Imperial College London
Department of Computing

Robot Learning of Manipulation Skills with Minimal Human Intervention

Ya-Yen Tsai

Submitted in part fulfilment of the requirements
for the degree of Doctor of Philosophy in Computing
at Imperial College London, 2024

Copyright Declaration

Statement of Originality

I hereby declare that this thesis and the work herein detailed were composed and originated by myself, except where appropriately referenced and credited.

Abstract

Manipulation is vital in human daily activities. It makes interaction with an environment possible from simple tasks like grasping or writing to complex ones like assembly and surgery. Despite humans' versatility, there is continuous research in robotics to alleviate humans from monotonous or repetitive tasks. Presently, robotic manipulation is predominantly employed in well-structured and controlled industrial settings. As the interactions are not so complex, robots in these environments are typically programmed with task-specific code. However, applying robotic manipulation in everyday environments is considerably more intricate than in industrial settings. Relying solely on human programmers is impractical and hence, robots need to develop the ability to autonomously reason about the environment, learn new skills, and adapt to unfamiliar surroundings.

This work delves into a promising approach to robot learning that combines human priors with self-exploratory training. Human priors, such as demonstrations, offer valuable insights into tasks, effectively reducing the learning time for robots. Subsequently, self-exploratory training empowers robots to enhance their skills, enabling them to reason about environment states, adapt to new environments, surpass human demonstrations, and gain deeper insights into tasks through interactions with their surroundings. While providing significant human priors can expedite learning, it also runs the risk of introducing bias that restricts robots from surpassing human performance. Conversely, excessive self-exploratory training may prolong exploration time and impede the development of natural and human-like behaviours in robots. Therefore, this thesis aims to find a suitable balance between human involvement and robot learning. Specifically, we address critical questions concerning the selection and utilisation of human priors to facilitate effective learning. We also seek to determine the optimal types and quantities of human priors that should be provided, as well as how these priors should be integrated into the learning process to maximise the robot's capabilities.

Acknowledgements

First, I would like to express my sincerest gratitude to my supervisor, Prof. Edward Johns, for his continuous support and unconditional guidance. His advice has been invaluable in completing my PhD. He has helped me improve and develop my research skills and has always been very patient and enthusiastic about research. I enjoyed having constructive and motivational conversations with him. He always guided me to find and focus on the most critical problems. I appreciate him for always being so supportive and for his guidance throughout every project. His wise advice made me a better scientist.

I would like to thank Prof. Guang-Zhong Yang for welcoming me to the Hamlyn family and providing academic support in my early PhD life. He gave me the opportunity to develop my research at the centre and provided me with valuable experience of free exploration in my research field. I would also like to thank him for cultivating a fantastic working environment within the Hamlyn Centre and organising various events. This allowed me to collaborate with brilliant minds and meet people from different backgrounds. I am also very grateful to The Hamlyn Centre at Imperial College London for supporting me financially during my studies.

My appreciation extends to all of my friends and members of the Hamlyn Centre, who made my PhD journey so much more enjoyable, fun and fruitful. Thank you all for your physical and emotional support, especially during the hard time in Covid-19. They all played a special role in the successful completion of this thesis.

A special thanks to Dr. Bidan Huang for her invaluable guidance and support during my academic life at Imperial. She has played an important role in sparking and fostering my passion for robotics. Throughout my academic journey, she has generously shared her wealth of research insights and expertise. Her thoughtful advice and feedback have proven helpful for my personal and professional growth. I am also thankful for the opportunity to intern at Tencent under her mentorship. This experience has been truly enriching and memorable.

More importantly, I would like to thank my parents and brother for their endless love, encouragement and emotional support which have accompanied me through every challenge. I am deeply grateful to my family for their support throughout my studies. I also would like to acknowledge everyone else who contributed to making this achievement possible, including my friends, teachers, mentors and colleagues. Their guidance, assistance and belief in me have been instrumental to my success.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The classical way of programming robots requires a human programmer to reason in advance and design an appropriate robot controller. It has been widely adopted to program robots for manufacturing purposes. In a well-controlled and structured manufacturing environment, robots are deployed to work with the same objects and carry out repetitive manipulations. This reduces the number of scenarios robots need to handle and the complexity of programming. However, the disadvantage of such an approach becomes more apparent as environments become more complicated. In domains such as hospitals, service industries, and households, environments are often unstructured and changing. There are too many variations and uncertainties that robots need to handle. Programming robots to work in such environments with explicit programming may be extremely laborious. For unforeseen situations, a human-engineered controller may be impossible to implement in advance. Robots, therefore, must be equipped with more advanced intelligence to learn novel skills and adapt to changes in environments without too much reliance on humans.

Robot learning has recently been introduced to answer the question of *how a robot should learn to interact with unstructured environments*. Researchers use machine learning techniques to help robots reason about the states of environments based on

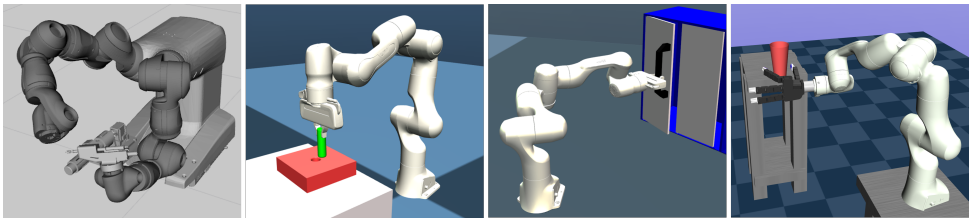


Figure 1.1: Examples of a robot performing different robotic manipulations. From left to right, these manipulations are sewing, peg-in-hole, door opening and grasping.

their perception and make decisions. These approaches are data-driven and focus on the problems of data collection and learning from data. Common robot learning approaches include Imitation Learning (IL)/Learning from Demonstration (LfD) and Reinforcement learning (RL). IL leverages human demonstrations to quickly learn a control policy without time-consuming exploration [80]. RL, on the other hand, learns through self-exploratory data collection and training without human-provided data. Prior works have shown much success using robot learning to achieve robotic manipulations like Rubik's cube [5] that are difficult to do with explicit programming.

Despite much success, robot learning still faces many technical challenges in robotic manipulation. As depicted in Fig. 1.1, in unstructured environments, robots can interact with objects of different physical properties like shapes, sizes, locations, dynamics, and constraints. Robots can also manipulate them in different ways. Working with high-dimensional state and action spaces requires a significant amount of training data to learn the complicated state-action mappings. In practice, collecting this training data is tedious and time-consuming. A limited amount of data can constrain robots from learning a generalised or robust policy. In addition, robots often lack sufficient sensory information to make full observations of their environment. For example, in a peg-in-hole task, a robot has to estimate the hole's location from partial observations made through vision, force, and proprioceptive feedback. This can be complicated because multiple states of an environment may exhibit similar partial observations. Furthermore, uncertainties are prevalent in an unstructured

environment and can come from factors like noise in sensory feedback and communication delays between hardware and human intervention. Uncertainties could result in robots performing sewing in the wrong locations, colliding with an environment, pulling a door in the wrong directions, or dropping an object during grasping. Corrective motions are needed in the event of uncertainties but are difficult to train in advance. Finally, there are also some challenges specific to current robot learning approaches. For IL, the performance of robots can be easily undermined by scarce and suboptimal demonstrations. Without additional experience, robots cannot respond well to variations in environments. For RL, learning often faces problems of inefficient exploration, lengthy training periods, hand-engineered reward functions, and a reality gap [80].

The thesis explores possible solutions to the problems of robot learning in robotic manipulation. We leveraged an idea that combines the learning strategies from IL and RL. We proposed to use human demonstrations as guidance for a robot to learn not just the skills used to perform a task but also to get clues about task constraints and the dynamics of an environment. These give insights to robots about where to explore for an optimal policy, what uncertainties may be encountered at test time, and what ought to be performed in a task. These are task-relevant priors already acquired by human demonstrators that can speed up the training process of robots and alleviate the demand for a significant amount of training data. Self-exploratory training is then adopted to improve the robustness of the learned policy and to encourage the exploitation of proprioceptive sensory feedback, such as force and tactile. By iteratively interacting with environments, robots can explore beyond what has been provided by the demonstrations and improve their adaptability to new tasks or uncertainties. In this thesis, we have shown that by combining IL and self-exploratory training, robots can learn to reason about the states of an environment, acquire novel skills, and adapt to new environments effectively without much assistance from humans.

1.1 Contributions

In summary, the contributions of the thesis are as follows:

- *Constrained-Space Optimisation and Reinforcement Learning for Complex Tasks.* LfD can transfer human manipulation skills to robots efficiently, but scarce and imperfect demonstrations can have a negative impact on learning performance. In this chapter, demonstrations are instead used to form a constrained space. Through interactions within this space, a RL agent learns the demonstrated manipulation skills. This approach prevents the policy from exploring invalid regions and implicitly shapes the policy to learn human-like manipulation skills. By carefully defining a reward function, an agent can learn to optimise a policy that aligns with human intentions. This approach has been verified with a robotic suturing task. The results have shown improved performance compared to solely learning from demonstrations.

- *One-Shot Imitation Learning for Contact-Rich Manipulation.* Many assembly tasks, such as screwing and insertion, are achieved by two actions: tool-object alignment and tool manipulation. In an unstructured environment, robot perception and reasoning are essential to aligning tool and object. However, providing a large number of human demonstrations to learn the reasoning is impractical. In this chapter, a novel method for learning contact-rich manipulation tasks from only a single human demonstration is proposed. This is inspired by the way in which humans leverage force sensing to perform the contact-rich part of a task, such as the "wiggling" motion in a key insertion task that aims to align the key with the hole. In tool-object alignment, the robot wiggles around the object and uses a predictor, trained in a self-supervised manner, to align the end-effector precisely with the object. In tool manipulation, the robot then replays the single demonstration. In experiments

conducted in both simulated and real-world settings, this method has proven effective in addressing contact-rich tasks like peg-in-hole and door-opening by learning from a single human demonstration.

- *DROID: Minimising the Reality Gap using Single-Shot Human Demonstration.* RL has demonstrated great success in the past several years. Simulation provides a good alternative to real-world data collection, but the reality gap can affect the sim-to-real transfer. This is caused by the discrepancy between the dynamics of the simulated and real environments. In prior works, Domain Randomisation has been proposed to address the reality gap for both robotic locomotion and manipulation tasks, but it does not provide a good way to identify the range of parameters. In this chapter, Domain Randomisation Optimisation IDentification (DROID) is proposed. It's a novel framework that exploits single-shot human demonstration to identify the simulator's distribution of dynamics parameters and applies it to training a policy for a task. The results demonstrate that the proposed framework can identify the dynamics of the real world. A policy learned in simulation can also be successfully transferred to reality.

- *Minimising the Reality Gap in Tactile Based Learning.*

  Tactile feedback plays an essential role in human manipulation. Recently, roboticists have been exploring how to equip robots with tactile sensing capabilities and tactile-based manipulation skills using a state-of-the-art learning approach. However, there are many difficulties. Developing and simulating tactile sensors are often treated as separate problems, which makes tactile simulation notoriously difficult. Currently, simulations are either highly accurate but computationally intensive or faster but less accurate. Neither is ideal for learning tactile-based policies from data. This chapter presents a tactile sensor designed with sim-to-real transfer in mind. It consists of an array of

taxels with independent sensing characteristics. A framework is introduced to model the sensor and close the sim-to-real gap. Each taxel is modelled as a mass-spring-damper system for fast, easy simulation. The parameters associated with each system are identified using DROID. Encoders are then proposed to further close the gap in the latent space of tactile signals between simulated and real sensors. Experiments have demonstrated how robots can learn tactile-based manipulation skills without human demonstrations and the effectiveness of the proposed framework.

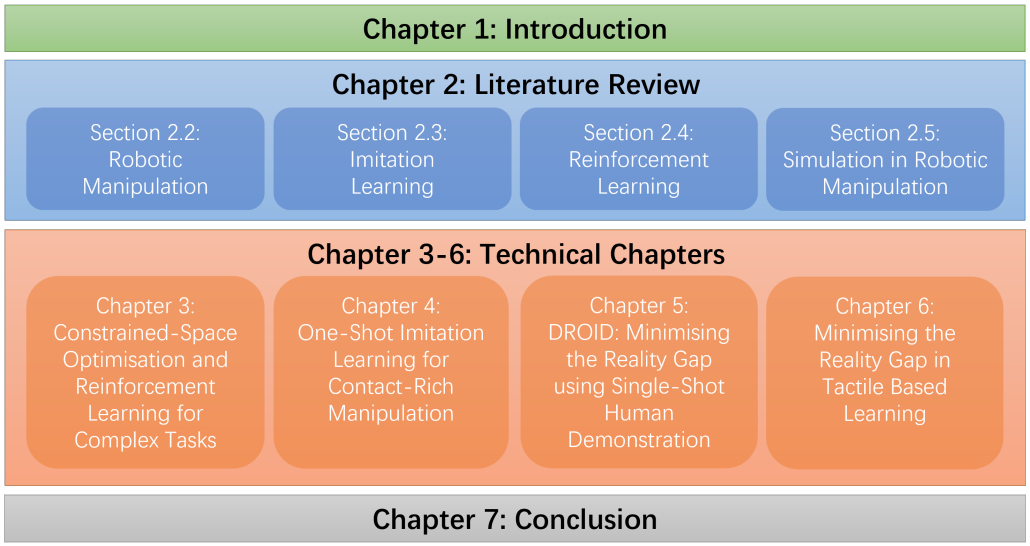


Figure 1.2: Overview of the thesis.

1.1.1 Outlines of the thesis

Fig. 1.2 shows the overview of the thesis. Each chapter is summarised as follows:

- Chapter 1 introduces the core ideas and motivation of the thesis.
- Chapter 2 reviews the relevant literature the proposed methods are based on or inspired from. The literature covers manipulation, robot learning approaches and simulation.
- Chapter 3 is the first technical chapter. It presents the work that addresses the problem of learning from scarce and suboptimal human demonstrations.
- Chapter 4 is the second technical chapter. It presents a novel approach for a robot to learn contact-rich manipulation skills using a single human demonstration.
- Chapter 5 is the third technical chapter. It presents the work that addresses the problem of the reality gap when transferring a policy from simulation to the real world.

- Chapter 6 is the last technical chapter. It presents a novel approach for robots to learn human-like manipulation with tactile feedback.

- Chapter 7 summarises the thesis and suggests potential directions for future research.

1.2 Publications

The following two papers have both been published in RA-L and presented in ICRA in 2020 and 2021. The first work was supervised by Prof. Guang-Zhong Yang before he left Imperial College London at the end of 2020. The original work has been adapted to form Chapter 3 of this thesis. The second work was done during an internship. It was co-supervised by the academic supervisor Prof. Edward Johns and the industrial supervisor Dr. Bidan Huang. The original work has been adapted to form Chapter 5 of this thesis.

- **Tsai, Y. Y.**, Xiao, B., Johns, E., and Yang, G. Z. (2020). Constrained-space optimization and reinforcement learning for complex tasks. IEEE Robotics and Automation Letters, 5(2), 683-690.
- **Tsai, Y. Y**., Xu, H., Ding, Z., Zhang, C., Johns, E., and Huang, B. (2021). Droid: Minimizing the reality gap using single-shot human demonstration. IEEE Robotics and Automation Letters, 6(2), 3168-3175.

Chapter 2

Literature Review

2.1 Introduction

Robotic manipulation has been widely adopted in the traditional production process [30]. It offers high endurance, speed, and precision compared to human manipulation and hence it is designed to handle repetitive pick and place tasks in structured industrial environments [59]. The state-of-the-art robotic manipulation still lacks the intelligence to handle the dynamics and uncertainties in daily unstructured environments. It often requires explicit programming by human programmers to provide the reasoning in perception and adequate control [88]. However, programming robots with lines of code can be time-consuming, and designing an appropriate robot controller to handle uncertainties can be extremely complicated.

To enable robots with intelligence, robot learning has been proposed as a way to provide robots with a degree of task autonomy and adaptability. One major area of research focuses on transferring human knowledge and learned skills to robots. This transfer of human intelligence to robots can involve providing clues such as control strategies, task constraints, environment dynamics, and task goals that reflect how

humans approach similar tasks [61, 104]. The specific types of information that need to be transferred depend largely on the particular tasks the robot will perform and the embodiment of the robot. For many robotic manipulation tasks where the robot's body resembles a human arm and hand, robots can learn human-like motions directly from visual observations of humans performing the same tasks. This approach allows robots to learn complex motor skills by watching humans, similar to how humans can learn by observing and imitating others [110]. In tasks where the robot's embodiment differs significantly from humans, alternative methods like teleoperation can be used. With teleoperation, a human operator can directly guide the robot's end-effector through the desired motions, and the robot can learn the precise movements of its own arm/hand from this process [23, 67].

In some situations, robots may not directly imitate human motor skills due to differences in embodiment. For example, in soccer penalty kicks, a human player relies on visual cues from the goalkeeper's stance and gaze to decide where to aim their kick, then coordinates their entire body to produce the desired speed and angle of contact with the ball. In this case, while direct imitation may not be feasible for a non-anthropomorphic robot, the robot can still learn the cognitive aspects of the skill from human demonstrations. Using approaches like apprenticeship learning or inverse reinforcement learning, the robot can infer the overall strategies and goals behind the human's actions. For the penalty kick, this involves understanding that the human aims for open areas of the goal based on reading the goalkeeper's positioning and attention. After learning the task objectives, the robot can then develop its own control policies, specialised for its unique embodiment. Rather than kicking with a leg and foot, the wheeled robot may adapt by pivoting its base and using an arm to strike the ball. By focusing on reproducing demonstrated task goals rather than motions, robots with distinct embodiments can learn skills their own way, benefiting from human guidance while customising behaviours for their capabilities.

This chapter provides a comprehensive review of the relevant literature on robot learning and robotic manipulation. It covers the background knowledge and context necessary for this research. The chapter is structured into four broad areas: robotic manipulation, imitation learning, reinforcement learning, and simulation for robotic manipulation. The first section introduces robotic manipulation and its challenges, especially in unstructured environments with uncertainties. It reviews approaches to enhancing the adaptability and robustness of robotic manipulation systems, such as soft robotics and adaptive control. The second section surveys IL, which enables robots to learn complex skills from demonstrations. It discusses how robots derive policies from demonstrations and the limitations of IL. The third section covers RL, including the preliminaries and challenges in applying RL for robotic manipulation, including working with high-dimensional data, long training times, and the sim-to-real gap. Some works that combine IL and RL have also been reviewed. The fourth section surveys the use of simulation for robotic manipulation. Sim-to-real techniques such as system identification and domain randomisation for transferring skills learned in simulation to the real world are also covered.

2.2 Robotic Manipulation

Humans can perform a wide range of manipulations, but are not suitable for tasks that involve tedious, repetitive and dangerous manipulations. Hence, robots have been introduced and developed to work with these tasks [30, 59]. However, robotic manipulation is challenging for many reasons. Firstly, robots often possess different mechanics and perceptions from humans. Programming robots to achieve optimal task performance with respect to their embodiment is therefore difficult. Secondly, the presence of uncertainties can lead to unexpected interactions, and corrective actions are not possible to be programmed in advance by humans. This may prevent robots from making contact with objects in the desired ways. For robotic manip-

ulation, these uncertainties may come from an inaccurate model, pose estimation, and control of robots [58, 77, 144]. The current application of robotic manipulation addresses these in two ways. One is to deploy robots in a structured and controlled environment [90]. In this environment, robots are programmed to follow fixed motions and carry out quasistatic manipulation with the same objects repeatedly. This is often referred to as automation and is commonly used in the manufacturing industry and for medical imaging purpose [4]. For an unstructured environment, on the other hand, robotic manipulation is mostly achieved through teleoperation [90]. Based on the feedback from sensors placed on the robotic hardware, human operators leverage the master device to perform manipulation and handle uncertainties remotely [94, 105]. Teleoperation is commonly used in surgical robotics, underwater robotics, and space vehicles to provide precise manipulation in a confined space or to access environments unfavourable to humans.

Many research works have attempted to address uncertainties in an unstructured environment for robotic manipulation without the use of teleoperation. Early works proposed to implement force controllers, such as impedance control or fuzzy control, on robotic manipulators to improve robots' adaptability [8, 28, 145]. These controllers allow robots to adjust control strategies based on different interactive forces made with an environment. Force controllers have been studied to solve tasks such as peg-in-hole [101] and grinding [152]. However, these methods often rely on the careful design of force controllers and are usually less generalisable to different tasks. Other works have defined a series of pre-programmed manipulation actions to constrain the motion of objects, thus reducing the uncertainties in manipulations [16]. But, different shapes of objects may require different constrained motions and designs, which may require expert knowledge. Instead of solving uncertainties through algorithms and controllers, prior work has also attempted to address the challenges with soft robotic grippers. These focus on using material softness and mechanical compliance to lower the complexity of control [124]. For example, grippers with

fin-ray structure can conform to an object surface during contact [123]. This structure allows grippers to handle objects like a ball, office supplies, and vegetables with simple actuation [32, 54]. Soft grippers based on fluid elastomer actuators (FEA), on the other hand, have expandable and asymmetric structures. When a fluid like liquid or gas is exerted, grippers inflate and bend, allowing high forces to be generated [2, 48, 124]. However, limitations of soft grippers often include a lack of long-term robustness, undesired deformation during grasping, and difficulty of miniaturisation [32, 124].

Recently, many robot learning approaches have been extensively studied to handle uncertainties in grasping, pick-and-place and in-hand manipulation. These approaches learn control strategies from demonstrations [3, 155], through self - exploration [50] or by combining conventional controllers with RL methods [70]. Control strategies that learn from demonstrations require humans to explicitly perform corrective actions to account for uncertainties. It is unrealistic to cover all the possible corrective actions with these learning strategies. Other approaches, such as RL require time-consuming trial and error and carefully designed reward functions to learn such strategies.

In this thesis, the focus is on enhancing the adaptability and autonomy of robotic arms and dexterous robotic hands through robot learning techniques. The tasks primarily involve kinematic control and quasi-static interactions between objects and robots. Quasi-static tasks are characterised by small velocities and accelerations, where dynamic effects like inertia and momentum are negligible. They can typically be modelled using kinematic equations without needing complex dynamic models. Kinematic control and quasi-static manipulation provide an ideal testbed for developing robot learning approaches before progressing to more advanced dynamic tasks. They reduce the complexity of sensing, modelling, and control, allowing the research to be focused on higher-level problems such as generalisation, adaptation,

and sim-to-real transfer.

2.3 Imitation Learning

IL techniques are introduced to provide a natural and intuitive way for robots to learn new skills without explicit programming. Robots are provided with human demonstrations, and they imitate human behaviours by learning a mapping between observed states and actions [63]. The advantages of IL compared to explicit programming are that the demonstrations can be efficiently provided to cover diverse scenarios in a task, and transferring the human prior knowledge to a robot can prevent it from learning the same behaviour from scratch, which in turn, speeds up the learning process [44, 71, 143]. This approach has demonstrated much success in range of applications in humanoid robots [12], robotic manipulation [43], navigation [19] and games [115].

The core of IL lies in collecting demonstrations from humans and policy derivation. The quality and quantity of demonstrations determine the insights into the task to be learned and the number of scenarios the robots can learn to handle. The policy derivation governs the control strategy and adaptability of the learners. In the rest of the section, we cover different demonstration collection methods, ways to learn the control strategy from the collected observation-action pairs, and some limitations of IL. In the following context, we refer to the agent who performs demonstrations as a demonstrator, expert, or teacher, and another agent who learns from the former as a learner or student, who is typically a robot.

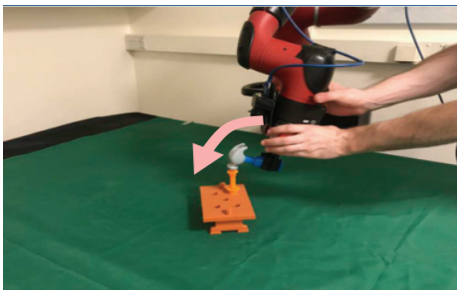
2.3.1 Collecting Human Demonstrations for Robot Learners

A human demonstration is collected by sensors that capture, for example, the motion, forces or muscle activities of a demonstrator performing a task. Sensors can be placed on a demonstrator, learner or located elsewhere. Collecting human demonstrations often concerns a correspondence problem [20, 100]. It occurs as a result of dissimilar physical configurations and perceptions between a demonstrator and learner, and so the demonstration performed may not be directly imitable by a learner [18]. The correspondence can be divided into two sequential mappings, a record mapping and an embodiment mapping [10]. The former concerns the perceptual equivalence, i.e. whether the exact motions and states performed are fully captured in the data, while the latter concerns the physical equivalence, i.e. whether the recorded data could be fully observed or executed by the learner. The data collection approaches can be divided into four methods, namely kinaesthetic, shadowing, sensors on the teacher and external observations, as illustrated in Fig. 2.1 and Fig. 2.2.

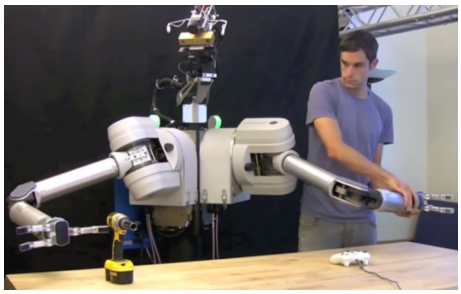
In the following, we discuss these in the context of recording mapping and embodiment mapping. The mappings are explained with the assumption that the states are fully observable, i.e. the sensors can fully capture what the demonstrator performs and the recorded data is fully observable by the learner. Hence, the demonstrations are recorded as state-action pairs. In reality, the data captured from the sensors may be subject to factors like incompleteness, uncertainties, noise, and latency. These result in states being only partially observable, and therefore, the demonstrations are recorded as observation-action pairs. Learning from these data requires an agent to additionally recover the underlying states from, for example, the history of the observations. How to recover such information is still an open question and is not within the scope of our discussions.

Embodiment Mapping \ Record Mapping	Identity	Non-Identity
Identity	Kinaesthetics	Shadowing
Non-Identity	Sensors On Teacher	External Observation

Figure 2.1: A recording mapping and embodiment mapping matrix [10].

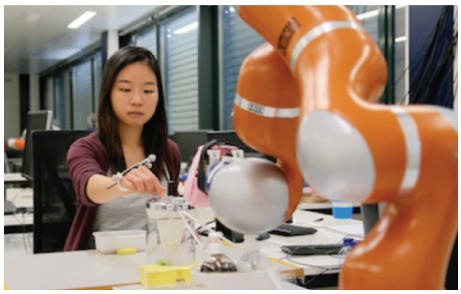


(a) Kinaesthetic approach [71]



(b) Shadowing approach (Source: https://faculty.cc.gatech.edu/%7Echernova/CS7633/slides/LfDOverview.pdf)

(c) Sensors on teacher (Source: https://faculty.cc.gatech.edu/%7Echernova/CS7633/slides/LfDOverview.pdf)



(d) External Observation [113]

Figure 2.2: Different types of LfD methods.

Kinaesthetic teaching is a common method for demonstrating desired motions to robots. In this approach, the teacher physically guides the robot through the desired motions while the robot's sensors record the demonstrations [33, 85]. As the states, s_d and actions, a_d of demonstrations are recorded on the robot, both the record mapping and embodiment mapping are direct, and therefore, these pairs are fed as commands to the robot states, s_r and actions a_r [57]. Another way to perform demonstrations on the robot is via teleoperation. The demonstrator uses a master device like a joystick or haptic device to control the motions of the slave device, while the entire motions are also captured on the slave [33, 85]. This again can achieve direct record and embodiment mappings. On the other hand, the demonstrations can also be recorded using the sensors placed on the demonstrator [24, 65]. This often requires the teacher to equip with wearable sensors. Since the recorded s_d and a_d differ to what the robots would observe or execute, an additional function is needed to map these pairs to s_r and a_r.

Record mapping can be indirect if the states s_d and a_d cannot be fully recorded. One example of such a demonstration approach is called shadowing. In this case, the demonstrations are recorded by observing and mimicking the demonstrator's behaviour while the motions are recorded using the equipped sensors. In the prior work [103], the authors proposed to first learn a mapping between the posture of a demonstrator and a humanoid robot. Then, a humanoid can learn new arm gestures by simply observing and mimicking the demonstrator's behaviour. Alternatively, the demonstrations can also be recorded using external sensors like cameras [36, 136] that are not placed on either the demonstrator nor the robot. However, this approach usually suffers from noise, occlusions, and rapid movement that can be introduced in the observations of human actions. Since the sensors are external to both the demonstrator and the robot, the record and embodiment mappings are indirect. The recorded data must be interpreted and then re-targeted to the embodiment of the robot [108].

2.3.2 Policy Derivation

Policy derivation from demonstrations often involves learning a mapping between states and actions [10, 76]. A state describes the current situation of an agent, and in this context, it is assumed to be fully observable. It can include information like pose, velocity or force acting on the agent [63] or the kinematics and dynamics of the environment. The mapping can be learned with a classifier or a regressor. A classifier is trained on a set of state-action pairs obtained from the demonstrated examples. The trained classifier outputs the most appropriate robot action class for a given state input. A robot action class can be a basic motion or a motion primitive. Regression is similar to classification in terms of state-action mapping, but the trained regressor maps states to continuous actions instead. Regression is often applied to low-level motion control mapping.

One popular approach for learning state-action mappings is to use Gaussian mixture models (GMMs) to encode motion primitives and then apply Gaussian Mixture Regression (GMR) to generate smooth trajectories. Calinon et al. employed this concept to encode both the demonstrations and task constraints in joint and task spaces [25]. By combining the encoded constraints with Jacobian-based optimisation in the null space, they developed skills that could be generalised to new situations. Dynamic Movement Primitives (DMPs), proposed by Ijspeert et al. [65], is another popular approach for learning the mappings. It uses differential equations and a nonlinear term to model each primitive as a nonlinear dynamical system. It is robust to variability in the demonstrations and allows adaptation in response to changes in the environment or task goals. Multiple equations describing motion primitives can then be combined to form a smooth trajectory [64,106]. For nonlinear and high-dimensional problems, others have attempted to use variations of local regression based on k-Nearest Neighbours, locally weighted regression [13], receptive field weighted regression [118] and locally weighted projection regression [140] to

learn the mappings.

2.3.3 Challenges

There are several key challenges with IL. First, finding the optimal model complexity to learn a policy is critical but difficult. This can also be referred to as a problem of bias/variance trade-off [21]. If the model is too simple, it will have high bias, meaning the policy cannot adequately capture the relationships between states and actions in the training data. An overly simple model will underfit the data, introducing errors even when the training data is not too noisy. On the other hand, if the model is too complex, it risks overfitting due to noise in the training data, resulting in high variance and poor generalisation. Finding the right level of model complexity involves balancing this bias-variance trade-off. Another challenge arises from learning from suboptimal demonstrations. Suboptimal demonstrations contain unsuccessful actions, large variations, noise, and redundancy. They can lead to learning mappings between states and unsuccessful actions or multiple applicable actions, increasing the chance of failure [10]. Suboptimal demonstrations may also lack key features, preventing the learner from extracting useful information. Reasons for suboptimal demonstrations may come from demonstrators being unfamiliar with the demonstrated task, data collection equipment being inaccurate and noisy, or problems associated with correspondence, as mentioned earlier. A limited amount of human demonstrations can also impact the learned policy, especially for algorithms that naively learn the underlying characteristics and distributions of the demonstrated data to form part of their learning policies [14]. Lastly, pure IL lacks flexibility in incorporating additional objectives and exploring more efficient and robust policies.

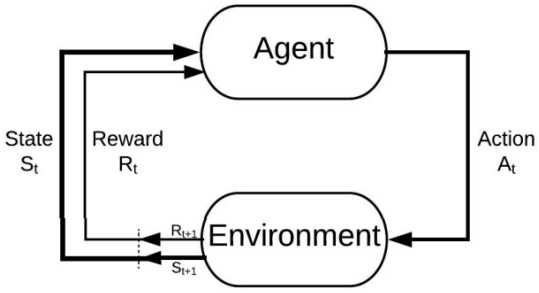


Figure 2.3: A standard RL framework showing how an agent interacts with an environment [129].

2.4 Reinforcement Learning

In RL, an agent learns a policy through iterative interactions with an environment and receives a reward from the environment, as illustrated in Fig. 2.3. It iteratively improves the policy and maximises the accumulative return given the defined reward. RL differs from supervised or unsupervised learning. It employs this exploration and exploitation mechanism to learn a policy for a novel task; hence, it does not require laborious data labelling or human-provided training data. Different states, actions, and associated rewards can be explored through iterative interaction with an environment. Hence, an optimal policy can be learned by solving the corresponding Markov Decision Process (MDP) [129]. The success in Atari games [91] has driven the research of RL even further to Deep Reinforcement Learning (DRL), and many other prior works have later demonstrated much success of RL in games with much higher complexity and dimensions [125–127]. Without the need for a large number of demonstrations, RL has shown its potential in finding an optimal policy in many applications [38, 46, 55, 125].

RL was originally studied for solving just a single task rather than multiple tasks. More recently, the concept of meta-RL has been proposed to allow RL to quickly adapt its learned policies to a new task using only a limited amount of experience. Instead of training a separate policy per task, it is trained on a family of tasks, al-

lowing it to implicitly encode the common task structure among different tasks [42].

Enlightened by the prior works, RL has been applied in numerous robot learning scenes. Works by OpenAI have demonstrated that the skills first learned through iterative interaction with the environment in the simulation can be deployed to the five-fingered robotic hand to handle dexterous object manipulation and solve a Rubik's cube [5, 9]. Other prior works exploited the feasibility of using DRL in controlling and manipulating a robotic arm. Gu et al. [50] trained the agent asynchronously across multiple robotic arms in an attempt to learn a door-opening task, while Zeng et al. [149] used DRL to train the robot to learn complex synergies between pushing and grasping.

2.4.1 Preliminaries of MDPs, Q-learning and Deep Q- learning

In the following section, we review the preliminaries of MDPs, Q-learning and Deep Q-learning.

Markov Decision Processes An MDP is a mathematical framework based on a state set $\mathcal{S}$ associated with a finite action set $\mathcal{A}$. If and only if state $\mathbf{s}$ in $\mathcal{S}$ captures all the relative information from history, state $\mathbf{s}$ is Markovian. The transition function between two Markov states can be defined as: $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. A Markov Reward Process (MRP) is defined as the tuple $< \mathcal{S}, f, r, \gamma >$, where r is a scalar reward function and γ is a discount factor. The scalar reward function is defined as $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. The parameter $\gamma \in [0, 1]$ determines how one values the importance of current reward and future rewards. Besides, γ also stabilises the total return in an infinite time-step case. An MRP with control input/action is defined as an MDP: $< \mathcal{S}, \mathcal{A}, f, r, \gamma >$. The objective of an RL agent is to find the optimal policy that

maximises the accumulated rewards (return) R. Considering the observed states $\mathbf{s} \in \mathcal{S}$ and admissible control input $\mathbf{a} \in \mathcal{A}$, the overall learning problem can be formed into MDPs. To evaluate the value function of the policy, the RL agent will try to solve the MDP. When the value function is obtained, the RL agent can make optimal decisions according to it, thus fulfilling the decision-making objective.

Q-learning In RL, the behaviour of the RL agent is determined by the policy defined as $\pi(\mathbf{a}|\mathbf{s}) = P(\mathbf{a}_\omega = \mathbf{a}|\mathbf{s}_\omega = \mathbf{s})$. The policy $\pi(\mathbf{a}|\mathbf{s})$ represents the probability distribution of the action picking $\mathbf{a}_\omega$ according to the observed system state $\mathbf{s}_\omega$. r_ω is the reward obtained after yielding the action at the ω-th time-step. The agent takes the observed state $\mathbf{s}_\omega$ as input and gives the action $\mathbf{a}_\omega$, and receives the reward r_ω at time-step ω. Associated with the reward function, the state-action value $Q(\mathbf{s}_\omega, \mathbf{a}_\omega)$ is defined as the expectation of return R_ω which starts with state $\mathbf{s}_\omega$ and action $\mathbf{a}_\omega$. The state-action value function under policy π can be calculated as follows:

$$\begin{aligned} Q^\pi(\mathbf{s}, \mathbf{a}) &= \mathbb{E}_\pi[R_\omega|\mathbf{s}_\omega = \mathbf{s}, \mathbf{a}_\omega = \mathbf{a}] \\ &= \mathbb{E}_\pi[r_\omega + \gamma r_{\omega+1} + \gamma^2 r_{\omega+2} + \dots |\mathbf{s}_\omega = \mathbf{s}, \mathbf{a}_\omega = \mathbf{a}] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{\omega+k}|\mathbf{s}_\omega = \mathbf{s}, \mathbf{a}_\omega = \mathbf{a}] \end{aligned} \tag{2.1}$$

and the state-action value function can be also rewritten as the Bellman Equation (BE):

$$Q^\pi(\mathbf{s}_\omega, \mathbf{a}_\omega) = \mathbb{E}_\pi[r_\omega + \gamma \mathbb{E}_\pi[Q^\pi(\mathbf{s}_{\omega+1}, \mathbf{a}_{\omega+1})]]. \tag{2.2}$$

The goal of Q-learning is to find the optimal decision-making policy π, which maximises the state-action value function. The optimal state-action value function can be defined as:

$$Q^*(\mathbf{s}, \mathbf{a}) = \max_\pi \mathbb{E}_\pi[r_\omega + \gamma r_{\omega+1} + \gamma^2 r_{\omega+2} + \dots |\mathbf{s}_\omega = \mathbf{s}, \mathbf{a}_\omega = \mathbf{a}]. \tag{2.3}$$

To maximise the return obtained by the RL agent, the Q-learning algorithm updates the state-action values according to the reward function in an off-policy style as:

$$Q(\mathbf{s}_\omega, \mathbf{a}_\omega) \leftarrow (1-\alpha)Q(\mathbf{s}_\omega, \mathbf{a}_\omega) + \alpha(r_\omega + \gamma \max_{\mathbf{a}_k}\{Q(\mathbf{s}_{\omega+1}, \mathbf{a}_k)\}), \tag{2.4}$$

where $\alpha \in (0, 1]$ is the learning rate, which determines the learning speed of Q-learning.

To ensure that Q-learning explores extensively without diverging, the adaptive ϵ-greedy policy is adopted in this thesis. To change the value of the exploration rate ϵ in an adaptive way, we start with a large ϵ and then reduce it gradually after episodes are completed.

$$\mathbf{a}_\omega = \begin{cases} \arg\max_{\mathbf{a}_k}\{Q(\mathbf{s}_\omega, \mathbf{a}_k)\}, \text{ with probability: } 1-\epsilon, \\ \text{random } \mathbf{a}_k, \text{ with probability: } \epsilon. \end{cases} \tag{2.5}$$

To summarise, the Q-learning algorithm can be viewed in Algorithm 1.

Algorithm 1 Algorithm for Q-learning

```
Set the state-action values randomly
for Every episode do
    Initialise state s_0
    for Current timestep ω do
        Pick action a_ω through ε-greedy policy and observe the next state s_{ω+1}
        Update Q(s_ω, a_ω) ← (1 − α)Q(s_ω, a_ω) + α(r_ω + γ max_{a_k}{Q(s_{ω+1}, a_k)})
        s_ω ← s_{ω+1}
    end for
end for
```

Deep Q-learning When the state set $\mathcal{S}$ and action set $\mathcal{A}$ are too large or continuous, calculation of the exact state-action value function $Q(\mathbf{s}, \mathbf{a})$ becomes difficult and may be impossible in many cases. One alternative is to use a function approxi-

mator to replace the exact state-action value function. As a universal approximator, neural networks with nonlinear activation functions can be used for the Q-network to approximate $Q(\mathbf{s}, \mathbf{a})$ [56].

In the above algorithm, the observed state $\mathbf{s}$ is the input of the deep neural network while the output $\hat{Q}(\mathbf{s}, \mathbf{a}_k)$ approximates the exact state-action value $Q(\mathbf{s}, \mathbf{a}_k)$, $k = 1$, $2, \ldots, m$. By choosing the action according to the largest state-action value $\hat{Q}(\mathbf{s}, \mathbf{a}_k)$, the RL agent can make the optimal decision.

To simplify the notations, the set of all parameters (including all the weights and biases) in the deep Q-network will be written as θ^D in the rest of this chapter. The deep Q-network implicitly determines the policy, in which the action $\mathbf{a}_k$ is picked according to the observation $\mathbf{s}_\omega$ at timestep ω. The chosen action drives the current state $\mathbf{s}_\omega$ to the next state $\mathbf{s}_{\omega+1}$, and the reward r_ω will be received from the environment based on the reward function. The target network then takes the state $\mathbf{s}_{\omega+1}$ and the control input set $\mathcal{A}$ as the input to calculate the output, $\max_{\mathbf{a}_k}\{\hat{Q}(\mathbf{s}_{\omega+1}, \mathbf{a}_\omega|\theta^D)\}$. When $\max_{\mathbf{a}_k}\{\hat{Q}(\mathbf{s}_{\omega+1}, \mathbf{a}_\omega|\theta^D)\}$ and r_ω are available, the state-action value $\hat{Q}(\mathbf{s}_\omega, \mathbf{a}_\omega|\theta^D)$ of the current state-action pair can be updated according to the target network and obtained reward. Considering the importance of experience replay during the training of the RL agent [92], the state transitions associated with rewards are stored in the experience buffer as the tuple $< \mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i >$. $\mathbf{s}_i$ is the current state, $\mathbf{a}_i$ is the chosen action according to the current state, $\mathbf{s}'_i$ is the next state and r_i is the reward received. The algorithm for updating the weights of the deep Q-network is presented in detail in Algorithm 2.

Algorithm 2 Update of the parameters in the deep Q-network

1: Initialise the parameters in the deep Q-network θ^D:

$$\theta^D \leftarrow \theta^{D_0}$$

2: Create the experience buffer (EB)
3: Constants in the training: Num_Episode, Num_Time_Step, Num_Replay_Times
4: **for** `Episode = 1:Num_Episode` **do**
5: Choose the initial state $\mathbf{s}$ randomly
6: **for** `j_time = 1:Num_Time_Step` **do**
7: Generate action $\mathbf{a}_\omega$ through policy π determined by the current deep Q-network
8: Transit to the next state $\mathbf{s}_{\omega+1}$ and observe the reward $r_\omega = r(\mathbf{s}_\omega, \mathbf{a}_\omega, \mathbf{s}_{\omega+1})$
9: Store the tuple $< \mathbf{s}_\omega, \mathbf{a}_\omega, \mathbf{s}_{\omega+1}, r_\omega >$ in the EB
10: $\mathbf{s}_\omega \leftarrow \mathbf{s}_{\omega+1}$
11: Randomly sample a mini-batch of N tuples $< \mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i >$, $i = 1, \ldots, N$ from EB
12: Obtain the Q-target as:

$$\bar{Q}_i = r_i + \gamma \max_{\mathbf{a}_k} \{Q(\mathbf{s}'_i, \mathbf{a}_k | \theta^D)\}$$

13: Calculate the cost function:

$$J(\theta^D) = \frac{1}{2N} \sum_{i=1}^{N} (\bar{Q}_i - Q(\mathbf{s}_i, \mathbf{a}_i) | \theta^D)^2$$

14: Calculate the mini-batch gradient of θ^D:

$$\nabla_{\theta^D} J(\theta^D) = -\frac{1}{N} \sum_{i=1}^{N} (\bar{Q}_i - Q(\mathbf{x}_i, \mathbf{a}_i) | \theta^D) \frac{\partial Q(\mathbf{s}_i, \mathbf{a}_i | \theta^D)}{\partial \theta^D}$$

15: Update the weights of the deep Q-network through gradient descent:

$$\theta^D \leftarrow \theta^D - \alpha^D \nabla_{\theta^D} J(\theta^D),$$

16: **end for**
17: **end for**

2.4.2 Combining Imitation Learning and Reinforcement Learning

Many prior works focused on learning an optimal policy using RL from scratch or without previous knowledge of the task it was learning. Other works focused on giving hints about the task to be learned through human demonstrations. This may help to design a better reward function, train an agent faster, and learn a better control policy than solely learning from human demonstrations. For example, Guenter et al. aimed to solve the imitation learning problem, where complete solutions may not be provided through demonstrations alone [51]. The authors proposed a method to build a probabilistic encoding from a small number of demonstrations. This encoding is then used to modulate reinforcement learning, allowing the robot to generate new solutions to novel obstacle avoidance and grasping tasks that it has not seen before. Vecerik et al. [139], on the other hand, proposed a Deep Deterministic Policy Gradient from Demonstrations (DDPGfD). The framework incorporates the experience of demonstrations into RL's replay buffer and prioritises it such that the more important transitions are sampled more frequently. This allows the agent to learn behaviours similar to those of the demonstrators while requiring less demonstration data. Similar approaches have also been proposed by Nair et al. [97], but the authors aimed to overcome the inefficient exploration strategies of RL. This work then additionally introduced an auxiliary loss and a Q-filter to improve the learning performance. Gao et al. [46], on the other hand, use RL to address learning from imperfect demonstrations and ensure robust behaviours can be learned. Unlike other works, the authors proposed a Normalised Actor-Critic (NAC) that initialises a policy first from human demonstrations and then refines it by iteratively interacting with environments based on the manually defined reward. Rajeswaran et al. [111] demonstrated that with the help of a small number of human demonstrations, RL could work better for tasks that involve high-dimensional state

and action spaces. This approach has been shown to reduce the sample complexity and training time of DRL for multi-fingered manipulation tasks, and a policy learned can perform robust and natural movements.

In IL, a robot can be constrained to tasks that are only demonstrated by a teacher. This makes it incapable of handling new situations that have not been demonstrated. Guenter et al. [51] incorporate RL into their system to permit some alternations to its original trajectory when it encounters unseen states. In the presence of perturbation, RL plays a role in exploration to find alternatives to accomplish a given task. When facing such a situation, GMM-modelled trajectories will be re-centred to form a smooth solution that avoids such obstacles.

Instead of using the demonstrations to initialise the policy, Kim et al. [78] proposed Approximate Policy Iteration with Demonstration (APID). It uses the information from a few and/or suboptimal demonstrations as imposed constraints to an optimisation problem, making it less prone to noisy demonstrations. Kang et al. [73] proposed a Policy Optimisation from Demonstration (POfD), which utilised the scarce and imperfect demonstrations to limit the exploration region to be around that of the expert policy. The author has shown that a better policy could be achieved by including the demonstration-guided exploration term in its learning objective.

Inverse Reinforcement Learning (IRL) has been proposed to derive the preference of an agent in the form of a reward function based on its observed behaviour [150]. This avoids the need for manual reward engineering when a demonstration of a desired behaviour can be easily obtained. Abbeel et al. [1] for example, leveraged this idea to recover an unknown reward function from demonstrations and mimic different driving styles in a car driving example. Chung et al. [31], on the other hand, applied IRL to predict pedestrian intention and enable a service robot to mimic pedestrian-like motions. However, the challenges of IRL involve the derivation of multiple possible reward functions from the same observations, the generalisation of

the function to the unseen demonstrations, and the growth in the complexity of the reward function with an increase in the problem size [11, 102, 151].

2.5 Simulation in Robotic Manipulation

Data-driven approaches like RL have recently gained much success in robot learning [9, 82, 96]. However, the application of RL to robotics suffers from expensive training data collection in real robots [84]. Simulation, on the other hand, has gained huge popularity for RL training [9, 22, 47, 134]. It offers a cheap and efficient data collection alternative and overcomes challenges such as self-inflicted damage that may occur during exploratory data collection in reality. In robotic manipulation, prior works have proposed to simulate perceptions like force and vision, but not much has been explored for simulating tactile sensors for manipulation tasks. Various tactile sensors have been developed, for example, the BioTac sensor [45], which relies on the underlying incompressible conductive fluid to impose the changes in impedance of electrodes during contacts; resistive sensors [35, 116] and capacitive sensors [81, 120], which convert the physical deformation into electrical signals; and the vision-based sensors such as GelSight [148], TacTip [141] and GelSlim [37], which utilise a camera to capture the deformation of a membrane in the form of an image. Despite different sensing mechanisms in reality, most tactile sensor simulations focus on simulating the sensor deformation or force distribution at the contact surface. Two of the most common simulation approaches are FEM-based and image-based.

Finite element methods (FEM) model a tactile sensor by decomposing it into smaller elements [98, 99, 114, 122]. While the model is more realistic, it is computationally expensive and hard to provide large-scale training data for learning. For vision-based tactile sensors, the deformation is captured as an image, and the simulation requires correctly rendering the changes in the image [49]. These approaches focus

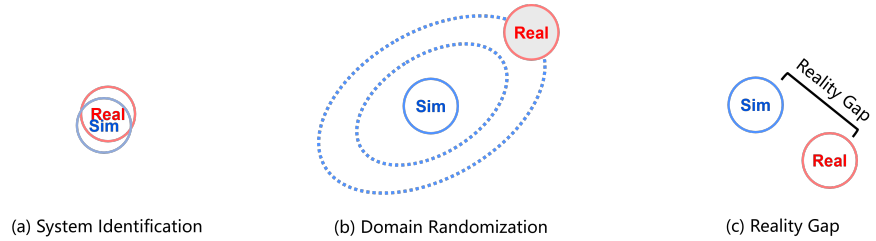


Figure 2.4: Common techniques for crossing the reality gap involve (a) System identification and (b) Domain randomisation. A red circle represents the data distribution of a task in the real world, and a blue circle represents the data distribution of a task in simulation. Here, the distance between the two circles (c) represents the reality gap.

on recovering static tactile images for tasks such as object 3D reconstruction. Other works model a tactile sensor using a surface mesh [69, 93], where contact force propagation is considered only at the model surface, [74] use a point spread function, and Ding et al. [34] define a set of pushing and pulling forces around the point of contact to simulate the deformation at the model surface.

2.5.1 Sim-to-Real

The RL approach is known for data inefficiency. Working with RL in simulation for policy training introduces a derived issue such as the reality gap [135]. The policy learned in simulation can fail in the real world due to a mismatch in the kinematics and dynamics models between the two environments. It hinders the policy learned in simulation from transferring well to the real world. Two common strategies have been studied in prior works to bridge the reality gap: System Identification (SI), and Domain Randomisation (DR) [138].

SI has been a popular approach for sim-to-real transfer [7, 68, 75, 86, 119, 147]. It focuses on finding the exact model of the real world so the physical behaviours match between the real and simulated systems. This could be done by constructing the simulation through direct measurement of the environment parameters in the real world [130] or by collecting real-world data for optimising the simulated model parameters [40, 131, 146]. However, correctly identifying the system's parameters is challenging. Many parameters cannot be explicitly measured, especially dynamics-related ones like friction, stiffness, and damping. In addition, the system parameters could be high-dimensional and coupled. This further increases the difficulties in achieving accurate and precise SI results [41, 138, 154].

Rather than identifying the environment parameters, DR creates multiple simulated environments by randomising the system parameters within given ranges during policy training. This improves the policy's generalisability and robustness against the reality gap. Recently, it has achieved significant progress for sim-to-real transfer in robotics [5, 6, 9, 27, 66, 107, 112, 131, 137, 138]. DR can achieve better sim-to-real performance than SI [131]. DR covers a greater range of parameters in the simulation containing the real values during RL training. Prior works optimised system parameters with simulated and real trajectories collected using hand-designed policies [9]. Other works determine parameters by automatically adjusting the bound-

aries of uniform randomisation distributions according to model performances [5]. While effective, these works suffer from a common drawback. Existing work [138] has demonstrated its demanding engineering efforts in adjusting the randomisation ranges, which is difficult and not intuitive. Hand-tuning the parameter ranges could easily cause overestimated values and lead to training RL in an invalid environment and learning a suboptimal policy. How to quickly choose the randomisation ranges for different parameters and achieve effective policy generalisation still remains a challenge.

Besides the two main approaches, a variant of DR, Adaptive DR, has also been studied. It was proposed to optimise the parameter distributions and minimise the chance of training RL in invalid environments. Other works use techniques such as approximate Bayesian computation [15], Bayesian Optimisation, or a relative entropy policy search [109] to estimate or optimise distributions of system parameters [27, 95, 112].

Chapter 3

Constrained-Space Optimisation and Reinforcement Learning for Complex Tasks

Robots can effectively acquire new skills from human demonstrations. However, learning from limited and imperfect demonstrations can negatively impact learning performance and is often overlooked. This chapter examines an approach that learns improved sewing motions for stent-graft manufacturing from suboptimal demonstrations and optimises by RL. In contrast to the traditional learning approach, the suboptimal demonstrations provide only loose guidance in the robot's kinematic movements, while the RL agent can learn to improve its performance based on a carefully designed reward function that aligns with the intentions of the demonstrators.

The original work where this chapter has been adapted from is published in:

Tsai, Y. Y., Xiao, B., Johns, E., and Yang, G. Z. (2020). Constrained-space Optimisation and Reinforcement Learning for Complex Tasks, *in* 'IEEE Robotics

and Automation Letters'.

3.1 Introduction

LfD is an appealing approach compared to traditional robot programming. It provides an intuitive way to transfer human skills to robots and a quick way to acquire novel skills. However, it comes with some limitations. One of the main challenges is the dependence of the robot's learning performance on the quality of the provided demonstrations. A good demonstration is essential for a robot to learn a task optimally, but obtaining it can be difficult. Demonstrations may often contain undesirable characteristics such as jerky motions, lengthy completion times, and unwanted movements, which can also be learned by the robot. Another limitation lies in the difference in embodiment between the demonstrator and the robot. This comes from the differences in kinematics, dynamics, sensors, and constraints and may result in learning infeasible or inefficient behaviours when direct cloning is applied. Lastly, LfD approaches typically have less flexibility to incorporate extra constraints or subgoals. This restricts the demonstrator's ability to further shape the desired behaviour of the robot.

In this chapter, we explore the automation of sewing using a robot in stent-graft manufacturing. This sewing task is a delicate and intricate task that poses challenges in obtaining optimal demonstrations. The robot considered is a 7 DoF robot that shares different kinematics, sensing capabilities and hardware constraints with a human demonstrator. In this chapter, we develop a solution that leverages suboptimal demonstrations and RL to learn an improved automation of this sewing task. Note that while learning sewing skills may also be done with classical control approaches, this work seeks alternative solutions to avoid time-consuming hand-crafted controllers.

The reasoning behind learning from suboptimal demonstrations is that they are more accessible, especially for tasks that involve delicate and fine manipulations.

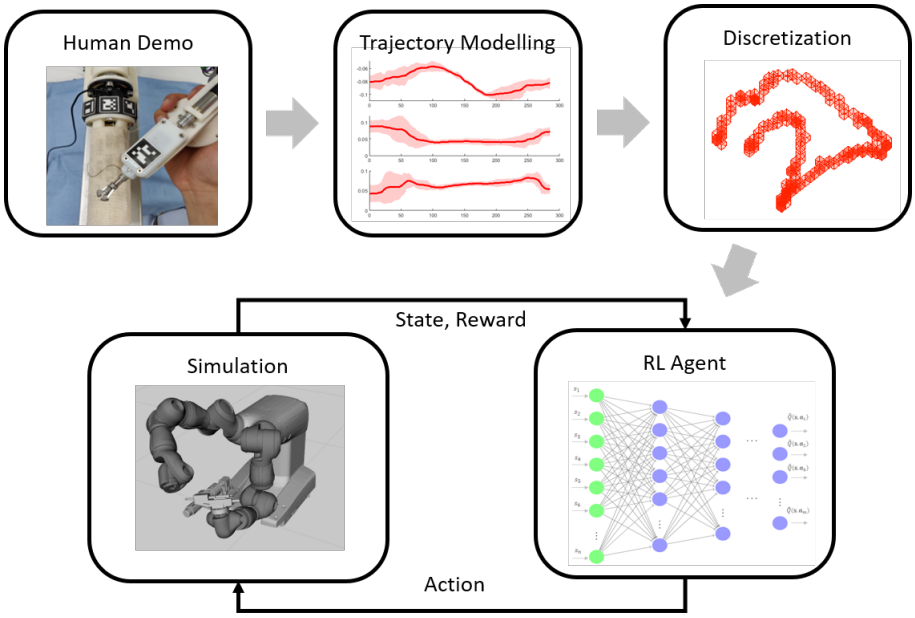


Figure 3.1: An overview of the proposed framework illustrating how trajectory modelling is discretised and learned through an RL agent.

These demonstrations can provide useful insights about how to complete a task and what the task constraints are, even if they do not offer an optimal solution. Rather than directly learning from demonstrations, the mean and variance of the demonstrated trajectories can be used to create bounded and discretised spaces in Cartesian space. This can help to both guide the robot and limit it from exploring unsafe and invalid regions during later learning. On the other hand, by leveraging RL, this approach can further incorporate additional objectives that align with the demonstrator's intentions during learning. This increases flexibility and results in improved performance in the sewing task compared to simply behaviour cloning.

While prior work, such as [25], has considered similar ideas, there are two significant differences. First, this work focuses on learning an improved skill from suboptimal demonstrations, which are demonstrations that could complete a task but may incorporate undesired behaviours for the robot. Second, the learning is optimised using RL and it allows additional objectives to be imposed, which has not been addressed in prior works.

3.2 Methodology

The presented framework does not directly learn a policy from demonstrations. Instead, the framework proposes to model the suboptimal demonstrations as a task constraint in a discretised Cartesian space. The space is constructed from only a limited set of demonstrations but captures different characteristics and variations of human motions beyond those being demonstrated. To then learn the actual policy of a sewing task, the DRL is adopted. The RL approach offers the advantage of finding an optimal solution given the reward function. This allows a demonstrator to include extra objectives that align with the intentions of the demonstrators during policy learning. The RL agent is trained by performing exploration within the bounded space constructed earlier. This enhances the overall learning efficiency by preventing the agent from exploring invalid regions. Explicitly constraining the exploration space can also greatly simplify the design of the reward function. The learning is performed in a physical simulation, where the RL agent finds an optimal policy that maximises the cumulative reward. Note that in this framework, we formulate the learning of the sewing skill from suboptimal demonstrations with discrete observation and action spaces. This aims to simplify the learning process and focus only on the main concept of learning a policy from a constraint space. In the rest of this section, we will discuss in detail the proposed framework, which includes human demonstrations, trajectory modelling and trajectory optimisation.

3.2.1 Human Demonstration and Problem Formulation

For this work, the setup of human demonstration emulates a robot task for stent graft manufacturing [62, 133], which consists of three components: a stereo camera, a suturing device [60] and a mandrel device as illustrated in Fig. 3.2. The stereo camera is to capture the motions of the suturing device and the mandrel device

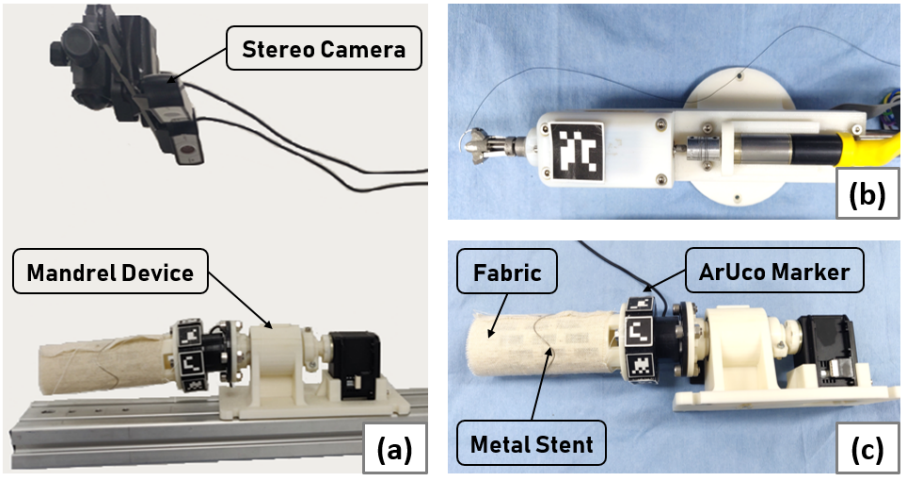
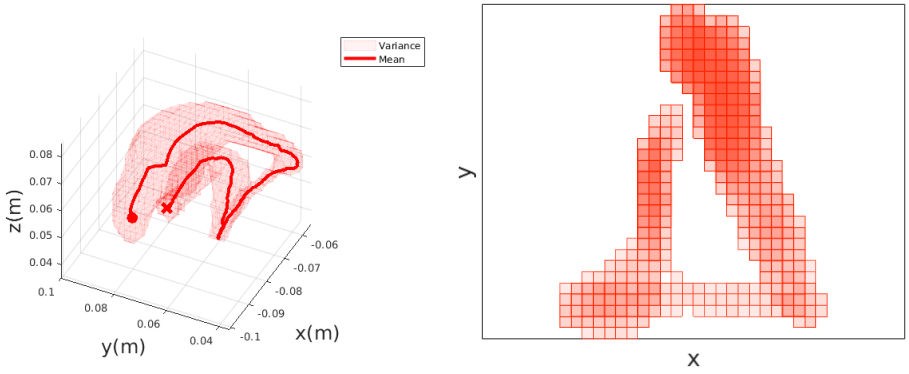


Figure 3.2: (a) The setup of the human demonstration, which uses the stereo camera to track the markers attached to a device (b) The handheld suturing device and (c) The mandrel device.

during the demonstration. The suturing device is a handheld device that facilitates single-handed stitching by passing its needle between two ends at the tip. The mandrel device consists of a cylindrical supporting structure covered by a piece of fabric and a metal stent, with slot windows to allow for needle piercing.

Demonstrations of stitching motions are provided to the robot for learning purposes. A demonstration involves manipulating the hand-held suturing device and performing a single stitch on a location of the mandrel that binds the stent and the fabric securely. The stitching process consists of three motion primitives: approach the stitch slot, pierce in and pass the needle, and return to the initial pose.

Here, we consider learning from imperfect demonstrations. We collect demonstrations from non-experts, who have only practised for 10 minutes. Hence, the demonstrations may, for example, involve undesired behaviours like motion redundancy and jerkiness. During a demonstration, the motions of the suturing and mandrel devices are tracked using ArUco markers and recorded at 20 Hz using the stereo camera. The tracking error of a marker is less than 1mm. A demonstration is recorded as a time-indexed trajectory, $\mathcal{S} = \{\gamma_m, \gamma_s\}$, which includes a time series of 6 DoF mandrel poses and suturing devices poses, both under the camera frame.



(a) Continuous 3D trajectory (b) Discretised 3D trajectory

Figure 3.3: Comparison of (a) continuous and the (b) top view of the discretised task space computed from the mean and the variance of the demonstrated trajectories. The solid circle and the cross represent the starting and the end points, respectively.

Each pose consists of the translation component, $T = [x, y, z]$ and the rotation component, $R = [\theta x, \theta y, \theta z]$, in Euler-Rodrigues representation. Fig. 3.4 shows the 6 DoF of all the temporally aligned trajectories.

3.2.2 Trajectory Modelling

Trajectory modelling involves calculating the mean and variance of the suturing device's pose relative to the mandrel at each timestep using all the suboptimal demonstrations. This information is used to compute the task constraint, which is represented as a bound space in Cartesian space. The bound space determines the states and action candidates in the subsequent RL optimisation.

Multiple trajectories are pre-processed before modelling. The first part of the pre-processing filters the noises and sudden hand movements by smoothing the trajectories using a moving average approach. Then, the trajectories of the suturing device are transformed to the mandrel frame to eliminate the relative movements between the devices and the camera across different demonstrations. Therefore, each collected trajectory is converted into a 6 DoF temporal trajectory $\gamma_{ms} \in \mathbb{R}^6$ consists of the translation component, $T_{ms} = [x_{ms}, y_{ms}, z_{ms}]$ and the rotation component,

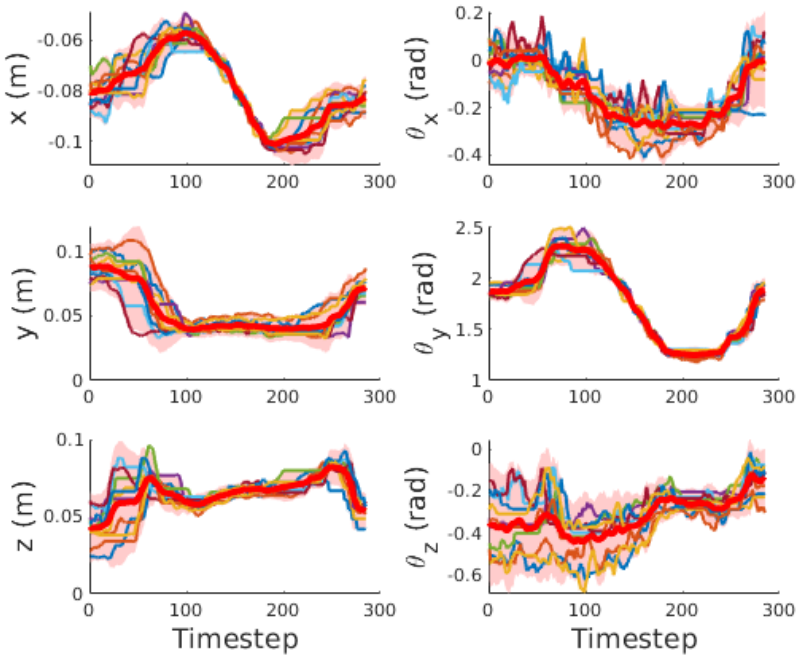


Figure 3.4: 10 sets of demonstrated trajectories were temporally aligned using DTW. The 3 translation components of the aligned trajectories are shown on the left, and the 3 orientation components of the aligned trajectories are shown on the right. The red line on each plot shows the means of the trajectories, while the shaded region represents the variances of the trajectories at each timestep.

$R_{ms} = [\theta x_{ms}, \theta y_{ms}, \theta z_{ms}]$, in Euler-Rodrigues representation.

To determine the mean and variance of poses at each time step, they are first aligned temporally via the Dynamic Time Warping (DTW) approach, similar to [26]. DTW works by aligning the time series trajectories in a way that minimises the distance between the points in Cartesian space in the trajectories using an Euclidean distance as a similarity measure. The alignment minimises the total distance between all pairs of points on two trajectories. The resulting alignment is a warping path that maps the points from one trajectory to another. The process is repeated across all pairs of demonstrated trajectories. After temporal alignment, all trajectories are aligned to the same temporal length, which allows the mean and variance at each time step to be computed. Finally, a trajectory composed of all the means and variances is denoted as $\bar{\gamma} = \{\bar{\mu}, \bar{\sigma}\}$. This mean trajectory along with the variance is illustrated as the red line and shaded region in Fig. 3.4.

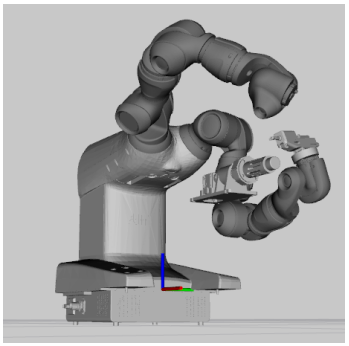


Figure 3.5: The simulation is used to simulate the physics and visualise the robot's movement.

3.2.3 Trajectory Optimisation

In practice, learning from human-demonstrated trajectories can be challenging due to various factors such as suboptimal demonstrations or the use of external cameras, which can result in unwanted movements. To address this, trajectory optimisation is employed. We construct a bounded exploration space using demonstrations and apply RL along with a carefully designed reward function in simulation to discover an efficient and safe trajectory within the defined space.

The left arm of ABB IRB14000 YuMi robot is used for learning the single-handed sewing task from suboptimal demonstrations as shown in Fig. 3.5. The learning is conducted in Gazebo simulation and visualised in Rviz. As the focus of learning is only on kinematics, a simulation can be constructed so that it aligns well with the real-world setting given models and kinematics of the suturing device, mandrel and robots. The simulation is adopted to simulate the interactions and collisions between objects and compute the inverse kinematics of the robot.

To construct a constraint space for subsequent learning, a Cartesian space is voxelised into cubes, each of which has a length of 0.002m. Using the mean and variances from $\bar{\gamma}$, we identift regions enclosed by the demonstrated trajectories as free spaces, while areas outside of which are identified as obstacles. This process is iterated through all timesteps, resulting in a 3D constraint space in Cartesian space as

shown in Fig. 3.3(b). It is worth noting that motion primitives with larger variances indicate multiple potential trajectories, allowing the policy to explore a broader space and identify optimal paths. Conversely, primitives with smaller variances imply movements requiring higher precision, and hence the policy should deviate less from the mean trajectory.

The RL agent is represented by a deep neural network that takes a state as the input and outputs a discretised action. A state is defined as the location of a suture device in Cartesian space relative to the mandrel frame. At each timestep, t, each action corresponds to a point in space around $\bar{\mu}_{t+1}$. There are a total of 126 action candidates. 125 different actions are uniformly distributed within $\bar{\mu}_{t+1} \pm \bar{\sigma}_{t+1}$. An additional action corresponds to the location of the next computed mean, i.e. $\bar{\mu}_{t+1}$. For each timestep, the RL agent either takes a random action or the optimal action according to the current policy. As the action candidates always come from points bound by $\bar{\mu}_{t+1} \pm \bar{\sigma}_{t+1}$, this implicitly drives the agent towards the end of $\bar{\gamma}$. The rotation component at time t is obtained from the rotation component of $\bar{\mu}_t$. The location and the orientation components form the pose at each timestep, and the corresponding joint positions of the robot are computed via inverse kinematics. At each timestep, the sub-trajectory connecting actions at $t-1$ and t are checked with the voxelised Cartesian space constructed earlier for collision. The computed joint positions are also checked for collision within the simulation. If a collision is detected, the current iteration is restarted. An episode begins at $\bar{\mu}_0$ and concludes at $\bar{\mu}_N$, with N is representing the final timestep in $\bar{\gamma}$. The RL agent undergoes training using a mini-batch sampled from the experience buffer at each step.

While the goal of this work is to work towards autonomous sewing in stent graft manufacturing, it shares many similar objectives with many surgical subtasks such as suturing, needle passing, and knot tying. Previous studies have defined various metrics to assess the performance of these subtasks. For example, Liang et al.

proposed evaluating surgical subtasks based on kinematic analysis of a robot's movements including metrics such as movement time, speed and motion jerkiness [87]. Similar metrics have also been proposed in [128] to evaluate robot-assisted surgery skills. To evaluate the performance of experts and novices in robotic surgical tasks, metrics like time to completion and total distance travelled by a robot's end-effector have also been considered in [72]. In this work, we assumed the demonstrators of this task share common objectives with the surgical subtasks, therefore, some of these metrics are considered in the reward function of RL as follows:

$$r(\mathbf{s}, \mathbf{s}') = -\frac{\Delta\Phi}{7} - \frac{|\mathbf{s}' - \mathbf{s}|}{|(\mathbf{s}'_{\text{mean}} - \mathbf{s}_{\text{mean}})|} - \angle(\mathbf{s}, \mathbf{s}') + 10 \tag{3.1}$$

The reward function consists of 4 terms. The first term is the average of $\Delta\Phi$, which is the sum of the absolute values of differences across seven robot joint angles between the current state, s, and the next state, s'. The second term compares the pose length between s and s', to that of the mean trajectory, $|s'_{\text{mean}} - s_{\text{mean}}|$. The third term, $\angle(\mathbf{s}, \mathbf{s}')$, is the angle between two vectors, obtained from the previous state, the current state and the next state. The first two terms aim to penalise actions that resulted in greater joint changes and larger displacement between two consecutive states. The third term penalises actions that result in a sharp turn between states. The last term is determined heuristically to facilitate the convergence of the agent. The optimal trajectory is defined based on its overall trajectory length in both the joint and end-effector spaces and the smoothness of the end-effector.

After training, the optimal trajectory is generated by continuously taking the optimal action of the current state from the start point to the end of the mean trajectory. The corresponding joint trajectory is determined through inverse kinematics.

3.3 Experiments And Results

3.3.1 Experimental Setup

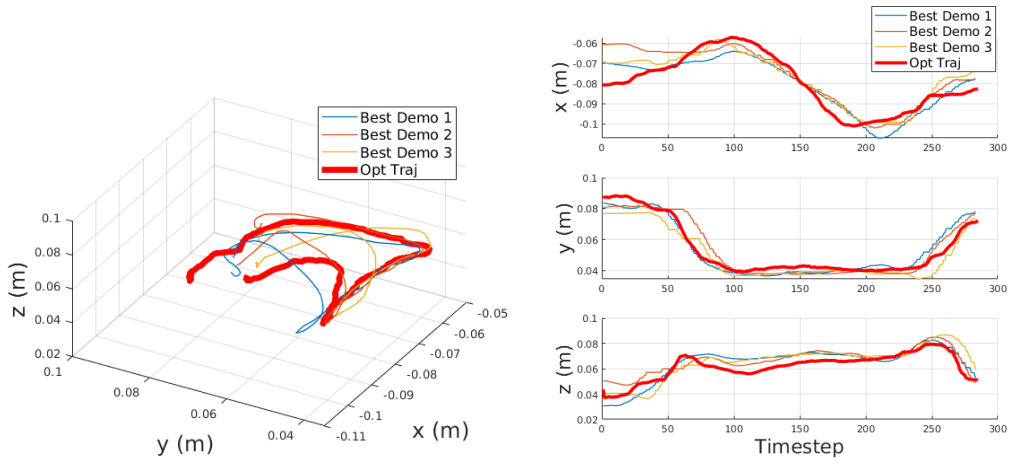
Two experiments were conducted to evaluate the performance of the proposed framework. The first experiment assessed the sensitivity of the learned policy to the inclusion of the terms in the reward function. The second experiment focused on evaluating the improvements achieved by the optimised policy compared to the baselines learned directly from suboptimal demonstrations. To evaluate the performance of the learned policy, two metrics were used: overall trajectory length in Cartesian space and trajectory length in the robot's joint space. Both metrics are commonly employed in evaluating surgical subtasks in prior works [72, 87, 128] and can provide valuable insights into the performance of the framework.

13 sets of human demonstrations were collected from 5 novice demonstrators using the experimental setup shown in Fig. 3.2(a) to validate the proposed framework. The demonstrators were shown a successful stitching demonstration and given 10 minutes to practice. Each demonstrator was instructed to perform a successful single-handed stitching demonstration at a specific location on the mandrel. This demonstration consisted of the three motion primitives as mentioned and the entire motion was recorded by the stereo camera. The demonstrated trajectories had various temporal lengths, ranging from 87 to 163 steps, with an average of 120 steps.

In each experiment, the evaluation was performed on 10 independent trials. Each trial involved learning an optimal policy from different sets of human demonstrations sampled from the collected demonstrations. All the demonstrated trajectories were smoothed using a simple moving average method with a window size of 10. Then, demonstrated trajectories from each trial were independently preprocessed and modelled to form a 3D constraint space in Cartesian space, represented as

$\bar{\gamma} = \{\bar{\mu}, \bar{\sigma}\}$.

The RL agent was trained to learn a mapping between states and actions. It consisted of two fully connected hidden layers. The first hidden layer consisted of 400 hidden units, while the others had 200 hidden units, both using the ReLU activation function. During learning, the RL agent started from$\bar{\mu}_0$. At each step, it selected one action from a total of 126 discretised actions, uniformly distributed from the $\bar{\mu}_{t+1} \pm \bar{\sigma}_{t+1}$. When an action resulted in a collision, the episode was restarted. The policy was updated iteratively until convergence. At the end of the training process, the optimal path was generated by continuously selecting actions with the highest Q-value at each timestep until the end pose was reached. Hyperparameters of RL such as the learning rate, discount factor, mini-batch size, number of episodes and number of timesteps in one episode were tuned heuristically and summarised in Tab. 3.1.



(a) (b)

Figure 3.6: Comparison of the end-effector trajectories between the 3 best human demonstrations from the testing dataset and the optimised one. (a) shows the comparison in the Cartesian space and (b) compares the differences in each DoF between the trajectories temporally.

Table 3.1: A summary of parameters used to train the RL agent.

Parameter	Symbol	Values
Step Size	δ	0.002
Discount Factor	γ	0.99
Episode	*Episode*	200
Steps	*Step*	284
Exploration Rate	ϵ	$0.75\frac{\text{Num\_Episode}-\text{Episode}}{\text{Num\_Episode}}$
Mini-Batch Size	N	32

3.3.2 Results and Discussion

In the first experiment, we examined the impact of different terms in the reward function on the performance. Quantitative analyses were performed on the two metrics mentioned and the policy's performance using different weights assigned to each term were examined. It's worth noting that the last term of the reward function served as a constant. It was used solely to adjust the reward bias and did not influence the final optimised trajectory. Therefore, we maintained its constant value for this term throughout the sensitivity test.

To assess the sensitivity of the learned policy to each reward term, the weight associated with one term was varied, while keeping the weights of other terms fixed. The performance evaluation was conducted over 10 independent trials, with each trial consisting of 5 different sets of human demonstrations. Every sensitivity test was trained on the same set of independent trials to minimise the bias in the distribution of the training data..

The results are summarised in Tab. 3.2. w1, w2 and w3 are the weights associated with the first three terms in Eqn. 3.1. The first term is to penalise for action that results in larger joint angle differences, the second is to penalise for longer trajectory in Cartesian space and the third is to penalise for learning unsmooth trajectory.

From Tab. 3.2(a), it is evident that there is an inverse relationship between $w1$ and

trajectory lengths in joint space as $w1$ decreases. This observation aligns with our expectations and indicates that the policy has been optimised based on the desired objectives of the demonstrator. However, some abnormalities can be found when $w1$ exceeds two. Specifically, a large value of $w1$ appears to negatively impact the length of trajectories in joint space. One possible reason for this abnormality is an overemphasis on minimising the trajectory length in joint space. This may lead to more aggressive and abrupt movements in joint space.

In the first three sets of results from Tab. 3.2(b), an increase in $w2$ corresponds to shorter trajectory lengths in Cartesian space, which again aligns with our expectations. However, there is a slight decrease in the trajectory length in Cartesian space when $w2$ is smaller or equal to one. This observation can be attributed to a relatively reduced impact of $w2$ compared to other terms during learning.

In Tab.3.2(c), the results demonstrate that an increase in $w3$ results in greater penalisation for actions that yield unsmoothed movements. Consequently, it leads to longer trajectories in both Cartesian and joint space. These findings are again consistent with the expectations. To ensure the learning of smooth trajectories, it is necessary for three consecutive points on a trajectory to be as colinear as possible. This requirement encourages the robot to follow a more continuous and fluid path, resulting in longer trajectories.

In the second experiment, the effectiveness of the optimised policy relative to the baselines learned directly from suboptimal demonstrations was assessed. The evaluation was conducted over 10 independent trials, each of which used 5 different human demonstrations. Weight combination, $w1 = 1, w2 = 10, w3 = 1$, was used in the reward function to train a policy as this was the optimal weight combination that resulted in the best-performing policy from the last experiment. For testing purposes, three of the best human demonstrations were excluded from the training dataset. These three demonstrations were selected as they had the shortest trajectory lengths

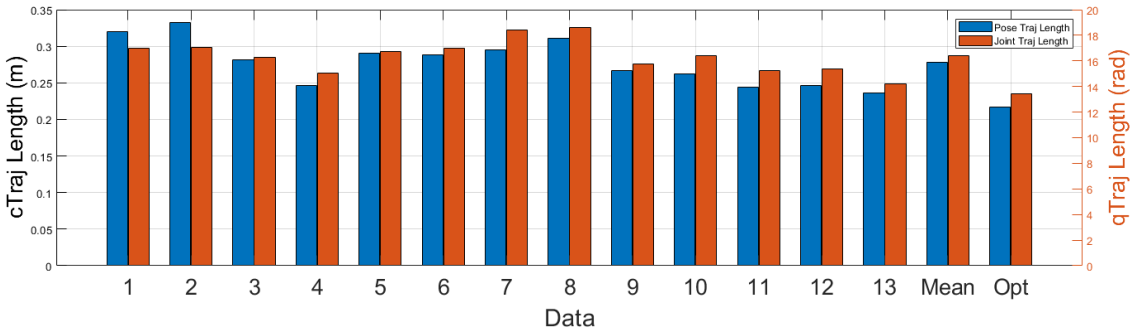


Figure 3.7: Comparisons of trajectory length between human demonstrated trajectories, the mean of the demonstrated trajectory and the mean of the optimised trajectory in Cartesian space and the robot's joint space. Note that cTraj Length and qTracj Length correspond to the trajectory length in Cartesian space and the robot's joint space respectively. The dataset comprises 13 demonstrations, with the first 10 demonstrations serving as training data to construct the bounded space. Demonstrations 11 to 13 are the best human demonstrations for testing and evaluation.

in both Cartesian space and joint space. Note that the human demonstrations only recorded the trajectories in Cartesian space. Their joint position trajectories were obtained by calculating inverse kinematics. In this experiment, we compared the learned trajectories with all the collected demonstrations and the mean trajectory $\bar{\mu}$ in terms of their lengths in both Cartesian space and joint space.

The experimental results are shown in Fig. 3.7. Among all the demonstrations, the mean of the learned trajectory exhibited the best performance. After evaluation on 10 independent trials, we obtained lengths of $0.2172m \pm 0.0478$ in Cartesian space and $13.4603rad \pm 0.2253$ in joint space, while the best human demonstration had lengths of $0.2440m$ in Cartesian space and $14.2188rad$ in joint space. Visual comparisons between the best demonstrations and the optimised trajectory are presented in Fig. 3.6. As clearly shown, the optimised trajectory not only appeared to have a shorter length but also exhibited smoother motion compared to the three best human demonstrations. The quantitative analysis validates the effectiveness of the proposed framework in learning an optimal trajectory from suboptimal demonstrations. By incorporating additional objectives in policy learning, we see that

the learned trajectory can be fine-tuned according to the demonstrator's preference and outperform directly learning from demonstrations in terms of both trajectory lengths and smoothness.

Table 3.2: Results of the sensitivity test. These compare the trajectory lengths in Cartesian space (cTraj Len) and joint space (qTraj Len) when training the RL policy using different weight combinations for each term in the reward function. w1, w2, w3 are weights associated with the first three terms in the reward function.

(a)

w1	**w2**	**w3**	**cTraj Length**	**qTraj Length**	**smoothness**
10	1	1	0.2208 ± 0.0326	13.5379 ± 0.3046	22.7287 ± 4.7063
4	1	1	0.2076 ± 0.0158	13.4718 ± 0.2188	21.0044 ± 1.4047
2	1	1	0.1971 ± 0.0166	13.4494 ± 0.2604	20.5045 ± 1.6609
1	1	1	0.1992 ± 0.0113	13.4568 ± 0.2543	21.5851 ± 2.7356
0.5	1	1	0.2026 ± 0.0191	13.4676 ± 0.2756	20.5108 ± 2.6054
0	1	1	0.2101 ± 0.0254	13.5476 ± 0.3884	20.1153 ± 1.7105

(b)

w1	**w2**	**w3**	**cTraj Length**	**qTraj Length**	**smoothness**
1	10	1	0.2090 ± 0.0159	13.4603 ± 0.2253	21.7162 ± 4.7767
1	4	1	0.2124 ± 0.0242	13.4671 ± 0.2187	21.9504 ± 3.3410
1	2	1	0.2106 ± 0.0153	13.5436 ± 0.2792	22.8224 ± 3.6675
1	1	1	0.1992 ± 0.0113	13.4568 ± 0.2543	21.5851 ± 2.7356
1	0.5	1	0.2040 ± 0.0150	13.4618 ± 0.1506	21.0464 ± 3.9734
1	0	1	0.1927 ± 0.0083	13.3910 ± 0.2379	20.4052 ± 2.9759

(c)

w1	**w2**	**w3**	**cTraj Length**	**qTraj Length**	**smoothness**
1	1	10	0.2549 ± 0.0473	13.8107 ± 0.1100	26.3503 ± 5.5383
1	1	4	0.2144 ± 0.0240	13.5222 ± 0.2801	21.7033 ± 4.2326
1	1	2	0.2007 ± 0.0098	13.4850 ± 0.0128	21.7829 ± 2.9467
1	1	1	0.1992 ± 0.0113	13.4568 ± 0.2543	21.5851 ± 2.7356
1	1	0.5	0.2054 ± 0.0124	13.4722 ± 0.1023	21.4827 ± 4.2276
1	1	0	0.1945 ± 0.0064	13.4553 ± 0.2785	20.2863 ± 0.9836

3.4 Conclusions

This chapter introduces a novel LfD framework with two key contributions. Firstly, the proposed approach extends the traditional LfD method by enabling a robot to learn from scarce and suboptimal demonstrations. Instead of directly learning the state-action mapping, the demonstrations are used as task constraints that limit the robot's exploration during policy learning. Secondly, the framework incorporates additional objectives to provide demonstrators with flexibility in fine-tuning the learned skill or trajectory. This is not feasible with the conventional behaviour cloning approach. To validate the proposed framework, experiments were conducted on a sewing task. Results demonstrated improved performance compared to learning directly from human-demonstrated trajectories. This has been shown with reduced trajectory length and improved smoothness in the learned trajectory. This approach also shows promise for applications where only limited training data is available for efficient learning.

Chapter 4

One-Shot Imitation Learning for Contact-Rich Manipulation

In Chapter 3, we considered robot learning in a structured environment. Robots can learn manipulation skills from demonstrations but do not have to adapt to new tasks or environments. In this chapter, the focus is on learning a contact-rich manipulation task in an unstructured environment, where robots have to adapt to the changing environment. Learning adaptability from demonstrations may be difficult and impractical. This work explores the self-supervised strategy for the robot to reason about the state of the environment and adapt a single demonstrated motion sequence to complete a contact-rich manipulation task.

An introduction is presented in Section 4.1 followed by the method in Section 4.2. Section 4.3 details the experiments and results. Finally, the work is summarised in Section 4.4.

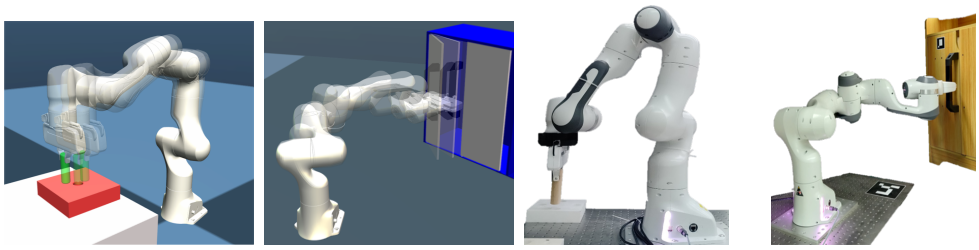


Figure 4.1: Two common contact-rich tasks are presented and used to evaluate the proposed method. (a) Peg-in-hole task in simulation. (b) Door opening task in simulation. (c) Peg-in-hole task in reality. (d) Door opening task in reality.

4.1 Introduction

Contact-rich manipulation tasks often concern how to localise and interact with an object. These tasks typically come with some form of constraint, determining the ways in which the object can be contacted and interacted with. Nonetheless, humans can still leverage a rather simple strategy to interact with the object without much prior knowledge of it and a vast amount of training data. To interact with an object by hand, a visual inspection can first be made to guide the hand towards the target; then, by making physical contact with it, the target can be felt, allowing the adjustment to be made by the hand relative to the object. Finally, when the hand is aligned with the object, a suitable contact-rich manipulation can be executed. A typical example is a key insertion task. With the key in hand, we can first take a visual estimation of the hole. Then, by making the key into contact with the hole through a "wiggling" motion, the location of the hole can be inferred. This allows the key to be further adjusted and aligned with the hole before insertion takes place. Inspired by the way in which humans use perception to accomplish a contact-rich task, in this chapter, we study whether this approach can be efficient and practical for a robot and generalisable to different tasks.

A recent and closely related work is [71]. While the work only uses vision for object pose estimation and focuses on daily manipulation tasks, the author has shown that it is feasible to teach a robot using an IL approach and break down the learning

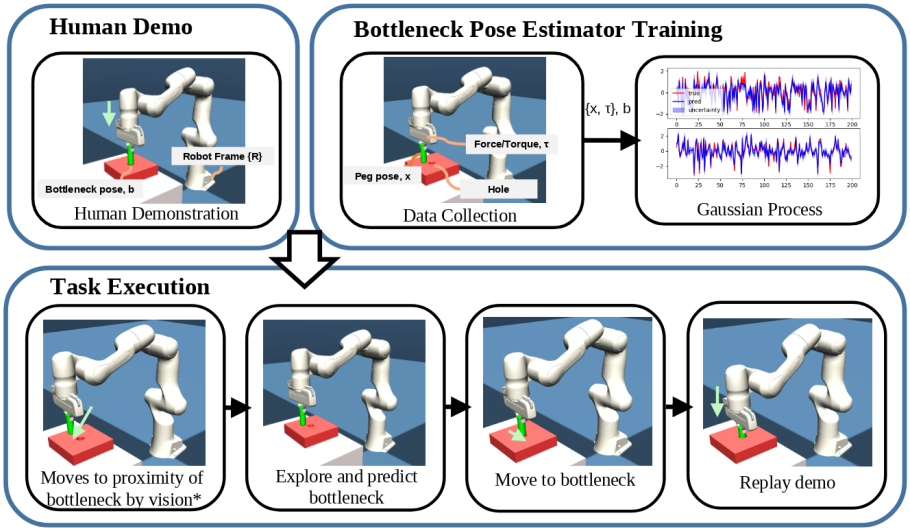


Figure 4.2: This shows an overview of the proposed framework. In a human demonstration, the robot is guided kinaesthetically to interact with the object from the bottleneck pose. A bottleneck pose estimator is trained using a self-supervised approach and GP. (*)Before task execution, we assume the end-effector is moved to the proximity of the bottleneck by vision-based estimation [71]. Then, through a random exploration, GP estimates the bottleneck based on its observations. Finally, it moves to the bottleneck and replays the demonstration.

problem into pose estimation and demonstration replay without any prior knowledge of the object being interacting with. This is based on the idea that if the robot's end-effector can align with an object during testing, in the same way as during the demonstration, then the demonstrated end-effector actions can simply be replayed to solve the task. This work shows that the alignment can be achieved using a wrist-mounted camera and training a self-supervised pose estimator of the bottleneck, defined as the point where physical interaction between the robot and the object begins. However, the camera-based markerless pose estimation via regression can have errors of around 1 cm, rendering this method unsuitable for tasks requiring precise control, such as many contact-rich tasks [71].

Inspired by the way humans perform contact-rich tasks and the method in [71], we propose a generic framework that also leverages IL to teach a robot contact-rich tasks. Given a vision-based initialisation from [71], the robot performs a similar "wiggling" motion as humans to explore the local area around this end-effector

pose and make physical contact with the object.Using historical end-effector 3 DoF position (m) and 6 DoF force/torque (N and Nm), a pre-trained bottleneck estimator predicts the bottleneck pose. Provided with only a single demonstration, the end-effector can follow the same actions from this point onward to achieve fine-grained physical interaction with the object. This framework uses a Gaussian Process (GP) for bottleneck prediction. Trained in a self-supervised manner, it can capture the nuance in the low-dimensional proprioception and force data and make a prediction of the bottleneck, together with uncertainty. Our method does not rely on much prior knowledge of the object and a significant amount of training data to learn to interact with the object. Our experiment shows that this method endows the robot to learn contact-rich tasks from just a single demonstration and can accomplish common peg-in-hole and door-opening tasks with success rates of 80%.

4.2 Methodology

4.2.1 Summary of Method

IL leverages human demonstration to provide an efficient way for the robot to learn complicated manipulation skills, ideally without requiring significant prior task knowledge. We consider a particular form of IL [71], where the robot learning process can be broken down into bottleneck alignment and a demonstration replay. In this context, the bottleneck is referred to as the end-effector's pose relative to the object at the start of the demonstration. In the prior work [71], the author shows that a vision-based bottleneck estimator can be trained to guide the robot's end-effector to the bottleneck at test time. The demonstrated actions can then be followed to accomplish many everyday manipulation tasks. This work demonstrates that visual perception alone can allow a robot to estimate an object's pose from a distance without requiring direct contact. This means the robot would be able to interact with randomly placed objects in its environment. However, the approach using high-dimensional visual inputs and convolutional neural networks for training the pose estimator resulted in prediction errors of about 1 cm. For many manipulation tasks that require high precision or direct contact, this level of error would be unacceptable.

Inspired by the prior work, a generic IL framework is proposed for a contact-rich task. The overall pipeline is illustrated in Fig. 4.2. Considered a target object of an unknown pose in a contact-rich task, we assume that by using visual observations and the method described in [71], the robot's end-effector can first reach the vicinity of the bottleneck relative to the target object. With an error of up to 1 cm from the ground truth bottleneck, we then propose to use low-dimensional contact-type feedback, i.e., proprioception and force data, to feel around the bottleneck and correct the estimation made by vision. A GP pose estimator is used for the estimation

purpose, as GP works well with low-dimensional data and in dealing with uncertainties. Trained in a self-supervised approach, the pre-trained estimator can infer the pose of the bottleneck from contact feedback within the vicinity of the bottleneck. By using the uncertainties from the GP's predictions, we can weighted average all the past estimations made. This allows the error in the aggregated estimate to be decreased as the robot collects more data. Finally, with only a single demonstration, the same actions can be replayed from the estimated bottleneck to interact with the target object. This framework consists of four main components, namely human demonstration, data collection, bottleneck pose estimator, and task execution, and will be detailed in the rest of the section.

4.2.2 Human Demonstration

A human demonstration is performed kinesthetically to provide the reference trajectory in Cartesian space and to bring the robot into interaction with an object. Prior to the demonstration, the robot's end-effector, x_{d0}, aligns with the object's bottleneck pose, b. During the demonstration, the robot then records a sequence of its end effector's pose, x_d, i.e., trajectory. The trajectory is collected relative to the end-effector's local frame so that it can be readily replayed on different bottleneck poses at test time as opposed to working with the joint space trajectory.

4.2.3 Data Collection

In data collection, the contact observations and the corresponding bottleneck poses are collected for training the bottleneck pose estimator. The observation of the estimator involves 3 DoF position and orientation and 6 DoF force/torque measured with respect to the end-effector frame. The position and orientation are measured in metres and radians. The force/torque are measured in N and Nm respectively.

The estimator is trained to estimate the bottleneck pose, with the output in metres. The observed pose is expressed relative to the initial pose of the end-effector. We prevent the use of an absolute observed pose because, at test time, the object pose is subject to changes. Using an absolute pose prevents the method from generalising to different object poses.

The robot automatically collects the data through random exploration. By making multiple physical contacts at different locations around the bottleneck and with the target object, the robot obtains and records the proprioception, x and force/torque sensory feedback, τ. Since the pose of the object and the bottleneck remain unchanged during the demonstration, the robot can also obtain the labelled target, i.e. pose of the bottleneck. At test time, we assume the robot's end-effector can move to the vicinity of the bottleneck by a vision-based pose estimator, and therefore the boundary to which the data is collected is governed by the prediction error of this estimator, i.e. $0.01m$ around the ground truth bottleneck as reported in [71].

Note that the proposed approach taken for exploration is inspired by how humans, for example, leverage random exploration to insert a key into a lock. While it does require sufficient exploration data to identify the bottleneck accurately, it is simple to implement and has proved adequate for the tasks in this work. A more complex exploration can be easily integrated into the proposed framework. For example, an exploration strategy conditioned on the vision-based bottleneck prior may shorten the exploration time and improve the accuracy of bottleneck prediction. However, this is beyond the scope of the study and will be addressed in future work. Also, in this chapter, our focus is on the exploration and estimation of the bottleneck in 3D position. In future work, we will consider more complex physical interactions involving orientation.

4.2.4 Bottleneck Pose Estimator

Since we have a partial observation problem, a sequence of proprioception and force/torque sensors, $\boldsymbol{z} = \{\boldsymbol{x}, \boldsymbol{\tau}\}$, is mapped to the bottleneck pose b'. In this work, observations from two previous timesteps are used to predict the bottleneck. The estimator is trained using GP. Given the training inputs and the associated labelled target, GP can perform estimation on input data with uncertainty. In this mapping problem, we intend to find a probability distribution over functions that maps $\boldsymbol{z} \mapsto f(\boldsymbol{z}) = b'$, defined by the prior mean, μ, and the covariance function, $\mathcal{K}$

$$f(\boldsymbol{z}) \sim \mathcal{GP}(\mu(\boldsymbol{z}), \mathcal{K}(\boldsymbol{z}, \boldsymbol{z'})) \tag{4.1}$$

For convenience, the prior mean is set to be the zero function and the following anisotropic kernel is adopted:

$$\mathcal{K}(\boldsymbol{\theta}, \boldsymbol{\theta'}) = \sigma_f^2 exp(-\frac{1}{2\boldsymbol{L}^2}(\theta - \theta')^2) + \sigma_\epsilon^2 \delta_{\boldsymbol{\theta},\boldsymbol{\theta'}} \tag{4.2}$$

This consists of radial-basis function kernel and Gaussian noise, where σ_f^2 is the signal variance, $\mathbf{L}$ is the vector of length scale and σ_ϵ^2 is the variance of the noise. The squared length scale controls the smoothness and scale of the kernel. To model the GP predictive distribution, a GP prior $p(f|\boldsymbol{z})$ is conditioned on the training data, D, which contains the vectors of $\boldsymbol{z_D}$, and the associated vectors of output, the bottleneck, $\boldsymbol{b'_D}$, i.e. $D = \{\boldsymbol{z_D}, \boldsymbol{b'_D}\}$:

$$p(f_*|z_*, D) = \mathcal{N}(\mu, \sigma^2) \tag{4.3}$$

where f_* and z_* are the prediction and the test input, respectively.

4.2.5 Task Execution

At test time, the object is randomly positioned, and hence the bottleneck is unknown and differs from the demonstrated pose. Following the previous work, we assume the bottleneck pose could be first estimated by vision, and the robot can be guided to the proximity of the bottleneck with a prediction error of up to $0.01m$. This pose, x_0^t, is where the proposed framework will be carried out.

Taking N steps of random contact exploration around x_0^t, the GP estimator makes $N-1$ sets of estimation, b'_*, and uncertainty, σ_b. For GP, high uncertainty can be given to estimation if 1) the data input at test time varies a lot from any of the data inputs at training data or 2) the observations are made in free space movement, and provide uninformative feedback about the bottleneck. Therefore, taking advantage of the uncertainties, we intend to upweight the estimations made from useful proprioception and force inputs, i.e. data that captures the physical properties around the bottleneck, and downweight the estimations made from feedback during uninformative free space movement. The final estimation of the bottleneck, $\hat{b'_*}$, is the weighted average of all the b'_*, and is calculated as follows:

$$\hat{b}'_* = \frac{\sum_n^{N-2}(1-\sigma_b^n)b_*'^n}{\sum_n^{N-2}(1-\sigma_b^n)} \tag{4.4}$$

Given $\hat{b'_*}$, the robot then moves towards it in free space and subsequently, the demonstrated end-effector velocity is replayed to interact with the object.

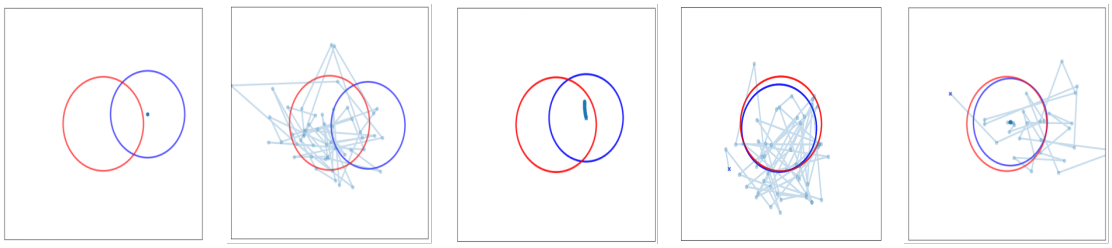
4.3 Experiments and Results

The proposed framework was evaluated on the two typical contact-rich tasks, peg-in-hole and door opening, both in simulation and real settings, as shown in Fig. 4.1. In simulation, a detailed comparison in performance was made against 4 other baselines, namely pure human demonstration (HD), human demonstration with random exploration (HD+Rand), reinforcement learning (RL), and human demonstration with a multi-layer perceptron estimator (HD+MLP). In real, we evaluated the feasibility of our method. All the experiments were conducted on a 7 DoF Franka Emika robot arm with a two-fingered gripper. It can measure sensory feedback, including 6 DoF proprioception and 6 DoF force/torque in the end-effector frame. A compliance control was implemented on the robot to prevent hard collisions when making contact with the object and the environment.

4.3.1 Task Setup

For each task, the bottleneck was chosen to be where the physical interaction between the robot and object starts. The demonstration was performed kinesthetically by bringing the robot's end-effector from this bottleneck to interact with the target object and accomplish the task. Prior to all training and testing, we assumed the robot's end-effector had already reached the vicinity of the bottleneck by vision, but with a prediction error of up to $0.01m$ to the ground truth bottleneck according to the method in [71]. Note that during training and demonstration, the pose of the object and the bottleneck remained unchanged, but at testing, the object's position was subject to movement.

The simulation was created in MuJoCo [132] and shared the same setting as the real-world environment. In the peg-in-hole task, the robot's gripper vertically held a peg with a diameter of $0.029m$, and the goal was to insert the peg into the hole with



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

Figure 4.3: This compares the policy execution process between the proposed method and the other baselines. For illustration purposes, we only show the execution process in the x-y plane. (a) Human demonstration without random exploration, (b) Human demonstration with random exploration, (c) PPO, (d) Human demonstration with an MLP estimator, and (e) Human demonstration with a GP estimator. In each plot, the red circle indicates the location of the hole, the blue circle indicates the final position of the peg, and the blue dot line indicates the trajectory taken to the final peg position.

a diameter of $0.033m$. A single demonstration provided the downward actions from the bottleneck to insert the peg in the hole. At test time, the hole was randomly positioned, but with vision initialisation, the alignment between the end-effector and true bottleneck can be made with an error of up to 1 cm. For the door-opening task, the bottleneck was at the centre of the doorknob, and the demonstration followed an arc path to open the door from this point. At test time, the door was randomly positioned. We, again, assume that the end-effector could be guided to the vicinity of the door knob by vision, with an error of up to 1 cm. The task was considered successful if the door could be opened by more than 30°. This is a critical angle at which the door cannot be further opened if the bottleneck is not aligned well.

4.3.2 Results and Discussions

Simulation

Two studies were conducted in simulation to evaluate the proposed HD+GP approach. In the first study, we examined whether HD+GP could successfully complete the tasks when the target object was placed at different random locations in

each trial. We also evaluated how effective HD+GP's estimator was in inferring the bottleneck location compared to other baselines. In the second study, we investigated whether the number of exploration steps could be dynamically adjusted based on HD+GP's estimation results. Given that GP produces estimations based on uncertainty, which depends on how informative the observations are, we wanted to determine whether variations in HD+GP's bottleneck estimations could converge according to observation quality. Four baselines were chosen to compare against the proposed method for both tasks, and are described as follows.

Pure human demonstration (HD) was the original baseline from [71]. At test time, since we assumed the robot's end-effector was guided to the vicinity of the bottleneck by vision-based estimation, the demonstration actions were directly replayed from this point onwards without further correcting the end-effector's pose.

Human demonstration with random exploration (HD+Rand) was implemented to test whether the active localisation of the bottleneck in our method is superior to just random movements without any additional training. During testing, the robot underwent 100 exploratory steps, each step was uniformly sampled within a range of -0.01 to 0.01m from its starting position x_0^t. After the exploration, the robot followed the demonstrated trajectory.

Proximal Policy Optimisation (PPO). PPO was selected as a RL baseline to compare against the proposed method due to its strong performance on a variety of problems. The goal of this work was to evaluate PPO's data efficiency in comparison to the proposed approach. The PPO agent was trained in the same environment as the demonstration, where the pose of the object (the hole and door) remained unchanged as in the demonstration. Each episode began with vision-based initialisation and the agent was deployed to complete a task within 100 steps. The agent was trained to select an action of the end-effector that maximised the total reward based on the observation. The observation included the robot's end-effector pose

and the 6 DoF force/torque relative to its state at the beginning of the episode. The reward function was, $r = exp(-d)$, where d is the distance between the current and the target end-effector pose. The target end-effector pose corresponds to where the end-effector should end up relative to the vision-based initialisation upon successful completion of the task. The agent was trained on a total of 1200 timesteps, which was the total amount of training data used to train the proposed GP estimator. No domain randomisation technique was applied to vary the training conditions during training. At test time, the learned PPO policy took 100 timesteps to interact with the target object and complete the task, starting from the vision-based initialisation of x_0^t.

Human demonstration with an MLP estimator (HD+MLP) took a similar approach as the proposed framework to train the estimator and interact with the object, except that the GP estimator was replaced with an MLP estimator. The estimator consisted of two fully connected layers with leaky ReLU and Tanh. It was trained using a self-supervised approach. The estimator took the robot's end-effector pose and the 6 DoF force/torque from the past two timesteps as inputs and estimated the bottleneck pose. During training, a single demonstration was provided starting at the known bottleneck position. With this bottleneck position, the robot then collected 1200 timesteps of observations around the bottleneck and the labelled target, i.e. bottleneck pose. Since only partial observations were available, the estimator was trained to take three historical observations to estimate the bottleneck pose. During testing, the robot randomly explored around x_0^t for 100 steps, and at each step, the MLP estimator made an estimation of the bottleneck based on the observation. The final estimation was computed by averaging all the past estimations made. Lastly, the robot moved to where the estimation was and executed the demonstration actions.

Human demonstration with a GP estimator (HD+GP) is the proposed method.

The GP estimator was trained in the same way as HD+MLP. Both of these estimators share the same inputs and outputs and are trained on 1200 timesteps of observations. During testing, HD+GP also took a random exploration for 100 steps around x_0^t and made one estimation with uncertainty. The key difference between HD+MLP and HD+GP was how the final pose was computed after exploration. HD+MLP simply averaged all 100 past estimations equally. HD+GP instead performed a weighted average of the per-step estimations, weighting each by its associated uncertainty from the GP. After computing the final bottleneck pose estimate, the robot moved to the predicted bottleneck and replayed the demonstrated trajectory.

In the first study, each method was evaluated over 10 independent trials, with 10 repetitions per trial. For each trial, a single successful demonstration was provided starting from the bottleneck for each task across all the methods, except for PPO, which did not use any demonstration. For each repetition, the target object (hole or door knob) was placed in a different position relative to the robot. The robot's end-effector started from a random initial position near the bottleneck to emulate a vision-based initialisation. The distance between the initial end-effector pose and the bottleneck was within 1 cm. In this work, two metrics were used to evaluate the performance of each method. 1) The distance between the robot's end-effector and the bottleneck at the end of the task. This distance indicates how well the end-effector could align with the bottleneck, which reflects how accurately the method's estimator was trained. A smaller distance suggests better alignment and training. 2) The success rate. The percentage of trials in which the method could complete the full task. A higher success rate indicates a more robust performance of the method to the uncertainty of the environment. For the door-opening task, an additional metric was measured, i.e. the maximum degrees of door angles that could be opened. This metric evaluates the influence of the end-effector's alignment with the bottleneck on the range of door angles that could be opened.

Results In Fig. 4.3, we illustrate the strategy taken for each approach in the peg-in-hole task. The original baseline (HD) and HD+Rand were not additionally trained to correct the prediction error from the vision-based estimation. Hence, at test time, these approaches failed to guide the robot to the bottleneck and accomplish the task. While PPO was trained to infer such information implicitly, the training data was insufficient for it to complete this task. On the contrary, both HD+MLP and HD+GP can correctly infer the bottleneck with sufficient accuracy that the demonstrated actions can be carried out to accomplish the task.

Tab. 4.1 summarises the quantitative evaluation and comparison of the different methods on the two manipulation tasks. This table compares the bottleneck pose error distance, success rate and the maximum door opening angle. The bottleneck pose error distance measures the difference between the reached bottleneck pose by each method and the ground truth bottleneck pose. It evaluates how precisely each method can infer the bottleneck pose from partial observations. The success rate quantifies the robustness of each method against the robustness of a method to the uncertainty of the environment. For the door opening task, the maximum door angle attained provides an additional metric to evaluate performance. This angle measures how wide the door could be opened during the test runs. It demonstrates how precisely the end-effector was aligned with the door handle bottleneck. More accurate estimation and better alignment allow the door to be opened to wider angles during the task.

For the peg-in-hole task, HD, HD+Rand, and PPO often failed to reach the correct bottleneck pose, with the mean distance between the final pose reached and the ground truth being over 4cm. This is clearly larger than the tight clearance between the peg and hole. In contrast, both HD+MLP and HD+GP achieved a low mean error distance and a high success rate. This demonstrates that the trained estimators for these methods could successfully infer the proper bottleneck pose by leveraging

random exploration. As expected, HD+GP slightly outperformed HD+MLP, since HD+GP utilises the uncertainties to adjust its estimations, while HD+MLP simply averages all estimated poses. The performance of HD+MLP and HD+GP demonstrates the value of a learned pose estimator versus pure random trial-and-error for this precision insertion task. HD+MLP and HD+GP also demonstrated greater data efficiency in learning compared to the PPO method, which learned the task completely from scratch.

For the door opening task, the results in Tab. 4.1 show that all the methods utilising human demonstrations could open the door in most trials. Compared to HD, HD+Rand and HD+MLP, HD+GP achieved the best alignment with the bottleneck (doorknob), as indicated by the lowest mean distance. Due to the arc-shaped demonstrated trajectory, poor alignment could cause the end-effector to lose grasp of the knob, or result in the robot deviating the robot from the desired path and dragging the door and cabinet. These issues were most prevalent in HD and HD+Rand, which had the highest mean distances. In terms of success rate, HD+MLP and HD+GP performed the best, with HD+GP slightly superior. For maximum door angles opened, it is important to note that with perfect alignment, the demonstrated motion could open doors up to 50°. HD and HD+Rand achieved higher mean angles but lower success rates, indicating larger variations in the door angles opened. Their performance depended heavily on the quality of vision initialisation and random exploration. In contrast, HD+MLP and HD+GP showed more consistent results. The PPO baseline, on the other hand, achieved the poorest performance. This was attributed to limited training data and poor grasping of the doorknob.

The simulation results demonstrate that HD+MLP and HD+GP achieved similar and superior success rates compared to the other baseline methods. This indicates that the IL approach of decomposing the learning problem into pose estimation and demonstration replay enabled the robot to learn a contact-rich task. By training

an estimator to infer the complete hidden state from partial visual observations, HD+MLP and HD+GP could efficiently perform demonstrated tasks despite environmental uncertainties. The high success rates demonstrate the value of segmenting the contact-rich learning pipeline into visual state estimation, contact-rich state estimation, and demonstration replay.

Note that, overall, the results showed that the door-opening task achieved a higher average success rate compared to the peg-in-hole insertion task across all the methods. The difference in performance between the two tasks can be attributed to several factors. The peg-in-hole task posed more difficulties due to the tight clearance between the peg and the sides of the hole, requiring the robot's end-effector to be precisely aligned and navigate to the bottleneck. In contrast, the door opening task presented the robot with multiple possible bottlenecks that could lead to a successful demonstration replay. Specifically, anywhere along the long axis of the door knob could result in pulling the door open. This increased the possibility for the baseline algorithms and the proposed method to stumble upon the correct alignment by chance.

In the second study, to determine whether the estimation could converge, we calculated the weighted average estimation based on the past estimations and uncertainties at each step of exploration. The variations in this weighted average estimation are plotted against the number of exploration steps in Fig. 4.4. The observation data were obtained from the peg-in-hole trials. Initially, with a limited number of quality observations, the variation in the estimation is large. As more data were included, the weighted averaged estimation result started to converge after 50 steps. This implies that a threshold can be set for the variation in the weighted average estimation. The robot can either be encouraged for further exploration or terminated early for demonstration replay, allowing the proposed method to be more efficient and robust against poor observations.

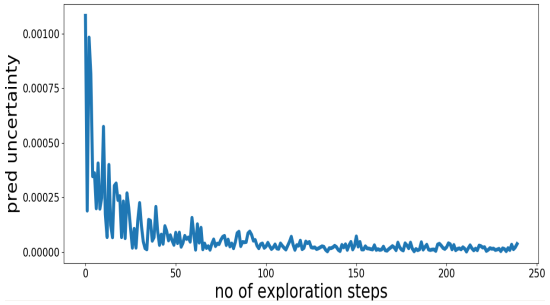


Figure 4.4: This presents the bottleneck estimations of HD+GP against the number of exploration steps.

Table 4.1: This table compares the performance of each method for each task in simulation quantitatively.

	Peg-in-Hole (Sim)		**Door Opening (Sim)**		
	Distance to GT Bottleneck (m)	Success Rate (%)	Door Opening Angle (°)	Distance to GT Bottleneck(m)	Success Rate(%)
HD	0.042±0.0015	1	36.38±7.60	0.021±0.0043	73
HD + Rand	0.044±0.0040	9	31.64±12.20	0.019±0.0064	66
PPO	0.056±0.0032	0	18.85±9.90	0.61±0.027	6
HD + MLP	0.0016±0.0013	74	34.81±4.73	0.017±0.0042	84
HD + GP	0.0013±0.0013	82	39.16±4.01	0.011±0.0037	88

Real Robot Experiments

The proposed method was validated on the peg-in-hole and door-opening tasks in real robot experiments. The real experiments shared the same environmental settings and conditions as the simulation. A human demonstration was kinaesthetically provided to teach the robot how to interact with an object from the bottleneck, and the GP estimator was trained from 1200 timesteps of exploration around the bottleneck of the demonstrated task in a self-supervised manner. This was equivalent to about 20 minutes of data collection. At test time, to emulate the vision initialisation with an error of up to $0.01m$, we randomly selected a point around the ground truth bottleneck with their maximum distance being $0.01m$ as the starting point for the robot's end-effector. The robot then made 50 steps of exploration observation to infer the bottleneck. With the end-effector positioned at the predicted bottleneck, the demonstrated velocity trajectory was then replayed to interact with the object.

10 tests were carried out for each task, and a test was considered successful if the peg could be successfully inserted into the hole or the door could be opened by more than 30°. For the peg-in-hole task, we achieved a success rate of 80%, with the clearance between the peg and the hole being $0.004m$. In the two failed cases, the peg was only off by 0.001 to $0.002m$. The main reason for the failure was that the peg collided with the surface around the hole while moving towards the bottleneck. The same success rate was also obtained for the door-opening task. The failed cases were caused by the poor estimation of the doorknob, which resulted in the gripper slipping away from the doorknob when replaying the demonstration. This was likely due to the limited quality of observation, which was caused by fewer contacts made between the gripper and the knob during exploration.

4.4 Conclusions

Inspired by how humans interact with an object, a novel method using IL is introduced for the robot to acquire a manipulation skill for a contact-rich task. We break down this learning into a bottleneck pose estimation and trajectory replay process. In this framework, the end-effector of the robot is required to first align with the bottleneck pose relative to the object before the demonstration can be replayed to interact with an object. A GP estimator, trained in a self-supervised learning approach, can make an estimation of the bottleneck from contact-type sensory feedback. The prediction is sufficiently accurate to allow the robot to make a fine physical interaction with the object. In the experiment, an extensive study was carried out in simulation, and a detailed comparison was made with the four baselines. Our method has obtained the highest success rates among the others across all the tasks. In the real robot experiments, the proposed method was shown to be practical. Training on real sensory feedback, the robot achieved a success rate of 80% in both peg-in-hole and door-opening tasks.

Chapter 5

DROID: Minimising the Reality Gap using Single-Shot Human Demonstration

In the previous chapter, we have shown that learning through self-exploration can endow robots with adaptability while minimising human intervention. One stream of robot learning approaches is RL. RL has demonstrated great success in the past several years. However, it is known for being data-demanding, and most works focus on learning a policy in only a simulated environment. One of RL's main challenges is transferring the learned policies from a simulated environment to the real world. The discrepancy between the two environments often causes the failure of sim-to-real transfer. In this chapter, we continue to explore the use of human demonstrations. The aim is to construct a simulated environment that better reflects the real one, so robots can learn efficiently in simulation and the policy can be transferred to reality.

Section 5.1 provides an introduction on simulation and sim-to-real transfer. The method of the proposed learning approach is presented in Section 5.2. Section 5.3 describes the experiments and results. Finally, the work is summarised in Sec-

tion 5.4.

The original work from which this chapter has been adapted is published in:

Tsai, Y. Y., Xu, H., Ding, Z., Zhang, C., Johns, E., and Huang, B. (2021). Droid: Minimising The Reality Gap Using Single-shot Human Demonstration, *in* 'IEEE Robotics and Automation Letters'.

5.1 Introduction

RL has been widely applied to decision-making, control, and planning. In the field of robot learning, many works have adopted RL as the robot controlling policy to improve its learning efficiency and performance [79, 89, 117]. Recent works have demonstrated that RL can be used to control a dexterous robotic hand or a robotic arm to solve tasks that require complicated manipulation skills, such as solving a Rubik's cube [5, 9] or opening a door [137].

RL uses self-exploration to find the optimal policy. Typically, this requires a considerable amount of trial-and-error, which is time-consuming and can easily result in hardware damage if executed on the physical robot. A less costly and safer approach is learning the policy via simulation. However, the discrepancy between the real world and the simulation models could hinder the policy from directly being deployed in the real world, especially when the task is contact-rich. In the literature, this *sim-to-real* problem is referred to as the reality gap, which remains an open issue to date.

System Identification (SI) and *Domain Randomisation* (DR) are the two common approaches to cross the reality gap. While SI tries to reduce the reality gap by identifying the real-world physics parameters, DR tries to increase robustness to the reality gap by training in randomised simulated environments. However, these methods still struggle to accurately obtain real-world parameters or adequately choose randomisation ranges without any real data [138]. This can result in learning a biased policy or a policy failing to converge.

To address this, we propose a novel framework, Domain Randomisation Optimisation IDentification (DROID), to automatically optimise the environment parameter distribution with a combination of DR and SI approaches. Rather than determining the DR range using intuition or tedious tuning, we exploit human demonstrations

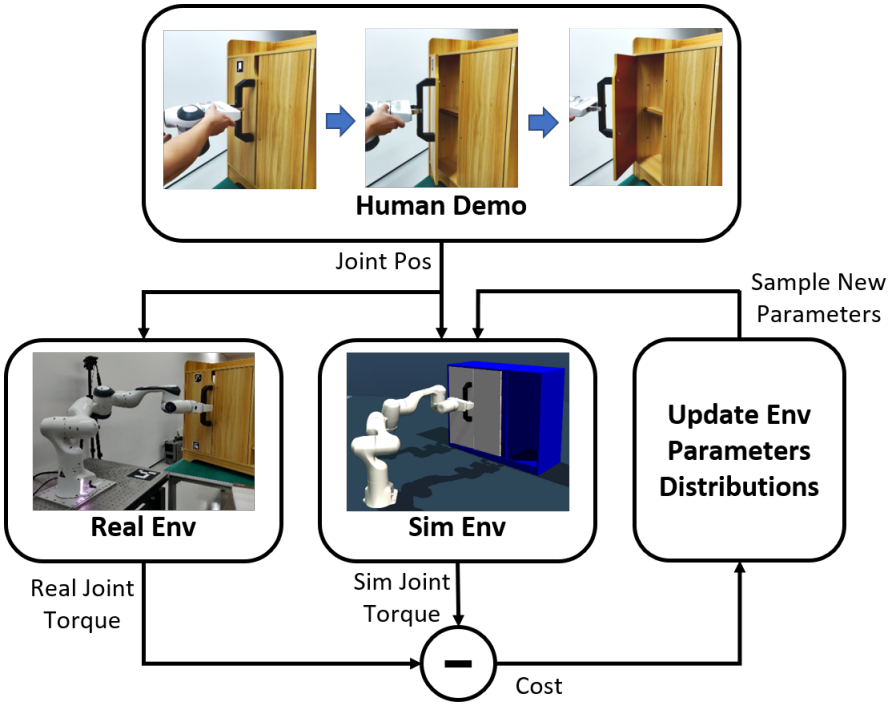


Figure 5.1: The overview of the proposed framework, DROID, used to minimise the reality gap.

and a robot's dynamics feedback to determine the appropriate parameter distribution. Through this, we identify the simulator's optimal parameter range as a statistical model, which can then be sampled during training with RL. An overview of this method is shown in Fig. 5.1.

With DROID, the learned policy can be transferred directly to the real world; thus, learning efficiency is significantly improved, and unsafe interactions with the environment are avoided. Our experimental results show that after parameter optimisation and identification, a much higher success rate can be achieved with DROID on a real-world door-opening task compared with standard DR or SI. We also show how the learned dynamics can be directly used to train policies for different but related tasks.

5.2 Methodology

Learning in a simulated environment is convenient, but transferring the learned results to the real environment requires an accurate model of that environment. Simulation typically uses mathematical models to compute the interaction force and torque between objects. These models rely on pre-defined dynamics-related parameters such as friction, stiffness, and damping. Unlike kinematics-related parameters, many of them are not easily accessible and, hence, are often difficult to measure and identify. Therefore, tasks that involve these parameters experience difficulty in sim-to-real transfer.

Estimating the dynamics parameters of a robot typically involves collecting real-world data through an excitation trajectory and fitting the parameters accordingly. This trajectory is carefully hand-designed to excite the robot at appropriate joint positions, and with various velocities and accelerations while meeting constraints like joint limitations and self-collision. However, designing a suitable trajectory can be challenging, particularly for high-order systems and highly dynamic tasks. In this work, our focus is not to accurately recover the dynamics of the real world, but instead to find appropriate distributions of dynamics parameters for sim-to-real purposes.

Working with a highly dynamic system may introduce additional complexity in, for example, sensor perception, robot learning, dynamics modelling in simulation and sim-to-real transfer. This makes it difficult to isolate the problem of the sim-to-real and validate the proposed method. Therefore, in this work, the validation is on a quasi-static contact-rich door-opening task, which involves a system of a 7 DoF robotic arm and a 1 DoF cabinet door. Sim-to-real involving more complex systems and highly dynamic tasks will be considered in future research.

The framework proposed to account for the sim-to-real gap is called DROID, which

determines the distributions of the real-world dynamics parameters through human demonstrations. Building on the concept of interaction force, we implicitly perceive the dynamics of the real-world system from the feedback of the robot and use this information to determine the distribution of the parameters in the simulation. This gives us a reasonable set of parameters for the DR in RL and hence results in a successful policy transfer.

DROID is composed of three phases: the human demonstration (Section 5.2.1), the parameter identification and optimisation (Section 5.2.2), and the policy learning with optimised DR (Section 5.2.3). In the first phase, the human demonstrates a contact task in the real world. The robot clones human behaviours multiple times and records the data involving dynamics feedback. In the second phase, a robot in the simulator repeats the same behaviours and records the same set of data. The data from the real system and the simulator are then used to identify the distribution of the task-relevant parameters. Through an iterative approach, we can gradually update the parameter distribution to minimise the differences between the two perceived feedbacks until the obtained simulated environments better reflect the real world. In the final phase, a policy is trained with DR based on the optimised parameter distribution. The resulting policy can be transferred to the real world with good performance. Note that this study focuses on minimising the reality gap, and the human demonstrations only serve the purpose of identifying the task-relevant parameter distribution. Learning the policy from human demonstration is not within the scope of this work. In the following sections, we will go into more detail on how each part is implemented.

5.2.1 Single-Shot Human Demonstration

In this first phase, we collect data reflecting dynamics relevant to the task. To this end, a human demonstrates the task once in the real world to provide a robot

joint position trajectory, $\boldsymbol{q_d}$, through kinesthetic guidance. This demonstration is safe to be executed by the robot and allows the robot to repeat it multiple times automatically. By repeating the $\boldsymbol{q_d}$ to interact with the environment, the dynamics information can be perceived by the joint torque sensor feedback, $\boldsymbol{\tau_r}$, of the robot in the real world. Such feedback is later used as a reference to identify and update the parameter distribution in the simulation. Different from the approaches that make the robot randomly interact with the environment, in DROID, we rely on a human to provide a trajectory that is safe for the robot to identify the system dynamics. This only requires a single-shot demonstration. We focus on the task-relevant parameter distributions and limit the random exploration of the robot in the real world. This minimises the risk and saves time in the real robot experiment.

5.2.2 Parameter Optimisation and Identification

The goal in this phase is to correctly identify the parameter distribution that minimises the discrepancy between the simulated and real systems. Rather than finding the specific value for each parameter, we determine their distributions due to the presence of noise and uncertainties in the real world. We can then train the RL to find a policy that works within this distribution, which is the key problem solved in DROID. Falsely defined distribution will lead to failure in sim-to-real transfer, and the policy trained under an unreasonable DR can suffer bad performance. Typically, DR samples values from fixed ranges due to its simplicity and scalability [107]. However, this may be less ideal to represent the noises or model complex distributions of parameters in the real world. Similar to many prior works [27, 29], we address this by using multivariate normal distribution in this work, which should be a more realistic representation of the distribution of values. By randomly sampling system parameters from this distribution, an RL agent can be trained on parameters that are more likely to appear in the real world, leading to improved sim-to-real perfor-

mance. This distribution is modelled as $\Phi(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and the $\boldsymbol{\Sigma}$ are the mean and covariance, and system parameters $\boldsymbol{\phi}$ are randomly sampled from Φ. In the later part of this section, we describe how we ensure the distribution is within a valid range for each parameter.

During the human demonstration phase, data was collected from the real system. Taking the identical steps, we can program the robot to repeat the same task in the simulation via a PD control and hence obtain the torque sensor feedback $\boldsymbol{\tau_s}$ from the simulation. Since the value of the real-world parameter $\boldsymbol{\phi}'$ is not easily accessible, we align the simulated and real environments by aligning the robot behaviours in them. Here, we define the "behaviour" as the robot action and perception pairs, i.e. the motion trajectory $\boldsymbol{q_d}$ and the torque sensing $\boldsymbol{\tau_s}$. Changing $\boldsymbol{\phi}$ changes the dynamics of the simulation and the perceived torque feedback from the robot. Sampling different $\boldsymbol{\phi}$ from its distribution Φ, we observe the different behaviours under different environments in the simulator and identify the ones that are most similar to the real-world behaviour. We update Φ based on the robot behaviours.

We formulate this as a distribution optimisation problem and update Φ iteratively based on the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) approach [53] with the following objective function:

$$\mathcal{J}(\boldsymbol{\phi}) = \frac{1}{N}\sum_{n=1}^{N}(\|\boldsymbol{\tau_s}(\boldsymbol{\phi}) - \boldsymbol{\tau_r}^n(\boldsymbol{\phi}')\| + c), \boldsymbol{\phi} \sim \Phi \tag{5.1}$$

where N is the total number of trajectories from the real robot, and c is to penalise the situation when the robot fails to grip the doorknob during the door opening process in the simulation. This occasionally happens when for example, the friction coefficients of the fingers become too low or when the joint damping is set to be invalid.

The overview of the optimisation process is summarised in Algorithm 3. It begins

with an initial guess of $\Phi_{init} = \mathcal{N}(\boldsymbol{\mu}_{init}, \boldsymbol{\Sigma}_{init})$. The covariance may be initialised in several ways. One naive approach is to set it to the identity matrix, which assumes independent parameter distributions with equal variance. Another approach is to initialise the diagonal entries of the covariance matrix with small random values drawn from a normal distribution (e.g. N(1, 0.1)) and set the off-diagonal terms to zero. At each iteration, M sets of environment parameters, $\boldsymbol{\phi}$, are sampled from the current distribution Φ. If a sampled parameter is not within physically valid ranges, it is resampled. Each $\boldsymbol{\phi}$ creates a unique simulated environment, where the robot executes the demonstrated trajectory and obtains the torque sensor feedback, $\boldsymbol{\tau_s}$. $\boldsymbol{\tau_s}$ is compared to N real-world torque trajectories to compute the cost $\mathcal{J}(\boldsymbol{\phi})$ as shown in Eqn. 5.1. Each $\boldsymbol{\phi}$ is hence assigned a cost. CMA-ES then updates Φ from the x best $\boldsymbol{\phi}$ with the lowest costs. The update process of Φ (*i.e.*, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) and hyper-parameters of CMA-ES follow the standard procedure. These steps repeat until Φ converges to Φ^*.

Algorithm 3 Optimising parameter distribution

Initialise hyper-parameters of CMA-ES
Initialise Φ with $\mathcal{N}(\boldsymbol{\mu}_{init}, \boldsymbol{\Sigma}_{init})$
while not converged **do**
    for $m = 1 : M$ **do**
        Sample $\boldsymbol{\phi_m}$ from Φ
        Robot perform task in simulation with $\boldsymbol{\phi_m}$
        Collect $\boldsymbol{\tau_s}(\boldsymbol{\phi_m})$
        Calculate $\mathcal{J}(\boldsymbol{\phi_m})$ in Eqn.5.1 with N real trajectories $\{\boldsymbol{\tau_r}^1(\phi'), ..., \boldsymbol{\tau_r}^N(\phi')\}$
    end for
    Select x best $\boldsymbol{\phi}$ from $\{\boldsymbol{\phi_1}, ..., \boldsymbol{\phi_M}\}$ by $\min \mathcal{J}(\boldsymbol{\phi})$
    Update $\Phi = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the selected $\boldsymbol{\phi}$ set
    Update hyper-parameters of CMA-ES
end while
return optimised Φ^*

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

Figure 5.2: (a) and (b) show the hardware setup in the real world and in simulation. The AruCo markers attached to the table and the cabinet are used for tracking purposes. (c) shows four different locations of the doorknob, each separated by $5cm$ along the lever arm of the door. (d) shows three variants of the door. From left to right are the original door, the door with one spring and the door with two springs attached to the hinge. The springs were used to emulate doors with different dynamics.

5.2.3 Policy Learning

In this chapter, we adopt a RL framework to learn an optimal control policy π_θ, parameterised by θ, through a policy gradient. This is formulated as a MDP which is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : p\left(s_{t+1}|s_t, a_t\right)$ is a distribution of state transitions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, ρ_0 the initial state distribution and $\gamma \in (0, 1)$ is the discount factor. The optimal policy π_θ^* aims to maximise the cumulative reward over episodes:

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_t r\left(s_t\right)\right]. \tag{5.2}$$

Proximal Policy Optimisation [121] is deployed for robot learning purpose. It updates the policy by using the surrogate objective:

$$L(\theta) = E_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \tag{5.3}$$

where, $\hat{A}_t$ is the estimate of the advantage function at timestep t, and $r_t(\theta)$ denotes the ratio between the current policy and the previous policy. The clipped term keeps the ratio inside the interval $[1-\epsilon, 1+\epsilon]$, and the minimum of the clipped and unclipped terms is used for the expectation. This provides a lower bound, or a

pessimistic bound, on the unclipped term.

As the model of the real-world environment is described by Φ^*, sufficient simulation environments are hence required to be sampled from the distribution in order to reflect the reality. Therefore, the RL agent is trained on multiple simulated environments sampled from Φ^*. By doing so, we hope to maximise the similarity in state transition between the simulated and real environments and enhance the transferability of the learned policy to the real-world application.

The structure of the reward function follows closely to the one defined in [137] and is presented as follows:

$$r = \begin{cases} \omega_1 \cdot r_{door} + \omega_2 \cdot r_{ori} \\ \quad +\omega_3 \cdot r_{dist} + \omega_4 \cdot r_{log\_dist} + \omega_5 \cdot r_{slip}, \text{ if } \lambda < 30°, \\ \\ \omega_1 \cdot r_{door} + \omega_2 \cdot r_{ori} + \omega_5 \cdot r_{slip}, \text{ otherwise} \end{cases} \tag{5.4}$$

where λ is the hinge angle of the door, ranging from 0° (closed door) to 90° (completely opened door). The reward function has five terms. The r_{door} rewards the action that results in the increase to the door hinge. The r_{ori} rewards the relative orientation between the doorknob and gripper. A higher reward is given if the relative orientation is closer to orthogonal. The r_{dist} and r_{log_dist} are associated with the relative displacement between the doorknob and gripper. A higher reward is given for the short displacement. One major difference between our door-opening strategy and the DoorGym's lies in that they use hooks to open the door while our robot is trained to firmly grip the knob handle during the entire task. This significantly increases the complexity of the dynamics involved. Therefore, an additional penalty term r_{slip} is added in the reward function to prevent the fingers from slipping. $\omega_1, \omega_2, \omega_3, \omega_4$, and ω_5 are five coefficients for normalising each reward term.

5.3 Experiments and Results

We evaluate our work through a door-opening task involving dynamics. The main focus of our experiments is to assess how well the RL policy learned with DROID can be transferred to the robot in the real world. This evaluation consists of three experiments.

The first experiment validated that DROID can identify parameter distributions for environments with different dynamics. In the second part, we trained a policy for the door-opening task with DR, but the randomised ranges for parameters are governed by the optimised parameter distributions. The evaluation of the policy's performance was done by comparing its robustness and effectiveness in sim-to-real transfer with other approaches (standard DR and without DR). Finally, we tested the generalisability of the RL policy learned from the optimised parameter distributions.

5.3.1 Experimental Setup

The experiment consisted of a real and a simulated system. In the real system, a cabinet door, a camera and a 7-DoF Franka Emika robot arm were used, as illustrated in Fig. 5.2(a)(b). The Franka robot was equipped with 7-DoF joint torque sensors allowing it to record torque feedback while interacting with the environment. A two-finger gripper was mounted at the end effector for grasping and manipulation purposes. The cabinet was the target of interest in which we attempted to identify its dynamics parameter distributions. The fixed camera provided the relative pose information between the robot and the door and the measurement of the door angle, which can be obtained via tracking the visual markers attached to the door and optical table during the experiment.

For the simulation part, the MuJoCo platform [132] was deployed to perform the RL training. The simulated environment was defined by kinematic and dynamic parameters. The kinematics included a kinematic tree that defined the relative poses between objects and object geometries. The kinematics and geometries of the robot were provided officially while the kinematic parameters of the cabinet were measured explicitly. For the dynamics parameters, the approximations of some robot's dynamics parameters were officially provided, such as the robot's joint friction coefficients, and the gripper's damping coefficient. Dynamics parameters such as the masses of the door and knob were explicitly measured. The inertia of the door and the robot were estimated from their CAD models. Other dynamics parameters, such as the robot's joint damping coefficient, gripper's friction coefficient, door joint's friction loss, damping, and stiffness coefficients were unknown and difficult to measure. In the experiment, the distributions of these dynamics parameters were identified. Here, the joint friction loss of the door refers to the energy lost due to friction as it rotates. The joint damping coefficient relates to the dissipation of energy due to viscous friction and sliding and torsional frictions describe the frictions between two parallel surfaces and two coaxial surfaces. Note that in the experiment, we did not evaluate the performance of the proposed method by comparing the identified parameters with their true values, instead, we evaluated the similarity of the robot behaviours in simulation and reality.

5.3.2 Parameter distribution identification

In the first experiment, we evaluated the identified distributions using DROID. This verified whether DROID could find the distribution of parameters that reflects the interaction behaviour encountered in the real-world door-opening task.

For the given cabinet door (Fig. 5.2), the real robot followed the human demonstration to obtain ten sets of $\boldsymbol{\tau_r}$ that reflected the dynamics of the interactions with the

door. We first made an initial guess of the Φ_{init} for the parameters of interest. The estimation was made by referencing Franka's officially provided values and DoorGym's parameters [137] with the exceptions being the door and knob masses which were directly measured. The initial covariance matrix was set to be a diagonal matrix with small values, which are shown in Tab. 5.1 under $diag(\Sigma_{init})$. At the first iteration, 30 simulated environments were sampled from the initial distributions. Parameters sampled outside the physically valid ranges were resampled. With each environment, the robot cloned the human demonstration by following $\boldsymbol{q_d}$ to obtain $\boldsymbol{\tau_s}$. The associated cost for each trial was calculated using Eqn. 5.1. The higher the discrepancy in the torques, the higher the cost. For the simulation that failed to successfully open the door due to factors like grasp slipping, an extra penalty of 10 was added to the cost. CMA-ES algorithm took the top five best candidates of $\boldsymbol{\phi}$ to update the means and the covariances and hence the Φ. In a new iteration, the updated Φ was used to generate another 30 new simulations and this process was iterated until convergence.

Through the optimisation process described above, we have estimated the statistical distributions of the parameters for the real settings (Tab. 5.1) based on 10 independent trials. The table shows the mean and standard deviation results for the initial guess and final optimised distributions for each parameter. It is important to note that the goal of this work is not to recover the true values of the dynamics parameters themselves but rather to determine appropriate parameter distributions such that the resulting simulated environment exhibits similar quasi-static interaction to the real world. Fig. 5.5 illustrates the optimisation process and results. Fig. 5.5(a) shows the joint torque trajectories of the robot obtained in the simulation before and after the optimisation and the joint torque trajectory obtained in the real world. This only displays one of the joints for illustration. As can be seen, the differences between the red (after optimisation) and the black (real robot) lines are much smaller than the differences between the blue (before optimisation) and the black. This sug-

gests that the proposed approach can indeed minimise the reality gap. The same conclusion can be drawn from Fig. 5.5(c), which plots the cost against the iteration when identifying parameter distribution using CMA-ES. After 30 iterations, the average cost gradually converged to a lower value. This indicates that the simulations sampled from the optimised distribution lead to a smaller reality gap than those sampled from the initialised distributions. The means and the variances of three parameters at each iteration are shown in Fig. 5.5(b), and this shows the distribution of these parameters converged to fixed ranges. The quantitative results comparing the unoptimised and optimised distributions are summarised in Tab. 5.1. We have applied this result to the sim-to-real transfer experiment, which is detailed in the next section.

Furthermore, validation was carried out to determine the variations in estimating the parameter distributions using a different human demonstration. A different human demonstration was provided with an alternative robot pose as illustrated in Fig. 5.3 for validation. Note that due to the limitations of the robot workspace, only the two presented poses could be applied to interact with and open the door. The parameter distribution identification process was repeated using this pose. The experimental result comparing the estimated parameter distribution using the two human demonstrations is shown in Fig. 5.4. As can be seen there are some minor variations across different parameters. We attribute the reason to using the sampled-based derivative free CMA-ES optimisation approach, which gave the results in different local optima. Despite the differences, most parameter distributions estimated using the alternative pose converged to similar distributions. Hence, we showed that a single human demonstration may be sufficient for parameter estimation.

To further verify the approach, we have conducted two sets of controlled experiments by changing the dynamics of the real door.

The original door was modified to include additional springs to change its dynamics.

(a) (b)

Figure 5.3: Two different initial robot poses were used in the human demonstration. (a) is the initial robot pose used to estimate parameter distribution in Tab. 5.1. (b) is another initial robot pose used to validate the possibility of a bias in parameter estimation.

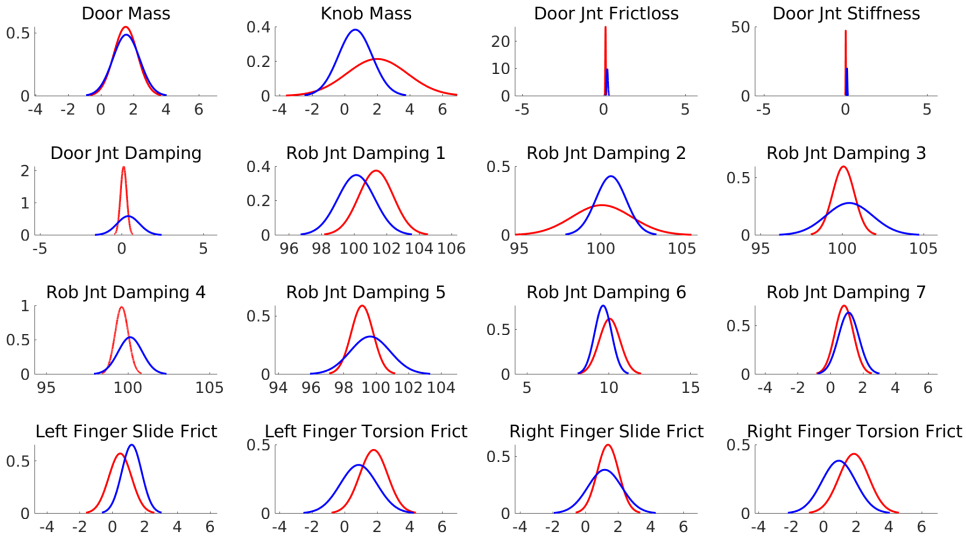


Figure 5.4: This compares the parameter distributions estimated using two different initial poses and human demonstrations. Red and blue curves represent the parameter distribution estimated using robot poses (a) and (b) shown in Fig. 5.3.

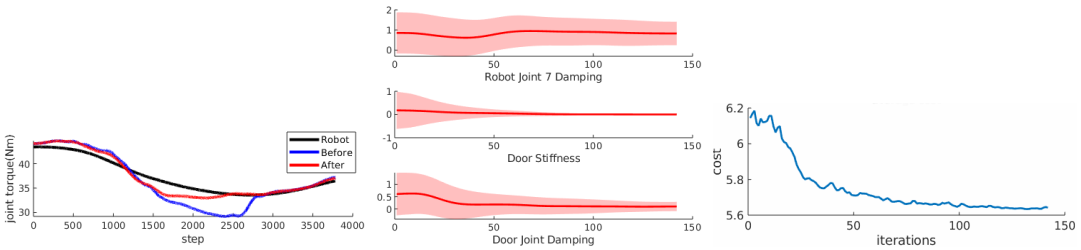
Fig. 5.2(d) illustrates the three variants of the door, without any spring, with one spring and with two springs attached to the hinge. These springs were identical and shared similar dynamics.

The optimisation results for different door dynamics are also summarised in Tab. 5.1. The robot dynamics were not much affected, but the environment dynamics varied a lot. This is because we only changed the door dynamics, and the algorithm was able to identify the change correctly. Doors equipped with different amounts of springs would expect to have different distributions for the door joint friction loss, stiffness and damping. For the friction loss, the door with two springs was estimated to have higher friction loss than the other two scenarios. The joint stiffness increased with the number of springs. However, some anomalies were observed. For example, higher joint damping values and knob masses were estimated when the doors were equipped with springs compared to the door without the spring. The results differed from expectations. We attribute this discrepancy to the fact that we were dealing with an undetermined system, where we attempted to estimate more parameters from just 7 DoF torque trajectories. Fully recovering the true dynamics parameters in this case is a very challenging problem as it results in non-unique solutions. While the estimated parameter values did not truly reflect reality, our finding shows that the proposed approach could still detect changes to the dynamics via variations in the estimated parameter distributions. Overall, the proposed framework still showed reasonable results and can effectively capture differences in the changes of dynamics in reality.

Based on the above results, the proposed DROID framework has demonstrated feasibility in determining the parameter distributions of the real-world dynamics for sim-to-real transfer. In theory, the reality gap is smaller after optimisation. The framework has also been shown to be feasible for identifying tasks with different dynamics. Most outcomes have appeared to be reasonable with a few exceptions,

Table 5.1: This compares the initial and final parameter distributions after optimisation and identification. A distribution is defined by μ and σ. Additionally, this also compares the optimised distributions for doors equipped with one or two springs.

	DoorGym		Door without spring		Door with 1 spring		Door with 2 springs	
	μ_{init}	$diag(\Sigma_{init})$	μ_{opt}	$diag(\Sigma_{opt})$	μ_{opt}	$diag(\Sigma_{opt})$	μ_{opt}	$diag(\Sigma_{opt})$
Door Properties								
Door Mass(kg)	1.144	0.5	1.50±0.67	0.73±0.30	0.92±0.60	0.52±0.15	2.54±0.60	1.23±0.15
Knob Mass(kg)	0.199	0.1	1.98±0.88	1.86±0.59	2.31±0.64	0.42±0.16	3.99±0.64	1.29±0.16
Friction Loss	0.05	0.025	0.10±0.25	0.02±0.02	0.0±0.05	0.0±0.0	0.56±0.05	0.05±0.01
Joint Stiffness	0.01	0.005	0.002±0.55	0.008±0.03	1.06±0.35	0.0±0.0	1.24±0.35	0.06±0.0
Joint Damping	2.0	1.0	0.10±0.26	0.19±0.08	0.71±0.20	0.01±0.02	0.6±0.20	0.18±0.02
Robot Properties								
Joint Damping (7DoF)	[100, 100, 100, 100, 100, 10, 0.4]	[2.0, 2.0, 2.0, 2.0, 2.0, 1.0, 0.2]	[101.35±1.42, 100.06±0.56, 100.08±2.08, 99.61±0.23, 99.14±2.20, 10.05±0.80, 0.83±0.29]	[1.06±0.35, 1.83±0.55, 0.67±0.36, 0.41±0.27, 0.68±0.12, 0.64±0.25, 0.57±0.27]	[98.11±0.26, 98.69±0.69, 100.38±0.39, 99.64±0.71, 101.01±0.38, 9.72±1.08, 1.17±0.29]	[0.41±0.17, 0.63±0.23, 0.65±0.14, 0.15±0.07, 0.96±0.12, 0.04±0.02, 1.19±0.31]	[98.63±0.26, 99.28±0.69, 95.83±0.39, 100.12±0.71, 95.71±0.38, 11.55±1.08, 1.54±0.29]	[1.23±0.17, 0.71±0.23, 1.42±0.14, 0.81±0.07, 0.77±0.12, 0.46±0.05, 0.67±0.31]
Gripper Properties (Left Right)								
Sliding Friction	[0.5, 0.5]	[0.25, 0.25]	[0.47±1.07, 1.78±0.78]	[0.70±0.15, 0.86±0.20]	[1.37±0.64, 0.76±0.85]	[0.35±0.14, 0.39±0.08]	[3.04±0.64, 2.68±0.85]	[0.46±0.14, 0.54±0.08]
Torsional Friction	[0.5, 0.5]	[0.25, 0.25]	[1.38±0.24, 1.84±0.19]	[0.66±0.13, 0.92±0.05]	[0.79±0.66, 2.29±0.99]	[0.42±0.07, 0.80±0.17]	[1.04±0.66, 2.0±0.99]	[0.72±0.07, 0.83±0.17]



(a) (b) (c)

Figure 5.5: (a) shows the comparison between a robot's joint torque trajectory in the real world and the joint torque trajectories before and after the optimisation in simulation. (b) shows the changes in means (the solid lines) and the variances (the shaded areas) of the three parameters over iterations during optimisation. A red line represents the mean, and a shaded region represents the variance. (c) shows the cost over iterations when using CMAES to optimise the parameter distributions.

which may be attributed to factors such as imperfect demonstration resulting in loss of grip or noises present in joint torque sensors in the real world. We have used the optimised result in sim-to-real practice and detailed it in Section 5.3.3.

5.3.3 Sim-to-real transfer with optimised DR

In many prior works, DR methods have been shown to be effective in addressing the reality gap. Selecting the randomisation ranges remains challenging, and training the RL agent on overestimated ranges may be inevitable and could lead to poor learning performance. In the last experiment, we successfully identified the parameter distributions, but we have yet to demonstrate that the proposed method is

effective in learning adequate policy and transferring it to the real world. In this experiment, we compared the learning performance of policies learned from three methods, namely normal DR, DROID without DR and DROID with DR. These policies were trained to open the door without springs in the simulation and then directly transferred to the real world to open the same door. The normal DR approach was trained on the estimated parameter distributions shown on the second and third columns in Tab. 5.1. DROID without DR, similar to SI, was trained using only fixed parameters. In this case, the parameters used were the optimised $\boldsymbol{\mu}$ listed in the fourth column of Tab. 5.1. Finally, DROID with DR was trained based on the optimised distributions, i.e. fourth and fifth columns of Tab. 5.1.

For the sake of fairness, all of them were trained using a model-free on-policy approach, *i.e.* PPO, and shared identical RL hyperparameters, architectures of networks, observations, and reward functions, with the parameter ranges being the only differences. In the RL setting, the observation included the 7 DoF joint positions, velocities and torques of the arm, and the 3 DoF relative positions between the robot gripper and the knob. These combined formed a 24 DoF state space. The action was continuous and the action space consisted of 9 DoF, specifying the 7 robot joint positions and 2 gripper widths. The critic and actor models were represented by a neural network with two hidden layers of size 64, following Tanh activation functions. We performed our simulations on the MuJoCo physics engine with a timestep of 0.001s. During the training, each episode took a maximum of 512 steps. We employed the ADAM optimiser with a step size of 0.001 and a mini-batch of 64 episodes to update the policy and value networks.

Three sets of policies using different sim-to-real techniques were trained: a typical domain randomisation technique, system identification technique, and the DROID technique. Each of these sets was trained and evaluated in the real-world for 10 times. The results of these sim-to-real techniques are compared and reported in

Tab. 5.2. This table compares three metrics to evaluate the identified parameters and the performance of the three techniques in both simulation and reality: success rate, maximum door opening angle achieved, and the number of steps required in the door opening procedure. The success rate was selected as a metric for evaluating the sim-to-real gap because, in the absence of a reality gap, the policy's performance in both simulation and reality should be comparable. By measuring the discrepancy in success rates between the two environments, we can determine the size of the gap. The door opening angle was considered as another evaluation metric. As the hinge angle increases, the system's dynamics become more important in the door-opening process. A better sim-to-real technique should demonstrate a larger door opening angle. Finally, the number of steps taken to open the door was examined, specifically, we focused on the door angle opened per step. When the sim-to-real gap is small, the performance in reality should resemble that in the simulation.

The door-opening angles were recorded, and the histogram is shown in Fig. 5.6. We define a case to be successful when the hinge angle of the door is larger than 30° in a real-world trial. From these results, we can see the latter two approaches (DROID with DR and without DR) outperformed the standard DR. In the DROID with DR setting, the optimised means and variances for DR (80% success rate) can achieve significant advantages in real-world tests over the normal DR (20% success rate) which used initialised means and variances. We also notice that the policies trained without DR, which used optimised means μ_{opt} as system parameters in simulation, can achieve the highest success rate of 86.7% in reality. It is even higher than the DROID with the DR method. This suggests that the optimised μ_{opt} can achieve a good sim-to-real transfer. The fact that DROID without DR results in a higher success rate, but lower door-opening angles than DROID with DR is interesting. We attribute this to the relatively weak dependency on system dynamics when the door was opened with small angles (*e.g.*, 30°).

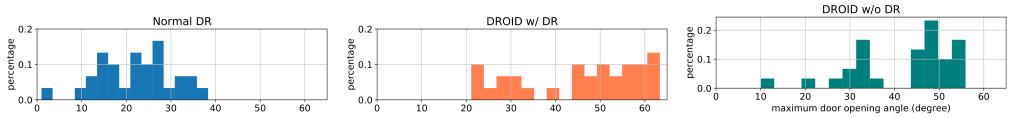


Figure 5.6: Comparison of the door-opening performance in reality for three methods (from left to right): normal DR, DROID with DR and DROID without DR. The horizontal axis is the maximal opening angle of the door in degrees. The vertical axis is the percentile value for each bin. It compares the distributions of maximal angles over 30 runs for each method.

Table 5.2: Results comparisons for three different Sim-to-Real techniques, namely Domain Randomisation, DROID without DR and DROID.

		DR (μ_{init}, Σ_{init})	DROID w/o DR (μ_{opt})	DROID w/ DR (μ_{opt}, Σ_{opt})
Success Rate (angle>30°)	sim	**100**%	**100**%	**100**%
	real	20%	**86.7**%	80%
Open Angle (mean±std)	sim	**91.3** ± 0.4	70.8 ± 16.0	91.2 ± 0.2
	real	22.4 ± 8.2	42.2 ± 11.1	**45.4** ± 13.6
Num. Steps (mean±std)	sim	96.3 ± 17.2	60.3 ± 2.6	**38.7** ± 14.1
	real	68.7 ± 8.4	73.6 ± 7.0	**68.3** ± 11.0

As for the door angle opened, we show that DROID with DR can achieve the door opening with more concentration on larger maximal hinge angles, *e.g.* around 60°, compared to normal DR and DROID without DR. Tab. 5.2 also demonstrates that DROID with DR can achieve the highest average door-opening angle among all three methods, even though it did not perform as well as the normal DR method in simulation. This suggests that the inclusion of optimised parameters distributions in DROID with DR allows the policy to better adapt to real environments.

In terms of door angle opened per step, our results show that DR achieved an average of 0.95° per step in simulation and 0.33° per step in reality, while DROID without DR achieved an average of 1.17° per step in simulation and 0.57° per step in reality. In comparison, DROID obtained an average of 2.35° per step in simulation and 0.66° per step in reality. This showed that its performance was relatively more consistent across both environments.

5.3.4 Generalisation of the Learned Policy

As mentioned in the previous section, we only used one human demonstration to determine the parameter distribution. This demonstration was, however, not used during RL because it could only serve a door with the same dimensions. In this experiment, we tested the policy with three additional doorknob locations in the simulation and in the real system, as shown in Fig 5.2(c), to emulate doors with different dimensions. These locations were set to be $5cm$ apart along the lever arm of the door. With DROID, the trained policy was able to pull the doorknob at different locations and open the door successfully. As such, we have verified that once the parameter distributions are identified using a human demonstration, different RL policies can be trained within the optimised distributions to accomplish tasks other than the one demonstrated.

5.4 Conclusions

In this chapter, we propose a novel and generic framework, DROID to minimise the reality gap between simulated and real environments. The approach is designed to be applicable to a range of contact-rich manipulation tasks, such as door opening. By executing a human demonstration trajectory in both simulation and reality, the differences in their dynamics can be minimised by iteratively updating and identifying the distributions of the real dynamics parameters. Using a door-opening task as an example, we have verified its capability to identify reasonable parameter distributions and thus reduce the reality gap. A successful RL policy can then be obtained by training in this distribution and directly transferring to the real world. The sim-to-real performance has been shown to be superior to training with typical DR and SI approaches. Finally, we also demonstrated that a generalised RL policy could be trained to accomplish different tasks, given that the system dynamics remain unchanged.

Chapter 6

Minimising the Reality Gap in Tactile Based Learning

In Chapter 5, we showed that simulation provides an efficient alternative to the real world for data collection. In this chapter, we further explore the possibility of simulation. Humans use tactile feedback to reason about an environment's state and achieve precise manipulation. Many tactile sensors are therefore developed for a robot to learn human-like manipulability. This chapter introduces a novel way for a robot to learn precise manipulation without explicit human demonstration, using a proposed tactile-based simulator. A robot learns manipulation skills through an expert-student framework, where the expert acquires its policy based on a human-designed controller and engineered objective function, and the student, i.e. the robot, then imitates the expert and later learns through self-exploratory training.

An introduction to tactile simulation is detailed in Section 6.1. The tactile simulator and policy used to learn a manipulation task are then described in Section 6.2 and 6.2.4. Section 6.3 is the experiments and results. Lastly, Section 6.4 presents the summary of the work.

6.1 Introduction

In an effort to achieve human-like manipulation, tactile sensors have been studied extensively in the robotics community. And in recent years, robot learning techniques, such as RL and IL, have been used for tactile-based policy learning. However, the reliance on large amounts of data means that these methods often require training to be done in simulation, followed by sim-to-real transfer.

However, simulation of dynamics is challenging [138], and particularly so for tactile sensors. For example, a tactile sensor that relies on an incompressible conductive fluid as a sensing medium may require modelling the fluid behaviour. As another example, a tactile sensor with a deformable surface [83] is challenging to simulate since most widely available simulators for robot learning, such as Bullet, MuJoCo etc., are typically designed for rigid body simulation. Most importantly, existing sensors are typically not designed with simulation in mind, and instead are designed to maximise the reliability of real-world tactile data. We argue that, in the era of data-driven methods where sim-to-real is so important, tactile sensors should be designed specifically with sim-to-real in mind.

To this end, in this paper, we introduce a new tactile sensor that is designed specifically to be easy to simulate. The sensor consists of an array of taxels that move along a single axis following an external force. This allows for easy simulation by modelling each taxel as an independent mass-spring-damper system, rather than requiring the full deformation of the entire sensor to be jointly modelled. With this sensor, we then studied a range of different methods for achieving sim-to-real transfer to determine the most appropriate strategy for the new sensor. These include kinematics and dynamics calibration and learning a latent space that is invariant across simulation and reality. Finally, we evaluated these methods on a real-world grasping task, with policies trained purely in simulation using RL and transferred

directly to the real world.

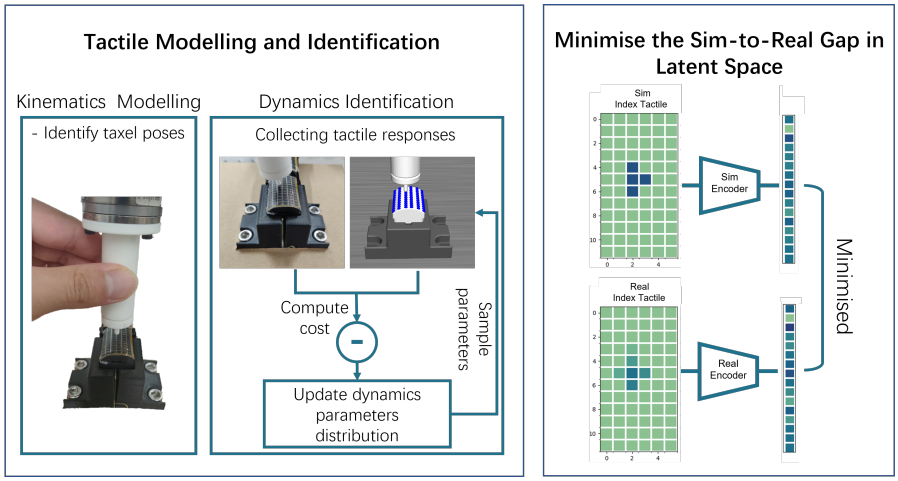


Figure 6.1: Our novel tactile sensor, with an overview of the best-performing sim-to-real strategy from our experiments. In tactile modelling, we first measure the pose of each taxel. This allows us to construct a tactile model, where each taxel is described using a mass-spring-damper system. We leverage an evolutionary strategy to identify the dynamic parameter distribution of each taxel. The distribution forms the DR ranges used to train a control policy. During policy learning, we learn the embedding of the simulated tactile signals and reduce the 12-by-6 dimensions into a latent space with a dimension of 16. To cross the sim-to-real, we then minimise the difference in the embedding of the simulated and real tactile signals.

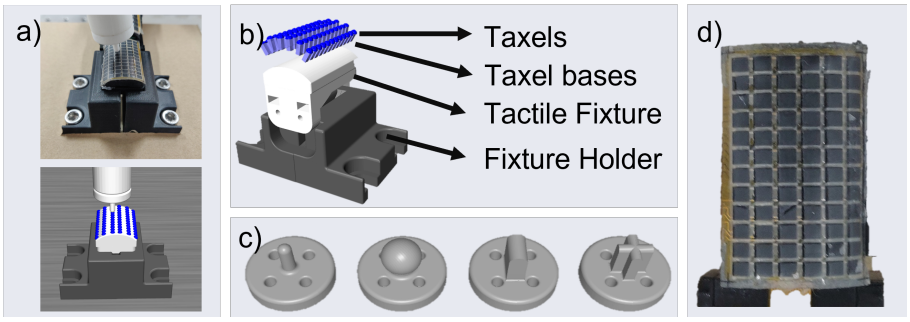


Figure 6.2: a) Tactile calibration setup in simulation and reality b) Tactile sensor model c) Types of indentors used to indent and calibrate the tactile sensors in simulation and reality. The indentors include a small sphere, a large sphere, a flathead and a cross d) A real tactile sensor consists of 12-by-6 taxels in array form.

6.2 Methodology

6.2.1 Tactile Sensor

The tactile sensor is in the form of a flexible sheet. It consists of 12x6 piezoresistive elements arranged in an array form developed in-house as shown in Fig. 6.2(d). Each element, or taxel, can be schematically represented as an independent variable resistor, with the resistance being inversely proportional to the force acting on it. The acquisition circuit measures the current through each taxel at a rate of 200 Hz. During initialisation, a baseline value is acquired and used to offset all subsequent measurements.

From top to bottom, there is a tactile sensor, a curved rubber pad, and a 3D-printed rigid base, i.e. the tactile fixture. These parts are aligned and assembled by hand. The pad is deformable and can emulate the soft finger pulp of a human's hand. The pad deforms upon contact to maximise the overall contact area and intensify tactile signals.

6.2.2 Tactile Sensor Modelling in Simulation

The pad is deformable and can emulate the soft finger pulp of a human's hand. It deforms upon contact to maximise the overall contact area and intensify tactile signals. Each taxel can sense independently in reality and is modelled as a discrete unit in simulation. The contact behaviour of a taxel is governed by a mass-spring-damper system, and the reason is twofold. Firstly, it allows the deformation of the tactile sensor and pad to be simulated in a rigid-body simulator. Secondly, it allows different contact behaviours to be defined with different dynamics parameters of a system. To build the sensor model, we perform kinematics calibration and dynamics calibration. Kinematics calibration determines the pose of each taxel relative to the tactile fixture, so the kinematic model of the tactile sensor can be constructed in simulation. Dynamics calibration is then performed for each taxel. It determines the distribution of dynamics parameters associated with each mass-spring-damper system. It is done by performing an indentation action on a taxel in both simulation and reality. The indentation trajectory is offline-planned using the pose of each taxel obtained from the last step. To ensure appropriate excitation of the tactile sensor for calibration, we conducted a preliminary experiment to understand the speed at which the robot finger contacts objects during grasping. We then used a similar speed to execute the Franka robot for calibration. Given a set of dynamics parameter distributions, when the difference in a taxel's real and simulated responses under the same contact action is minimised, a taxel is calibrated.

Kinematics Calibration Kinematics calibration is performed to account for the misalignment between the tactile sensor and the rubber pad during the assembly and to register the pose of each taxel to the tactile fixture. This step is essential to determining the actual sensing points of the tactile sensor in simulation. A calibration platform, as illustrated in Fig. 6.2(a), is constructed in real to explicitly

measure and determine the poses of taxels, using a Franka robot as the measuring tool. In this setup, a robot arm is equipped with a 3D-printed spherical indentor, and the tactile fixture is secured by a fixture holder. Both the robot base and the holder are fixed on an optical table.

To measure the pose of a taxel, the robot end-effector is hand-guided to touch the tip of the indentor to a taxel. Using the robot arm as a measuring tool, we can record and calculate each taxel's pose relative to the robot base and tactile fixture. Since the 12-by-6 taxels are uniformly arranged, we do not have to measure every single one. As shown in Fig. 6.2(d), we only measure the poses of taxels at rows 1, 5, 9, and 12. The poses of the remaining taxels are determined by interpolation between these measured rows. Multiple measurements are taken for each sampled taxel, and averaged to reduce errors.

Dynamics Identification While all the taxels share the same sensing mechanism and are manufactured in the same way, each of them possesses slightly different sensitivity and deformation. It is necessary to determine a set of dynamics parameters associated with the mass-spring-damper system of each taxel that describes the deformation and tactile response in simulation. Since the identification of the real dynamics parameters is very difficult, our goal is instead to determine an optimal dynamics parameter distribution for each taxel. It is identified, so the responses of a taxel in simulation and reality are similar when undergoing the same indentation action. The distribution will be later used during policy learning. It serves as DR ranges for the policy to be trained on, improving its adaptation to different environments.

In MuJoCo simulation, the tactile sensor is modelled as illustrated in Fig. 6.2(b). A taxel is connected to its base via a sliding joint. A base is positioned underneath a taxel inside the rubber pad. When a taxel is pressed, it is brought into contact with

its base, and the contact force is generated. Mujoco supports a touch sensor, which calculates and measures the sum of normal forces from contacts made between two geometric bodies. This sensor is placed on a taxel to measure the contacts made between the taxel and its base. The sliding joint is to allow for deformation when a taxel is pressed. The parameters associated with the joint can be used to emulate different forces required to press a taxel or release a taxel from its base. These also describe different taxel sensitivity. The deformation behaviour associated with a system is described by the four parameters, namely the mass of each taxel, stiffness coefficient, damping coefficient, and the displacement between the taxel and taxel base along the sliding direction.

The goal of dynamics identification is to indent the same taxel in simulation and reality, and use the differences in their tactile responses to estimate the suitable parameter distributions to describe the dynamics of this taxel. To achieve this, we leverage the same real-robot setup as in the previous kinematic calibration step, which is depicted in the top image of Fig. 6.2 a). The setup in the simulation is constructed by explicitly measuring the relative pose between the robot base and tactile fixture and using the officially provided Franka model. By combining this setup with the tactile sensor model described above, we have the same experiment setup in simulation as in reality, with the only difference being the dynamics parameters of the tactile sensor. Given the setup and the identified pose of each taxel, we then plan a robot indentation trajectory. This brings the indentor to the top of a taxel, with its tip being orthogonal to the taxel's surface and presses on a taxel by 1mm$\pm$0.1mm with the indentor before retreating.

To identify the parameters for a taxel, we adopt the stochastic and gradient-free approach - CMA-ES [53]. This approach iteratively looks for the optimal parameter distributions that best describe the contact behaviour of a taxel in reality and is done repeatedly for each taxel. Given a taxel, the identification process starts with

the initial guess of a parameter distribution $\Omega_{init} = \mathcal{N}(\boldsymbol{\mu}_{init}, \boldsymbol{\Sigma}_{init})$, which is manually determined. For each iteration, N sets of parameters are sampled from the current distribution Ω. Each set of parameters constructs a simulation environment, and it describes the contact behaviour of a taxel. The indentation is carried out in simulation and reality by the position-controlled Franka robot using the offline-planned trajectory and a spherical indentor. During indentation, the time-indexed taxel responses, $\mathbf{S_{real}}$ and $\mathbf{S_{sim}}$, are recorded and normalised using the maximum values. A taxel response in simulation comes from the measurement of a touch sensor attached to each taxel, and a taxel response in reality comes from the changes in current across the resistor of the taxel. Since simulation and reality measure taxel responses in different units, normalisation is performed. Using the two normalised time-indexed responses, the cost associated with a set of parameters can be calculated with Eqn. 6.1. The CMA-ES then updates Ω (*i.e.*, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$) and hyper-parameters from the b best set of parameters that have the lowest costs. A new set of parameters is then resampled from the updated Ω' for a new round of iteration. The process is iterated until convergence or a pre-defined maximum iteration is reached. This identification process is repeated for each taxel. The optimal set of $\Omega^* = \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ for each taxel forms the DR ranges. In later policy learning, an agent is trained on these ranges to increase its adaptability to different environments and minimise the reality gap.

The cost is defined by the following objective function, J:

$$
\begin{aligned}
&J = w_1 \cdot \textstyle\sum_{t=0}^{T_c} \|S_{sim,t} - S_{real,t}\| + w_2 \cdot \sum_{n=0}^{N} \mathrm{f}(p_n) \\
&\textit{where,} \\
&\mathrm{f}(p_n) = \begin{cases} \exp(2 \cdot p_n / p_n^{max}), & \text{if } p_n > p_n^{max} \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{6.1}
$$

where w_i is the weighing factor, t is the time index, T_c is the duration of contact, p_n is the n-th parameter, and $p_{max,n}$ is the upper limit for the n-th dynamics parameter. The second term of the objective function is to limit CMA-ES from sampling invalid values.

6.2.3 Learning The Embedding of Tactile Signals

The goal of learning representations of tactile signals is to isolate the most important features of the tactile signals while reducing the mismatch between simulation and reality for those key features. The high dimensional raw tactile data increases the complexity of the observation space for policy learning, and can enlarge sim-to-real mismatches, due to inconsistencies across multiple tactile modalities between simulation and the real world. Encoders are therefore employed to mitigate the reality gap by concentrating on the most crucial aspects of tactile signals. Two encoders are used to encode the tactile signals from simulation and reality to the same latent space. The reality gap is bridged by mapping the real tactile responses to simulated tactile embeddings.

Two encoders, the sim encoder and the real encoder, are introduced. The sim encoder extracts and encodes all the raw tactile signals $\mathbf{S_{sim}}$ from tactile sensor models in the simulation into a latent space. The models used are the calibrated models as described in the last step. The sim encoder is trained as a standard autoencoder. At training, the autoencoder first encodes all the raw tactile signals S_{sim} to the embedding, then reconstructs them to the same S_{sim}. The autoencoder is trained on the tactile signals obtained from exploratory data collection during policy training. During data collection, the dynamics parameters of tactile models in the simulation are randomised by sampling from the identified distributions obtained in the previous steps. In other words, the training data for the sim encoders comprises tactile responses from tactile models with different dynamic responses. During encoder

training, mini-batches of size 128 are sampled and used as the input and target of the autoencoder. The mean squared error between the input and reconstructed output serves as the loss function to update the autoencoder. In contrast, the real encoder is trained to encode S_{real} to the same latent space as the sim encoder, but via a supervised learning approach. It shares the same architecture as the sim encoder. The training data comprises tactile responses collected by indenting the real tactile sensor with various indentors, as shown in Fig. 6.2(c). This ensures different distributions of taxels are activated under an indentation, allowing the real encoder to generalise across different types and shapes of contacts. During data collection, the same indentations are performed in both simulation and real-world to obtain paired samples of $\mathbf{S}_{\mathbf{sim}}^{\mathbf{train}}$ and $\mathbf{S}_{\mathbf{real}}^{\mathbf{train}}$. During training, mini-batches of size 128 are sampled. The sim encoder encodes $\mathbf{S}_{\mathbf{sim}}^{\mathbf{train}}$ into $\mathbf{S}_{\mathbf{sim}}^{\mathbf{enc,train}}$, which serves as the target output for the real encoder. The real encoder encodes $\mathbf{S}_{\mathbf{sim}}^{\mathbf{train}}$ into $\mathbf{S}_{\mathbf{real}}^{\mathbf{enc,train}}$. The mean squared error between $\mathbf{S}_{\mathbf{sim}}^{\mathbf{enc,train}}$ and $\mathbf{S}_{\mathbf{real}}^{\mathbf{enc,train}}$ is calculated and used to update the real encoder.

6.2.4 Expert-Student Learning Framework

This section describes the expert-student learning framework used to learn the tactile-based policy for a grasping task.

An expert-student learning framework is a data-driven approach that combines model-based and model-free RL to learn a robotic grasping skill. Compared to solely model-based or model-free RL, this framework offers some advantages. First, it does not have to explicitly learn the dynamics of an environment, which can be very complex and subject to inaccuracy. Then, by leveraging an expert, an agent can accelerate the learning and thus shorten the training duration. This problem is formulated as a MDP defined by the tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}(s'|s, a)$ describes the transition from a state s and an action a to the next state s', $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and γ is a discount factor.

Our method consists of an expert policy, π_e, and a student policy, π_s. It draws some similarities to the prior works [17, 82] in that the expert policy leverages the privileged state, while the student policy only has access to the proprioceptive history. However, we modify the framework. The expert policy is implemented as a model predictive controller (MPC), which leverages the simulation as the dynamics model and a defined cost function to determine the optimal action to interact with the environment. The student is implemented as a model-free RL, which first imitates the expert and then improves upon itself through self-exploratory training. The student learns the optimal policy π_s^*, and selects an action that maximises the overall rewards over a trajectory based on R. Finally, only π_s^* is transferred to real. With the proposed expert-student framework, an optimal control policy can be quickly learned from the expert policy without exploring unnecessary state space. The student policy can further refine its learned policy based on the predefined reward.

The implementation of expert policy follows closely the model predictive path integral (MPPI) approach [142]. Without any training data, MPC can generate a good policy. However, it relies heavily on an accurate dynamics model to describe the transition of MDP and requires a much longer computation time to perform an iterative forward calculation at each step to determine an optimal action. Since we do not want to transfer the expert policy to reality, the simulator is used as the dynamics model. The MPC selects an optimal action based on the full state of the environment without adding noise. The state, s, of the task includes the pose of the hand, the joints of each finger, the raw tactile signals and the pose of an object. The grasp considered in this work is quasi-static and hence velocities are not included.

The expert policy is leveraged to increase the learning performance of the student policy. When executing the expert policy, the $\{o, a, o', r, d\}$ tuple is recorded and stored in the replay buffer of the student policy, hence guiding the student at the start of policy learning. The problem considered is a Partial Observation Markov Decision Problem (POMDP), where the student policy makes a decision based on noisy sensor readings. Prior works have studied extensively the POMDP and used the sequence of observations to determine an optimal action in RL [39, 52, 153] and hence the scope of this work will not delve into a detailed study and analysis of POMDPs in RL. In this work, the student policy takes history observation to determine an action. The observation, o, is saved instead of the state, s and o includes the history of s except that the raw tactile signals are converted to a latent representation by the sim encoder, and the rest is added with uniform noises for better sim-to-real transfer. The bounds to which the noises are sampled are determined empirically. o' indicates the next observation, which is based on s'. The robot's action, a, consists of the 3 DoF Cartesian position of the end-effector, the 8 joint positions of the fingers, and a binary ungrasp signal. The end-effector and fingers are position-controlled. If the ungrasp signal is true, the hand opens the fingers first before moving the arm to execute the grasp action. Here, r is the

reward associated with o and a, and d indicates whether the episode is done.

To select a good action at each step, the MPC first randomly samples M sets of an action sequence from the predefined action limits. Each action sequence has a length of N, representing the receding horizon, and is processed with a low-pass filter to ensure a smooth action. To determine the optimal action from M, each action sequence is run in a simulation starting with the same condition. This forward simulation is performed N times to calculate the receding state and the total score associated with each action sequence. The total score is the sum of the scores from each step in the receding horizon, and each score, c, is calculated based on the cost function. The best action sequence is computed by weighted averaged all the sampled action sequences using their associated score. It is done so that the actions with the lowest cost are weighted more. The most recent action in the best action sequence is then selected to interact with the environment. The $\{s, a, s', c, d\}$ tuple collected from the expert policy is converted to $\{o, a, o', r, d\}$ and stored in the replay buffer of the student policy as described earlier.

The student policy follows the standard Twin Delayed DDPG policy (TD3). The policy is initialised from the sets of tuples collected by the expert policy stored in its replay buffer. From thereafter, the student follows the standard procedure to perform its own exploratory training. During training, the raw, tactile signals in the simulation are converted to latent representations by the first encoder. At test time, the student policy is transferred to reality, and the tactile signals, in reality, are converted to latent representations by the second encoder.

6.2.5 Cost and Reward

The expert policy evaluates sampled actions based on a cost function, C, defined as follows:

$$
\begin{aligned}
&C = C_{tactile} + C_{object} + C_{hand} + C_{lift}^{object} + C_{lift}^{hand} \\
&C_{tactile} = \begin{cases} 5 & S'_{index} = 0 \text{ or } S'_{thumb} = 0 \\ -2 - \frac{1}{72}(S'_{index} + S'_{thumb}) & \text{otherwise} \end{cases} \\
&C_{object} = 100 * \|P_{object}^{init} - P_{object}^{curr}\| \\
&C_{hand} = 300 * (\|P_{object}^{curr} - P_{index}^{curr}\| + \|P_{object}^{curr} - P_{thumb}^{curr}\|) \\
&C_{lift}^{hand} = \begin{cases} 5 & P_{hand}^{curr} < P_{hand}^{init} \\ 0 & otherwise \end{cases} \\
&C_{lift}^{object} = \begin{cases} 5 & P_{object,z}^{curr} < P_{object,z}^{init} \\ -200 * (P_{object,z}^{curr} - P_{object,z}^{init}) & otherwise \end{cases}
\end{aligned}
\tag{6.2}
$$

where $S'_{index} = \sum_{i=1}^{72} S_i^{index} and S'_{thumb} = \sum_{i=1}^{72} S_i^{thumb}$ are the summations of all the index and thumb tactile signals respectively. P_{object}^{init}, P_{object}^{curr}, P_{hand}^{init}, P_{hand}^{curr} represent the initial and current position of the object and initial and current position of the hand. The subscript z indicates the position in the z-direction, i.e. an upward direction.

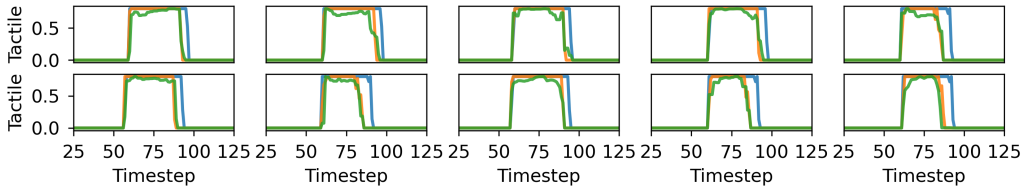
The first term, $c_{tactile}$, encourages more contact between the fingers and the objects for firm grasping. There are a total of 72 tactile signals on each finger. When contact occurs, the tactile signals from all the fingers are aggregated and scaled to

comparable ranges as other cost terms. Note that when all the tactile signals are activated, this cost term becomes zero. The second term, c_{object}, is to prevent the object from being moved too far away from its initial position during the grasping process. The third term, c_{hand}, encourages the fingers to grasp the centre of mass of an object and hence achieve a more stable and precise grasp. The fourth and final terms, c_{lift}^{hand} and c_{lift}^{object} are to encourage the hand to lift an object as high as possible until the end of an episode.

The student shares the common goal with the expert but instead selects an action that maximises the total rewards over a trajectory based on the reward function. The reward function is obtained from the cost function through negation and scaling. The additional scaling in the function further enhances the RL learning stability. The relationship between the reward and cost functions is as follow:

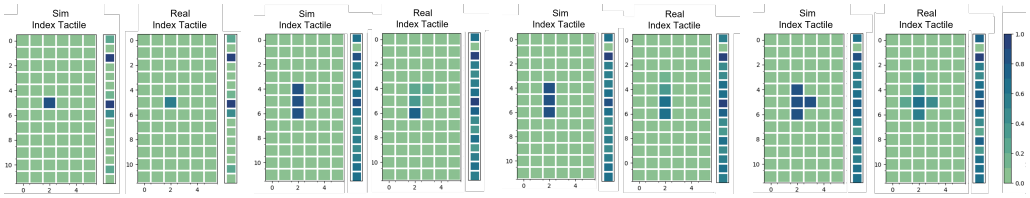
$$\mathcal{R} = 5 - C/10$$

6.3 Experiments And Results



(a)

Figure 6.3: Comparison of taxel signals from simulation and reality. This shows the taxel responses of the taxels on the first two rows of a tactile sensor. Each subplot shows and compares the simulated and real signals of a taxel collected from the same indentation. The green lines indicate the real tactile signals respectively. The blue and orange lines indicate the simulated tactile signals before and after calibration. The y axis represents the tactile signals, and the x axis represents the duration during indentation.



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

Figure 6.4: Comparison of taxel responses between the simulation and reality when the tactile sensor was pressed by different indentors. Each subplot shows a pair of 12-by-6 matrices indicating the location and intensity of the indentation and a 16-by-1 matrix indicating the corresponding latent representation. The raw and encoded tactile signals are standardised to 0 and 1 based on their maximum and minimum values. (a) small sphere indentor (b) large sphere indentor (c) flathead indentor (d) cross indentor.

Two experiments were designed to answer the following questions: 1) How well can the proposed tactile model simulate our tactile sensor? 2) How well can the simulation-learned policy perform in reality using the proposed tactile model? 3) How well can a grasping policy be learned with tactile and with different tactile representations? 4) Is it sufficient to cross the tactile sim-to-real gap in only tactile modelling, latent space or both?

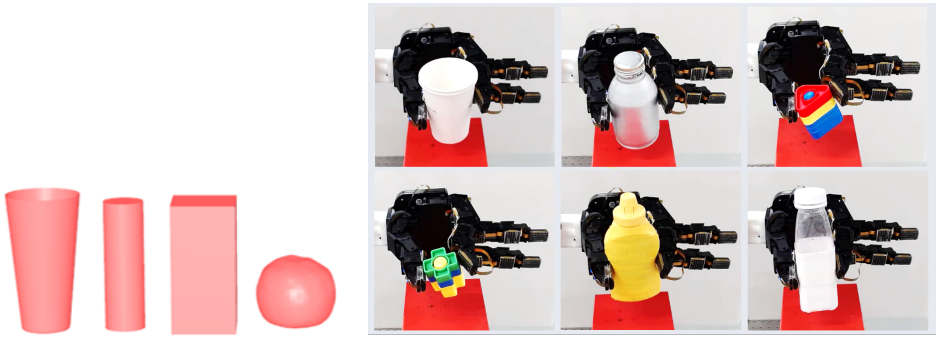
6.3.1 How well can our tactile model simulate the tactile sensor?

The tactile sensor model was evaluated by comparing the tactile responses in simulation and reality from undertaking the same indentation action. Fig. 6.3 shows examples of taxel responses when taxels were pressed by the small sphere indentor. The green lines indicate the real tactile signals. The blue and orange lines indicate the simulated tactile signals before and after calibration. Simulated tactile responses followed the real ones closely regarding the ascending time, maximum values, and contact duration, as shown in Fig. 6.3.

We also evaluated the embedding of the simulated and real tactile responses when the tactile sensor was indented by different indentors. The results are presented in Fig. 6.4, next to each 12-by-6 matrix. The colour bar indicates the tactile intensity. For each type of indentor, the same corresponding taxels in simulation and reality were activated. In raw signals, they differ by tactile intensity, and this may be attributed to factors like the nonlinear relationship between indentation and tactile signals or inaccurate modelling. By extracting and representing the raw signals in the embedding, the tactile signals from simulation and reality can be brought into a similar distribution and intensity.

6.3.2 Grasping Task

Our experiment considered a robotic grasping and picking task using a 7 DoF Franka robot arm equipped with a multi-DoF Allegro hand. Each finger of the hand has 4 DoF. Only the thumb and index fingers were used for grasping and picking purposes, as illustrated in Fig. 6.5. The fingertips of the index and thumb were replaced with our tactile fixtures covered with tactile sensors. The goal was to perform a precision grasp without using the palm to assist the grasp and pick up an object from a table.



(a) (b)

Figure 6.5: (a) are objects used for policy training and (b) are objects for testing.

An expert-student learning framework was adopted to train a policy. The expert policy was implemented using a model predictive controller, which leverages the simulation to select an optimal action to interact with the environment. The student was implemented as a Twin Delayed DDPG (TD3), which first imitated the expert and then improved upon itself through self-exploratory training. The policy determined the action of the Franka end-effector and the finger joints based on the historical observations of the estimated object pose and proprioceptive with or without tactile feedback. The policy was trained on four objects with different shapes and sizes as shown in Fig.6.5(a) in simulation and evaluated on six objects in Fig.6.5(b) unknown to the policy. During the training and evaluation, the initial position of an object was randomised within the workspace of the finger joints. We assumed the location of the object had been estimated prior through vision, but with errors of up to 2cm. Each object was evaluated 50 times in simulation and 20 times in reality.

Five baselines and the proposed approach were implemented and evaluated. They shared the same policy structure and training conditions, except for the tactile sensor model used during training and the tactile representation in the observation.

Baseline 1 Baseline1 did not use any tactile feedback. Its observation only included historical observation of robot end-effector position, finger joints of index and thumb and the estimation of object position.

Baseline 2 Baseline2 was trained on a hand-calibrated tactile model. This model used $\boldsymbol{\mu}_{init}$ for the dynamics parameters of each taxel and the parameters were determined heuristically. During policy training, the dynamics parameters of each taxel were randomised within Ω_{init}. This policy used tactile signals in the observation.

Baseline 3 Baseline3 was trained on the calibrated tactile model without DR. Each taxel used its optimised dynamics parameters, $\boldsymbol{\mu}_{*}$. This policy also used tactile signals in the observation.

Baseline 4 Similar to Baseline2, Baseline4 was trained on a hand-calibrated tactile model with DR. The policy took the encoded tactile signals in the observation instead. The simulated and real signals were encoded by only the first encoder.

Baseline 5 Baseline5 used the calibrated tactile models. The first encoder was trained to encode both the simulated and real tactile signals. The policy took the encoded tactile signals in the observation.

Proposed The proposed approach was trained on the calibrated tactile model in the simulation. During policy training, the dynamics parameters of each taxel were randomised within Ω^{*}. The tactile feedback was encoded as described in Sec. 6.2.3.

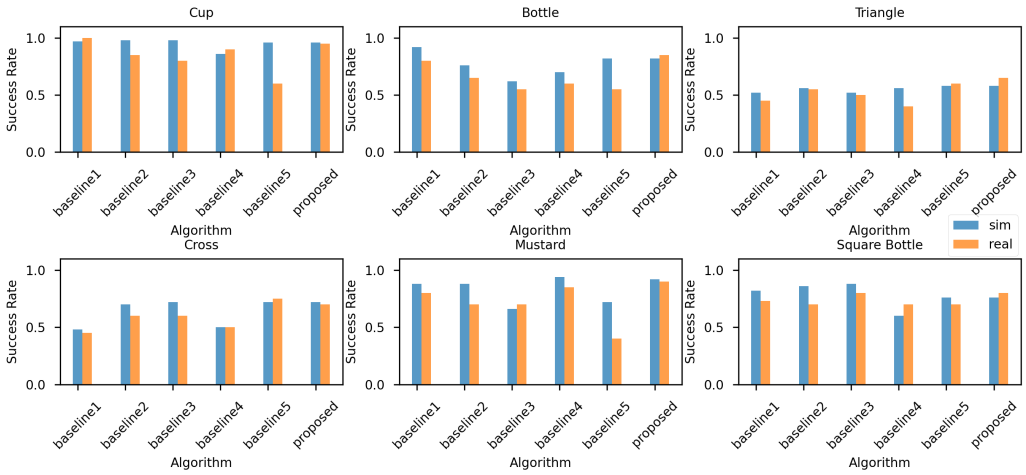


Figure 6.6: Comparison of grasping success rate in simulation and reality

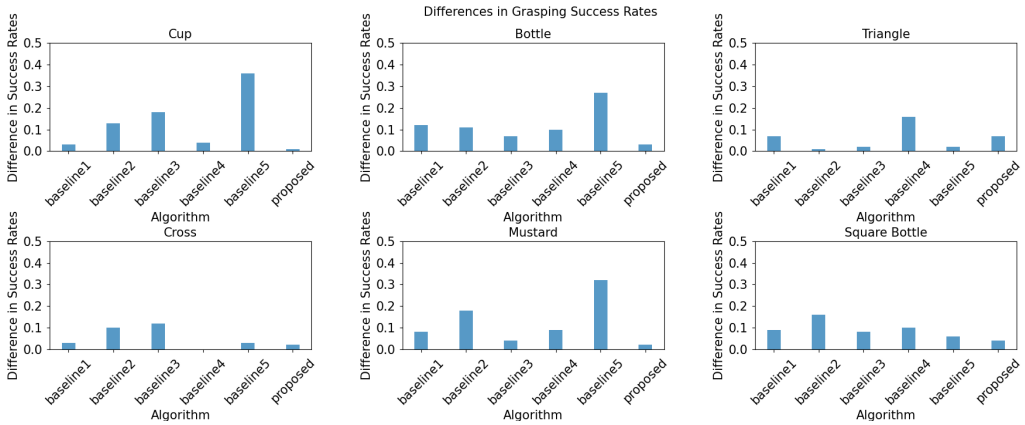


Figure 6.7: A comparison of difference in grasping success rates in simulation and the real world across various objects and baselines.

Table 6.1: Statistics of the difference in success rates obtained from simulation and reality for policies across all objects

	baseline1	baseline2	baseline3	baseline4	baseline5	proposed
Difference in success rates	0.07± 0.03	0.12± 0.05	0.09± 0.05	0.08± 0.05	0.18± 0.14	0.03± 0.02

How well can the simulation-learned policy perform in reality using the proposed tactile model?

Fig. 6.6 shows the success rates for each object in simulation and reality while Fig. 6.7 shows the discrepancies in success rates between the simulation and reality across different objects and policies. Additionally, Tab. 6.1 summarises the errors shown in Fig. 6.7 for all objects. It includes the mean and standard deviation of the differences in success rates across all objects. Here, the task was defined as successful if the robot could grasp and pick up an object from the table and hold it until the end of the episode without dropping it.

Notably, Baseline1 exhibits competitive results in sim-to-real transfer. It had a smaller observation space and reality gap due to the lack of tactile signals compared to other policies. The primary source of the reality gap came from the readings of actuators, which possess high accuracy. Therefore, Baseline1 performed quite well even when only using a simple domain randomisation technique. The same

conclusion can also be drawn from the small mean value in Tab. 6.1, which implicitly implies the differences between the simulation and reality.

As for the proposed approach, the learned policy achieved similar grasping performance in both simulation and real environments. From the statistics in Tab. 6.1, we can see that the average difference in the success rates is the lowest compared to all the other baselines. These results imply two implications. First, it suggests a small reality gap exists between simulated and real-world environments even when the high dimensional tactile feedback is included. This indicates the proposed tactile model and the alignment in tactile embedding could improve the sim-to-real transfer of a tactile-based policy. Second, the smallest variation in the difference in the success rates suggests that the learned tactile-based policy is more robust across different objects. However, we also see that the policy may occasionally fail a grasp due to the absence of contact during exploration, slippage, and unstable grasps. Out of all the objects, the triangular prism was the hardest due to the smaller contact area between the object and the tactile sensors, which can easily result in the object slipping away or towards the palm of the hand. These failures were seen in both the simulation and reality and could be improved with longer policy training and were not attributed to the inaccuracy of the tactile model.

How well can a grasping policy learn with tactile and with different tactile representations?

Without tactile feedback, Baseline1 can grasp unknown objects, but its performance was not consistent across all the objects. The policy almost always performed the same actions regardless of what and where the objects were, and it never adjusted its grasp in the event of an unstable grasp. Hence, the success rate was highly dependent on the type of object and the initial position of an object. Without inferring the object position, this strategy achieved high success rates for large objects, such as a

cup, bottle, mustard bottle, and square bottle, but failed hard on smaller objects, such as a triangle and cross. On the other hand, for a policy that leverages tactile sensors as feedback, an agent could attempt to adjust its grasp to achieve a more stable grasp. This is illustrated in Fig. 6.8, which shows the contacts between the tactile sensors and the object while grasping a cup using the learned policy of the proposed approach. We can see that the centre of contact is moving towards the centre of the tactile sensors during grasping to encourage a more stable grasp in both simulation and reality. Note that the change in the centre of contact is not simply due to increased grasping force. An increased grasping force would result in a larger patch of tactile sensors being activated due to the deformable tactile sensors. And the figure also shows that the maximum grasping force remains relatively unchanged at t = 9, 14, 19. Hence, as expected, this implies the importance of tactile feedback in robotic manipulation.

Baseline2 and Baseline3 were trained using raw tactile signals in the observation, and Baseline4 and Baseline5 were trained to take the encoded tactile feedback. However, from our results, we could not see an advantage of using an encoded tactile representation over using raw tactile signals in terms of grasping success rates. We suspect that without explicitly training an encoder, Baseline2 and Baseline3 may still learn to do similar encoding during policy learning.

Is it sufficient to cross the tactile sim-to-real gap in only tactile modelling, latent space or both?

To examine the sim-to-real performance, we compared the success rates in simulation and reality. Baseline2 and Baseline4 were trained using the hand-calibrated tactile sensor and DR. The differences between the grasping success rates in simulation and reality are still quite large. Hand-calibrated tactile sensors with DR appeared to be insufficient for policy sim-to-real. The excessively large DR range not only

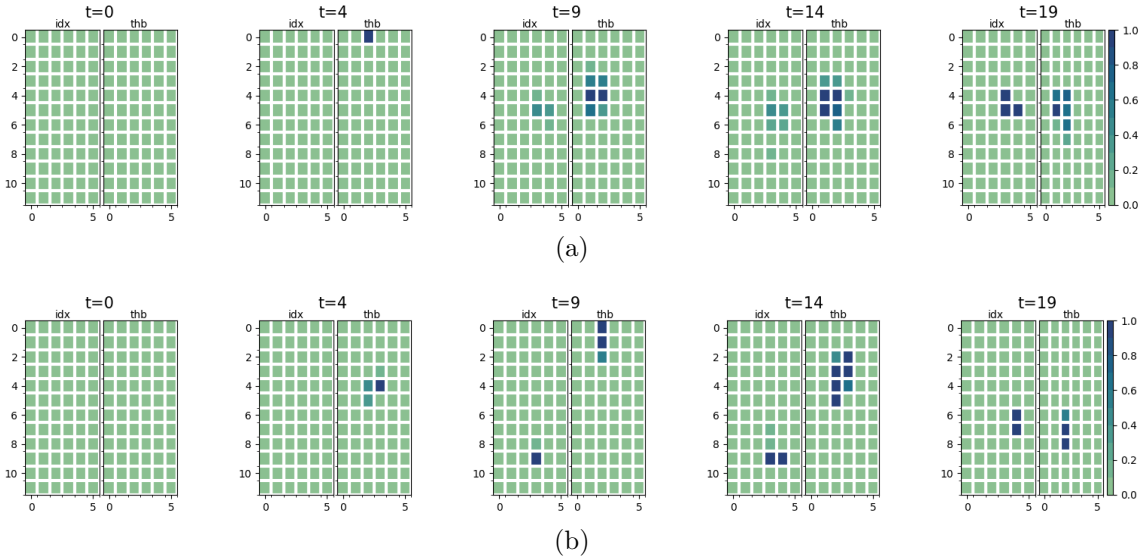


Figure 6.8: Tactile signals from simulation (a) and reality (b) on the index and thumb at five different timesteps while grasping a cup. The intensity of the tactile signal is shown on the colour bar.

increased the training difficulty, but also deteriorated the grasping performance. Baseline3 and Baseline5 were trained using the calibrated tactile sensor without using DR. Baseline3 appeared more consistent in performance than Baseline5 in both environments. We suspect that the difference in the distribution of the simulated and real tactile feedback may have been amplified by the $s_{\hat{sim}}$ in the latent space for Baseline5. The proposed method achieved the most consistent grasping performance across all objects. It was trained on the calibrated tactile sensor with DR and was trained to minimise the difference in tactile embedding. From the results, we can see that while using encoded tactile feedback may not necessarily improve the grasping success rate, minimising the sim-to-real gap in the latent space can greatly improve the sim-to-real results.

6.4 Conclusions

In this paper, we have demonstrated a simple and efficient modelling approach for tactile-based learning and sim-to-real transfer. The sensor we used consists of an array of taxels, and each taxel is modelled as a discrete system to emulate the deformation during contact. The dynamics parameters' distributions are estimated using the real tactile responses as the reference. We introduce one encoder to extract physical embedding from raw tactile signals and another encoder to map the real tactile signals to the same latent space as the previous one. Through a grasping task, we have shown that the proposed framework can easily integrate into a policy learning method to grasp different objects. From our experiments, we demonstrated that our modelling and sim-to-real approach can achieve good performance in both simulation and reality compared to other baselines. Our technique has been shown to work well on our developed tactile sensor array and is generalisable to most tactile sensor arrays.

Chapter 7

Conclusion

7.1 Summary

In industrial applications, robots are programmed to carry out repetitive manipulation at high speeds and with high accuracy, freeing humans from having to do laborious tasks. However, in many other areas, current robotic manipulation still cannot meet the task requirements. Robots still lack advanced mechanics, perception and intelligence to interact with an unstructured environment. This thesis focuses on the research of robotic intelligence with the ultimate goal of achieving autonomous robotic manipulation. We explore a way to endow robots with the abilities to reason from their perceptions, learn novel skills efficiently and adapt to new environments.

In this thesis, we presented different methods and ideas of using human demonstrations and a robot's self-exploratory strategies to improve the efficiency and performance of robot learning for manipulation tasks. Human demonstrations can provide information about a task and adequate control for robots to manipulate an environment. Self-exploratory training, on the other hand, allows robots to improve their

strategies through interaction with environments and prevents robots from heavily relying on human intervention. It, thus, relieves burdens on human programmers and avoids learning human-biased task constraints or strategies. Our work answered the problem of how to use human demonstrations to facilitate learning with supporting experiments and results.

7.2 Conclusion, Limitations and Future Work

Robots still face many challenges in an unstructured environment. Solutions to address the dynamics and uncertainties of this environment are still far from ideal. In this thesis, we focused on the challenges of learning robotic manipulation. It includes a thorough literature review on robotic manipulation and robot learning approaches and the proposed solutions to some of the challenges in robotic manipulation.

Throughout the thesis, we leveraged the idea of human demonstrations and self-exploratory training to facilitate robot learning. We learned that human demonstrations could provide insights into not only control policy but also task constraints and environment dynamics. This information does not have to rely on many expert demonstrations, so it places significantly fewer burdens on human demonstrators. The proposed works have shown that task constraints can limit robots from exploring unnecessary spaces, hence alleviating the problems of high-dimensional state and action spaces. By leveraging a human demonstration and the robot's proprioceptive sensors, a robot can safely interact with an environment and identify the dynamics and variations of an environment. We also came to the same conclusion as prior works that learning directly from demonstrations can quickly acquire skill policies but is not sufficient. Additional experiences were essential for robots to achieve improved and robust manipulation. Our work has shown that subsequent self-exploratory training after learning from demonstrations could be important.

With additional exploratory training, robots can then learn to work with uncertainties and have the ability to moderately correct themselves from errors in perception and control. Lastly, we have also demonstrated that self-exploratory training could encourage robots to exploit their proprioceptive sensory information. The equipped sensory feedback, such as force and tactile sensors, allowed them to reason about the environments and can ease the problems associated with partial observation.

However, the works presented in this thesis were not without limitations. The primary focus was on manipulation where interactions between the robot and the environment were quasi-static. There are several factors that may limit the proposed approaches to handling highly dynamic systems. First, the low sampling rate of perception and control may limit the agent's ability to sense and respond to rapid changes. Second, highly dynamic systems are often are complex, difficult to model and sensitive to disturbances. The presented works mostly rely on simulations to learn policies. However, the simulations may involve simplified dynamics models and estimated dynamics parameters. While in our work, this may be sufficient to cross the reality gap for quasi-static interactions, the sim-to-real transfer may be challenging for dynamic interactions. Apart from the discussed limitations, there are also some specific to the proposed method in each chapter. The work presented in Chapter 3 demonstrated a framework that can obtain an optimal policy from suboptimal demonstrations using RL. However, its performance could be highly dependent on the exploration space, which is governed by the variations and quality of the demonstrations. In Chapter 4, a random exploration strategy was applied to explore informative contact feedback and to help infer a bottleneck. The strategy was adopted due to its simplicity, but it may require a long exploration time to ensure informative contact feedback is obtained. The orientation of the bottleneck was not considered, but it is as important as the position information in the real application. In Chapter 5, the CMAES approach has been adopted to estimate the high dimensional dynamics parameters of the environment. While efficient, the

sampling-based approach may lead to slight variations in results across different iterations. Also, recovering multiple parameters from a single-valued cost may be challenging, and could sometimes lead to incorrect estimation. Lastly, in Chapter 6, the learned policy may not handle objects well when contacts are made to parts of the robotic hand not covered with tactile sensors. The sparsity of the sensors was found to be a major limitation on the usable workspace during grasping and learning more complex manipulation skills.

Finally, robot learning still faces many challenges that the proposed solutions could not address but have constantly emerged. One of the main challenges comes from the hardware of robots, such as noises and errors from sensors, imprecise robotic actuation, a lack of sufficient DoF and forces on a robotic manipulator and communication latency. These may not be solely addressed at the software level and require breakthroughs in different aspects. Furthermore, current robot learning approaches are mostly data-driven and require a large amount of training data in order to learn new and robust skills. This is still very time-consuming, even with the help of simulation, hence rendering the learning inefficient. Lastly, robots to date still lack a good strategy for dealing with unprecedented situations naturally and recovering from failure. Many of these situations may not be foreseen, and hence, this strategy cannot be simply acquired with data-driven approaches or through explicit programming. Humans, on the other hand, outperform robots in many of these areas. Studying the way in which humans work with these problems may provide potential solutions to these challenges.

7.3 Epilogue

This thesis examined methods that combine human priors and a self-learning approach to help robots learn human-like manipulation. Our works have shown that

using this idea, humans can be relieved from laborious explicit programming, robots can learn new skills more efficiently, and learn to handle a range of environments compared to solely learning from human demonstrations or through RL. We hope this work can provide useful insights into future research directions for the robotics community.

Bibliography

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

[2] Ahmed Abo-Ismail. On the development of a new pneumatic versatile gripper. In *Proceedings of the JFPS International Symposium on Fluid Power*, pages 701–706. The Japan Fluid Power System Society, 1993.

[3] Fares J Abu-Dakka, Bojan Nemec, Aljaž Kramberger, Anders Glent Buch, Norbert Krüger, and Ales Ude. Solving peg-in-hole tasks by human demonstration and exception strategies. *Industrial Robot: An International Journal*, 2014.

[4] John R Adler Jr, Steven D Chang, Martin J Murphy, James Doty, Paul Geis, and Stephen L Hancock. The cyberknife: a frameless robotic system for radiosurgery. *Stereotactic and functional neurosurgery*, 69(1-4):124–128, 1997.

[5] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[6] Raghad Alghonaim and Edward Johns. Benchmarking domain randomisation for visual sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12802–12808. IEEE, 2021.

[7] Adam Allevato, Elaine Schaertl Short, Mitch Pryor, and Andrea Thomaz. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning*, pages 445–455, 2020.

[8] Robert J Anderson and Mark W Spong. Hybrid impedance control of robotic manipulators. *IEEE Journal on Robotics and Automation*, 4(5):549–556, 1988.

[9] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[10] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[11] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[12] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202, 2008.

[13] Christopher G Atkeson. Using locally weighted regression for robot learning. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 958–963. IEEE, 1991.

[14] Christopher G Atkeson and Stefan Schaal. Learning tasks from a single demonstration. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1706–1712. IEEE, 1997.

[15] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[16] Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. Surprisingly robust in-hand manipulation: An empirical study. *arXiv preprint arXiv:2201.11503*, 2022.

[17] Thomas Bi, Carmelo Sferrazza, and Raffaello D'Andrea. Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation. *IEEE Robotics and Automation Letters*, 6(3):5761–5768, 2021.

[18] Aude G Billard, Sylvain Calinon, and Florent Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384, 2006.

[19] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[20] Marcel Brass and Cecilia Heyes. Imitation: is cognitive neuroscience solving the correspondence problem? *Trends in cognitive sciences*, 9(10):489–495, 2005.

[21] Erica Briscoe and Jacob Feldman. Conceptual complexity and the bias/variance tradeoff. *Cognition*, 118(1):2–16, 2011.

[22] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[23] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[24] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262, 2007.

[25] Sylvain Calinon and Aude Billard. A probabilistic programming by demonstration framework handling constraints in joint space and task space. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 367–372. Ieee, 2008.

[26] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007.

[27] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[28] Jie Chen, Cédric Richard, and Ali H Sayed. Adaptive clustering for multitask diffusion networks. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 200–204. IEEE, 2015.

[29] Yuanpei Chen, Chao Zeng, Zhiping Wang, Peng Lu, and Chenguang Yang. Zero-shot sim-to-real transfer of reinforcement learning framework for robotics manipulation with demonstration and force feedback. *Robotica*, 41(3):1015–1024, 2023.

[30] Ping Yong Chua, T Ilschner, and Darwin G Caldwell. Robotic manipulation of food products–a review. *Industrial Robot: An International Journal*, 2003.

[31] Shu-Yun Chung and Han-Pang Huang. A mobile robot that understands pedestrian spatial behaviors. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5861–5866. IEEE, 2010.

[32] Whitney Crooks, Shane Rozen-Levy, Barry Trimmer, Chris Rogers, and William Messner. Passive gripper inspired by manduca sexta and the fin ray® effect. *International Journal of Advanced Robotic Systems*, 14(4):1729881417721155, 2017.

[33] Joseph DelPreto, Jeffrey I Lipton, Lindsay Sanneman, Aidan J Fay, Christopher Fourie, Changhyun Choi, and Daniela Rus. Helping robots learn: a human-robot master-apprentice model using demonstrations via virtual reality teleoperation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10226–10233. IEEE, 2020.

[34] Zihan Ding, Nathan F Lepora, and Edward Johns. Sim-to-real transfer for optical tactile sensing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1639–1645. IEEE, 2020.

[35] Zihan Ding, Ya-Yen Tsai, Wang Wei Lee, and Bidan Huang. Sim-to-real transfer for robotic manipulation with tactile sensory. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6778–6785. IEEE, 2021.

[36] Martin Do, Pedram Azad, Tamim Asfour, and Rudiger Dillmann. Imitation of human motion on a humanoid robot using non-linear optimization. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 545–552. IEEE, 2008.

[37] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1927–1934. IEEE, 2018.

[38] Sayna Ebrahimi, Anna Rohrbach, and Trevor Darrell. Gradient-free policy architecture search and adaptation. *arXiv preprint arXiv:1710.05958*, 2017.

[39] Maxim Egorov. Deep reinforcement learning with pomdps. Technical report, Tech. Rep.(Technical Report, Stanford University, 2015), 2015.

[40] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 39–46, 2013.

[41] Nima Fazeli, Russ Tedrake, and Alberto Rodriguez. Identifiability analysis of planar rigid-body frictional contact. In *Robotics Research*, pages 665–682. Springer, 2018.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[43] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[44] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[45] Jeremy A Fishel and Gerald E Loeb. Sensing tactile microvibrations with the biotac—comparison with human sensitivity. In *2012 4th IEEE RAS & EMBS international conference on biomedical robotics and biomechatronics (BioRob)*, pages 1122–1127. IEEE, 2012.

[46] Yang Gao, Ji Lin, Fisher Yu, Sergey Levine, Trevor Darrell, et al. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[47] Guillermo Garcia-Hernando, Edward Johns, and Tae-Kyun Kim. Physics-based dexterous manipulations with estimated hand poses and residual reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9561–9568. IEEE, 2020.

[48] Paul Glick, Srinivasan A Suresh, Donald Ruffatto, Mark Cutkosky, Michael T Tolley, and Aaron Parness. A soft robotic gripper with gecko-inspired adhesive. *IEEE Robotics and Automation Letters*, 3(2):903–910, 2018.

[49] Daniel Fernandes Gomes, Paolo Paoletti, and Shan Luo. Generation of gelsight tactile images for sim2real learning. *IEEE Robotics and Automation Letters*, 6(2):4177–4184, 2021.

[50] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[51] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[52] Mehmet Haklidir and Hakan Temeltaş. Guided soft actor critic: A guided deep reinforcement learning approach for partially observable markov decision processes. *IEEE Access*, 9:159672–159683, 2021.

[53] Nikolaus Hansen. The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer, 2006.

[54] Jochen Hemming, C Wouter Bac, Bart AJ van Tuijl, Ruud Barth, Jan Bontsema, EJ Pekkeriet, and Eldert Van Henten. A robot for harvesting sweet-pepper in greenhouses. 2014.

[55] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[56] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.*, 3(5):551–560, 1990.

[57] Matthew Howard, David J Braun, and Sethu Vijayakumar. Transferring human impedance behavior to heterogeneous variable impedance actuators. *IEEE Transactions on Robotics*, 29(4):847–862, 2013.

[58] Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E Gil Jones. Contact-reactive grasping of objects with partial shape information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1228–1235. IEEE, 2010.

[59] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. Cloud robotics: architecture, challenges and applications. *IEEE network*, 26(3):21–28, 2012.

[60] Yang Hu, Lin Zhang, Carlo A Seneci, Wei Li, Mohamed EMK Abdelaziz, and Guang-Zhong Yang. Design, fabrication, and testing a semiautomatic sewing device for personalized stent graft manufacturing. *IEEE/ASME Transactions on Mechatronics*, 24(2):517–526, 2019.

[61] Bidan Huang, Yiming Yang, Ya-Yen Tsai, and Guang-Zhong Yang. A reconfigurable multirobot cooperation workcell for personalized manufacturing. *IEEE Transactions on Automation Science and Engineering*, 2021.

[62] Bidan Huang, Menglong Ye, Su-Lin Lee, and Guang-Zhong Yang. A vision-guided multi-robot cooperation framework for learning-by-demonstration and task reproduction. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4797–4804. IEEE, 2017.

[63] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

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

[65] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[66] Stephen James, Andrew J Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

[67] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[68] Rae Jeong, Jackie Kay, Francesco Romano, Thomas Lampe, Tom Rothorl, Abbas Abdolmaleki, Tom Erez, Yuval Tassa, and Francesco Nori. Modelling generalized forces with reinforcement learning for sim-to-real transfer. *arXiv preprint arXiv:1910.09471*, 2019.

[69] Jimmy A Joergensen, Lars-Peter Ellekilde, and Henrik G Petersen. Robworksim-an open simulator for sensor based grasping. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8. VDE, 2010.

[70] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[71] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[72] Timothy N Judkins, Dmitry Oleynikov, and Nick Stergiou. Objective evaluation of expert and novice performance during robotic surgical training tasks. *Surgical endoscopy*, 23:590–597, 2009.

[73] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *International Conference on Machine Learning*, pages 2474–2483, 2018.

[74] Zhanat Kappassov, Juan-Antonio Corrales-Ramon, and Véronique Perdereau. Simulation of tactile sensing arrays for physical interaction tasks. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 196–201. IEEE, 2020.

[75] Manuel Kaspar, Juan David Munoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. *arXiv preprint arXiv:2002.11635*, 2020.

[76] Yohannes Kassahun, Bingbin Yu, Abraham Temesgen Tibebu, Danail Stoyanov, Stamatia Giannarou, Jan Hendrik Metzen, and Emmanuel Vander Poorten. Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions. *International journal of computer assisted radiology and surgery*, 11(4):553–568, 2016.

[77] Ben Kehoe, Dmitry Berenson, and Ken Goldberg. Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps. In *2012 IEEE International Conference on Robotics and Automation*, pages 576–583. IEEE, 2012.

[78] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In *Advances in Neural Information Processing Systems*, pages 2859–2867, 2013.

[79] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[80] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *The Journal of Machine Learning Research*, 22(1):1395–1476, 2021.

[81] Hyung-Kew Lee, Sun-Il Chang, and Euisik Yoon. A flexible polymer tactile sensor: Fabrication and modular expandability for large area deployment. *Journal of microelectromechanical systems*, 15(6):1681–1686, 2006.

[82] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[83] Nathan F. Lepora. Soft biomimetic optical tactile sensing with the tactip: A review. *IEEE Sensors Journal*, 21(19):21131–21143, 2021.

[84] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[85] Jacky Liang, Jeffrey Mahler, Michael Laskey, Pusong Li, and Ken Goldberg. Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2017.

[86] Jacky Liang, Saumya Saxena, and Oliver Kroemer. Learning active task-oriented exploration policies for bridging the sim-to-real gap. *arXiv preprint arXiv:2006.01952*, 2020.

[87] Ke Liang, Yuan Xing, Jianmin Li, Shuxin Wang, Aimin Li, and Jinhua Li. Motion control skill assessment based on kinematic analysis of robotic end-effector movements. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 14(1):e1845, 2018.

[88] S Lichiardopol. A survey on teleoperation. *Technische Universitat Eindhoven, DCT report*, 20:40–60, 2007.

[89] Long Ji Lin. Scaling up reinforcement learning for robot control. In *ICML*, 1993.

[90] Matthew T Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1), 2018.

[91] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[92] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[93] Sami Moisio, Beatriz León, Pasi Korkealaakso, and Antonio Morales. Model of tactile sensors using soft contacts and its application in robot grasping simulation. *Robotics and Autonomous Systems*, 61(1):1–12, 2013.

[94] Riccardo Muradore and Paolo Fiorini. A review of bilateral teleoperation algorithms. *Acta Polytechnica Hungarica*, 13(1):191–208, 2016.

[95] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Bayesian domain randomization for sim-to-real transfer. *arXiv preprint arXiv:2003.02471*, 2020.

[96] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[97] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[98] Yashraj S Narang, Balakumar Sundaralingam, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals for the syntouch biotac. *The International Journal of Robotics Research*, 40(12-14):1467–1487, 2021.

[99] Yashraj S Narang, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals via a physics-based and data-driven framework. *arXiv preprint arXiv:2006.03777*, 2020.

[100] Chrystopher L Nehaniv, Kerstin Dautenhahn, et al. The correspondence problem. *Imitation in animals and artifacts*, 41, 2002.

[101] Wyatt S Newman, Yonghong Zhao, and Y-H Pao. Interpretation of force and moment signals for compliant peg-in-hole assembly. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 571–576. IEEE, 2001.

[102] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[103] Masaki Ogino, Hideki Toichi, Yuichiro Yoshikawa, and Minoru Asada. Interaction rule learning with a human partner based on an imitation faculty with a simple visuo-motor mapping. *Robotics and Autonomous Systems*, 54(5):414–418, 2006.

[104] Takayuki Osa, Amir M Ghalamzan Esfahani, Rustam Stolkin, Rudolf Lioutikov, Jan Peters, and Gerhard Neumann. Guiding trajectory optimization by demonstrated distributions. *IEEE Robotics and Automation Letters*, 2(2):819–826, 2017.

[105] Daniel S Pamungkas and Koren Ward. Tele-operation of a robot arm with electro tactile feedback. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 704–709. IEEE, 2013.

[106] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[107] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1–8. IEEE, 2018.

[108] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[109] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, volume 10, pages 1607–1612. Atlanta, 2010.

[110] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

[111] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[112] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[113] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[114] S Laurie Ricker and RE Ellis. 2-d finite-element models of tactile sensors. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 941–947. IEEE, 1993.

[115] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

[116] R Russell. Compliant-skin tactile sensor. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 1645–1648. IEEE, 1987.

[117] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pages 262–270. PMLR, 2017.

[118] Stefan Schaal and Christopher G Atkeson. Receptive field weighted regression. *ATR Human Information Processing Laboratories, Tech. Rep. TR-H-209*, 1997.

[119] Dario Schafroth, Christian Bermes, Samir Bouabdallah, and Roland Siegwart. Modeling, system identification and robust control of a coaxial micro helicopter. *Control Engineering Practice*, 18(7):700–711, 2010.

[120] Alexander Schmitz, Perla Maiolino, Marco Maggiali, Lorenzo Natale, Giorgio Cannata, and Giorgio Metta. Methods and technologies for the implementation of large-scale robot tactile sensors. *IEEE Transactions on Robotics*, 27(3):389–400, 2011.

[121] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[122] Carmelo Sferrazza, Adam Wahlsten, Camill Trueeb, and Raffaello D'Andrea. Ground truth force distribution for learning-based tactile sensing: A finite element approach. *IEEE Access*, 7:173438–173449, 2019.

[123] Ji Hyeon Shin, Jong Geon Park, Dong Il Kim, and Hae Sung Yoon. A universal soft gripper with the optimized fin ray finger. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 8:889–899, 2021.

[124] Jun Shintake, Vito Cacucciolo, Dario Floreano, and Herbert Shea. Soft robotic grippers. *Advanced materials*, 30(29):1707035, 2018.

[125] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[126] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[127] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[128] Abed Soleymani, Xingyu Li, and Mahdi Tavakoli. A domain-adapted machine learning approach for visual evaluation and interpretation of robot-assisted surgery skills. *IEEE Robotics and Automation Letters*, 7(3):8202–8208, 2022.

[129] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

[130] J. Tan, Z. Xie, B. Boots, and C. K. Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2729–2736, 2016.

[131] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

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

[133] Ya-Yen Tsai, Bidan Huang, Yao Guo, and Guang-Zhong Yang. Transfer learning for surgical task segmentation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9166–9172. IEEE, 2019.

[134] Ya-Yen Tsai, Bo Xiao, Edward Johns, and Guang-Zhong Yang. Constrained-space optimization and reinforcement learning for complex tasks. *IEEE Robotics and Automation Letters*, 5(2):683–690, 2020.

[135] Ya-Yen Tsai, Hui Xu, Zihan Ding, Chong Zhang, Edward Johns, and Bidan Huang. Droid: Minimizing the reality gap using single-shot human demonstration. *IEEE Robotics and Automation Letters*, 6(2):3168–3175, 2021.

[136] Aleš Ude, Christopher G Atkeson, and Marcia Riley. Programming full-body movements for humanoid robots by observation. *Robotics and autonomous systems*, 47(2-3):93–108, 2004.

[137] Yusuke Urakami, Alec Hodgkinson, Casey Carlin, Randall Leu, Luca Rigazio, and Pieter Abbeel. Doorgym: A scalable door opening environment and baseline agent. *arXiv preprint arXiv:1908.01887*, 2019.

[138] Eugene Valassakis, Zihan Ding, and Edward Johns. Crossing the gap: A deep dive into zero-shot sim-to-real transfer for dynamics. *arXiv preprint arXiv:2008.06686*, 2020.

[139] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[140] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293, 2000.

[141] Benjamin Ward-Cherrier, Nicholas Pestell, Luke Cramphorn, Benjamin Winstone, Maria Elena Giannaccini, Jonathan Rossiter, and Nathan F Lepora. The tactip family: Soft optical tactile sensors with 3d-printed biomimetic morphologies. *Soft robotics*, 5(2):216–227, 2018.

[142] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[143] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *International Conference on Machine Learning*, pages 6818–6827. PMLR, 2019.

[144] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[145] Di Xiao, Bijoy K Ghosh, Ning Xi, and Tzyh Jong Tarn. Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment. *IEEE Transactions on control systems technology*, 8(4):635–645, 2000.

[146] Wenhao Yu, Visak CV Kumar, Greg Turk, and C Karen Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510. IEEE, 2019.

[147] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[148] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

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

[150] Shao Zhifei and Er Meng Joo. A review of inverse reinforcement learning theory and recent advances. In *2012 IEEE congress on evolutionary computation*, pages 1–8. IEEE, 2012.

[151] Shao Zhifei and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, 2012.

[152] Dahu Zhu, Xiaozhi Feng, Xiaohu Xu, Zeyuan Yang, Wenlong Li, Sijie Yan, and Han Ding. Robotic grinding of complex components: a step towards efficient and intelligent machining–challenges, solutions, and applications. *Robotics and Computer-Integrated Manufacturing*, 65:101908, 2020.

[153] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.

[154] Shaojun Zhu, David Surovik, Kostas Bekris, and Abdeslam Boularias. Efficient model identification for tensegrity locomotion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2985–2990. IEEE, 2018.

[155] Zuyuan Zhu and Huosheng Hu. Robot learning from demonstration in robotic assembly: A survey. *Robotics*, 7(2):17, 2018.