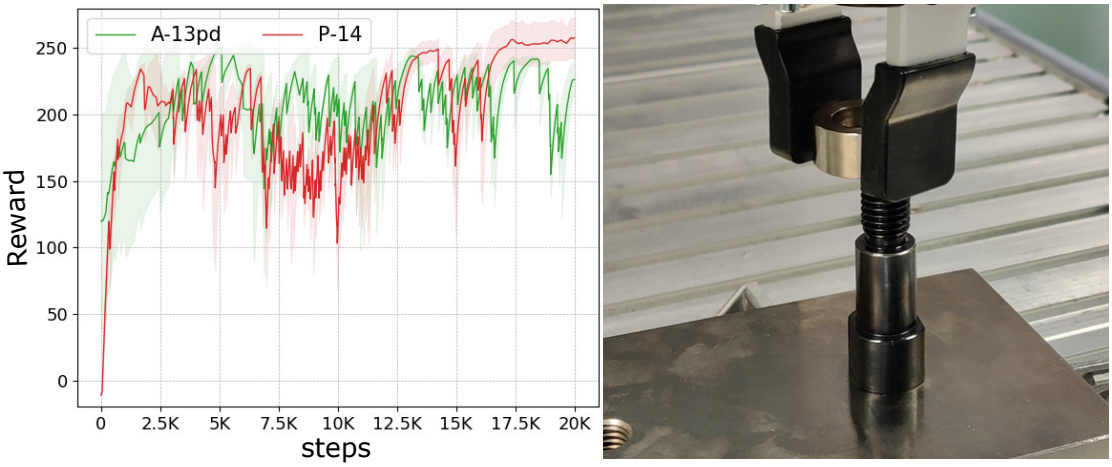


Figure 3.7: Ring-insertion task. Hole clearance of 0.2 mm. Cumulative reward per step of 20,000-steps training sessions of A-13pd and P-14 policy models.

3.3.4 Real robot experiments

Our proposed method was validated on real hardware using two high-precision assembly tasks. The first task involves an insertion task of a metallic ring into a bolt with a clearance of 0.2 mm, as shown in Figure 3.7. The second task is a more precise insertion task of the metallic peg into a pulley, with a clearance of 0.05 mm, as shown in Figure 3.8. Another robotic arm holds the pulley, and the center of the pulley is slightly flexible, which makes contact less stiff than the ring-insertion task. However, since the clearance is smaller, the peg is likely to get stuck if the peg is not adequately aligned, increasing the difficulty of solving the task. The best policy models from the previous experiment were used for training, P-14, and A-13pd. Both models were trained for 20,000 steps, twice. The episodes have a maximum length of 200 steps, about 10s.

Ring-insertion task results

From Figure 3.7, both models A-13pd and P-14 can quickly learn successful policies that solve the task. The high stiffness of the ring and bold makes the task more likely to result in a collision. The model P-14 produced an average of 45 collisions per training session, while A-13pd produced 34. Despite firmly grasping the ring with the robotic gripper,

the position/orientation of the ring can still slightly change. These slight changes can explain the drops in performance during the training session. However, the agents can adapt and learn to succeed in the task.

Peg-insertion task results

From Figure 3.8, we can see that it takes a lot more learning time to find a successful policy for both policy models compare to the ring-insertion task. While both policy models find a successful policy after about 13k steps, A-13pd achieved better consistent performance. As mentioned above, the physical interaction for this task is less stiff; thus, the average collisions per training session were fewer than in the ring-insertion task. For models A-13pd and P-14, the average number of collisions was 4 and 26, respectively.

The evolution of the policy model A-13pd, across a training session, is shown in Figure 3.9. The figure displays the observation per time step of only the insertion direction. The actions, $\mathbf{a}_x$ and $\mathbf{a}_p = [K_p^x, k_d]$ are also displayed. Observations and actions have been mapped to a range of [1, -1]. The peg-insertion task has three phases. A search phase before contact (Yellow). A search phase after initial contact (Red). An insertion phase (Green). On the left, the initial policy, we can clearly see that the insertion was not successful even after 200 steps, as well as a rather random selection of actions. On the contrary, on the right side, the task is being solved at around 130 steps. On top of that, the controller's parameters k_d and K_p^x have a clear response to the contact force perceived. After the first contact with the surface (Red), k_d and K_p^x are dramatically reduced, as a result, decreasing motion speed and reducing stiffness of the manipulator, which reduces the contact force. Then, when the peg is properly aligned (Green), k_d and K_p^x are increased to apply force to insert the peg -against the friction of the insertion- and to finish the task faster.

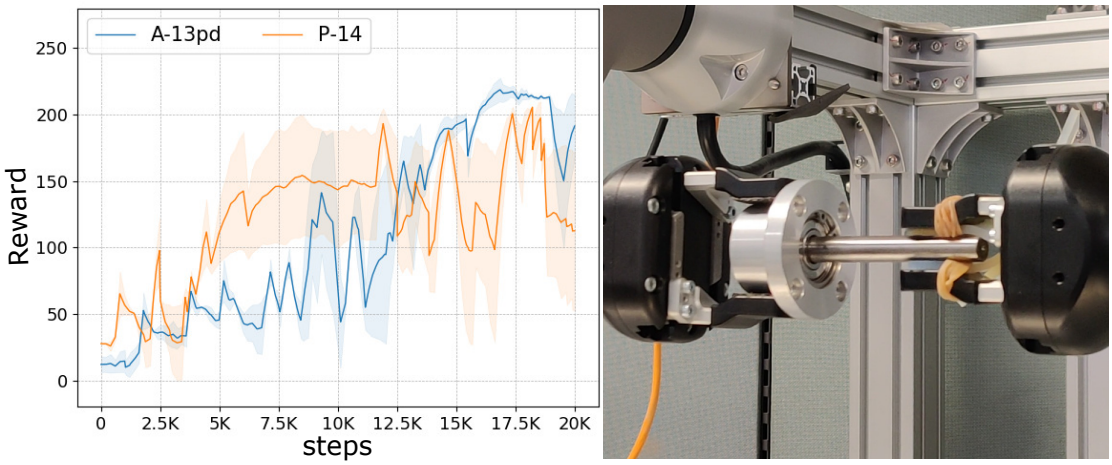


Figure 3.8: Peg-insertion task. Hole clearance of 0.05 mm. Cumulative reward per step of 20,000-steps training sessions of A-13pd and P-14 policy models.

3.4 Discussion

In this work, we have presented a framework for safely learning contact-rich manipulation tasks using reinforcement learning with a position-controlled robot manipulator. The agent learns a control policy that defines the motion trajectory, as well as fine-tuning the force control parameters of the manipulator's controller. We proposed two learning force control schemes based on two standard force control methods, parallel position/force control, and admittance control. To validate the effectiveness of our framework, we performed experiments in simulation and with a real robot.

First, we empirically study the trade-off between control complexity and learning performance by validating several policy models, each with a different action space, represented by a different number of adjustable force control parameters. Results show that the agent can learn optimal policies with all policy models considered, but the best results are achieved with the models A-13pd and P-14. These models yield the highest reward during training, proving to be the best trade-off between system complexity and learn-ability.

Second, results on a real robot showed the effectiveness of our method to safely learn high-precision assembly tasks on position-controlled robots. The first advantage is that

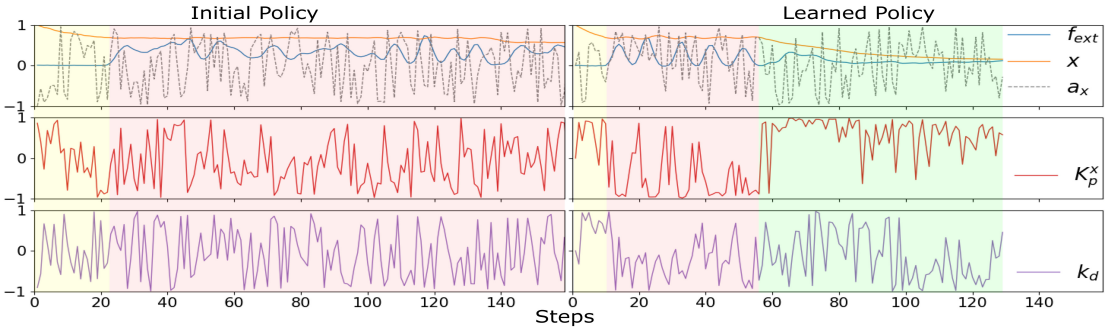


Figure 3.9: **A-13pd**: policy performance evolution on peg-insertion task. On the left, performance of the initial policy tried by agent. On the right, performance of the learned policy after training. All values correspond to the insertion direction only. Only 160 steps are displayed for space constraints. Insertion task divided into three phases: a search phase before contact (Yellow), a search phase after initial contact (Red) and an insertion phase (Green).

the fail-safe mechanism allows for training with minimal human supervision. The second advantage is that including information about the violation of safety constraints on the reward function helps speed up learning and reduce the overall number of collisions occurred during training.

Finally, in the usual peg insertion task, the motion trajectory is essential when the robot is in the air, while the force control parameters become essential when the peg is in contact with a surface or the hole. Results show that our framework can learn policies that behave accordingly on the different phases of the task. The learned policies can simultaneously define the motion trajectory and fine-tune the compliant controller to succeed in high-precision insertion tasks.

One of the limitations of our proposed method is that the performance is highly dependent on the choice of the controller's hyperparameters, more specifically, the base and range values of the controller's gains. In our experiments, we empirically defined said hyperparameters. However, to address said limitation, an interesting avenue for future research is to obtain these hyperparameters from human demonstrations, and then refine the force control parameters using RL. Additionally, for simplicity, we assume knowledge

of the goal pose of the end-effector for each task. However, vision could be used to get a rough estimation of the target pose to perform an end-to-end learning, from vision to low-level control, as proven in previous work [96].

Chapter 4

Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach

This thesis chapter originally appeared in the literature as

Beltran-Hernandez, C. C., Petit, D., Ramirez-Alpizar, I. G., & Harada, K. (2020). Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach. *Applied Sciences*, 10(19), 6923. Special Issue "Machine-Learning Techniques for Robotics".

4.1 Introduction

Autonomous robotic assembly is an essential component of industrial applications. Industrial robot manipulators are playing a more significant role in modern manufacturing industries with the goal of improving production efficiency and reducing costs. Though peg-in-hole assembly is a common industrial task that has been extensively researched,

safely solving complex high-precision assembly in an unstructured environment remains an open problem [94].

To reduce human involvement and increase robustness to uncertainties, the most recent research has been focused on learning assembly skills either from human demonstrations [105] or directly from interactions with the environment [8]. The present research focuses on the latter.

The main contribution of the work presented in this chapter is a robust learning-based framework for robotic peg-in-hole assembly given an uncertain goal position. Our method enables a position-controlled industrial robot manipulator to safely learn contact-rich manipulation tasks by controlling the nominal trajectory and, at the same time, learning variable force control gains for each phase of the task. The basis of this work is built upon the work described in Chapter 3. More specifically, the contributions of this chapter are:

- A robust policy representation based on time convolutional neural networks (TCN).
- Faster learning of control policies via domain transfer-learning techniques (sim2real) to greatly improve the training efficiency on real robots.
- Improved generalization capabilities of the learned control policies via domain randomization during the training phase on simulation. Although the effects of domain randomization have been researched [106, 10], to the best of our knowledge, we are the first to study the effects of sim2real with domain randomization on contact-rich real-robot applications with position-controlled robots.

The effectiveness of the proposed method is shown through extensive evaluation with a real robotic system on a variety of contact-rich peg-in-hole insertion tasks.

4.1.1 Problem Statement

Similar to Chapter 3, we considered a peg-in-hole assembly task that required the mating of two components. One of the components was grasped and manipulated by the robot

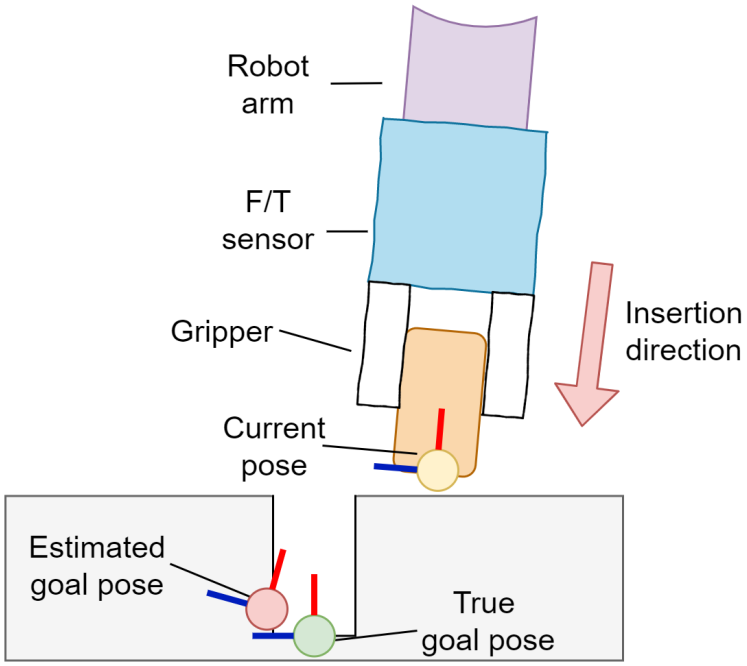


Figure 4.1: Insertion task with uncertain goal position.

manipulator, while the second component had a fixed position either via fixtures to an environment surface or by being held by a second robot manipulator. Figure 4.1 provides a 2D representation of the considered insertion tasks and the components assumed to be available to solve the task. The proposed method was designed for a position-controlled robot manipulator with a force/torque sensor at its wrist. Typically, these insertion tasks can be broadly divided into two main phases [62], search and insertion. During the search phase, the robot aligns the peg within the clearance region of the hole. In the beginning, the peg is located at a distance from the center of the hole in a random direction. The distance from the hole is assumed to be the "positional error". During the insertion phase, the robot adjusts the orientation of the peg with respect to the hole orientation, and pushes the peg to the desired position. We focused on both phases of the assembly task with the following assumptions:

- The manipulated object was already firmly grasped. However, slight changes of object orientation within the gripper were possible during manipulation.
- There was access to imperfect prediction of the target end-effector pose (as shown

in Figure 4.1) or a reference trajectory and its degree of uncertainty.

- The manipulated object was inserted in a direction parallel to the gripper's orientation.

We considered the second assumption fair given the advances in vision-recognition techniques, where the 6D pose of objects could be estimated from single RGB images [107, 108] or RGB images with depth maps (RGB-D) [109, 110]. The high accuracy of the predictions are in many cases enough for robot manipulation. Moreover, this second assumption included the specific case of using an assembly planner [111, 112], where even if the initial position of the objects is known, the inevitable error throughout the manipulation (e.g. pick-and-place, grasping, and regrasping) that makes the position/orientation of the manipulated objects uncertain during the insertion phase. A reference trajectory could be similarly obtained from demonstrations [52, 113, 114] when a complex motion is required to achieve the insertion. The last assumption allowed for defining a desired insertion force that may vary for different insertion tasks without loss of generalization.

4.2 Methodology

4.2.1 System Overview

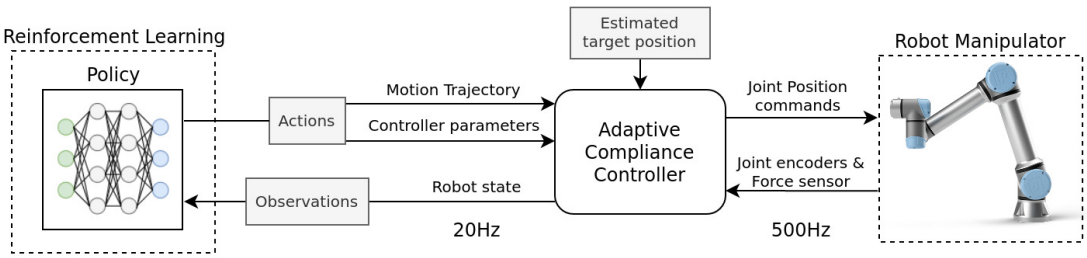


Figure 4.2: Our proposed framework. On the basis of estimated target position for an insertion task, our system learns a control policy that defines motion-trajectory and force-control parameters of an adaptive compliance controller to control an industrial robot manipulator.

Our proposed system aims to solve assembly tasks with an uncertain goal pose. Figure 4.2 shows the overall system architecture. There were two control loops. The inner loop was an adaptive compliance controller; we chose to use a parallel position-force controller that was proven to work well for this kind of contact-rich manipulation tasks [115]. The inner loop ran at a control frequency of 500 Hz, which is the maximum available in Universal Robots e-series robotic arms[1]. Details of the parallel controller are provided in Section 4.2.2. The outer loop was an RL control policy running at 20 Hz that provided subgoal positions and the parameters of the compliance controller. The outer loop's slower control frequency allowed for the policy to process the robot state and compute the next action to be taken by the manipulator, while the inner loop's precise high-frequency control would seek to achieve and maintain the subgoal provided by the policy. Details of the RL algorithm and the policy architecture are provided in Section 4.2.2. Lastly, the input to the system was estimated target position and orientation for the insertion task.

Motion commands $\mathbf{x}_c$ sent to the adaptive compliance controller corresponded to the pose of the robot's end effector. The pose was of the form $\mathbf{x} = [\mathbf{p}, \phi]$, where $\mathbf{p} \in \mathbb{R}^3$ is the position vector, and $\phi \in \mathbb{R}^4$ is the orientation vector. The orientation vector was described using Euler parameters (unit quaternions), denoted as $\phi = \{\eta, \varepsilon\}$, where $\eta \in \mathbb{R}$ is the scalar part of the quaternion and $\varepsilon \in \mathbb{R}^3$ the vector part.

4.2.2 Learning Adaptive-Compliance Control

The Reinforcement Learning (RL) method used in this chapter is the same as described in the previous chapter, Section 3.2.1. In this section, a novel policy representation is presented. Additionally, the reward function has been updated and the details described below.

[1]Robot details at https://www.universal-robots.com/e-series/

Multimodal Policy Architecture

The control policy was represented using neural networks, as shown in Figure 4.3. The policy input was the robot state. The robot state included the proprioception information of the manipulator and haptic information. Proprioception included the pose error between the current robot's end-effector position and predicted target pose $\mathbf{x}_e$, end-effector velocity $\dot{\mathbf{x}}$, desired insertion force F_g, and actions taken in the previous time step $\mathbf{a}_{t-1}$. Proprioception feedback was encoded with a neural network with 2 fully connected layers with activation function RELU to produce a 32-dimensional feature vector. For force-torque feedback, we considered the last 12 readings from the six-axis F/T sensor, filtered using a low-pass filter, as a 12 x 6 time series:

$$[F_{ext}^{0}, \ldots, F_{ext}^{12}], \quad \text{where} \quad F_{ext}^{i} = [F_x, F_y, F_z, M_x, M_y, M_z] \tag{4.1}$$

The F/T time series was fed to a temporal convolutional network (TCN) [116] to produce another 32-dimensional feature vector. The feature vectors from proprioception and haptic information were concatenated to obtain a 64-dimensional feature vector, and then fed to two fully connected layers to predict the next action.

The policy outputs actions for a parallel position-force controller. The policy produces two type of actions, $\mathbf{a} \doteq [\mathbf{a}_x, \mathbf{a}_p]$, where $\mathbf{a}_x = [\mathbf{p}, \phi]$ are position/orientation subgoals, and $\mathbf{a}_p$ are parameters of the parallel controller. The specific parameters controlled by $\mathbf{a}_p$ are described in Section 4.2.2.

Compliance Control in Task Space

Our proposed method uses a common force-control scheme combined with a reinforcement-learning policy to learn contact-rich manipulations with a rigid position-controlled robot. For the family of contact-rich manipulation tasks that require some sort of insertion, the parallel position-force control [14] performs better and can be learned faster than using an admittance control scheme when combined with an RL policy, as shown in Chapter 3. The details of the compliance controller has also been presented in Section 3.2.3

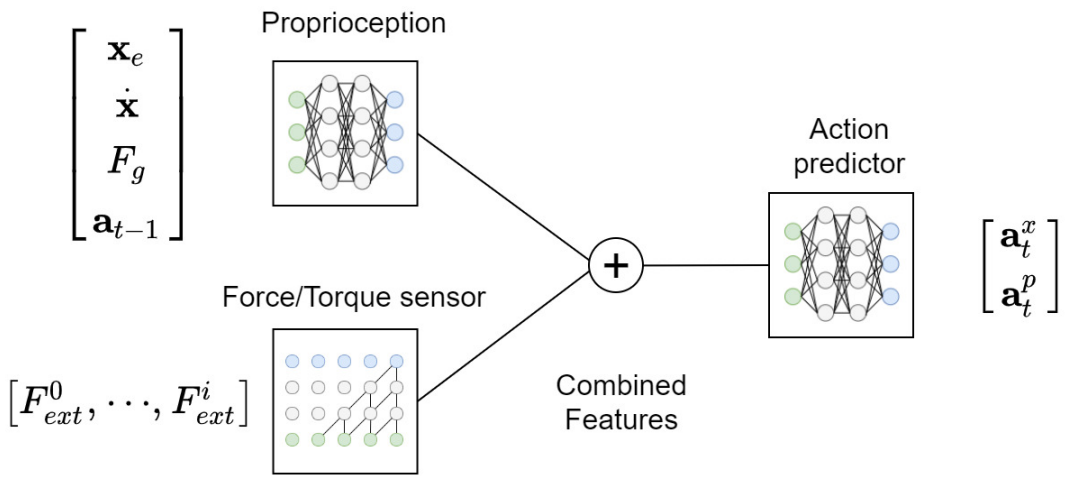


Figure 4.3: Control policy consisting of three networks. First, proprioception information is processed through a 2-layer neural network. Second, force/torque information is processed with a temporal convolutional network. Lastly, extracted features from first two networks are concatenated and processed on a 2-layer neural network to predict actions.

Task's reward function

For all considered insertion tasks, the same reward function was used:

$$r(\mathbf{s}, \mathbf{a}) = w_1 L_m(\|(F_{ext} - F_g)/F_{max}\|_2) + w_2 \kappa, \tag{4.2}$$

where F_g is the desired insertion force, F_{ext} is the contact force, and F_{max} is the defined allowed maximal contact force. $L_m(y) = y \mapsto x, x \in [1, 0]$ is a linear mapping in the range 1 to 0; thus, the closer to the goal and the lower the contact force, the higher the obtained reward. $||\cdot||_{1,2}$ is an L1,2 norm based on [72]. κ is a reward defined as follows:

$$\kappa = \begin{cases} 100 + ((1 - t/T) * 100), & \text{Task completed} \\ -50, & \text{Collision} \\ 0, & \text{Otherwise} \end{cases} \tag{4.3}$$

During training, the task was considered completed if the Euclidean distance between the robot's end-effector position and the true goal position was less than 1 mm. The agent was encouraged to complete the task as quickly as possible by providing an extra

reward for every unused time step with respect to the maximal number of time steps per episode T. Moreover, we imposed a collision constraint where the agent was penalized for colliding with the environment by giving it a negative reward and by finishing the episode early. This collision constraint encourages safer exploration, as shown in our previous work [115]. We defined a collision as exceeding force limit F_{max}. Therefore, a collision detector and geometric knowledge of the environment were not necessary. Lastly, each component was weighted via w; all ws were hyperparameters.

4.2.3 Speeding Up Learning

Two strategies were adopted to speed up the learning process. First, the exploitation of prior knowledge using the idea of residual reinforcement learning. Second, we used a physics simulator to train the robot on a peg-insertion task and transfer the learned policy directly to the real robot (sim2real).

Residual Reinforcement Learning

To speed up the learning of the control policy for insertion tasks that require complex manipulation, we used residual reinforcement learning [117, 118]. The goal is to leverage the training process by exploiting prior knowledge. With the assumption of an estimated target position or a reference trajectory, we could manually define a controller $\mathbf{x}_g$. Then, said controller's signal would be combined with policy action $\mathbf{a}_x$. The objective was to avoid training the policy from scratch, and avoid the exploration of the entire parameter space. The position command sent to the robot was

$$\mathbf{x}_c = (\mathbf{x}'_g + \mathbf{x}_f) + \mathbf{a}_x, \tag{4.4}$$

where $\mathbf{x}'_g$ is the reference trajectory process through a PD controller, $\mathbf{a}_x$ is the policy signal on the position, and $\mathbf{x}_f$ is the response to the contact force, as shown in Figure 3.3. The first two terms came from the parallel controller. Therefore, the policy would just need to learn to adjust the reference trajectory to achieve the task.

From Simulation to Real World

The proposed method works on the robot's end-effector Cartesian task-space, which makes it easier to transfer learning from simulation to the real robot or even between robots [119]. For most insertion tasks, a simple peg-insertion task was used for training on a physics simulator. We used simulator Gazebo 9 [104]. To close the reality gap between the physics simulator and real-world dynamics, we used domain randomization [84]. During training on the simulator, the following aspects were randomized:

- Initial/goal end-effector position: having random initial/goal positions helps the RL algorithm to find policies that generalize to a wide range of initial-position conditions.

- Object-surface stiffness: The RL agent also needs to learn to fine-tune the force-controller parameters to obtain a proper response to the contact force. Therefore, randomizing the stiffness of the manipulated objects helps it find policies that adapt to different dynamic conditions.

- Uncertainty error of goal pose prediction: On a real robot, the prediction of the target pose comes from noisy sensory information, either from a vision-detection system or from known prior manipulations (grasp and regrasp). Thus, during training on the simulation, we emulated this error by using normal Gaussian distribution with mean zero and standard deviation of a maximal distance error (for position and orientation).

- Desired insertion force: For different insertion tasks, a specific contact force is necessary for insertion to succeed. As we considered insertion force an input to the policy, during training, we randomized this value for each episode.

4.3 Experiments and results

4.3.1 Experiment Setup

Experimental validation was performed on a simulated environment using Gazebo simulator [104] version 9, and on real hardware using a Universal Robot 3 e-series with a control frequency of up to 500 Hz. The robotic arm had a force/torque sensor mounted at its end effector, and a Robotiq Hand-e parallel gripper. In both environments, training of the RL agent was performed on a computer with an Intel i9-9900k CPU and Nvidia RTX-2800 Super GPU. To control the robot agent, we used the Robot Operating System (ROS) [120] with the Universal Robot ROS Driver[2]. The experiment environment on the real robot is shown in Figure 4.4.

4.3.2 Training

During the training phase, the agent's task was to insert a cuboid peg into a task board on the simulated environment. The agent was trained for $500,000$ time steps, which, on average, takes about 5 hours to complete. During training, the environment was modified after each episode by randomizing one or several of the training conditions mentioned in Section 4.2.3. The range of values used for the randomization of the training conditions is shown in Table 4.1. The random goal position was selected from a defined set of possible insertion planes, as depicted in Figure 4.5.

After training on the simulation, the learned policy was refined by retraining on the real robot for 3% off the simulation time steps, which took about 20 minutes, to further account for the reality gap between simulated and real-world physics dynamics.

[2]ROS driver for Universal Robot robotic arms developed in collaboration between Universal Robots and the FZI Research Center for Information Technology https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

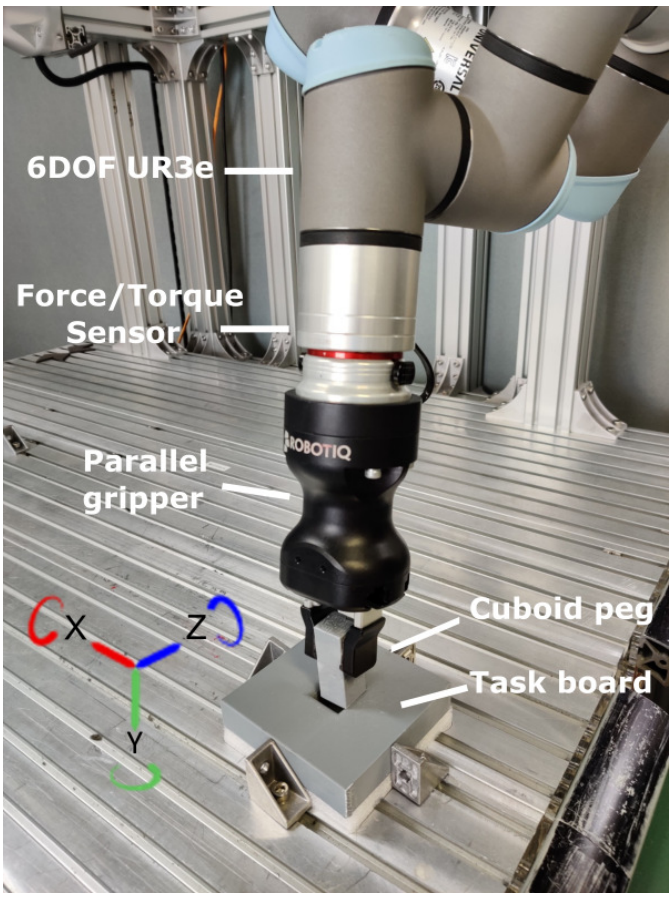


Figure 4.4: Real experiment environment with a 6-degree-of-freedom UR3e robotic arm. Cuboid peg and task board hole had a nonsmooth surface with 1.0 mm clearance.

4.3.3 Evaluation

The learned policy was initially evaluated on the real robot with a 3D-printed version of the cuboid peg in the hole-insertion task with the true goal pose. During evaluation, observations and actions were recorded. Figure 4.6 shows the performance of the learned policy (sim2real + retrain). The figure shows the relative position of the end effector with respect to the goal position, the contact force, and the actions taken by the policy for each Cartesian direction normalized to the range of [-1, 1], as described in Section 4.2.2. As shown in Fig. 4.1, the insertion direction was aligned with the y axis of the robot's coordinate system. In Figure 4.6, we highlighted three phases of the task. Blue corresponds to the search phase in free space before contact with the surface, yellow

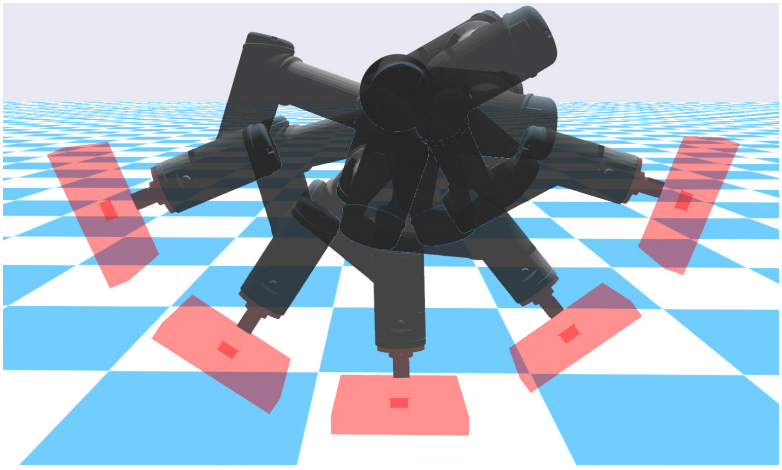


Figure 4.5: figure

Simulation environment. Overlay of randomizable goal positions.

Condition		**Value range**
Initial position (relative to goal)	Position (mm)	[-40, 40]
	Orientation (°)	[-10, 10]
Uncertainty error	Position (mm)	[-2, 2]
	Orientation (°)	[-5, 5]
Desire insertion force (N)		[0, 10]
Stiffness (in Gazebo: *surface/friction/ode/kp*)		$[7.0 \times 10^{-4}, 1.0 \times 10^{-5}]$

Table 4.1: Randomized training conditions.

is the search phase after initial contact with the environment, and green corresponds to the insertion phase. During the search phase, and particularly on the insertion direction (y axis), we could clearly observe that the learned policy properly reacted to contact with the environment by quickly adjusting the force control parameters. On top of that, during the insertion phase, the learned policy changed its strategy from just minimizing contact force to a mostly position-control strategy to complete insertion. This behavior is proper for this particular insertion task, as there is little resistance during the insertion phase, but it is not the desired behavior for other insertion tasks, as we discuss later in Section 4.3.4.

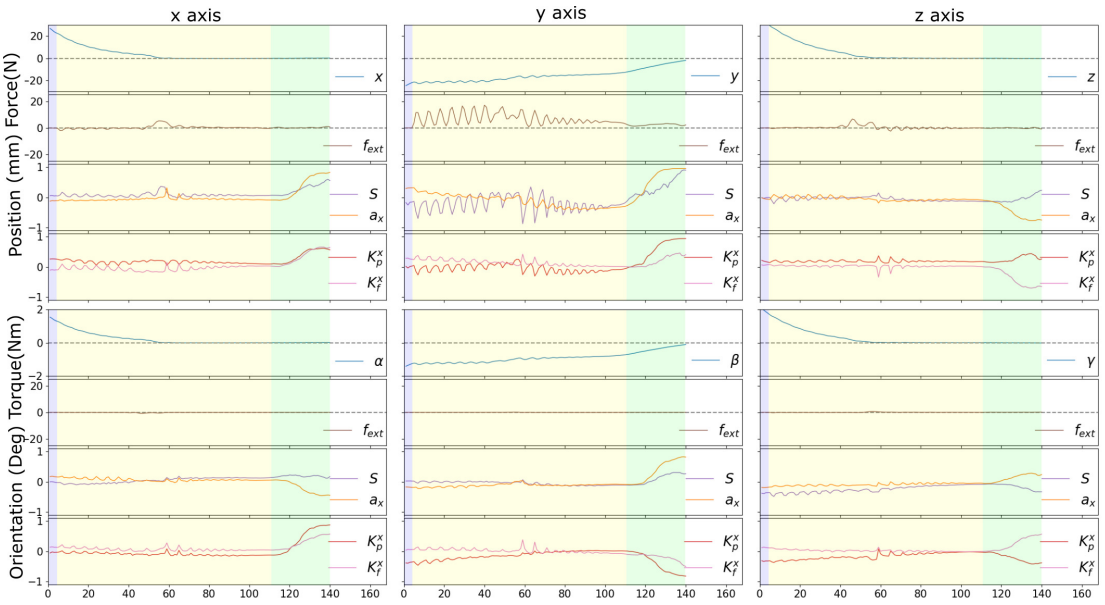


Figure 4.6: Performance of learned policy (sim2real + retrain) on 3D-printed cuboid-peg-insertion task. Insertion direction was aligned with y axis of robot coordinate system. Relative distance from robot's end effector to goal position and contact force is shown. The 24 policy actions besides the corresponding axis are also shown.

Additionally, we compared the performance of the learned policy as a combination of sim2real and refinement on the real robot versus just learning on the real robot or just directly transferring the learned policy from the simulation (sim2real) without further training. We evaluated these policies on a 3D-printed version of the cuboid-peg-insertion task. Policies were tested 20 times with a random initial position assuming a perfect estimation of the goal position (true goal). Table 4.2 shows the results of the evaluation. The three policies had a very high success rate, but the policy transfer from the simulation had difficulty with the real-world physics dynamics. As expected, the policy retrained from the simulation gave the best overall performance time.

4.3.4 Generalization

Now, to evaluate the generalization capabilities of our proposed learning framework, we use a series of environments with varying conditions.

Method	Success Rate	Avg. Time Steps	Avg. Time (sec)
Scratch	100%	109.6	5.48
Sim2real	95%	75.3	3.77
Ours	100%	65.6	3.28

Table 4.2: Comparison of learning from scratch, straightforward sim2real, and sim2real + retraining (Ours). Test performed on a 3D printed cuboid peg insertion task assuming knowledge of the true goal position.

Varying degrees of Uncertainty error

First, the learned policies are evaluated on the 3D printed cuboid peg insertion task where there is a degree of error on the estimation of the goal position. To clearly compare the performance of the different methods with different degrees of estimation error, we added and offset of position or orientation about the x-axis of the true goal pose. Nevertheless, for completeness we also evaluate the policies on goal poses with added random offset of translation, $[-1, 1]$ millimeters, and orientation, $[-5°, 5°]$, on all directions. On each case, the policies were tested 20 times from random initial positions. Results are shown in Table 4.3.

Estimation error / Success rate											
	Position					**Orientation**					
Method	**1mm**	**2mm**	**3mm**	**4mm**	**5mm**	**1°**	**2°**	**3°**	**4°**	**5°**	**Random**
Scratch	0.9	0.9	0.7	0.55	0.35	1.0	0.9	0.8	0.8	0.5	0.8
Sim2real	0.9	0.85	0.85	0.6	0.4	1.0	0.9	0.8	0.8	0.3	0.75
Ours	**1.0**	**1.0**	**0.95**	**0.65**	**0.6**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.9**

Table 4.3: Comparison of learning from scratch, straightforward sim2real and sim2real + retraining (Ours) with different degrees of goal-position uncertainty error. Test performed during 3D-printed cuboid-peg insertion task.

In all cases, the policy learned from the simulation with domain randomization and fine-tuned on the real robot gave the best results. If the difference between the physics dynamics on the simulation and the real world was too big, learning from scratch could yield better results than only transferring the policy from the simulation, as can be

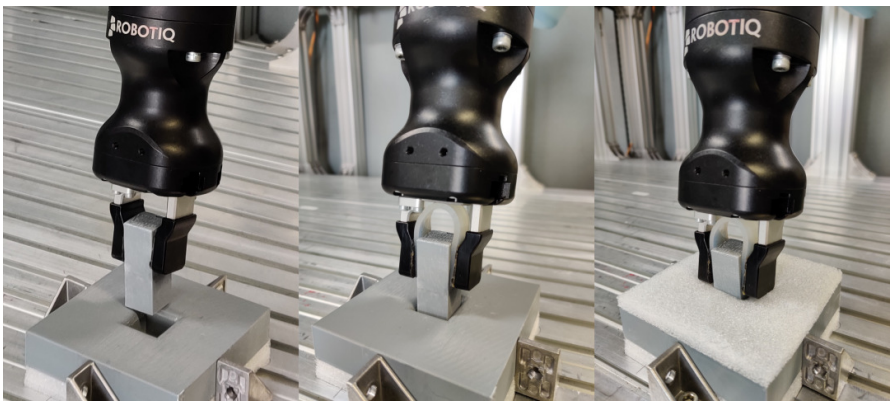


Figure 4.7: (left to right) High-, medium-, and low-stiffness environments.

Method/Stiffness	**High**	**Medium**	**Low**
Scratch	100%	70%	40%
Sim2real	95%	100%	100%
Ours	100%	100%	100%

Table 4.4: Success rate of 3D-printed-cuboid insertion task with different degrees of contact stiffness.

seen when the uncertainty error on orientation was too big (5°); where the friction with the environment makes the task much harder, such contact dynamics are difficult to simulate.

Varying Environment Stiffness

Second, the learned policy was also evaluated on different stiffness environments. Figure 4.7 shows the 3 environments considered for evaluation. High stiffness was the default environment. Medium stiffness was achieved by using a rubber band to hold the cuboid peg between the gripper fingers, adding a degree of static compliance. In addition to that, for the low-stiffness environment, a soft foam surface was added to further decrease stiffness. The policies were evaluated from 20 different initial positions, results are reported in Table 4.4.

Varying Insertion Tasks

Lastly, we evaluate the learned policy on a series of novel insertion tasks, none seen during training, to assess its generalization capabilities. These insertion tasks included challenges such as adapting to a very hard surface (high stiffness), requiring a minimal insertion force to perform the insertion, and a complex peg shape for mating the parts. The different insertion scenarios are depicted in Figure 4.8.

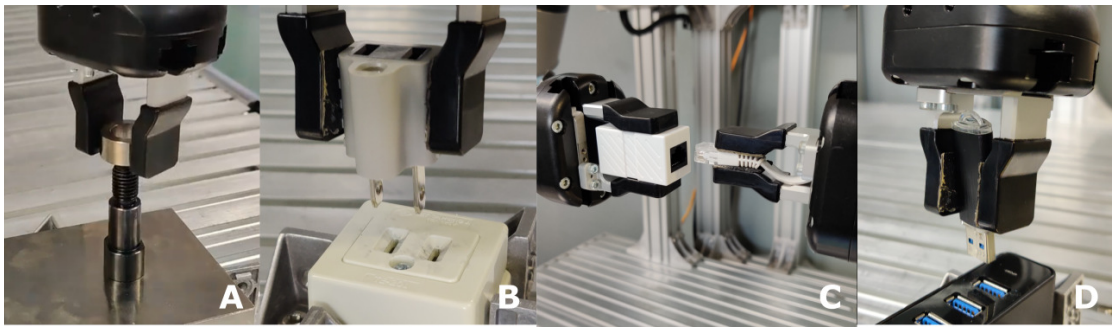


Figure 4.8: Several insertion tasks with different degrees of complexity. (A) Metal ring (high stiffness) with 0.2 mm of clearance. (B) Electric outlet requiring high insertion force. (C) Local-area-network (LAN) port, delicate with complex shape. (D) Universal serial bus (USB).

Task	Success rate	Insertion force
Ring	80%	5N
Electric Outlet (x)	75%	10N
Electric Outlet (y)	75%	10N
LAN port (x)	55%	5N
LAN port (y)	60%	5N
USB	80%	8N

Table 4.5: Success rate of learned policy on several insertion tasks.

For each task, the learned policy was executed 20 times from random initial positions and assuming perfect estimation of the goal position. Table 4.5 shows the success rate of the learned policy on these novel tasks, along with the desired insertion force

set for each task. As the insertion force was defined as a policy input, we could define specific desired insertion force for each task. Even though the policy was only trained by using the simpler cuboid-peg insertion task, mainly in simulation and shortly refined on a real robot with a 3D-printed version of the same task, the learned policy achieved a high success rate in novel and complex insertion tasks.

Compared to the cuboid-peg insertion task, on these novel insertion tasks, the peg was more likely to become stuck during the task's search phase, as the surrounding surface near the hole was not smooth and may have had crevices. The extra challenges were not present during the training phase, which reduced the capability of the learned policy to react in an appropriate way. The insertion task of the LAN port was the most challenging for the policy due to the complex shape of the LAN cable endpoint. If just one corner of the LAN adapter was stuck, the insertion could not be completed even if large force was applied.

Additionally, we tested the policy on different insertion planes for the electric outlet and the LAN port tasks. In both cases, success rate was similar due to training with the randomized insertion planes. However, the policy was slightly better with insertions on the y-axis plane due to retraining (on the real robot) only being done on this axis.

4.3.5 Ablation Studies

In this section, we evaluate the individual contribution of some components added to the proposed learning framework.

Learning from Scratch vs Sim2real

The inclusion of transfer learning from the simulation to the real robot for the proposed learning framework was evaluated. We compared the learning performance of training the agent on the real robot from scratch versus learning starting from a policy learned on simulation. Training from scratch was performed for 50,000 steps, while retraining from the simulation lasted 15,000 steps. Figure 4.9 shows the learning curve for both

training sessions. Learning from scratch required at least 50,000 steps to succeed at the tasks most of the time. In contrast, learning from the pretrained policy on the simulation achieved the same performance in under 5000 steps. The policy from the simulation still required some training to fine-tune the controller to real-world physics dynamics, which are difficult to simulate, as can be seen from the slow start and the drops in cumulative reward.

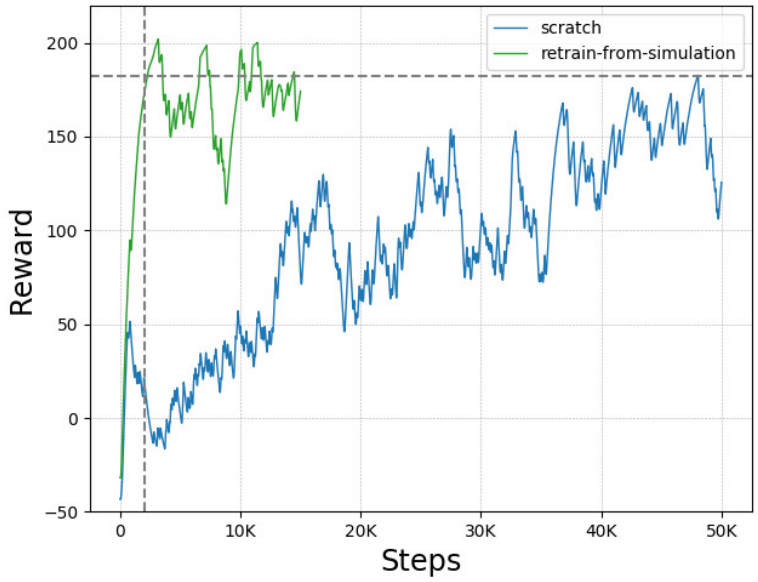


Figure 4.9: Comparison between learning from scratch and learning from a policy learned on simulation: learning curve for 3D-printed cuboid-peg insertion task on real robot with random initial positions.

Policy Architecture

We evaluated the contribution of the policy architecture introduced in our method (see Section 4.2.2) by comparing it to a policy with a simple neural network (NN) with two fully connected layers as used in previous work [115]. We trained both policies on the cuboid-peg insertion task on the simulation and compared their learning performance. Figure 4.10 shows the learning curve of both policy architectures for a training session of 70,000 time steps. From the figure, is clear that, with our newly proposed TCN-based policy, the agent was able to learn faster and exploit better rewards. The TCN-based

policy learned a successful policy (25,000) about 15,000 steps faster than the simple neural-network (NN)-based policy did (40,000). Additionally, the TCN-based policy converged to a higher cumulative reward than that of the simple NN-based policy.

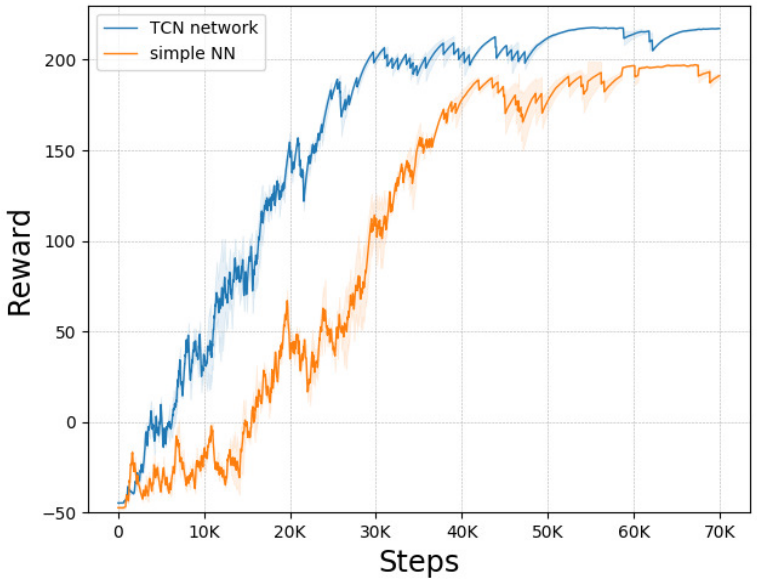


Figure 4.10: Comparison between policy architectures: learning curve for cuboid-peg insertion task with random initial positions.

Policy Inputs

Lastly, we evaluated the choice of inputs for the policy. We compared our proposed policy architecture with all inputs, as defined in Section 4.2.2, with two variants. First, we considered the policy without the inclusion of prior action $\mathbf{a}_{t-1}$. Second, we considered the policy without knowledge of desired insertion force F_g. The training environment was the cuboid-peg insertion task on the simulation with a random initial position and random desired insertion force. In the case of the policy that did not have F_g as input, the cost function still accounted for the desired insertion force.

Figure 4.11 shows the comparison of the learning curves. Most notable is the poor performance of the policy that lacked the knowledge of prior action $\mathbf{a}_{t-1}$. Prior-action information is critical for the agent to more quickly converge to an optimal policy. Addi-

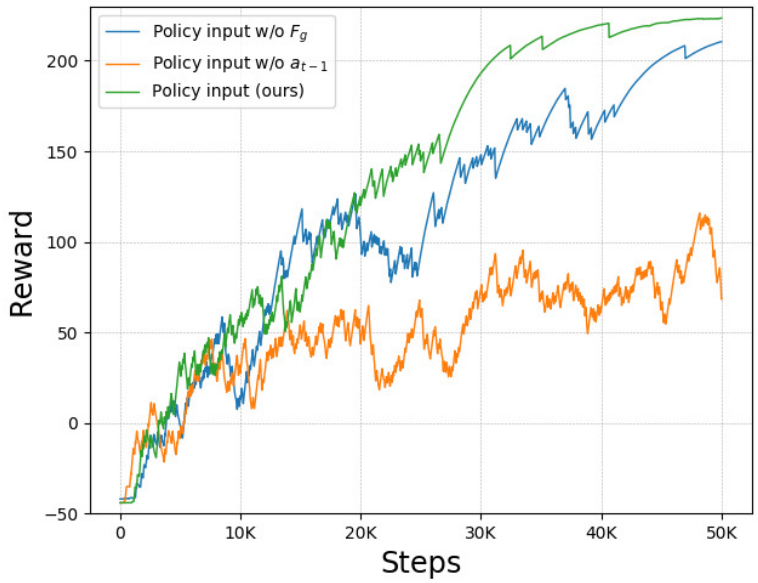


Figure 4.11: Comparison of policies with different inputs. Learning curve for cuboid-peg insertion task with random initial positions and random desired insertion force.

tionally, knowledge of F_g enables the agent to find policies that yield higher cumulative rewards, and to learn faster.

4.4 Discussion

We proposed a learning framework for position-controlled robot manipulators to solve contact-rich manipulation tasks. The proposed method allows for learning low-level high-dimensional control policies on real robotic systems. The effectiveness of the learned policies was shown through an extensive experiment study. We showed that the learned policies had a high success rate at performing the insertion task under the assumption of a perfect estimation of the goal position. The policy correctly learned the nominal trajectory and the appropriate force-control parameters to succeed at the task. The policy also achieved a high success rate under varying environmental conditions in terms of uncertainty of goal position, environmental stiffness, and novel insertion tasks.

While model free reinforcement-learning algorithm SAC was used in this work, the proposed framework can easily be adapted to other RL algorithms. The choice of SAC

was due to its sample efficiency as an off-policy algorithm. The pros and cons of using other learning algorithms would be interesting future work.

One limitation of our learning framework is the selection of the force-control parameter range (see Section 4.2.2). The choice of a wide range of values may allow for the policy to adapt to very different environments, but it also increases the difficulty of learning a task, as small variations in the action may cause undesired behaviors, as was the case during the first 20,000 to 30,000 steps of training (see Figure 4.10). On the other hand, a narrow range would make it easier and faster to learn a task, but it may not generalize well to different environments. Defining a range is much easier than manually finding the optimal parameters for each task, but it is still a manual process. Therefore, another interesting future study would be to use demonstrations to learn a rough estimation of the optimal force parameters to further reduce training times.

Chapter 5

Accelerating Robot Learning of Contact-Rich Manipulations: A Curriculum Learning Study

5.1 Introduction

Reinforcement Learning (RL) has been proven to be successful at learning complex behaviors to solve a variety of robotic contact-rich tasks [8, 36, 121]. However, RL solutions are still not widely adopted in real-world industrial tasks. One reason is that RL still requires an expensive and large amount of robot interaction with its environment to learn a successful policy. The more complex the target task is, the more interaction (samples) is required.

To tackle this problem, domain transfer methods such as Domain Randomization (DR) and Curriculum Learning (CL) have been introduced. The concept of CL, where the learning process can be made more efficient by following a curriculum that defines an order in which tasks should be learned, has been introduced in previous works [86, 122]. Additionally, DR of visual and physical properties of a task has been shown to improve the performance of tasks in novel domains [84, 10]. However, most of these results have

been only validated in simulated environments, from video games to robotic toy tasks, or in real-world toy environments. In this work, we tackle the problem of improving sample efficiency and performance when learning real-world complex industrial assembly tasks with rigid position-controlled robots. To this end, we seek to answer the question: Does the order in which the different environments (or tasks) are presented to the agent (through DR) affect the training sample efficiency and performance of the learned policy?

We hypothesize that on top of DR, guiding a RL agent's training with a curriculum (presenting tasks in increasing order of difficulty) towards the desired behavior can increase sample efficiency. The reasoning is that the curriculum helps reduce the overall exploration needed to achieve the desired goal while DR enhances domain transferability.

This chapter presents a study of the combination of CL with DR. More specifically, we compare different curricula designs and different approaches at sampling values for DR. As a result, we propose a novel method that significantly outperforms our previous work [123]. In [123], only DR is used to improve sim2real transferability without CL. Experimental results in simulation and real-world environments show that our novel method can be trained with only a fifth of the training samples required by our previous method and still successfully learn to solve the target insertion tasks. Furthermore, the learned policies transferred to the real world achieved high success rates (up to 86%) on industrial level insertion tasks, with tolerances of ± 0.01 mm, not seen during the training.

This chapter's contributions are as follows:

- A study of the application of Curriculum Learning to a learning framework for rigid robots solving contact-rich manipulation tasks.
- A novel learning framework combining curriculum learning with domain randomization to accelerate learning and domain transfer.
- An improved reward function to guide the learning of force sensor-based contact-rich manipulation tasks. The reward perceived by the agent is dynamically discounted by the curriculum's level of difficulty. For our target domain, the idea

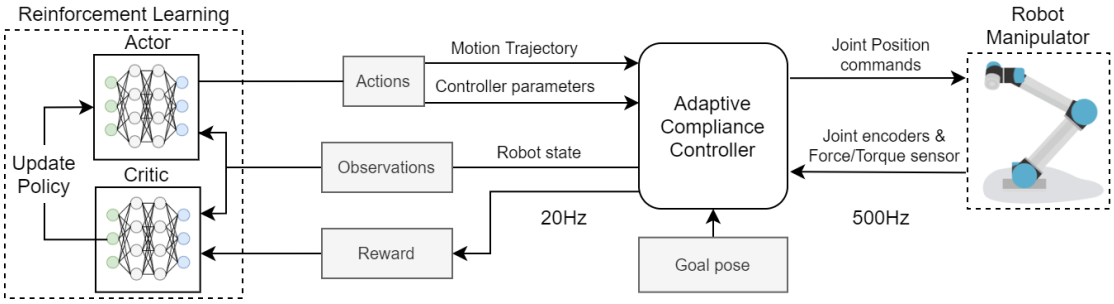


Figure 5.1: Overview of the system used for this study. The input is the goal pose, optionally the desired contact force can be defined, otherwise is considered as 0 N.

is to encourage the agent to learn to solve the hardest tasks as discussed in Section 5.2.5.

- An empirical study of the different methods considered in this chapter was conducted. Novel tasks not seen during training were used to validate the performance of each method, both in simulation and in the real world, including complex industrial insertion tasks. Additionally, we study the impact of different components of our proposed method in the Appendix.

Additionally, the system developed in this chapter has been open sourced [1] for benefit of the research community.

The rest of this chapter is organized as follows, related work is discussed in **??**. The case study for this work and the proposed method are explained in Section 5.2. Experimental results and comparisons with alternative methods and our previous work are shown in Section 5.3. Ablation studies are described in the Appendix.

[1] At https://github.com/cambel/robot-learning-cl-dr

5.2 Materials and Methods

5.2.1 Problem Statement

In the present study, we consider the peg-in-hole assembly task that requires the mating of two components. One of the components is grasped and manipulated by the robot manipulator, while the second component has a fixed position via fixtures to a support surface. The proposed method is designed for position-controlled robot manipulators with access to force/torque information at the robot's end-effector (e.g., F/T sensor at the robot wrist), especially those robots where low-level torque control is not available. Thus, sensor-based force control is necessary to realize contact-rich manipulation tasks for such a type of robot.

5.2.2 System Overview

Our propose method aims to improve the sample efficiency of the training phase. Figure 5.1 shows the overall system architecture which is based on the work presented in Chapter 4. There are two control loops. The inner loop has an adaptive compliance controller; we choose to use a parallel position-force controller that was proven to work well for this kind of contact-rich manipulation tasks, as shown in Chapter 3. The inner loop runs at a control frequency of 500 Hz, which is the maximum available in the Universal Robots e-series robotic arms[2]. The outer loop runs at a lower control frequency to account for the computation time required by the learning algorithm. The Reinforcement Learning (RL) method is the same as described previously in Section 3.2.1. Our system considers control of the 6 degrees of freedom of the Cartesian space at the robot's end-effector (position and orientation). To our previous learning control framework (see Section 4.2.2), we added the PID gains scheduling approach discussed in Section 5.2.3. Additionally, a new dense reward function is proposed and described in Section 5.2.2. Similarly, a DR method based on CL is implemented on top of this learning control framework, see Section 5.2.4 and Section 5.2.5.

[2]Robot details at https://www.universal-robots.com/e-series/

Reward function

On one hand, in our previous work [123], the reward function is defined only in terms of the contact force and distance between the current position of the robot's end-effector and the target position. The reason is to encourage the agent to get closer to the target position; the faster, the better, while discouraging any contact force. On the other hand, in this work, we propose the inclusion of the velocity of the robot's end-effector in the reward signal. While it is ideal that the agent achieves the task as fast as possible, high speeds are not desirable when the robot is close to the environment or in contact with it, as it can generate large contact forces. Thus, the proposed reward function has the following shape:

$$r(\mathbf{s}, \mathbf{a}) = w_1 r_{xv} + w_2 r_F + w_3 \rho \tag{5.1}$$

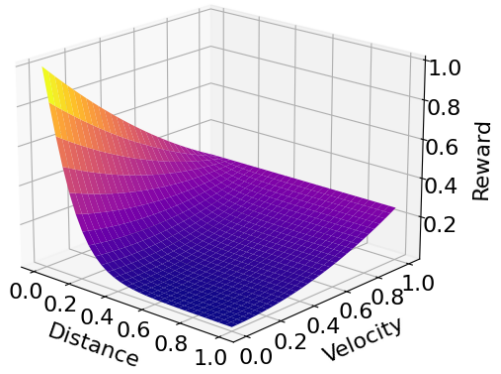
where r_{xv} is the component of the reward associated with the position and velocity of the robot's end-effector. r_{xv} aims to encourage the agent to get closer and keep closer to the target position. Besides, the agent is encourage to move faster, if it is far from the target pose, or slower when it is close to the target pose. r_{xv} is defined as:

$$r_{xv} = (1 - \tanh(5|x|)(1 - |\dot{x}|) + (|\dot{x}|/2)^2 \tag{5.2}$$

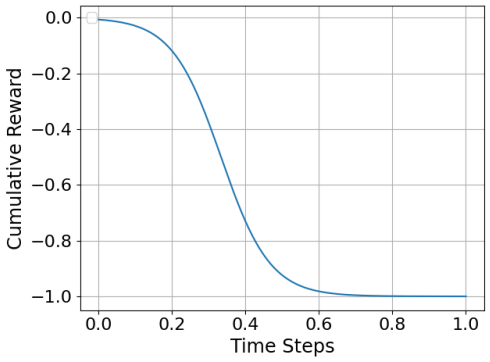
Where x is the distance between the robot's end-effector and the target position, and $\dot{x}$ is its velocity. A visualization of this reward component is shown in Figure 5.2a. The component of the reward Eq. (5.1) associated with the contact force is defined as:

$$r_F = -1/(1 + e^{-15|F_g - F_{ext}| + 5}) \tag{5.3}$$

where F_g is the desired insertion force, F_{ext} is the contact force. The reward is always negative as a discount reward to encourage minimal contact force. However, due to the nature of the task, contact with the environment is unavoidable, so an S shape function is proposed to allow small contact forces while strongly discouraging large ones. A visualization of this reward component is shown in Figure 5.2b The position, velocity and contact force are normalized by the maximum value allowed for each one. Finally,



(a) Visualization of the position-velocity-based component of the reward function. r_{xv} in Equation (5.2)



(b) Visualization of the contact-force-based component of the reward function. r_F in Equation (5.3)

Figure 5.2: Visualization of the reward function components.

ρ is defined as follows:

$$\rho = \begin{cases} 500, & \text{Task completed} \\ -200, & \text{Collision} \\ -1, & \text{Otherwise} \end{cases} \tag{5.4}$$

The task was considered completed if the Euclidean distance between the robot's end-effector and goal positions was less than 1 mm. The agent is encouraged to complete the task as quickly as possible by discounting the reward for every time step taken. Similar to our previous work [115], we imposed a collision constraint where the agent was penalized for colliding with the environment by giving it a large negative reward and immediately ending the episode. A *collision* is defined as exceeding the force limit F_{max}. F_{max} is a hyper-parameter that was defined as 50N in simulation or 30N in the real robot. Lastly, each component was weighted via w; all ws are hyperparameters. The performance of our new reward signal approach versus the reward signal proposed in our previous work [115] is shown in Section 5.4.1.

5.2.3 Compliance Control in Task Space

The agent's action space is based on our previous work , Section 3.2.3, which consists of learning the force control parameters of a traditional sensor-based force feedback con-

troller. More specifically, we learn the parameters of a parallel position-force controller.

The parallel position-force control requires the fine-tuning of three sets of parameters; gains of a PID for position tracking, gains of a PI for force tracking, and a selection matrix that defines the degree of control between force and position. The controller follows the control law shown in Eq. (5.5)

$$\mathbf{x}_c = S(K_p^x \mathbf{x}_e + K_d^x \dot{\mathbf{x}}_e + \mathbf{a}_x) + (I - S)(K_p^f F_e + K_i^f \int F_e dt), \tag{5.5}$$

where $F_e = F_g - F_{ext}$ (goal contact force minus sensed contact force), $\mathbf{x}_e = \mathbf{x}_g - \mathbf{x}$ (goal pose minus current pose), $\mathbf{a}_x$ represents an arbitrary translation/rotation given by the agent, a change to the control law presented in Section 3.2.3 is introduced in this chapter, where the action $\mathbf{a}_x$ is constrained by the selection matrix S. This change reduces the risk of executing a dangerous undesired behavior on the robotic system. $\mathbf{x}_c$ is the commanded positions to the robot. The selection matrix is

$$S = diag(s_1, ..., s_6), \quad s_j \in [0, 1]$$

The parameters to be learned are K_p^x, K_p^f, S, and $\mathbf{a}_x$. One for each of the 6 Cartesian degrees of freedom. The remaining parameters K_d^x and K_i^f were defined proportionally to the K_p^x and K_p^f respectively. Therefore, the action space consists of 24 parameters. Each parameter is bounded to a continuous range of valid values. More details are provided in Section 3.2.3.

PID Gains Scheduling

An additional concept is explored in this chapter, PID gains scheduling [124]. Parallel force control for sub-millimeter tolerance insertion tasks tends to get stuck in the region very close to the alignment of the peg onto the hole. In the presence of very small position errors, the PID position controller barely generates any signal. On the other hand, the PI controller overcome the position PID controller due to small contact forces or noise coming from the F/T sensor. Therefore, on insertion tasks with sub-millimeter tolerances, the force controller does not move towards the target pose due to small

resistance from the contact with the environment. To address this issue, we introduce PID gains scheduling, where once $\mathbf{x}_e$ has been reduced to a position error of less than 1 cm, the PID gains are then scaled up based on the position error $K_p^x = K_p^x * (1/\mathbf{x}_e)$. As $\mathbf{x}_e$ approaches zero, the value of K_p^x has a hyperbolic growth, thus the value of K_p^x is bounded to be maximum $kn = 10$ times its *current* value (the agent's chosen value). Section 5.4.2 provides a comparison between the use of a traditional PID versus the PID gains scheduling on our proposed method.

5.2.4 Domain Randomization

Domain randomization (DR) [84] is a popular method in robot learning to increase the generalization capabilities of policies trained in simulation, facilitating the transfer of the policy to a real-world robotic agent with minimal to no further refinement of the policy. In principle, the goal of DR is to provide enough variability to the simulated environment during training to generalize better to real-world conditions. In robot learning, DR randomizes a set of numerical parameters, N_r, of a physics simulator. With each parameter ψ_i being sample from a randomization space $\Psi \in \mathbb{R}^{N_r}$. Each parameter is bounded on a close interval $\{[\psi_i^{low}, \psi_i^{high}]\}_{i=1}^{N_r}$. For every episode, a new set of parameters is sample from the randomization space $\psi_i \in \Psi$. The most common approach is to draw sample uniformly from the randomization space. In this work, the randomized aspects of the peg-in-hole tasks are defined in Table 5.1.

5.2.5 Curriculum Learning

Curriculum Learning comes from the notion that the order in which information is organized and presented to a learner impacts the learner's performance and training speed. This idea can be observed in the way humans learn, starting with simple concepts and gradually progressing to more complicated problems[125, 122]. CL can also be observed in the way we train animals [126].

In this work, we follow the notion that starting with easier tasks can help the agent

Condition		Set
Initial position (relative to goal)	Position (mm)	[-50, 50]
	Orientation (°)	[-30, 30]
Peg shape		[Cylinder, Cuboid, Hexagon prism, Triangular prism]
Hole Clearance (mm)		$[3.0, 5 \times 10^{-1}]$
ϵ: Distance from full insertion (mm)		$[1.5 \times 10^{1}, 1]$
Friction (in Gazebo: *surface/friction/ode/mu*)		[1, 5]
Stiffness (in Gazebo: *surface/friction/ode/kp*)		$[5.0 \times 10^{-4}, 1.0 \times 10^{-6}]$

Table 5.1: Domain Randomization parameters and their maximum range of values

learn better when presented with more difficult tasks later on. We consider the CL problem in the context of DR, where the goal is to reduce the training time by guiding the learning process without loosing domain transferability. Then, the problem becomes how to select parameters from the randomized space Ψ to guide the agent's training. To this end, we consider four main approaches:

- Curriculum-based DR: The DR parameter's range of values is determined by the curriculum.
- The curriculum's evolution: a linear approach vs an adaptive approach.
- The DR sampling strategy: a Uniform distribution (UDR) vs a Gaussian distribution (GDR).
- A dynamic reward function based on the curriculum vs a standard reward function.

Curriculum-based Domain Randomization

We tackle the problem of defining a strategy to reduce the complexity of choosing a value for each randomization parameter. Though each parameter of the randomized space ψ can be considered a degree of freedom that can be controlled to define the training tasks,

adding new parameters would increases the difficulty of choosing the sequence of tasks to train the agent. Therefore, in order to simplify the problem while preserving the benefits of domain randomization, we propose the following approach: we represent the difficulty level $\mathbf{L}$ of a task as a numerical value in a close interval $[0, 1]$, from easiest to hardest. Then, a sub-set of each randomization parameter ψ_i is defined based on the difficulty level $\mathbf{L_{ep}}$ at the beginning of each episode during training:

$$\psi_i : [\psi_i^{low}, \psi_i^{low} + \psi_i^{high} * \mathbf{L_{ep}}] \tag{5.6}$$

where we assume that the parameter's set ψ_i is defined in ascending order, such that, at *low* and at *high*, the task is relatively the easiest and the hardest, respectively. The parameters considered in this work and their corresponding set are shown in Table 5.1.

Adaptive Curriculum Learning

We consider two approaches to update the curriculum difficulty; on the one hand, the naive approach is to monotonically increase the difficulty in a linear way, regardless of the agent's performance, i.e.,

$$\mathbf{L_{ep}} = ep/ep_{max} \tag{5.7}$$

with $\mathbf{L_{ep}}$ being a constraint equal to 1 if the current episode number exceeds a defined maximum number of episodes. On the other hand, we propose an adaptive curriculum based on the agent's performance $\mathbf{P}$ during the last few episodes. The agent's performance is computed as the success rate of the last few episodes. Based on the agent's performance, the curriculum's level is updated by a defined step size L_{step}. Two thresholds are also defined. If the agent's performances surpass L_{thld_up} or fall below L_{thld_down} such thresholds, then the curriculum's level is increased or decreased respectively. Algorithm 2 describe our adaptive curriculum approach.

Algorithm 2 Adaptive Curriculum Learning Evolution

$\mathbf{P} = 0$
for Every episode *ep* **do**
    Update ψ based on $\mathbf{L_{ep}}$ ▷ Eq. (5.6)
    Sample task from ψ
    // +1: success, -1: failure
    $\mathbf{P}$ += rollout current policy π on task
    Update policy
    if $\mathbf{P} \geq L_{thld\_up}$ **then**
        L_{ep} += L_{step}
        $\mathbf{P} = 0$ ▷ Consider newest rollouts
    else if $\mathbf{P} \leq L_{thld\_down}$ **then**
        L_{ep} −= L_{step}
        $\mathbf{P} = 0$ ▷ Consider newest rollouts

Domain Randomization Sampling Strategy

We consider the type of distribution from which the randomized parameters are sampled. Instead of the typical uniform distribution (UDR), we propose the use of a Gaussian distribution (GDR), $\mathcal{N}(\mu, \sigma^2)$, with the mean being centered around the current curriculum's level $\mathbf{L}_{ep}$, and variance is a hyperparameter. The reason behind this choice is to keep increasing the general difficulty of the task with the increment of the difficulty level, but with a small probability, the curriculum can generate an easier task than the difficulty level to reduce the catastrophic forgetting problem [127, 128].

Dynamic Reward Function

Lastly, for our target task domain, we consider desirable for the RL agent to learn to handle the *hardest* conditions to improve transferability to the real-world environment. To this end, we propose and evaluate a dynamically evolving reward with respect to the curriculum level difficulty. More specifically, the reward r, as defined in Section 5.2.2,

is scaled by the current difficulty level L_{ep}; thus, the full reward would be obtained only when the agent reaches and maintains the hardest level.

$$\mathbf{r_t^d} = r * \mathbf{L_{ep}} \tag{5.8}$$

where r_t^d stands for the dynamic reward at time t. In other words, at each time step, the reward obtained by the agent is a fraction of the full possible reward for reaching such state.

5.3 Experiments and results

Through the following experiments, we aimed to understand the performance of our proposed method compared to alternative approaches, in terms of sample efficiency and generalization. To that end, experiments were performed with novel tasks not seen during training in simulation and in the real-world environment, using insertion tasks with medium grade industrial-level tolerances ($\pm 0.01\ mm$).

The *baseline* used throughout these experiments was based on our previous work [123], which mainly focused on the use of DR to enhance domain transferability. This experimental section focus on comparing the different curricula designs, sampling strategies for DR, and curriculum-based dynamic reward. As such, our previous work [123] has been updated to include the new reward function and PID gain scheduling approach, proposed and described in Section 5.2.2 and 5.2.3 respectively, which is used as the *baseline* in this study. Ablation studies of these components of our proposed method are discussed in the Appendix.

5.3.1 Experimental Setup

A simulated environment was used both for training and validation. The Gazebo simulator [104] version 9 was used. The choice of simulation environment is discussed in Section 5.5. Two real-world environments were used for validation purpose only; no

further re-training was performed on the target domains[3]. The components of the real-world setup is described in Figure 5.3. Both environments consist of a Universal Robot 3 e-series robot arm with a control frequency up to 500 Hz. The robotic arm had a force/torque sensor mounted at its end-effector. In the simulation, the peg was considered as part of the robot's end-effector, as shown in Figure 5.4. The real-world robot simply used a Robotiq Hand-e parallel gripper. For the toy environments described in Section 5.3.5, this parallel gripper and a cuboid holder facilitate achieving a strong and stable grasp, similar to the simulation environment. However, for the industrial insertion tasks, we avoided the used of custom-made holders for the real-wold tasks, which increased the difficulty of the tasks as discussed in Section 5.3.5.

Our implementation of the RL agent that controlled both the simulated and real robot was developed on top of the Robot Operating System (ROS) [120] with the Universal Robot ROS Driver[4]. In both environments, training of the RL agent was performed on a computer with an Intel i9-10900X CPU and NVIDIA® Quadro RTX™ 8000 GPU. See the accompanying video [5]

5.3.2 Training

The training phase consisted of the repeated execution of the insertion task using a variety of peg shapes and physical parameters of the simulator, as described in Table 5.1. An *episode* was defined as a maximum of 1000 time steps, with each step being 50 ms. Early termination of an episode occurs under three conditions; 1) the target goal is

[3]Despite Gazebo's simulation of the high-stiffness robot being accurate, the robot controllers respond faster than the real robot (maybe due to safety speed reduction on the side of the real robot controller, which we have not modify). Therefore, a minimal calibration is required. From our experience, scaling the reference trajectory or the command send to the controller by a factor of two worked well enough. A rough similarity between the simulation and real robot controller is enough to enable straightforward sim2real transfer.

[4]ROS driver for Universal Robot robotic arms developed in collaboration between Universal Robots and the FZI Research Center for Information Technology https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

[5]Graphical abstract and experimental results: https://youtu.be/_FVQC5OcGjs

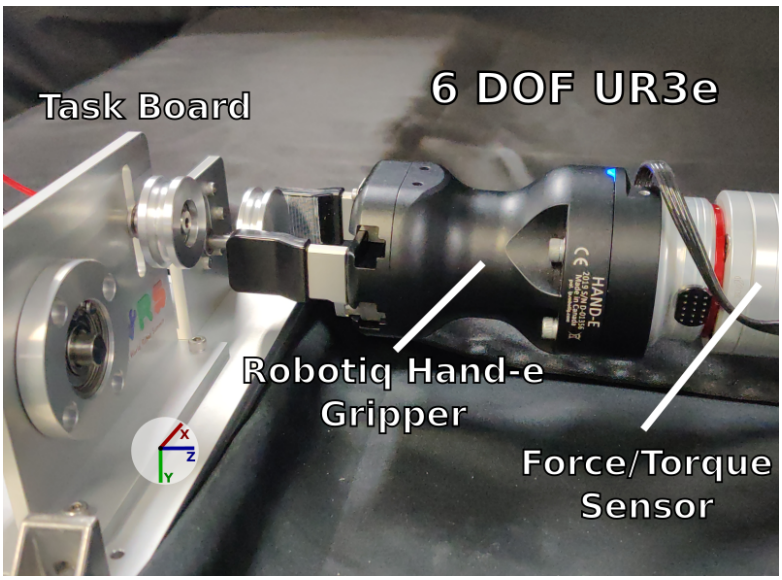


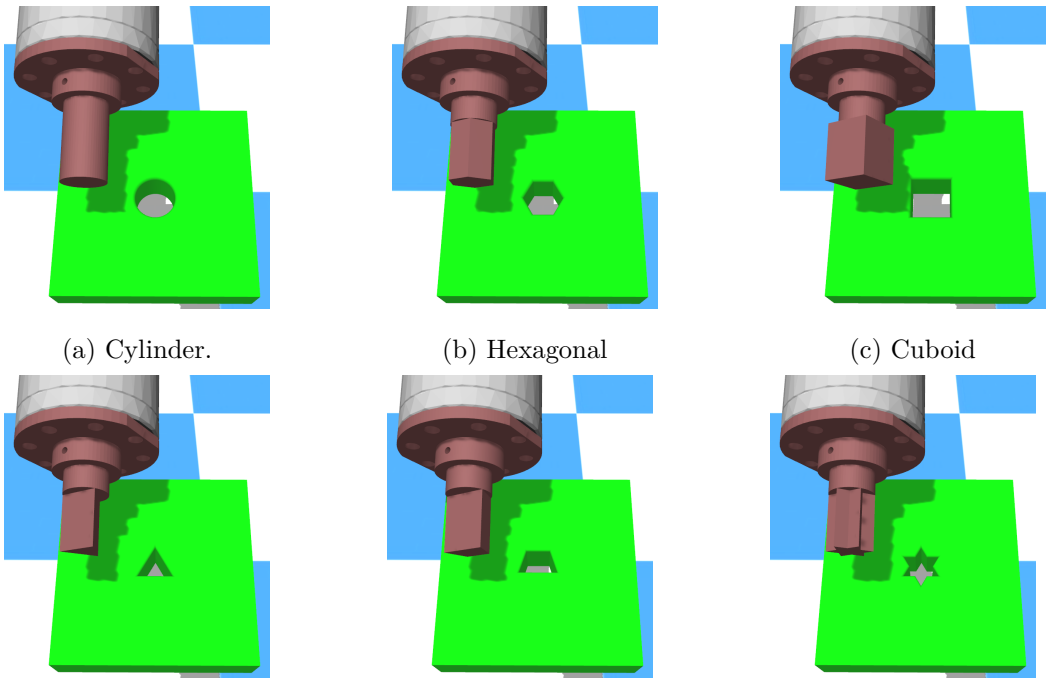
Figure 5.3: Real experiment environment with a 6-degree-of-freedom UR3e robotic arm. WRS2020 Task board is shown, along side the three insertion tasks used for validation, motor pulley, bearing, and shaft. Each task has industrial level sub-mm tolerances.

reached, the peg inserted, and within ϵ distance from the full insertion, as described in Table 5.1. 2) the robot collides with the task board, i.e., a large contact force is sensed at any point during the task (more than 50 N in simulation or more than 30 N with the real-world robot). 3) the agent gets stuck, thus, the cumulative reward decreases to less than a set value R_{min}.

5.3.3 Learning performance

First, we compare the learning performance of the approaches presented in Section 5.2.5, 5.2.5, and 5.2.5:

- *Baseline*: DR without curriculum learning (No Curriculum), as described in Section 5.3.
- Linear curriculum with Uniform distribution for DR (Linear Curriculum UDR).
- Linear curriculum with Gaussian distribution for DR (Linear Curriculum GDR).



(a) Cylinder. (b) Hexagonal (c) Cuboid

(d) Triangular (e) Trapezoid (f) Star

Figure 5.4: Simulated peg-in-hole environments. The cylinder, hexagonal prism, cuboid and triangular prism were used during training. The trapezoid prism and the star prism were used for testing.

- Adaptive curriculum with Uniform distribution for DR (Adp. Curriculum UDR).
- Adaptive curriculum with Gaussian distribution for DR (Adp. Curriculum GDR).

Each training session had a maximum of $100,000$ time steps, one-fifth of the training time required in our previous work [123]. As described in Section 5.2.4, each episode is generated with a different set of values for the randomization parameters. Figure 5.5 shows the cumulative reward per method during a complete training session. Each training session was repeated with different random seeds. The average value and standard deviation are shown as the bold line and shadow region, respectively. The results are a preliminary highlight of the significant improvement of applying Curriculum Learning compared to the *baseline*, which relied primarily on Domain Randomization alone. Furthermore, the adaptive curricula had a considerable performance above a simple linear increment of the curricula difficulty. Finally, using a Gaussian distribution instead of a

Uniform distribution for the sampling of Domain Randomization parameters also significantly improves the agents' performance during learning. The dynamic reward (DyRe) approach discussed in Section 5.2.5 is not included here as the scale of the reward is different.

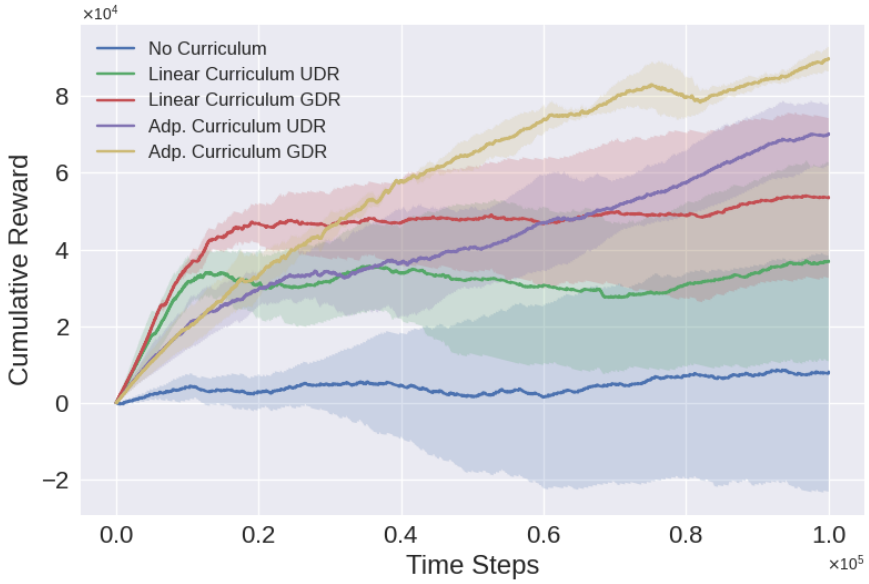


Figure 5.5: Learning curve comparison using the cumulative reward of the overall training session. Each method was trained three times. The results are aggregated as the average cumulative reward and corresponding standard deviation, represented by the bold line and the shadow region.

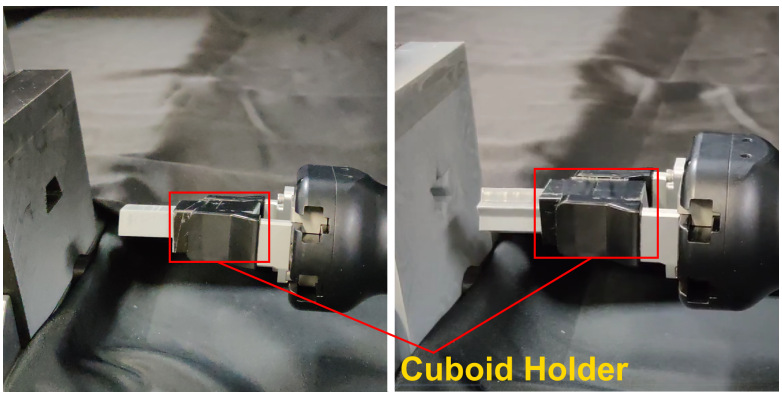
5.3.4 Evaluating learned Policies

Next, we evaluated the performance of the learned policies on novel conditions not seen during training. Each policy was executed 100 times with different initial conditions and randomized parameters (with a fixed random seed for a fair comparison). More specifically, the peg shapes used for testing were a trapezoid prism and star prism, as shown in Figure 5.4. The trapezoid introduces a non-symmetric-shaped peg. The star prism peg is more challenging due to its sharp corners that make the peg prone to getting stuck during the aligning phase, making the overall insertion task harder to complete during the allowed time limit of 50 seconds (same time limit as during training).

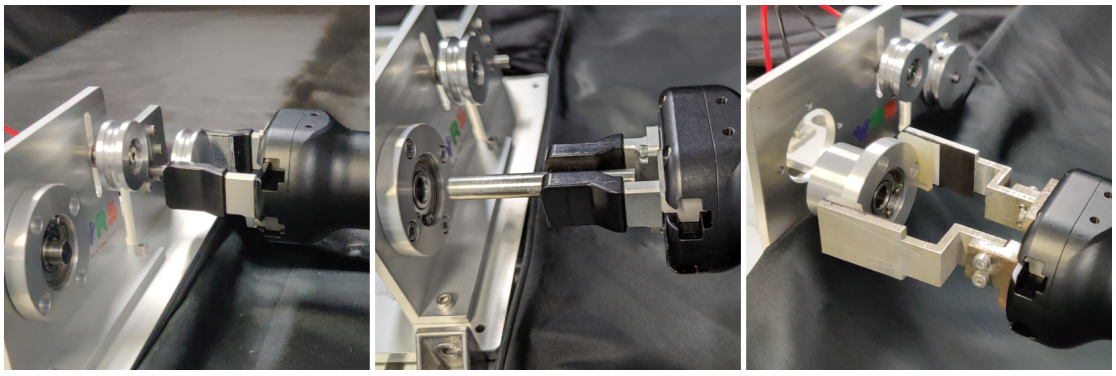
The results are shown in Table 5.2. They include a comparison of the overall success rate and the average time needed to complete the task; failure cases are not included in the computation of the average time. Two main conclusions can be drawn from these results; 1) A curriculum may seem to have a better learning performance, but the resulting policy may not transfer well to novel environments, as is the case with the *Linear Curricula* methods. Such linear curriculum approaches performed just slightly better than not using a curriculum at all. 2) The most successful methods are not necessarily the fastest. Our simulation environment did not handle very well friction between the peg and the task board, due to the high stiffness of the robot joints. In this almost friction-less world, the *Adp. Curriculum UDR* method is able to solve the tasks between 20% to 50% faster than our best method *Adp. Curriculum GDR DyRe*. However, the success rate of our method is at least 19% higher. The main reason for such results is that since contact force and collision avoidance have higher priority than speed during learning, our proposed method moves slower when the peg gets closer to the task board so the contact force is reduced. This conclusion is further supported by the real-world experimental data described next in Section 5.3.5.

Method	**Trapezoid Prism**		**Star Prism**	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
No Curriculum	0.88	9.585	0.627	11.304
Linear Curriculum UDR	1.00	9.817	0.696	9.152
Linera Curriculum GDR	1.00	11.650	0.775	11.780
Adp. Curriculum UDR	1.00	**6.881**	0.706	**6.875**
Adp. Curriculum GDR	1.00	8.460	0.794	8.013
Adp. Curriculum UDR DyRe	1.00	8.429	0.873	11.602
Adp. Curriculum GDR DyRe	1.00	8.400	**0.902**	11.544

Table 5.2: Evaluation of learned policies on novel conditions.



(a) Trapezoid and star prism pegs



(b) Motor Pulley, Shaft, and Bearing.

Figure 5.6: Real-world experimental scenarios. **Left**: 3D printed primitive shape-pegs, different from the ones used for training in simulation. **Right**: Industrial level insertion tasks from the WRS2020 Robotics Assembly Challenge.

5.3.5 Real-world experiments

We performed two sets of experiments to evaluate the transferability of the learned policies to the real world and to novel tasks. The experiments were performed using the *baseline* (No Curriculum) method and our newly proposed method (Adp. Curriculum DyRe GDR), which achieved the best results from the evaluation in simulation. The first set of tasks consisted of the same trapezoid and star prism-shaped pegs as the simulation experiments, which were not presented during training. A simplified peg was 3D printed using PLA material, as shown in Figure 5.6a. The second set of tasks consisted of novel

industrial level insertion tasks (See Figure 5.6b), similarly, these tasks were unseen during training in simulation. Both sets of tasks had sub-millimeter tolerances. Thirty trials were performed per method and task. The success rate and average completion time were measured where the initial position and orientation of the robot's end-effector at each trial were different and randomly sampled from a fixed random seed to fairly compare both methods. Each trial had a 500-time steps limit, i.e., 25 s.

Primitive Shaped Pegs

The 3D printed pegs were designed with a cuboid holder, as shown in Figure 5.6a, to increase the stability of the grasp. As a result, the stiffness of the contact is very high, as the task board was also firmly fixed to the workspace. Additionally, there is high friction due to the PLA material used for 3D printing and the imperfections on the printed surface. The high stiffness and friction made the task challenging. The results are shown in Table 5.3. As mentioned before, for this test, the *baseline* was also trained with only one-fifth of the samples shown to be needed [123] to learn a successful policy. Thus, the learned policy's less refined force control tends to apply too much force to complete the task quickly, but the high friction and the corners of the star-shaped hole cause the peg to get stuck easily. Therefore, the *baseline* method struggled to align the star prism peg and to get unstuck. On the other hand, our newly proposed approach successfully adapted to the real-world environment and succeeded at the novel tasks without further re-training the policies, just a straightforward sim-to-real transfer.

Industrial Level Insertion Tasks

The second set of tasks used for evaluation consisted of 3 insertion tasks with industrial level tolerances. These were chosen from the assembly task used in the Industrial Robots Assembly Challenge of the World Robot Summit 2020 edition [129]. The tasks, as shown in Figure 5.6b, were the insertion of a pulley into a motor shaft, a shaft into a bearing, and a bearing into a plate. Similar to the previous tasks, the learned policies were directly transferred from the simulation environment without further training. These

Method	Trapezoid Prism Peg		Star Prism Peg	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
No Curriculum	1.000	6.500	0.000	-
Adp. Curriculum DyRe GDR	**1.000**	**5.465**	**1.000**	**6.023**

Table 5.3: Evaluating learned policies on the real-world environment, using 2 toy scenarios not seen during training on simulation.

tasks are considerably more challenging as the grasp's stability significantly impacts the success. All three manipulated objects are round and grasped directly with a standard parallel gripper. Thus, torques applied along the direction of the grasp could easily change the object's orientation in the gripper. Small orientation changes significantly affect these very tight insertion tasks.

The results are shown in Table 5.4. Our newly proposed method (Adp. Curriculum DyRe GDR) achieved a high success rate in all the tasks. For the motor pulley and the shaft tasks, our method also solves the task faster by finding the right fit faster. Our method is less likely to get stuck as it applies less contact force as shown in Figure 5.7 and 5.8 In the case of the bearing task, the *baseline* method, when successful, is slightly faster as it tends to apply higher contact force and move faster once the parts are aligned. However, the same high contact force makes it harder to find the proper alignment, thus resulting in a very low success rate. As a result, our newly proposed method outperforms the *baseline* method, achieving a much higher success rate. These results are better appreciated in the supplemented video[6].

Learning Force Control

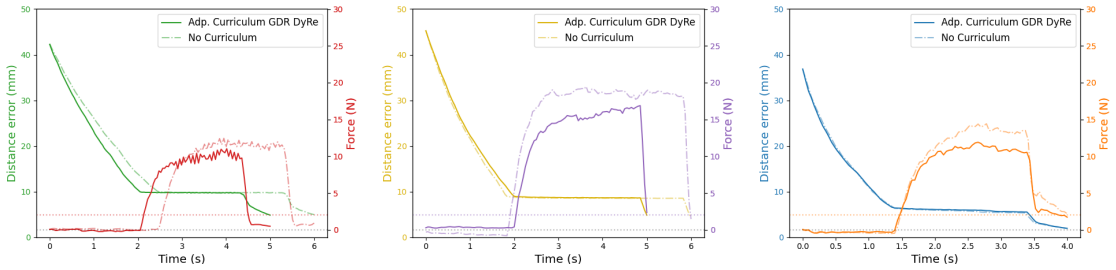
In addition to the success rate and time to completion, we compare the detailed performance of the two methods. Figure 5.7 shows the performance of both methods side

[6]Supplemental video: https://youtu.be/_FVQC5OcGjs

Method	Motor Pulley		Shaft		Bearing	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)	Success Rate	Avg. Time (s)
No Curriculum	0.400	8.258	0.667	9.199	0.267	6.819
Adp. Curriculum DyRe GDR	0.867	7.250	0.833	7.015	0.700	7.212

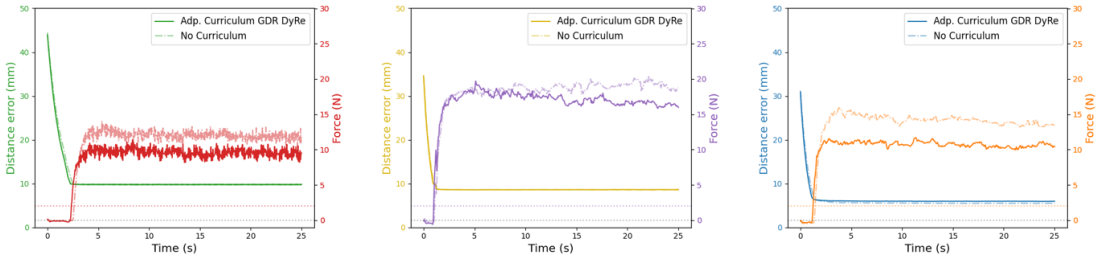
Table 5.4: Evaluating learned policies on the real-world environment, using 2 toy scenarios not seen during training on simulation.

by side for the three industrial insertion tasks. For simplicity, only the z-axis (i.e., the insertion direction), distance error (mm), and contact force (N) are displayed. As shown in Figure 5.7, our proposed method is more time-efficient and applies less contact force to the coupling part. Less contact force is desirable to avoid damage to either the assembly part or the robot. Similarly, Figure 5.8 shows the comparison of trials where both agents fail to complete the task on time. Though both agents failed, our method again shows a reduced exertion of contact force. In both cases, our method applies about 30% less contact force.



(a) Motor Pulley (b) Shaft (c) Bearing

Figure 5.7: Agents performance on WRS2020 insertion tasks. For clarity, only the z-axis (Insertion direction) distance error and contact force are displayed. Comparison was made for each task when both methods successfully completed the task.



(a) Motor Pulley (b) Shaft (c) Bearing

Figure 5.8: Performance of both methods where both failed to complete the task within the time limit.

5.4 Ablation studies

In this section we compare the performance of the proposed method where the improvements presented in this work; the new dense reward function (Sect. 5.2.2), and the addition of a PID gains scheduling to the force controller (Sect. 5.2.3). Similar to the experimental setup described in Section 5.3.4, each method was evaluate on simulation by executing their corresponding learned policy over a 100 trials for each task.

5.4.1 Reward Functions

The newly proposed dense reward function includes the robot's end-effector velocity combined with the error position. Our aim is to encourage the agent to move faster while being far from the target pose, but to move slower when closer to the target position to reduce the risk of high contact forces. For a fair comparison, the implementation of the *Old reward function* method and our proposed *New reward function* were identical except for the type of reward function. Both methods are based on our proposed method (Adp. Curriculum GDR DyRe), as described in Section 5.2.5. Figure 5.9 shows the comparison of the training session, and the overall cumulative reward for each method. In addition, both approaches were tested on two novel tasks on simulation, the same tasks described in Section 5.3.4. The results, displayed in Table 5.5, shows a significant improvement in performance. Our approach using the *New reward* achieved a higher

success rate. Additionally, our approach also was more efficient at solving the tasks. In average, the task are solved at least twice as fast.

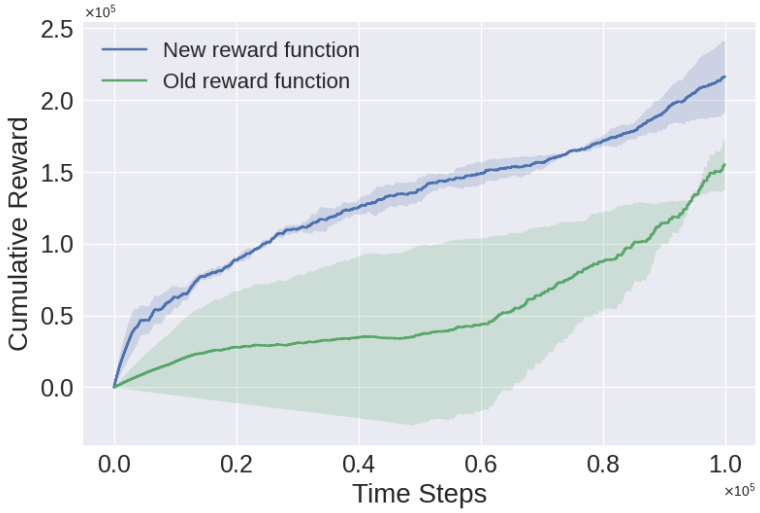


Figure 5.9: Learning curve comparison.

Method	Trapezoid Prism Peg		Star Prism Peg	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
Old Reward	0.98	14.979	0.85	12.271
New Reward	**1.000**	**5.465**	**1.000**	**6.023**

Table 5.5: Success rate on novel tasks on the simulated environment.

5.4.2 Force Controller Position PID types

Furthermore, we updated the PID position controller of our force controller to enhance the performance of the agent when the position error is very small. Similarly, both method were identical except for the implementation of the PID position controller and based on our proposed method (Adp. Curriculum GDR DyRe), as described in

Section 5.2.5. The results of evaluating each method on novel tasks not seen during training are shown in Figure 5.10, for the learning curve, and in Table 5.6. The results show a considerable improvement of performance when using our proposed *PID gain scheduling* approach. The success rate achieved by our *PID gain scheduling* approach is about twice the achieved with the *Normal PID* approach. On top of that, our *PID gain scheduling* approach can solve the tasks in less than half the time required by the *Normal PID* approach.

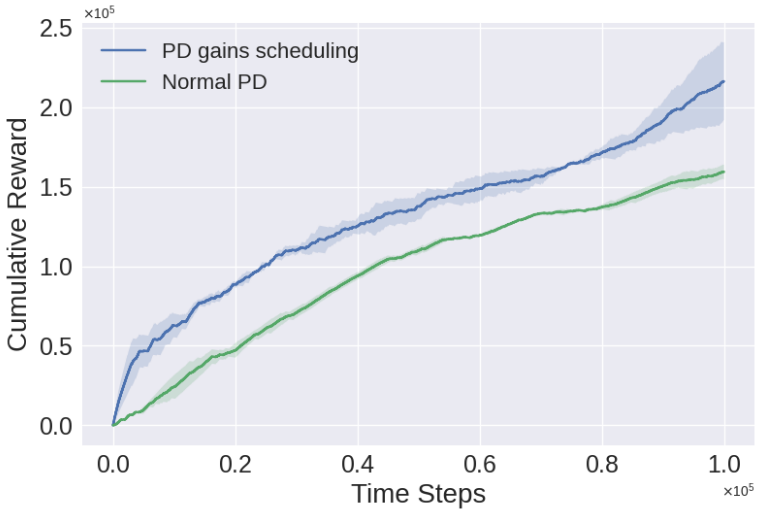


Figure 5.10: Learning curve comparison.

Method	Trapezoid Prism Peg		Star Prism Peg	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
Normal PID	0.780	12.798	0.500	13.949
PID gains Scheduling	**1.000**	**5.465**	**1.000**	**6.023**

Table 5.6: Success rate on novel tasks on the simulated environment.

5.5 Discussion

Training a reinforcement learning agent with a curriculum that starts from easier tasks with reduced risk of encountering fatal states (e.g., a collision during a manipulation task) improves the learning sample efficiency and overall performance. In this case study, in particular, we integrate CL to contact-rich peg insertion tasks. A task is defined by various physical parameters as described in Table 5.1. We aim to allow the agent to carefully explore more states by presenting tasks in increasing order of difficulty, e.g., by reducing the stiffness of the contact between the peg and the board, the agent is less likely to apply excessive contact force to the environment (i.e., a collision). At the same time, starting with easier tasks, such as a shorter distance from the initial position to goal one, reduces the overall exploration. The curriculum is design around a domain randomization approach to preserve and enhance the domain transferability benefits from DR. Nevertheless, the type of curricula is very relevant for achieving better performance, both in terms of sample efficiency and success rate, as seen in the results in Sect. 5.3.5.

From our previous work [123], we demonstrated that with sufficient domain randomization-based training in simulation (at least $500,000$ time steps or about 8 hours) and further retraining in the real-robot environment, it is possible to learn policies that adapt to novel domains successfully. The present chapter introduce a study on Curriculum Learning to tackle the problem of sample-efficiency, and the need to retrain in the target domain. The experimental results on the real-robot environment confirms that our newly proposed method is effective in learning contact-rich force control tasks. Despite that our proposed method was trained only in simulation with one-fifth of the training time used in our previous work [123] and without further retraining, it achieves a high success rate on novel tasks, including challenging industrial insertion tasks with sub-millimeter tolerances.

5.6 Conclusions

This chapter has studied the application of different approaches that combine Curriculum Learning with Domain Randomization to learn contact-rich manipulation tasks, particularly assembly tasks such as peg insertion. Based on such a study, we proposed to improve sample efficiency and generalization by training an agent purely in simulation, with the training being guided with CL and enhanced with DR. Additionally, this work introduced two enhancements to our learning framework, a new dense reward function and a PID gain scheduling approach, described in Section 5.2.2 and 5.2.3 respectively, and validated in the Appendix.

The learning framework proposed in this work is based on our previous work [123] where a combination of sim2real with DR was proposed. Our previous methods still required a considerably large amount of agent's interaction with its environment and additional refinement in the real-world environment to learn a robust policy. On the contrary, our novel approach can be trained purely in simulation with only toy insertion tasks. Empirical results showed that with our proposed method a successful policy can be learned using only one-fifth of the samples needed in our previous work. Such policies can be straightforwardly transferred to real-world environments and still achieve a high success rate, up to 86%, on novel complex industrial insertion tasks not seen during training.

Chapter 6

Discussion

6.1 Contributions

In this dissertation, a Reinforcement Learning (RL) framework for industrial position-controlled robotic manipulators has been proposed. The proposed framework has been designed to enable safe interactions of a real-world robotic agent. More specifically, the proposed learning framework enables robotic manipulators to learn how to solve high-precision contact rich manipulation task, such as insertion tasks. To solve such tasks, the robot needs to carefully control the interaction forces between the parts being manipulated. To that end, in Chapter 3, we have proposed a learning framework that combines RL with traditional force feedback control.

A drawback of Machine Learning methods, including RL, is the requirement of a large amount of data to learn. The demand for such data increases with the increment of the tasks complexity. The complex contact-rich tasks considered on this dissertation demand a considerable amount of data, i. e., experience, from the robotic agent to learn a successful policy. Despite that we have presented a learning framework that allows training of RL policies directly on real hardware, the requirement of large amount of interactions with the environment risk wear and tear of the robot. Similarly, the random nature of the initial policies and the need for exploration put the robot and its

environment at risk of damage. For this reason, the second objective of this dissertation was the proposal of methods to reduce such risk by exploiting the capabilities of simulation environment that can emulate to a great level of fidelity the dynamics of a robotic manipulator and the contacts between objects. Furthermore, simulation environments are easier to manipulate and change, which allow us to train the robotic agent on a wide variety of environmental conditions that in turn facilitates the learning of a more robust policy and the generalization to novel but similar tasks, not present during training. In Chapter 4, we have proposed a Sim2Real approach to learn policies in simulation and transfer them to a real-world environment, where the policies can be further trained (finetune) if needed. The proposed method has been evaluated on several conditions to show the robustness of the policy to uncertainty of the task. Additionally, the proposed method was evaluated on novel task not seen during training to demonstrate the generalization capabilities of our method.

Although, the method proposed in Chapter 4 significantly reduces the need to train policies directly on real hardware, the learning process itself still requires a considerable amount of experience to find a successful policy, which can take several hours. To address this problem, in Chapter 5, we focus on the concept of Curriculum Learning (CL). The idea is based on the notion that the order in which information is presented to a learner affects is ability to learn. In particular, by starting training with easier tasks, an agent can learn better and quicker when presented with harder and more complex tasks later on. Thus, in Chapter 5 a study of CL applied to our learning framework for contact-rich tasks is presented. Then, a new method based on CL is presented. The proposed method is evaluated on a variety of complex industrial-level insertion tasks. The effectiveness of the method is compared to the proposed method presented in Chapter 4. The results show a considerable improvement of the new method, by learning faster, up to five times faster, and better, even when evaluated on task not seen during the training phase.

6.2 Open Challenges and Future Work

The main limitation of the method presented in Chapter 5 is the assumption that the domain randomization parameter ranges are organized in order from *easy* to *difficult*. Prior knowledge is required to determine the *difficulty* of a physical parameters on a given task. Such prior knowledge is task-specific and not necessarily easy to obtain or even impossible to determine manually. For example, considering only the peg-in-hole task and the stiffness between the peg and the task board, a low-stiffness can intuitively seem easier to handle for a high-stiffness robot arm, as less careful force control is required to achieve the task without generating large contact forces. However, depending on the material and how low the stiffness is, the peg may get stuck, or the task board may be deformed to such an extent that the task becomes impossible to solve. To tackle this problem, an interesting future avenue is to add a layer of learning, following works such as [85, 130]. The main idea of this line of research work is to train a neural network to define the RL agent's tasks. In other words, the network learns to choose the best values for the randomization parameters at each episode to increase the performance of the learned policy. The approaches differ in how the new network is trained, and how the agent's performance is defined, such as the cumulative reward, success rate, or another evaluation metric. Following these self-learned curricula approaches reduces the burden on prior knowledge, though as discussed in [85], a self-learned curriculum can provide an insight into incompatibilities between the task and randomization ranges. Therefore, such approaches may allow the use of many other parameters of the physics simulator for domain randomization, potentially increasing the transferability to novel domains. Nevertheless, the possible downside is the requirement of longer training sessions due to the added complexity.

Another future avenue to further improve the presented work is the choice of a simulation environment. At the time of writing this chapter, there are various physics engine simulators available that simulate the contact dynamics between bodies with different degrees of accuracy, among other capabilities. Our choice of the Gazebo simulator was motivated by its realistic simulation of rigid position-controlled robots. Additionally, the

availability of ROS controllers that worked the same in the simulated and the real-world robot reduces the implementation burden and facilitates sim2real transfer. Nonetheless, other simulators provide better contact dynamics and are better adapted for Reinforcement Learning applications, such as Mujoco [131], or Nvidia Isaac Sim [132]. Working with such simulators would be a significant improvement, as Domain Randomization is also easier to implement for vision-based learning methods and physical parameters of the simulated environment.

References

[1] N. Yashiro, "Aging of the population in japan and its implications to the other asian countries," *Journal of Asian Economics*, vol. 8, no. 2, pp. 245–261, 1997.

[2] R. B. Freeman, "Is a great labor shortage coming? replacement demand in the global economy," 2006.

[3] L. Kugler, "Addressing labor shortages with automation," *Communications of the ACM*, vol. 65, no. 6, pp. 21–23, 2022.

[4] G. Boothroyd, *Assembly automation and product design*. CRC press, 2005.

[5] P. Akella, M. Peshkin, E. Colgate, W. Wannasuphoprasit, N. Nagesh, J. Wells, S. Holland, T. Pearson, and B. Peacock, "Cobots for the automobile assembly line," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 1. IEEE, 1999, pp. 728–733.

[6] B. Wang, "The future of manufacturing: a new perspective," *Engineering*, vol. 4, no. 5, pp. 722–728, 2018.

[7] I. Tomašević, D. Stojanović, D. Slović, B. Simeunović, and I. Jovanović, "Lean in high-mix/low-volume industry: a systematic literature review," *Production Planning & Control*, vol. 32, no. 12, pp. 1004–1019, 2021.

[8] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[11] B. Siciliano and L. Villani, *Robot force control*. Springer Science & Business Media, 2012, vol. 540.

[12] 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, 03 1982. [Online]. Available: https://doi.org/10.1115/1.3149634

[13] T. Tsuruoka, H. Fujioka, T. Moriyama, and H. Mayeda, "3d analysis of contact in peg-hole insertion," in *Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning (ISATP'97)-Towards Flexible and Agile Assembly and Manufacturing-*. IEEE, 1997, pp. 84–89.

[14] 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, 1993.

[15] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American Control Conference*, June 1984, pp. 304–313.

[16] H. Qiao, B. Dalay, and R. Parkin, "Fine motion strategies for robotic peg-hole insertion," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 209, no. 6, pp. 429–448, 1995.

[17] T. Arai, N. Yamanobe, Y. Maeda, H. Fujii, T. Kato, and T. Sato, "Increasing efficiency of force-controlled robotic assembly:–design of damping control parameters considering cycle time–," *CIRP annals*, vol. 55, no. 1, pp. 7–10, 2006.

[18] H.-C. Song, Y.-L. Kim, and J.-B. Song, "Automated guidance of peg-in-hole assembly tasks for complex-shaped parts," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4517–4522.

[19] K. Zhang, M. Shi, J. Xu, F. Liu, and K. Chen, "Force control for a rigid dual peg-in-hole assembly," *Assembly Automation*, 2017.

[20] H.-C. Song, Y.-L. Kim, and J.-B. Song, "Guidance algorithm for complex-shape peg-in-hole strategy based on geometrical information and force control," *Advanced Robotics*, vol. 30, no. 8, pp. 552–563, 2016.

[21] K. Zhang, J. Xu, H. Chen, J. Zhao, and K. Chen, "Jamming analysis and force control for flexible dual peg-in-hole assembly," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 3, pp. 1930–1939, 2018.

[22] M. Car, A. Ivanovic, M. Orsag, and S. Bogdan, "Impedance based force control for aerial robot peg-in-hole insertion tasks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6734–6739.

[23] V. Mallapragada, D. Erol, and N. Sarkar, "A new method of force control for unknown environments," *International Journal of Advanced Robotic Systems*, vol. 4, no. 3, p. 34, 2007.

[24] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Learning impedance control of antagonistic systems based on stochastic optimization principles," *The International Journal of Robotics Research*, vol. 30, no. 5, pp. 556–573, 2011.

[25] Y. Li and S. S. Ge, "Impedance learning for robots interacting with unknown environments," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1422–1432, 2013.

[26] M.-C. Chien and A.-C. Huang, "Adaptive impedance control of robot manipulators based on function approximation technique," *Robotica*, vol. 22, no. 4, pp. 395–403, 2004.

[27] M. Racca, J. Pajarinen, A. Montebelli, and V. Kyrki, "Learning in-contact control strategies from demonstration," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 688–695.

[28] Y. Fukumoto and K. Harada, "Force control law selection for elastic part assembly from human data and parameter optimization," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–7.

[29] G. Tesauro *et al.*, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[30] J. BAXTER, "A chess program that learns by combinin td (λ) with game-tree search," in *Proc. 15th Int. Conf. Machine Learning*, 1998, pp. 28–36.

[31] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 881–888.

[32] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[33] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, "Agent57: Outperforming the atari human benchmark," in *International Conference on Machine Learning*. PMLR, 2020, pp. 507–517.

[34] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[35] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[36] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[37] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[38] J. Peters, J. Kober, K. Mülling, O. Krämer, and G. Neumann, "Towards robot skill learning: From simple skills to table tennis," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 627–631.

[39] J. Tebbe, L. Krauch, Y. Gao, and A. Zell, "Sample-efficient reinforcement learning in robotic table tennis," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4171–4178.

[40] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[41] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental robotics IX*. Springer, 2006, pp. 363–372.

[42] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[43] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2619–2624.

[44] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[45] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, vol. 87. PMLR, 2018, pp. 651–673.

[46] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[47] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Learning to grasp with primitive shaped object policies," in *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2019, pp. 468–473.

[48] W. Gao, L. Graesser, K. Choromanski, X. Song, N. Lazic, P. Sanketi, V. Sindhwani, and N. Jaitly, "Robotic table tennis with model-free reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5556–5563.

[49] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1461–1466.

[50] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," *Robotics: Science and Systems VII*, vol. 7, pp. 57–64, 2011.

[51] 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)*. IEEE, 2017, pp. 3389–3396.

[52] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

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

[54] 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)*. IEEE, 2017, pp. 819–825.

[55] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[56] F. Li, Q. Jiang, S. Zhang, M. Wei, and R. Song, "Robot skill acquisition in assembly process using deep reinforcement learning," *Neurocomputing*, vol. 345, pp. 92–102, 2019.

[57] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2062–2069.

[58] 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)*. IEEE, 2019, pp. 3080–3087.

[59] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 811–817.

[60] H. Park, J.-H. Bae, J.-H. Park, M.-H. Baeg, and J. Park, "Intuitive peg-in-hole assembly strategy with a compliant manipulator," in *IEEE ISR 2013*. IEEE, 2013, pp. 1–5.

[61] S. R. Chhatpar and M. S. 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. IEEE, 2001, pp. 1465–1470.

[62] K. Sharma, V. Shirwalkar, and P. K. Pal, "Intelligent and environment-independent peg-in-hole search strategies," in *2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE)*. IEEE, 2013, pp. 1–6.

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

[64] J. Luo and H. Li, "A learning approach to robot-agnostic force-guided high precision assembly," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2151–2157.

[65] A. Y. Yasutomi, H. Mori, and T. Ogata, "A peg-in-hole task strategy for holes in concrete," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2205–2211.

[66] 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, 2011.

[67] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2397–2403.

[68] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.

[69] 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 preprint arXiv:1509.02971*, 2015.

[70] R. Martín-Martín, M. 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," in *International Conference of Intelligent Robots and Systems (IROS)*, 2019.

[71] C. Yang, C. Zeng, Y. Cong, N. Wang, and M. Wang, "A learning framework of adaptive manipulative skills from human to robot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1153–1161, 2018.

[72] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[73] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*, 2016, pp. 3406–3413.

[74] 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, 1997.

[75] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen, "Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole as-

sembly tasks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1658–1667, 2018.

[76] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[77] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, 2013, pp. 1–9.

[78] T. Ren, Y. Dong, D. Wu, and K. Chen, "Learning-based variable compliance control for robotic assembly," *Journal of Mechanisms and Robotics*, vol. 10, no. 6, 2018.

[79] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8943–8950.

[80] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[81] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proceedings of The 2nd Conference on Robot Learning*, vol. 87. PMLR, 29–31 Oct 2018, pp. 561–591.

[82] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, "Meta-reinforcement learning for robotic industrial insertion tasks," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9728–9735.

[83] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ Interna-*

tional Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 4383–4388.

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

[85] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[86] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[87] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, pp. 1–50, 2020.

[88] 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, 1996.

[89] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[90] S. Luo, H. Kasaei, and L. Schomaker, "Accelerating reinforcement learning for reaching using continuous curriculum learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[91] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International conference on machine learning*. PMLR, 2018, pp. 4344–4353.

[92] L. Leyendecker, M. Schmitz, H. A. Zhou, V. Samsonov, M. Rittstieg, and D. Lütticke, "Deep reinforcement learning for robotic control in high-dexterity assembly tasks-a reward curriculum approach," in *2021 Fifth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2021, pp. 35–42.

[93] A. Y. Yasutomi, H. Mori, and T. Ogata, "Curriculum-based offline network training for improvement of peg-in-hole task performance for holes in concrete," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 712–717.

[94] O. Kroemer, S. Niekum, and G. D. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *Journal of machine learning research*, vol. 22, no. 30, 2021.

[95] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[96] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[97] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *International Conference on Machine Learning (ICML)*, 2019.

[98] 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)*. IEEE, 2019, pp. 5844–5850.

[99] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[100] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl Θ2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[101] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, vol. abs/1801.01290, 2018.

[102] R. Campa and K. Camarillo, "Unit quaternions: A mathematical tool for modeling, path planning and control of robot manipulators," in *Robot Manipulators*, M. Ceccarelli, Ed. IntechOpen, 2008.

[103] D. A. Lawrence, "Impedance control stability properties in common implementations," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 1988, pp. 1185–1190 vol.2.

[104] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[105] M. Kyrarini, M. A. Haseeb, D. Ristić-Durrant, and A. Gräser, "Robot learning of industrial assembly task via human demonstrations," *Autonomous Robots*, vol. 43, no. 1, pp. 239–257, 2019.

[106] 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," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[107] S. Zakharov, I. Shugurov, and S. Ilic, "Dpod: 6d pose object detector and refiner," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1941–1950.

[108] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.

[109] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems (RSS) 2018*, 06 2018.

[110] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 880–888.

[111] K. Harada, K. Nakayama, W. Wan, K. Nagata, N. Yamanobe, and I. G. Ramirez-Alpizar, "Tool exchangeable grasp/assembly planner," in *International Conference on Intelligent Autonomous Systems*. Springer, 2018, pp. 799–811.

[112] E. Masehian and S. Ghandi, "Asppr: A new assembly sequence and path planner/replanner for monotone and nonmonotone assembly planning," *Computer-Aided Design*, p. 102828, 2020.

[113] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1025–1037.

[114] Y. Wang, K. Harada, and W. Wan, "Motion planning of skillful motions in assembly process through human demonstration," *Advanced Robotics*, vol. 0, no. 0, pp. 1–15, 2020. [Online]. Available: https://doi.org/10.1080/01691864.2020.1782260

[115] 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*, pp. 1–1, 2020.

[116] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[117] 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.

[118] T. Silver, K. R. Allen, J. B. Tenenbaum, and L. P. Kaelbling, "Residual policy learning," *ArXiv*, vol. abs/1812.06298, 2018.

[119] G. Bellegarda and K. Byl, "Training in task space to speed up and guide reinforcement learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2693–2699.

[120] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[121] H.-n. Wang, N. Liu, Y.-y. Zhang, D.-w. Feng, F. Huang, D.-s. Li, and Y.-m. Zhang, "Deep reinforcement learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, no. 12, pp. 1726–1744, 2020.

[122] K. A. Krueger and P. Dayan, "Flexible shaping: How learning in small steps helps," *Cognition*, vol. 110, no. 3, pp. 380–394, 2009.

[123] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[124] V. Veselỳ and A. Ilka, "Gain-scheduled pid controller design," *Journal of process control*, vol. 23, no. 8, pp. 1141–1148, 2013.

[125] G. B. Peterson, "A day of great illumination: Bf skinner's discovery of shaping," *Journal of the experimental analysis of behavior*, vol. 82, no. 3, pp. 317–328, 2004.

[126] B. F. Skinner, "Reinforcement today." *American Psychologist*, vol. 13, no. 3, p. 94, 1958.

[127] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[128] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[129] "Industrial robotics category assembly challenge rules and regulations," 2021, https://wrs.nedo.go.jp/wrs2020/challenge/download/Rules/DetailedRules_Assembly_EN.pdf (accessed on Apr 1th, 2022).

[130] M. Svetlik, M. Leonetti, J. Sinapov, R. Shah, N. Walker, and P. Stone, "Automatic curriculum graph generation for reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[131] 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*. IEEE, 2012, pp. 5026–5033.

[132] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.

Acknowledgements

First and foremost, I am extremely grateful to my supervisor Professor Kensuke Harada for accepting me as a student in the Robotic Manipulation laboratory for the last five years, as a research student, masters course student, and a Ph.D. student. His invaluable advice, continuous support, and patience helped me reach this stage. I am also deeply indebted to my Ph.D. advisors Dr. Ixchel Ramirez-Alpizar and Dr. Damien Petit, for their invaluable supervision, support, and tutelage during the course of my Ph.D. degree. I am also grateful to Professor Mario Arbulu for his selfless support and recommendations which help me start this journey in Japan. Additionally, this endeavor would not have been possible without the generous support from the Monbukagakusho Scholarship Program from the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT), who financed my research.

Special thanks to FUJIFILM for the financial support of my research projects. In particular, thanks to Kikuchi Shinichi and Takuya Nishi for the research collaboration and advice. I would also like to extend my sincere thanks to Dr. Felix Von Drigalski and Dr. Masashi Hamaya for their guidance and mentorship during my internship at OMRON SINIC X. To everyone from the team O2AC, thank you for welcoming me to such an amazing team which allowed me to have an incredible experience at the World Robot Summit 2020 Assembly Challenge.

I would also like to thanks Associate Professor Weiwei Wan, Assistant Professor Keisuke Koyama, and Specially Appointed Assistant Professor Takuya Kiyokawa for their help and suggestions on my research. To all the member of Harada Laboratory I have met during these five years, thank you very much for your support and for making my life as an international student easier.

I have many, many people to thank for the emotional support. I cannot begin to express my gratitude and appreciation for their friendship. To my friends, Nestor, Felipe, David, and Geraldine thank you so much for always being there to listen to me and for cheering me up with your memes. Special thanks to my host family Takeo and Chieko for adopting me like a child. I am very grateful for your unwavering support and encouragement.

Most importantly, none of this could have happened without my family. Special thanks to my parents Ernesto and Jeni, for their guidance and unconditional support without which I wouldn't be here. To my brother and sister, and my extended family, for your encouragement and support, thank you so much. To my partner Kanako and her family, I cannot thank you enough for all the patience, support, and cheering during the hard times, thank you so much.

Publications

Journal Papers:

1. **Beltran-Hernandez, C.C.**, Petit, D., Ramirez-Alpizar, I.G., & Harada, K. (2022). *Accelerating Robot Learning of Contact-Rich Manipulations: A Curriculum Learning Study.* In Review IEEE Transactions on Robotics. (Preprint: ArXiv, abs/2204.12844.)

2. **Beltran-Hernandez, C.C.**, Petit, D., Ramirez-Alpizar, I.G., & Harada, K. (2020). *Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep Reinforcement Learning Approach.* Applied Sciences. 10(19):6923.
   Special Issue "Machine-Learning Techniques for Robotics".
   DOI:10.3390/app10196923

3. **Beltran-Hernandez, C. C.**, Petit, D., Ramirez-Alpizar, I. G., Nishi, T., Kikuchi, S., Matsubara, T., & Harada, K. (2020). *Learning force control for contact-rich manipulation tasks with rigid position-controlled robots.* IEEE Robotics and Automation Letters, 5(4), 5709-5716.
   DOI: 10.1109/LRA.2020.3010739

4. Wang, Y., **Beltran-Hernandez, C. C.**, Wan, W., & Harada, K. (2021). *Hybrid trajectory and force learning of complex assembly tasks: A combined learning framework.* IEEE Access, 9, 60175-60186. DOI: 10.1109/ACCESS.2021.3073711

5. Wang, Y., **Beltran-Hernandez, C. C.**, Wan, W., & Harada, K. (2022). *An Adaptive Imitation Learning Framework for Robotic Complex Contact-Rich Insertion Tasks.* Frontiers in Robotics and AI, 414.
   DOI: 10.3389/frobt.2021.777363

6. von Drigalski, F. W. H. E., Kasaura, K., **Beltran-Hernandez, C. C.**, Hamaya, M., Tanaka, K., & Takamitsu, M. (2022). *Uncertainty-aware manipulation planning using gravity and environment geometry.* In Review. IEEE Robotics and Automation Letters. To appear in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2022.

7. von Drigalski, F. W. H. E., **Beltran-Hernandez, C. C.**, Nakashima, C., Hu, Z., Akizuki, S.; Ueshiba, T., Hashimoto, M., Kasaura, K., Domae, Y., Wan, W., & Harada, K. (2022). *Team O2AC at the World Robot Summit 2020: Towards Jigless, High-Precision Assembly* In Review. Advanced Robotics Special Issue on Industrial Robot Technology - Selected Papers from World Robot Summit 2020.

International Conference Papers (with peer-review):

1. **Beltran-Hernandez, C. C.**, Petit, D., Ramirez-Alpizar, I. G., Nishi, T., Kikuchi, S., Matsubara, T., & Harada, K. (2020). *Learning force control for contact-rich manipulation tasks with rigid position-controlled robots.*
   In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Published at IEEE Robotics and Automation Letters.

2. **Beltran-Hernandez, C. C.**, Petit, D., Ramirez-Alpizar, I. G., & Harada, K. (2019). *Learning to Grasp with Primitive Shaped Object Policies.*
   In IEEE/SICE International Symposium on System Integration (SII), Paris, France. DOI:10.1109/SII.2019.8700399

3. Wang, Y., **Beltran-Hernandez, C. C.**, Wan, W., & Harada, K. (2021). *Robotic imitation of human assembly skills using hybrid trajectory and force learning.* In 2021 IEEE International Conference on Robotics and Automation (ICRA) (pp. 11278-11284). IEEE.

4. Arbulu, M., Mateus, P., Wagner, M., **Beltran, C.**, & Harada, K. (2018, December). *Industry 4.0, Intelligent Visual Assisted Picking Approach.*
   In International Conference on Mining Intelligence and Knowledge Exploration (pp. 205-214). Springer, Cham. Romania. DOI:10.1007/978-3-030-05918-7_18

Local Conference papers (without peer-review):

1. **Beltran, C.**, Petit, D., Ramirez, I., Matsubara T., & Harada, K. (2019). *Hybrid position-force control with reinforcement learning.* In 第20回 計測自動制御学会 システムインテグレーション部門講演会 SI2019.

2. **Beltran, C.**, Petit, D., Ramirez, I., & Harada, K. (2019). *Learning to Grasp with Primitive Shaped Objects.* In The 1st International Symposium on Symbiotic Intelligent Systems, Osaka, Japan.

3. **Beltran, C.**, Petit, D., Ramirez, I., Matsubara T., & Harada, K. (2019). *Reinforcement Learning Framework for Real-World Robotic Arm.* In The 37th Annual Conference of the Robotics Society of Japan RSJ, Tokyo, Japan.

4. **Beltran, C.**, Petit, D., Ramirez, I., Wan W., & Harada, K. (2018). *Learning Grasp with Guided Policy Search.* In The 36th Annual Conference of the Robotics Society of Japan RSJ, Nagoya, Japan.

5. Wang, Y., **Beltran-Hernandez, C. C.**, Wan, W., & Harada, K. (2020). Completing Robotic Assembly Skills with Force Control via Combined Learning. In 日本ロボット学会学術講演会予稿集.

Patents:

1. Kensuke Harada (原田研介), **Cristian Camilo Beltran-Hernandez (クリスティアンカミロ ベルトランエルナンデス)**, Kikuchi Shinichi(菊池慎市), Takayuki Nishi (西敬之) (2019).
   ロボットの制御装置、制御方法、及びプログラム. 出願番号（国際出願番号）：特願2020-109801 / 公開番号（公開出願番号）：特開2021-091079.

Awards:

- As part of team O2AC, 3rd place at Industrial Assembly Challenge at World Robot Summit 2020, and Japanese Society of Artificial Intelligence Award.
- Best Oral Presentation (優秀講演賞), 第20回計測自動制御学会システムインテグレーション部門講演会 (SI2019)
- Finalist of Best Paper Award, in the IEEE/SICE International Symposium on System Integration (SII), 2019.
- Finalist of International Session Best Paper Award, in The 36th Annual Conference of the Robotics Society of Japan (RSJ), 2018.