Application of Reinforcement Learning in Robotic Disassembly Operations

by

Mo Qu

A thesis submitted to

The University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

School of Engineering

College of Engineering and Physical Sciences

University of Birmingham

September 2023

Abstract

Disassembly is a key step in remanufacturing. To increase the level of automation in disassembly, it is necessary to use robots that can learn to perform new tasks by themselves rather than having to be manually reprogrammed every time there is a different job. Reinforcement Learning (RL) is a machine learning technique that enables the robots to learn by trial and error rather than being explicitly programmed.

In this thesis, the application of RL to robotic disassembly operations has been studied. Firstly, a literature review on robotic disassembly and the application of RL in contact-rich tasks has been conducted in Chapter 2.

To physically implement RL in robotic disassembly, the task of removing a bolt from a door chain lock has been selected as a case study, and a robotic training platform has been built for this implementation in Chapter 3. This task is chosen because it can demonstrate the capabilities of RL to pathfinding and dealing with reaction forces without explicitly specifying the target coordinates or building a force feedback controller. The robustness of the learned policies against the imprecision of the robot is studied by a proposed method to actively lower the precision of the robots. It has been found that the robot can learn successfully even when the precision is lowered to as low as $\pm 0.5mm$. This work also investigates whether learned policies can be transferred among robots with different precisions. Experiments have been performed by training a robot with a certain precision on a task and replaying the learned skills on a robot with different precision. It has been found that skills learned by a low-precision robot can perform better on a robot with higher precision, and skills learned by a high-precision robot have worse performance on robots with lower precision, as it is suspected that the policies trained on high-precision robots have been overfitted to the precise robots.

In Chapter 4, the approach of using a digital-twin-assisted simulation-to-reality transfer to accelerate the learning performance of the RL has been investigated. To address the issue of identifying the system parameters, such as the stiffness and damping of the contact models, that are difficult to measure directly but are critical for building the digital twins of the environments, system identification method is used to minimise the discrepancy between the response generated from the physical and digital environments by using the Bees Algorithm. It is found that the proposed method effectively increases RL's learning performance. It is also found that it is possible to have worse performance with the sim-to-real transfer if the reality gap is not effectively addressed. However, increasing the size of the dataset and optimisation cycles have been demonstrated to reduce the reality gap and lead to successful sim-to-real transfers.

Based on the training task described in Chapters 4 and 5, a full factorial study has been conducted to identify patterns when selecting the appropriate hyper-parameters when applying the Deep Deterministic Policy Gradient (DDPG) algorithm to the robotic disassembly task. Four hyper-parameters that directly influence the decision-making Artificial Neural Network (ANN) update have been chosen for the study, with three levels assigned to each hyper-parameter. After running 241 simulations, it is found that for this particular task, the learning rates of the actor and critic networks are the most influential hyper-parameters, while the batch size and soft update rate have relatively limited influence.

Finally, the thesis is concluded in Chapter 6 with a summary of findings and suggested future research directions.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Duc Truong Pham, for providing me the opportunity to carry out this wonderful project, contributing his knowledge and experience and offering plenty of inspiring advice throughout this study.

I wish to express my deep appreciation to the School of Engineering at the University of Birmingham for providing financial support for this study.

I would like to acknowledge all the members/alumni of the AUTOREMAN Group who have helped me throughout my PhD study.

Finally, I would like to express my love and deepest gratitude to my parents and sister who are proud of me and have provided me with constant support, and to my dear wife who always encourages me and makes my PhD journey much more enjoyable. Many thanks to all my friends who have spent the years and shared memories with me.

List of Publications

- **Qu, M**., Wang, Y., & Pham, D. T. (2023). Robotic Disassembly Task Training and Skill Transfer Using Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, 19(11), 10934-10943. doi:10.1109/TII.2023.3242831
- **Qu, M.**, Pham, D.T., Altumi, F., Gbadebo, A., Hartono, N., Jiang, K., …, Wang Y. (To be published). Robotic Disassembly Platform for Disassembly of a Plug-In Hybrid Electric Vehicle Battery: a Case Study.
- Huang, J., Pham, D. T., Li, R., **Qu, M.**, Wang, Y., Kerin, M., . . . Zhou, Z. (2021). An experimental human-robot collaborative disassembly cell. *Computers & Industrial Engineering*, 155, 107189. doi:https://doi.org/10.1016/j.cie.2021.107189
- Huang, J., Pham, D. T., Wang, Y., **Qu, M.**, Ji, C., Su, S., . . . Zhou, Z. (2019). A case study in human–robot collaboration in the disassembly of press-fitted components. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 234(3), 654-664. doi:10.1177/0954405419883060
- Li, R., Pham, D. T., Huang, J., Tan, Y., **Qu, M.**, Wang, Y., . . . Zhou, Z. (2020). Unfastening of Hexagonal Headed Screws by a Collaborative Robot. *IEEE Transactions on Automation Science and Engineering*, 17(3), 1455-1468. doi:10.1109/TASE.2019.2958712
- Zhang, Y., Lu, H., Pham, D. T., Wang, Y., **Qu, M.**, Lim, J., & Su, S. (2019). Peg–hole disassembly using active compliance. *Royal Society Open Science*, 6(8), 190476. doi:10.1098/rsos.190476

Table of Contents

Contents

1. Introduction

1.1. Research background

Remanufacturing is a processing strategy to return an End-of-Life (EoL) product to be at least as good as an originally manufactured one. As one of the most important Circular Economy strategies, remanufacturing is promising in many aspects, such as reducing the overall cost of production, reducing negative environmental impact, improving material usage efficiency, and providing the opportunity to upgrade the products. The development of remanufacturing will undoubtedly bring benefits to more sustainable manufacturing.

The remanufacturing procedure usually contains disassembly, cleaning, inspection, repair or replace, reassembly, and testing. Among them, disassembly is the first and most crucial step for the following reasons. Firstly, a reliable way to retrieve the complete product cores is required to proceed with the rest of the steps. In other words, remanufacturing can be economically unjustifiable if the cores are severely damaged during disassembly. Secondly, remanufacturing is not the only EoL processing that requires disassembly. Other processing, such as recycling and repurposing, also benefit from reliable and efficient disassembly. In other words, advancement towards a reliable and efficient disassembly system is transferable to other strategies.

Disassembly is challenging to robotise due to variability in the condition of the returned products and the required dexterity in robotic manipulations. Furthermore, disassembly is usually more stochastic than assembly, as it has to contend with used products of uncertain shapes, sizes, and conditions. The recent advancement of industrial robots and related automation technologies, such as computer vision, sensors and machine learning, is promising

for developing automated disassembly systems when dealing with the uncertainties in variability and increasing dexterity.

The research directions of robotic disassembly can be broadly categorised into four levels: robotic disassembly system design (Shahbazi et al., 2021), product disassembly sequence planning (Ong et al., 2021), robot path planning (Gasparetto et al., 2015), and robotic disassembly operations (Zhang et al., 2019). Among them, the study of disassembly operations is the most fundamental because every other planning or design depends on the capability of robots to perform operations. Furthermore, most disassembly operations rely on the robots interacting with external objects. Therefore, external information, such as force and visual information, is utilised only in this level of study. Examples of operations include unscrewing, object manipulation, and separating parts with close contact. Example solutions to these operations include force controllers, active or passive compliance strategies and robotic tools.

One of the critical features identified from these operations is physical contact, and many problems originate from the intrinsic rigidity of industrial robots. Although robots with intrinsic passive compliances are available, the precision of this kind of robot is an issue as precise operations usually require the robots to have high stiffness in their joints. Therefore, the studies on operations with physical contact focus on understanding the fundamental physics of the problems, hoping to design tools and control strategies to ensure the safety of the robots and the parts.

However, this research direction usually involves carefully building a mechanics model of the task that requires extensive knowledge — for example, knowing the exact geometry of the parts, which is not practical in disassembly due to uncertainties and lack of information. Thus, another approach where the robot can learn from trial-and-error – Reinforcement Learning (RL)

– can bring benefits because it does not require extensive prior knowledge and a mechanistic model. RL has recently achieved much success in various areas. It has also been applied to industrial robots to learn controllers for assembly operations with contact. This research attempts to bring the methods of RL also to the disassembly operations.

1.2. Aim and objectives

This research aims to apply RL in learning disassembly operations and investigate the methods to increase the performance of the learning process and the learned controllers. The main objectives of the PhD programme include:

1. Building a robotic disassembly operation training platform with RL,
2. Characterise and demonstrate the application of RL with a case study of a robot learning to remove a pin from a door chain slot,
3. Investigating the ability of the off-policy algorithms to learn from past data,
4. Investigating the robustness of the RL learning method against the precision of the robot,
5. Investigating the transferability of the learned controller among robots with different precision,
6. Building a simulated disassembly operation training platform for RL,
7. Accelerating the learning process by Sim-to-Real transfer,
8. Optimising the simulated environment by Bees Algorithm (BA),
9. Hyper-parameter study of DDPG algorithm with three case studies of simulated disassembly operations.

1.3. Outline of the thesis

Chapter 1 introduces the background, aim and objectives of this research.

Chapter 2 reviews the state-of-the-art literature on topics of robotic disassembly and applications of RL in robotic disassembly.

Chapter 3 reports the development of a robotic disassembly operation training platform by demonstrating the robot learning to slide a pin along a door chain slot using RL algorithms. Also, this chapter reports investigations about the robustness and transferability of the learned controllers under the influence of the precision of the robot. This chapter corresponds to **objectives 1 to 5**.

Chapter 4 reports the study of accelerating the learning process by transferring a learned controller from a simulated environment to a real-world environment (sim-to-real). This chapter corresponds to **objectives 6 to 8**.

Chapter 5 describes the hyper-parameter study of the DDPG algorithm. This chapter corresponds to **objective 9**.

Chapter 6 concludes the thesis and suggests future research directions.

2. Literature Review

2.1. Robotic disassembly for remanufacturing

Remanufacturing is one of the main choices for End-of-Life (EoL) processing when a product has reached the end of its lifecycle (AUTOREMAN project, 2017). According to (British Standards Institution, 2009), Remanufacturing "returns a used product to at least its original performance with a warranty that is equivalent or better than that of the newly manufactured product". Therefore, remanufacturing should be differentiated from the other EoL processing, such as reconditioning, reusing or refurbishing, in terms of the quality of the returned products, whereas the others only require the returned products to have an equivalent or lower quality (British Standards Institution, 2010).

Remanufacturing is an industry that has many positive social-economical impacts, such as material efficiency, sustainability and job opportunities (Jansson, 2016; Liao et al., 2018; Mitchell & Morgan, 2015; Sundin & Lee, 2012). One study shows that remanufacturing a diesel engine could reduce 66% of energy consumption compared with an originally manufactured one and has a 97% reduction of ozone depletion potential (Liu et al., 2014). Another study analyses strategies for remanufacturing loading machines, reporting up to 80% less climate impact and 58% less cost (Xiao et al., 2018). Remanufacturing also provides an opportunity to upgrade the product (Fofou et al., 2021), which is argued to impact business models (Chierici & Copani, 2016).

As illustrated in Figure 2.1, remanufacturing returns the processed product with the highest standard. However, it does not mean the products should always be remanufactured despite the benefits. Other factors, such as cost, quantity and quality of the returned cores, should also be

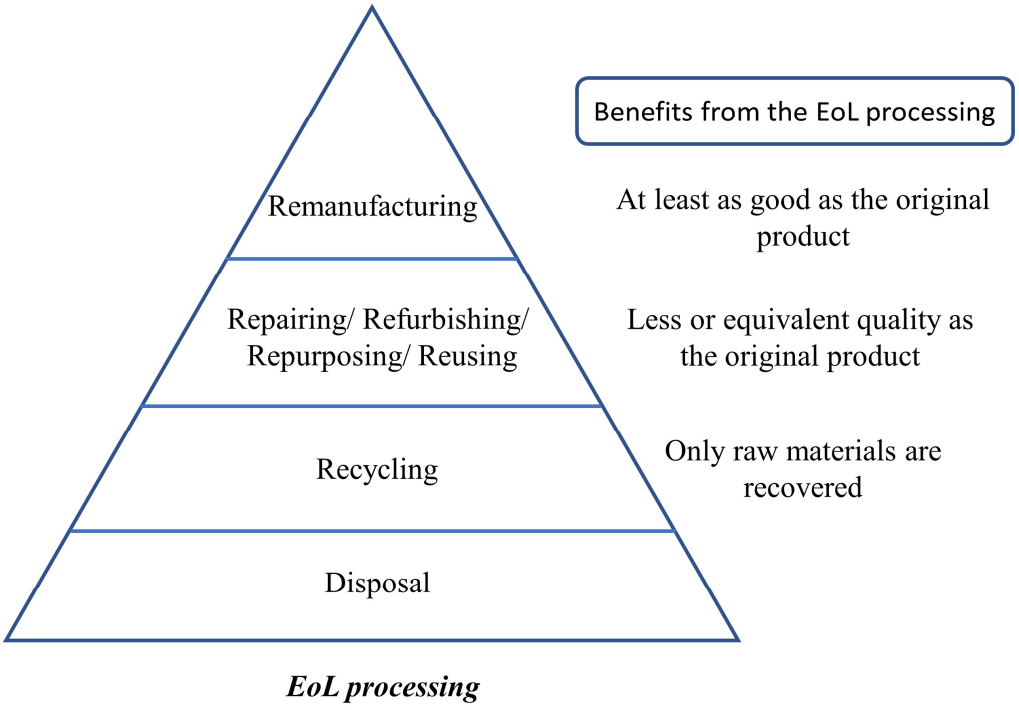


Figure 2.1: Hierarchy of benefits from the main EoL processing.

considered (Ijomah, 2009; Zhang et al., 2020b). Also, there are arguments about repurposing some EoL products on a large scale, such as Lithium-ion batteries, might introduce a lag of 5 to 10 years for the critical materials to be recycled and recirculated back to the market (Kamran et al., 2021). The logic is also applicable to remanufacturing. Some researchers have dedicated to evaluating the remanufacturability of the products (Amezquita et al., 1995; Fang et al., 2014; Zhang et al., 2020a), and there are some attempts to improve the remanufacturability through better product design (Ijomah et al., 2007; Shahbazi et al., 2021). However, with all the promising benefits, the study of remanufacturability itself proves that remanufacturing still requires more development to be a mature industry.

A complete remanufacturing cycle typically contains the following steps: disassembly, cleaning, inspection, repair, testing and re-assembly (British Standards Institution, 2010; Östlin, 2008). All steps require extra labour and cost. Among them, an efficient and reliable disassembly, as the first step of the remanufacturing process, must be developed to conduct the following steps on a large scale. With the advancement of Industry 4.0, automating disassembly is a promising research direction (Kerin & Pham, 2019, 2020), particularly with the rapid development of industrial robots.

2.2. Robotic disassembly

2.2.1. Challenges in robotic disassembly

Assembly and disassembly share many commonalities, but the challenges of automating disassembly are more extensive than their assembly automation counterpart (Vongbunyong & Chen, 2015). In disassembly, three main unique challenges are different from assembly.

Firstly, the uncertainties arising from the conditions of the returned EoL products must be carefully dealt with (Foo et al., 2022), as certainties are the key to a stable automation system. As for assembly, the uncertainties of the parts tend to be controlled to the minimum before being assembled (Daneshmand et al., 2022).

Secondly, a disassembly system must deal with an uncertain feed stream due to a much smaller input stream than an assembly system (Bogue, 2019) because the returned quantity of the returned products is hard to predict, but it can be anticipated and planned in an assembly system. To respond to the variety of input streams, a disassembly system that is flexible enough to deal with a range of returned products instead of just one kind is much more desired.

Thirdly, a product is generally easier to assemble than disassemble for a few reasons. The main reason is that the products should remain secured and intact during the use stage of the lifecycle. Sometimes, this is a matter of health and safety (Kong et al., 2018; Nedjalkov et al., 2016). Customers will not trust a product that is loosely assembled. Another reason is that the design for disassembly has been less considered for products developed years ago due to a lack of legislation and sustainable development awareness. Another part of the reason is that designing a product that is easy on both assembly and disassembly adds cost to the product, which is an unwise decision if the manufacturers do not need to be responsible for the disposal or maintenance of the product.

In summary, a disassembly system must face a system that contains higher uncertainty in terms of the variance of EoL conditions of the returned products and the diversity of the products than an assembly system. Sometimes, a disassembly system must face products that are harder to disassemble than assemble.

With the increased development of robotic technology, industrial robots in a disassembly system can solve some of the mentioned problems with less cost, which is not to say that using industrial robots can solve all the problems, but to say that by combining with other automation devices, such as specialised end-effectors or automation devices that are independent of the robot, industrial robots will be more critical as a part of the automation system. Hence, the scope of the literature review is limited to the field of robotic disassembly, which is a subset of disassembly automation.

2.2.2. Robotic disassembly in four levels

The studies about robotic disassembly can be categorised into these four levels:

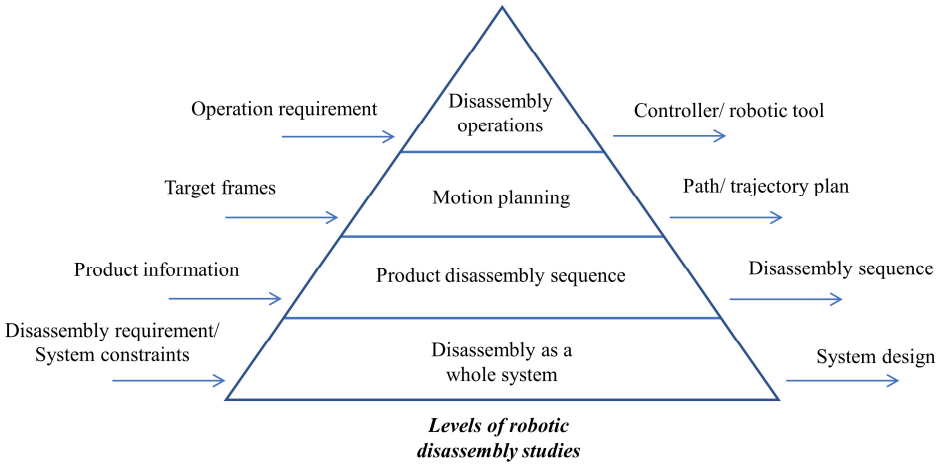


Figure 2.2: Levels of robotic disassembly studies and the corresponding problems to solve.

- **Level 1**: Studies that focus on developing a robotic disassembly cell from the perspective of developing a whole system. These studies need to be differentiated from those focusing only on parts of the robotic system.
- **Level 2**: Studies that focus on planning the sequence to disassemble a product.
- **Level 3**: Studies that focus on planning the motion of the robots.
- **Level 4**: Studies that focus on specific disassembly operations.

The principle of categorisation is based on the kind of information considered when choosing the approaches at each level, as illustrated in Figure 2.2.

Although the problems from these levels seem to have a top-down structure, it should not be mistaken that the design of a robotic disassembly system should follow the sequence of these levels. The decision from one level will inevitably affect the decision of another. For example, if the robot cannot fulfil a particular operation as planned, then either the disassembly sequence plan needs to be changed, or the function of the cell needs to be reviewed.

One major topic of study is not included in this four-level categorisation, but it also contributes extensively to robotic disassembly systems. It is the studies on design for disassembly (Soh et al., 2014) where the researchers include techniques that could ease disassembly operations, such as smart materials (Abuzied et al., 2020) or combining fasteners (one-to-many fasteners) (Carrell et al., 2009), from the design stage of the product. However, these techniques can only be applied to new generations of products. The current disassembly system development focuses on disassembling the retired products from the previous generations.

The rest of this section briefly reviews studies from the perspective of the above-mentioned four levels. The purpose of the review is not to be exhaustive or systematic but to elaborate on the features and limitations of the current studies by showing some representative studies. Some recent systematic reviews on studies of robotic disassembly have been done by (Hjorth & Chrysostomou, 2022; Poschmann et al., 2020).

Level 1 – Disassembly system design

On level 1, the system's functions and the equipment availability are considered. However, the designs made at this level are tightly linked with level 4, as the function of the system and the equipment required for a system are tightly linked with what operations can be done by the robots. Thus, some studies investigate the automation potential to disassemble products (Blankemeyer et al., 2021; Hellmuth et al., 2021).

A group of studies have focused on developing disassembly systems in which collaborative robots and operators work side-by-side in a Human-Robot Collaboration (HRC) way to achieve semi-autonomous disassembly (Ogenyi et al., 2021). The main disadvantage of a fully autonomous system is that it is generally harder to achieve because of the limitations of robotic

tools, accumulation of errors, and less adaptability compared with human operators (Hjorth & Chrysostomou, 2022). These can all be avoided by HRC.

Huang et al. built an HRC disassembly cell to disassemble an automotive water pump (Huang et al., 2019). The water pump consists of press-fitted parts, so a hydraulic press is used to separate them. In this case study, the human operator only needs to operate the hydraulic press. The robot is responsible for all the other tasks, including following the movement of the press while holding the pressed part. Despite the effectiveness shown in the case study, the product is relatively easy, as it has only six components.

Based on the previous work, Huang et al. built another HRC disassembly cell to disassemble a turbocharger, which is a more complicated product in terms of more components and types of operations involved. In this case study, the human operator is only responsible for unscrewing some bolts that the nut runner on the robot cannot access (Huang et al., 2021). However, this study does not consider the uncertainty introduced by the EoL conditions of the returned products.

However, the downside of HRC is also apparent. To work with robots side-by-side requires a higher safety standard (Michalos et al., 2015; Villani et al., 2018). Also, HRC can be seen as a way to trade the level of automation for the capability of dealing with uncertainties by combining the mechanical power of the robots and the cognitive ability and dexterity of humans. Thus, the efficiency is inevitably lower (Foo et al., 2022).

Level 2 – Product disassembly sequence planning

On level 2, the product information, such as the part list and connection relations, is provided to plan the disassembly sequence. Only high-level instructions are given, such as what part to

disassemble by which robot or what connectors should be removed first. In other words, the output only instructs the equipment what to do, not how they do it.

At this level, the cognitive ability of the robotic system, such as using vision sensing (DiFilippo & Jouaneh, 2018; Vongbunyong et al., 2013; Wegener et al., 2015), to obtain more information can contribute to better planning or replanning.

A systematic literature review is done by (Zhou et al., 2018), where the challenges and trends are clearly stated. A more recent review can be found in (Ong et al., 2021).

Level 3 – Robotic motion planning

On level 3, given the instructions, the robot plans every single movement. The robots might have been given some targeting locations, for example, moving from point A to point B, from the plan generated from level 2. Then, it is this level's responsibility to figure out the specific path or trajectory to move from point A to point B. However, suppose the robots do not know the targeting locations. In that case, the locations should be generated by the decision-making agent (human or computer) before planning the paths and trajectories. A systematic literature review can be found in (Gasparetto et al., 2015).

Level 4 – Robotic disassembly operations

On level 4, a specific operation needs to be carried out after the robot has been moved to a targeting location, which includes but is not limited to pick-and-place objects, unscrewing threaded fasteners (Li et al., 2020), removing an object from another (Zhang et al., 2019). All robotic operations need to encounter misalignment errors to some degree, for example, due to the repeatability of the robots or the precision of the vision systems. Thus, some operations will fail in some situations, such as objects slipping out of the gripper or bolts failing to be unscrewed, as the robot cannot successfully localise the bolts. Therefore, physical information,

such as vision and force information from cameras and force/torque sensors, is essential for some operations.

The disassembly operation study is the foundation of a robotic disassembly system because the decisions from every other level depend on the capability of robots to perform disassembly operations.

2.3. Application of reinforcement learning on robotic disassembly

Reinforcement Learning (RL) is a branch of machine learning algorithms specialising in making sequential decisions. The goal of RL is to make an agent make decisions (actions) based on observations, following the direction of having the best reward (an artificially designed function to rate the observations and actions) across a period of action-observation interactions. With the increased exposure to data and iterations of learning, an adequately designed RL agent should improve at making decisions to get more rewards to shape the agent's behaviour.

Although other ways exist to model the RL problems (Sutton & Barto, 2018), the Markov Decision Process (MDP) is the most popular way to model the problems, as shown in Figure 2.3. In an MDP, the agent is the one that takes the actions, and everything else that the agent interacts with is the environment. At each time step, t, the agent perceives the state, S_t, from the environment, represented by a set of features from the environment, and the agent selects an action, A_t, and interacts with the environment, while the environment responds with reward feedback, R_{t+1}, and changes from the current state to the next one, S_{t+1}.

This section reviews some representative studies of applying RL in robotic disassembly.

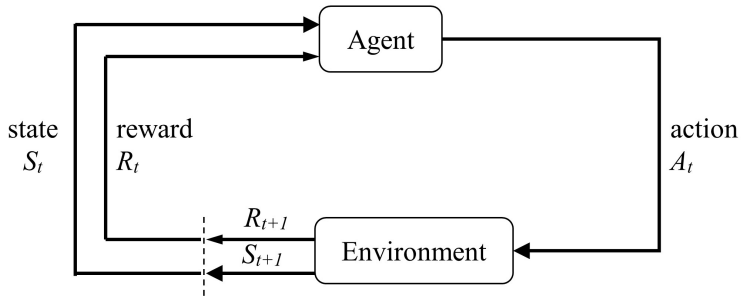


Figure 2.3: Basic agent-environment interaction in MDP (Sutton & Barto, 2018)

2.3.1. Reinforcement learning from an engineering perspective

This section reviews two promising research areas in improving robotic disassembly in general. The discussion focuses not on the advancement of RL algorithms but on what specific features of RL demonstrate the potential of helping robotic disassembly from an engineering perspective.

RL for continuous control

RL can select actions from both discrete and continuous action domains. Although some famous applications of RL are developed based on the problems in which the agent chooses discrete actions, such as playing video games (Mnih et al., 2013; Mnih et al., 2015) and the famous AlphaGo (Silver et al., 2017), a high proportion of the problems in robotics are solved by designing controllers, such as position controller or force controller (Mckerrow, 1991), that require input from continuous space. (Note that this is similar to the combinatorial and continuous problems in optimisation algorithms. RL has so many similarities with optimisation algorithms that a subfield of RL, Policy Search (PS), attempts to solve the problems by direct optimisation in the parameter space (Deisenroth et al., 2013; Sigaud & Stulp, 2019), and PS is widely used in robotic control problems (Chatzilygeroudis et al., 2020).

Although there has been successful implementation of RL in discrete action space (Inoue et al., 2017), choosing actions from a continuous action space is recognised to be more suitable for robotic tasks (Deisenroth et al., 2013; Kober et al., 2013).

Since robotic control is such a broad topic, it is hard to compare the algorithms fairly. Thus, some previous studies attempted to benchmark the RL algorithms in continuous control tasks (Duan et al., 2016; Henderson et al., 2018; Mahmood et al., 2018). However, it is still hard to draw universal conclusions from these kinds of comparison studies because one RL algorithm

might outperform others in one task but underperform in another one, or a change of the hyper-parameter can turn the results around (Islam et al., 2017; Mania et al., 2018).

Model-based RL algorithms

Unlike model-free RL algorithms, model-based RL algorithms learn directly from the transition data and a surrogate model that models the system's dynamic. This surrogate model is also trained by the same dataset that trains the learning agent, as shown in Figure 2.4 (Sutton & Barto, 2018).

In this way, the data efficiency is much higher than merely learning from experiences, which is a handy feature in robotic control because higher data efficiency means fewer trial-and-error interactions are required for the robots (Polydoros & Nalpantidis, 2017). Studies in simulated environments usually do not consider the importance of fewer interactions (Heess et al., 2017; Wang et al., 2019). For example, in one study (Levine et al., 2017), the training involved 1,700,000 grasp attempts from 14 robotic manipulators. Training with fewer interactions is needed (Mouret, 2016).

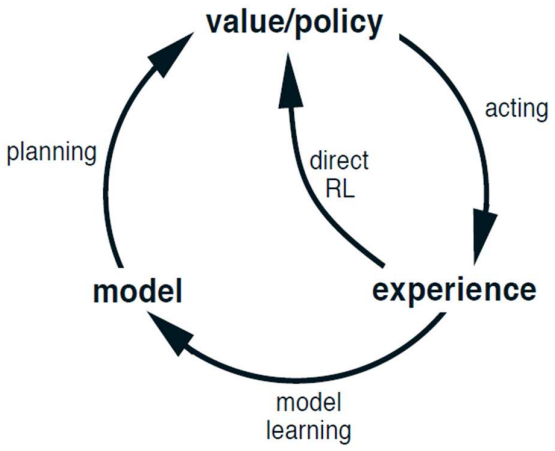


Figure 2.4: Illustration of model-based RL (Sutton & Barto, 2018).

However, the disadvantage of model-based RL is obvious: the performance highly relies on the accuracy of the prediction model. When training with real robotic tasks, the model must capture the task's randomness to make accurate predictions (Chua et al., 2018; Kurutach et al., 2018).

2.3.2. Reinforcement learning in robotic assembly/ disassembly operations

As discussed in Section 2.2, robotic disassembly operations are the most fundamental elements in a robotic disassembly system. They are often limited by the flexibility, precision and margin of error that the robots can offer.

To the date of this thesis, no published research paper discusses applying RL to robotic disassembly operation problems to the best of the author's knowledge. However, the existing literature focuses heavily on applying RL to robotic assembly problems. As discussed in Section 2.2, assembly and disassembly operations share some commonalities, especially in that they both need to address the issues associated with contacts. Therefore, the literature reviewed in this section is from the domain of assembly operations, hoping that the application on assembly operations can provide insights into disassembly operations.

Robotic assembly tasks are tasks about using robot manipulators to assemble components. Although other assembly operations exist, the typical operation can be represented as inserting a peg into a hole (Xu et al., 2019a). The peg-hole problems have been studied extensively, and solutions from different approaches have been offered. The main challenges of this insertion problem include (1) force information being complex and hard to predict and 2) positioning error due to inaccuracy from various sources, such as repeatability of the robot or the precision of the measurements (Whitney, 1982).

Nuttin et al. are among the first to propose the conceptual work of using RL in peg-hole assembly problems (Nuttin et al., 1997). The force controller adjusts its parameters by the

REINFORCE algorithm (Williams, 1992). However, the experiment is only performed in a simulated environment with CAD models.

More recent works start from Inoue et al., who are the first to apply RL to learn high-precision peg-hole insertion operation in a real-world setup (Inoue et al., 2017). They have trained multiple Long Short-Term Memory (LSTM) neural networks as they treat the insertion process as non-Markovian or partially observable (Bakker, 2001). They establish the basic architecture of the training system, the way to represent states and actions, the termination conditions, and the strategy of separating the operation into two stages: searching and insertion. Although separating the operation into two stages is not a novel idea for peg-hole insertion (Sharma et al., 2013), the effort to include this separation in an RL approach should still be highlighted. They have demonstrated that the RL approach is a competitive strategy for high-precision peg-hole insertion operations. The clearance between the peg and the hole is as small as $10\ \mu m$. They have also found that by using RL, the robot is able to perform peg-hole insertion where the precision of the manipulator is lower than the clearance without applying any compliance strategies. However, their approach is limited to the problem that is defined in a discrete action domain, which is not suitable when the solution space is too large, compared with the algorithms that can select actions from a continuous action domain, such as Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2016).

Thomas et al. improved the RL efficiency and success rate by combining motion planning algorithms in a simulated environment and learning an actual robot by manipulating the reward function (Thomas et al., 2018). They start by showing a motivating example of a ring-shaft assembly operation, where the central axis of the ring is far from the central axis of the shaft at the starting position so that the correct path is hard to be searched by a typical RL algorithm, indicating that the algorithm is trapped in local optima. Firstly, a reference trajectory is

generated in a simulated environment, given the geometry of the components from CAD models. Then, the generated reference trajectory guides the policy search in a direction that can avoid collisions and local optima of the policy search. Finally, to increase the generalisation capability of the controller across different task configurations (different initial and final positions of the object), a neural network that takes the state information and the reference trajectories as input is trained. Afterwards, the actions are selected based on the state information, and a fraction of the local trajectories centred around the current time step, t. This method decreases the online computational load as the motion planning algorithm has already encoded the global information. The authors have successfully demonstrated their approach in three tasks: 1) interlinking two U-shape blocks, 2) assembling a gear with a circular aperture to a shaft, and 3) inserting a round peg into a hole. However, the above approach can only be implemented if high-quality CAD models can be obtained before the training, but the models are not always available. Also, the demonstrating robot is an inherently compliant robot rather than a rigid industrial robot arm, which means some degree of compliance has been previously integrated. Therefore, applying the proposed approach to traditional robot manipulators requires further investigation.

Ren et al. solved the problem of selecting discrete actions from continuous using the DDPG algorithm (Lillicrap et al., 2016; Ren et al., 2018). They introduced compliant strategies in the RL settings for high-precision peg-hole insertion problems by adding Cartesian stiffness dimensions for impedance control into the actions space. This work proposes a "jamming indicator" that makes the RL process fully become an MDP. The authors have argued that the state information for each time step will then contain enough information to describe the state fully, and they have also tested the learned policy's generalisation ability. However, the "hole" being inserted by the peg is formed by two vertical and parallel walls, which essentially is a 2D hole instead of a 3D one, and whether such simplification can be generalised in actual peg-

hole insertion is questionable. The results of the learning curves appear to be very unstable, so the method or the hyper-parameter responsible for balancing exploration and exploitation should be improved.

A group of works used the Guided Policy Search (GPS) algorithm family to train the robots for assembly operations (Sergey & Vladlen, 2013). Luo et al. proposed using a Mirror Descent Guided Policy Search (MDGPS) algorithm (Luo et al., 2018; Montgomery & Levine, 2016), a variant of GPS, to learn the assembly problem where a round peg is inserted into a deformable hole with a smaller diameter. The GPS algorithm is a model-based on-policy algorithm that alternates between trajectory optimisation over local policies and then provides samples from local policies to the global policy. Compared with previous works, the authors argue that by providing an on-wrist F/T sensor and a rigid industrial robotic arm, the admittance control can also be integrated with the RL setting, meaning that not just the robots with inherent passive compliant joints or the robots that can issue joint torque commands can perform RL with contact-rich tasks (Ren et al., 2018). They have demonstrated that the deformable hole can be 5 mm smaller in diameter than the peg.

Another work by Luo et al. uses the same MDGPS algorithm. However, the application scenario changes from admittance controller to impedance controller and the action space changes from tool space to operational space (Luo et al., 2019). The demonstrating case is to assemble a set of gears that contains four contact-rich assembly operations: 1) inserting a round peg into a round hole, 2) fitting a gear wheel to the shaft, 3) inserting another gear wheel with squared peg-hole fitting, 4) align the two gear wheels. They have done an ablation study showing that the force information is less useful when used directly as the input to a global policy neural network model.

Another work from the GPS algorithm family was done by Fan et al. (Fan et al., 2019). It extends the GPS algorithm family to replace the traditional global policy with the DDPG algorithm (Lillicrap et al., 2016). It demonstrates the implementations with a Lego brick and U-shaped block assemblies.

Xu et al. extended the application of the DDPG algorithm (Lillicrap et al., 2016) to a dual-peg in-hole insertion problem by adding a fuzzy reward system and a feedback exploration strategy (Xu et al., 2019b). The fuzzy reward system tries to balance the contact force and operational time or possibly include more considerations by using fuzzy logic to shape the reward function. The feedback exploration strategy includes actively adjusting the exploration noise for both action space (Khamassi et al., 2017) and parameter space based on the reward. The exploration is decreased when the agent is performing better and vice versa. However, the feedback exploration strategy increases the computation load between each action step, thus decreasing the control frequency during the operation.

Schoettler et al. demonstrated that using a more intuitive reward function (Schoettler et al., 2020), for example, the $l1$ distance between the current image and the goal image or the sparse reward function, can increase the success rate and robustness against the error of the goal position of the training. They demonstrate the assembly operations by inserting three commonly found electric connectors, a) a USB connector, b) a D-Sub connector, and c) a Model-E connector. To overcome the vast space to exploration for optimising the sparse reward function, they have injected prior knowledge by using the Residual RL technique (Johannink et al., 2019; Silver et al., 2019) combined with the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) and the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm (Fujimoto et al., 2018). However, computing rewards from visual rather than force information

requires much more computation. In the case of contact-rich assembly, the advantage of using visual information over force information is not very clear.

Beltran-Hernandez et al. combined two traditional force control schemes, the PID parallel position/force control scheme (Chiaverini & Sciavicco, 1993) and the admittance control scheme, separately with the SAC algorithm (Haarnoja et al., 2018) to achieve peg-hole insertion task (Beltran-Hernandez et al., 2020b). They propose a "fail-safe mechanism" to decrease human involvement during the training. However, that mechanism relies on constantly checking the feasibility of the selected action, which should have been excluded from the action selection domain in the first place. Including it during the episode increases the unnecessary computation load.

Beltran-Hernandez et al. have also extended their work with an RL improvement technique called distributed prioritised experience replay (Beltran-Hernandez et al., 2020a; Horgan et al., 2018). The highlight of this paper is to implement the idea of simulation-to-real (sim-to-real) transfer (Chebotar et al., 2019) to improve the generalisation capability of the learned policies by pre-training the robot in a simulated environment. Thus, some factors are randomised for each training, including a) the initial/final position of the end-effector, b) surface stiffness, c) the error of predicted final goal, and d) desired insertion force. Another novelty is that instead of using the force/torque (F/T) information from the wrist sensor directly as the state input to the neural networks, the last 12 readings from the F/T sensor are converted from 12 time-series arrays to a 36-dimensional vector by a temporal convolutional network (Bai et al., 2018) with the other state information. This approach enables the feature extracted from the fluctuating of the force information to be used directly as state information for the current time step. The policy is trained on the assembly task of inserting a cuboid peg into a hole. Further generalisation capability studies evaluate the learned policies on the other assembly cases,

including a) ring-shaft assembly, b) 2-pin electric plug inserting into a socket, c) plugging in a LAN cable to a connector, d) plugging in a USB. Despite the excellent reported performance, not all the robots can access the controllers at the joint motor levels. Also, the training quality on actual operations will depend much on the simulation quality.

2.4. Summary

Remanufacturing is the only EoL processing that can return the used product to be at least as good as an originally manufactured one. Remanufacturing as an industry is promising in bringing many benefits, including cost reduction, less environmental damage, positive social impact, and the potential to upgrade business models. However, remanufacturing is limited by the quality and quantity of the returned product cores. Thus, disassembly is considered to be the critical step in remanufacturing.

Disassembly is currently a labour-intensive process due to the uncertainty and variation of the EoL products. The key to improving disassembly is to automate the process by industrial robots. Therefore, robots need to be intelligent and flexible.

Four groups of studies about developing a robotic disassembly system have been categorised: disassembly system design, product sequence planning, robotic motion planning, and disassembly operation methods. Among them, disassembly operation is the most fundamental part because the decisions from every other level depend on the capability of robots to perform disassembly operations.

Enabling the robots to learn from trial-and-error and existing data on disassembly operations is one meaningful way to make the robots flexible and robust, which can be achieved by integrating with the advancement of RL. However, although RL has been implemented in virtual environments on control tasks, the specific studies on applying RL in actual robotic operations for both assembly and disassembly still require further investigation. The current studies on RL mainly focus on developing theories or designs from an algorithm perspective.

However, the application aspect, such as investigating the relationship between learning performance and the physical features of the robots and using simulations to accelerate trainings, requires further investigation.

3. Robotic Disassembly Task Training and Skill Transfer Using Reinforcement Learning

3.1. Preliminaries

Disassembly is an early and key step in remanufacturing (Priyono et al., 2016). Although there has been extensive research into robotic assembly, relatively little has been done to address disassembly (Poschmann et al., 2020). This is because disassembly is challenging to robotise due to variability in the condition of the returned products and the required dexterity in robotic manipulations (Vongbunyong & Chen, 2015). Furthermore, disassembly is usually more stochastic than assembly, as it has to contend with used products of uncertain shapes, sizes, and conditions.

Solutions to various aspects of the disassembly automation problem have been proposed. A line of research focuses on modelling, planning and optimising the disassembly sequences (Guo et al., 2021). Some other works have proposed methods to develop robotic disassembly cells that are usually designed in a human-robot-collaborative way to increase the flexibility of the disassembly systems (Wegener et al., 2015). Another approach to increase the flexibility for robots to deal with the uncertainties in disassembly is by increasing the perception ability. For example, Vongbunyong et al. (Vongbunyong et al., 2013) have utilised vision systems to address variability in the product. However, the focus of these investigations has been on the whole product disassembly level. The specific robotic disassembly operations involve the physical interactions between the robot and the objects, such as unscrewing a nut (Li et al., 2020) or removing a peg from a hole (Zhang et al., 2019). Further research is needed to fundamentally increase the efficiency and capability of robotic disassembly cells.

A challenge in developing a robotic disassembly cell is rapidly planning the robot's control strategy for tasks that involve frequent contact (Li et al., 2020; Zhang et al., 2019), which is not widely studied in robotic disassembly but has been extensively studied in the field of robotic assembly. In assembly, a commonly studied contact-rich task is to insert a peg into a hole with a small clearance (Xu et al., 2019a). The classical approach is to obtain a mathematical description of the component geometries for predicting the evolution of the contact force. Then, based on the control strategies (rules), the robot adjusts its motion to minimise the load exerted on the part and the robot (Wan et al., 2017). Alternatively, passive compliance devices have been proposed, but most designs only work for fixed component geometries, which is impractical when robots have to deal with components of different dimensions (Whitney, 1982).

However, the coding and parameter-tuning process of the rule-based approach to generate skills can be time-consuming, and the task environment needs to be accurately measured and carefully documented (Li et al., 2020; Wan et al., 2017; Whitney, 1982; Xu et al., 2019a; Zhang et al., 2019). As opposed to obtaining the controller based on analysing the specific component geometries and parameter tuning, the data-driven approach, such as reinforcement learning (RL), can be used to learn a controller for contact-rich robotic disassembly operations.

RL is a form of machine learning where decision-making skills are acquired by trial and error (Sutton & Barto, 2018). Apart from the benefits of the data-driven approach described above, RL is attractive in situations where no data are accessible, as in the case of supervised learning (Ma et al., 2021), since RL allows learning from scratch. As described later in this chapter, some RL algorithms can also learn directly from an existing dataset. Apart from assembly, RL has achieved success in many other robotics problems, such as grasping (Levine et al., 2017) and motion planning (Li et al., 2022).

Applying RL to robotic manipulation tasks reduces programming complexity, coding time and dependence on the operator (Kober et al., 2013). For robotic operations learnt by trial and error, there is no need to fine-tune the feedback controller (Roveda et al., 2018) or develop a specific rule-based program. Inoue et al. (Inoue et al., 2017) applied RL to the assembly problem in real-life settings, but the method is limited to selecting actions from a discrete domain. Thomas et al. (Thomas et al., 2018) have shown that by utilising CAD models and motion planners, tasks can be learned and generalised, but the CAD models might not always be available or accurate, especially for disassembly tasks, where the condition of the product can be highly varied. Ren et al. (Ren et al., 2018) and Luo et al. (Luo et al., 2019) have studied RL applied to impedance control. However, impedance control may not be practicable in every scenario, as it requires a fast controller and rapid sensory feedback. Additionally, Xu et al. (Xu et al., 2019b) and Fan et al. (Fan et al., 2019) demonstrated that high-precision assembly tasks can be achieved using RL.

Apart from the articles mentioned in the previous paragraph that focus on generating robotic operational skills by RL, another aspect of the research is the generalisation of the learned skills through skill transfer (Kroemer et al., 2021). Various types of skill transfer have been proposed. For example, to bypass the common issues of RL associated with lack of data and unsafe exploration by robots, simulators have been utilised to generate skills for virtual robots; the virtually generated skills can subsequently be transferred to the physical world (Salvato et al., 2021). Another type aims to transfer the skills learned in one task to others within the task family (Pastor et al., 2009). Finally, skill transfer among robots with different configurations has also been demonstrated in the literature (Gupta et al., 2017).

However, to the best of the authors' knowledge, the influence of precision (repeatability) of the robot, or more generally, the mismatch between the action command and the actual actions

being taken, has not been studied in the context of RL and robotics. For example, the precision of a robot could change with time and thus could be different between when it learned a task and when it had to execute that task. In another scenario, the policies learnt by a master robot may be required to replicate on other robots with different precision. Thus, precision is an important factor when deploying industrial robots in manufacturing-related operations, as it is a fundamental equipment property. Without this understanding, a knowledge gap exists in how a robotic skill obtained through RL can be mass-implemented.

The contributions of this work are as follows: (i) developing an RL-based training platform for robot learning contact-rich disassembly operations, empirically studying (ii) the effect of changes in the precision of the robot on the performance of the controllers learned by RL methods, and (iii) the performance of the controller implemented by a robot if the controller is learned and transferred from a robot with a different precision.

The chapter is structured as follows. In Section 3.2, the problem of training a robot by RL is formulated. In Section 3.3, the training platform, a methodology to modify the robot's precision, and the experimental procedures are introduced. The experimental results are presented and discussed in Section 3.4. Section 3.5 concludes the chapter and suggests future research directions.

3.2. Problem Formulation

RL is a form of machine learning that improves the performance of a system on one or a set of tasks, $\mathcal{T}$, with an increased amount of data, $\mathcal{D}$, or experience, E) (Sutton & Barto, 2018). The RL problem can be modelled as a Markov Decision Process (MDP), where the state, $s_t \in S$, is the current description of the environment at the time step, t, and $a_t \in A$ denotes the action selected by the decision-making agent. In RL, the agent constantly makes decisions according to the state and the decision-making policy $\pi(a_t|s_t, \theta_\pi)$, which is a stochastic or deterministic mapping from states to actions, where θ_π denotes the parameters of the policy. By evaluating the received state feedback through a reward function, $r_t(s_t, a_t, s_{t+1})$, the RL algorithm can update the policy, π_θ, to maximise the expectation of the total reward, R, defined by

$$R = \sum_{t=0}^{T} \gamma^t \, r_t(s_t, a_t, s_{t+1}) \tag{3.1}$$

where $\gamma \in [0,1]$ is the discount factor, and T is the termination time step for an episodic training process. RL tasks can be divided into continuous and episodic learning tasks. As many robotic tasks, such as pick-and-place and pin-hole separation, are not continuous but involve resetting and rebooting, only episodic learning tasks are considered in this work.

A robotic disassembly task can also be modelled as an MDP. The state, s_t, can be defined by

$$s_t = \left[x, y, z, u, v, w, F_x, F_y, F_z, T_x, T_y, T_z\right] \tag{3.2}$$

where x, y and z denote the Cartesian coordinates of the robot flange, u, v and w denote the rotation angle of the robot flange, F_x, F_y, F_z, T_x, T_y and T_z denote the force and torque acting along the x, y and z axes, respectively, while the action, a_t, is defined by

$$a_t = [\Delta x, \Delta y, \Delta z, \Delta u, \Delta v, \Delta w] \quad (3.3)$$

which is the movement command sent to the robot from the controller at each time step. Similar ways of defining actions, states, and the reward function have been successfully implemented in other studies (Inoue et al., 2017; Ren et al., 2018; Xu et al., 2019b).

3.3. Robotic Disassembly Training System Setup and Robot Repeatability Modification

This section introduces the platform set up to train a robot to perform contact-rich disassembly tasks. The task of removing a door-chain bolt from a lock is used as a case study to validate the feasibility of the training platform. That task could be troublesome for robots with low precision or compliance due to the small clearances involved. The main challenge is the nonlinear evolution of the contact force during the task. For this study, a modified version of the popular Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2016) with delayed updates is selected to demonstrate the use of RL to enable a robot to learn the task. The reasons for adopting this algorithm will be explained, although alternative RL algorithms could equally be used (Fujimoto et al., 2018).

3.3.1. Training Task and Platform

3.3.1.1. Training Task - Sliding a Bolt along a Groove

Figure 3.1(a) shows the components of the door-chain lock used in this study. The aim of the task is to train a robot to slide the bolt along a door-chain groove to the end where the clearance is large enough to allow it to be pulled out. Although a person would have no difficulty sliding the bolt, when a traditionally programmed robot operates, the bolt could jam if the axis of the groove is different from that the robot has been programmed to move along. That situation could happen if, for example, the fixture holding the lock has been accidentally rotated. In other words, the traditional way of programming such tasks would require knowledge about whether the object's axis and prefer it to be aligned with one axis of the robot coordinate system. In contrast, the learning-based skill acquisition only requires a rough direction to move along, e.g. along the $+x$ direction of the robot flange coordinate, without the need to know the groove's axis.

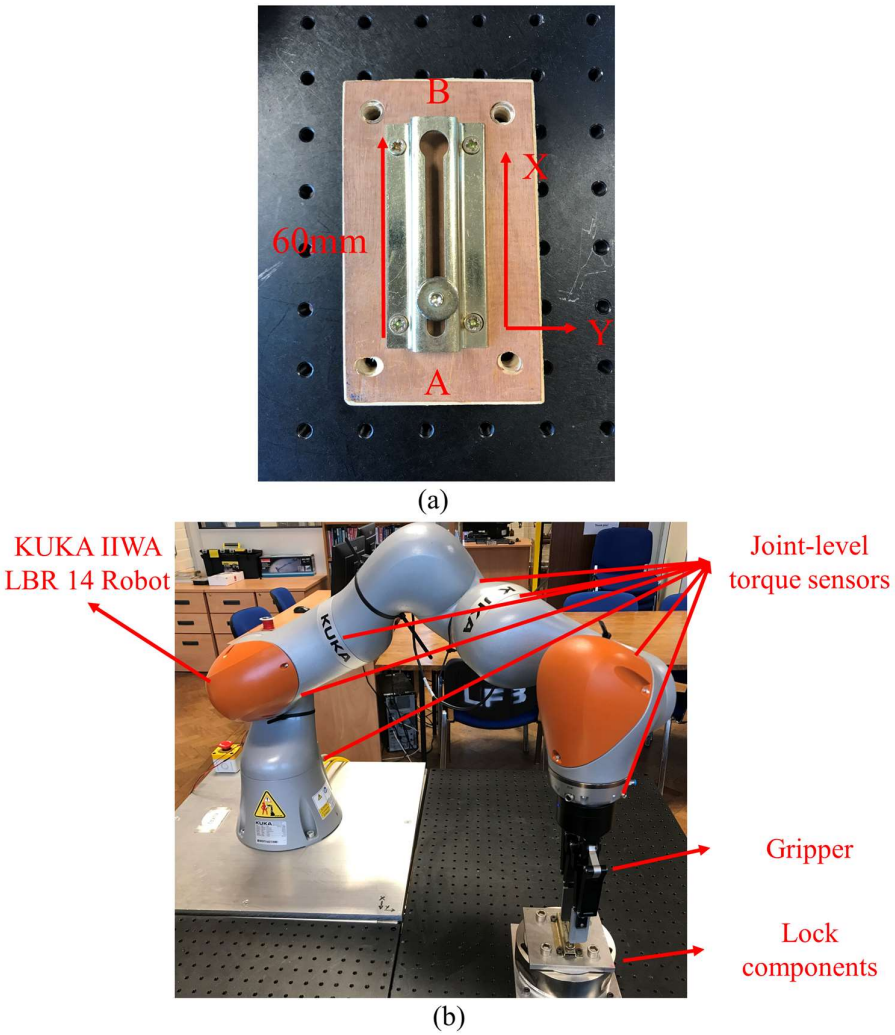


Figure 3.1: Task illustration of (a) the bolt and the groove, (b) training platform setup with the KUKA IIWA LBR robot and a Robotiq 2-finger gripper. The aim of the task is to slide the bolt to the removal point. The clearance between the pin and the groove is less than $1mm$.

The training process is considered episodic, where for each episode, the robot grips the bolt from the same starting position (point A in Figure 3.1(a)) and tries to move it to point B. Throughout the process, the force and torque information is constantly monitored to ensure that the robot does not experience excessive load so that the parts or the robot will not be damaged. The robot is encouraged to take actions that would maximise the accumulated discounted reward according to Equation (3.1), where the reward function, r, in this case, is defined as

$$r = w_{dis}(x_{t+1} - x_t) - w_f \frac{F_{norm}}{F_{max}} - w_t \frac{T_{norm}}{T_{max}} \tag{3.4}$$

where x_t denotes the coordinate of the robot flange along the x-axis at the current time step, and ${F_{norm}}^2 = {F_x}^2 + {F_y}^2 + {F_z}^2$ and ${T_{norm}}^2 = {T_x}^2 + {T_y}^2 + {T_z}^2$ denote the squared normalised force and torque experienced around the three axes. F_{max} and T_{max} are the maximum allowed force and torque on the flange of the robot during the task, and w_{dis}, w_f, and w_t are the coefficients of the reward terms. In this case, w_{dis}, w_f, and w_t are set at 0.6, 0.2, and 0.2, respectively. The purpose of the weights is to bring the terms of the reward functions into the same order of magnitude, and they can also be used to control the relative importance of each term depending on the preference of the practitioner. They can be selected arbitrarily as long as the result is that the terms in the reward function are on the same order of magnitude. If F_{norm} or T_{norm} is greater than F_{max} or T_{max}, respectively, or the maximum time step has been reached, the episode is terminated and labelled a failure. In this experiment, the safety force is $30N$, the safety torque is $5Nm$, and the maximum time step is 100.

3.3.1.2. Training Platform and Architecture

Fig. 1(b) shows the training platform setup. A KUKA iiwa LBR 14 robot (KUKA, 2022) is deployed in this case study. The robot has the feature that can estimate the force, F, and torque, T, applied at the robot flange through measurements from integrated joint-level torque sensors. This offers an alternative means to use an external 6-axis F/T sensor to acquire force/torque information. Although the information obtained using this method is relatively noisy, due to friction and error accumulation from multiple sensors, the method offers great simplicity during system integration. The experiment shows that this method is sufficient for this case study. A Robotiq two-finger gripper is used as the end effector to grip the bolt.

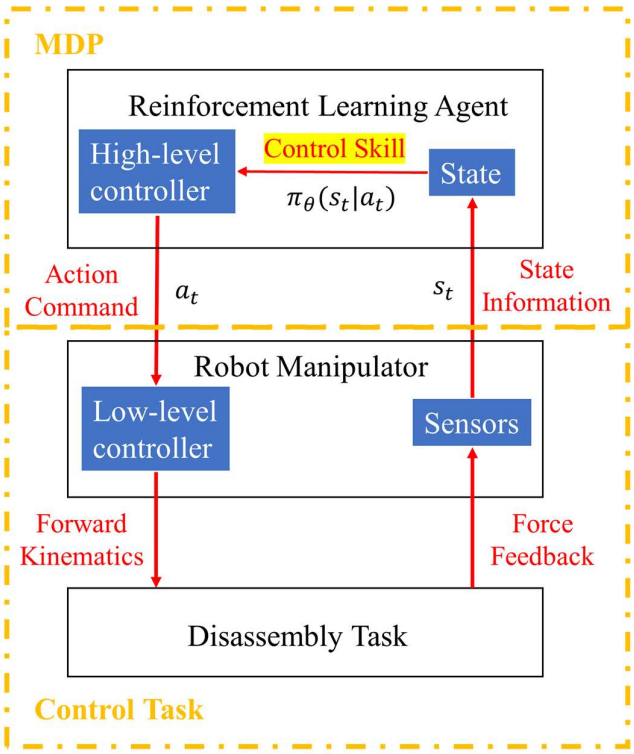


Figure 3.2: Architecture of the robot training platform. The main aim of the platform is to generate a control skill that a robot can take actions based on the state information received from the task by modelling the robotic disassembly task as an MDP.

Figure 3.2 shows the architecture of the robotic training platform. The RL unit in Figure 3.2 is a standard PC (Intel(R) Core(TM)i5-6500 CPU 3.20 GHz), and the PC communicates with the robot through the TCP/IP protocol. The RL unit selects actions based on the current policy and the state information gathered from the robot. It then sends an action command to the robot and waits for the feedback from this action. This communication process runs in cycles and stops when the termination condition of the episode has been reached.

3.3.2. Deep Deterministic Policy Gradient algorithm

The DDPG algorithm, originally proposed by Lillicrap et al. (Lillicrap et al., 2016), is a model-free off-policy RL algorithm based on the actor-critic architecture. In this case, DDPG is

selected among the other RL algorithms for two reasons. First, it can select actions from a continuous action domain. Second, as an off-policy learning algorithm, it has the capability to learn directly from a dataset instead of entirely depending on actual interactions with the environment. Thus, the DDPG algorithm (and other actor-critic off-policy algorithms) are more suitable for learning industrial robotic operations, which often require continuous control. Additionally, in industrial robotic operating environments, a dataset is usually available.

In the original paper (Lillicrap et al., 2016), the neural networks are updated after each time step. However, updating all the parameters of the algorithm after each time step will increase the computational load between each time step, thus reducing the response rate of the controller. This factor is often neglected in simulated control tasks but is important in real-world robotic learning. Therefore, the policy updates are delayed after each episode is finished, as shown in Algorithm 3.1.

Table 3.1 lists the main hyper-parameters used in this experiment. The hyper-parameters used in this work are obtained by trial and error, and the optimisation or evaluation of the performance of the algorithm for different hyper-parameters is not considered in this work. The neural networks used for estimating actor, $\pi(s|\theta^{\pi})$, and critic, $Q(s|\theta^{Q})$, are updated by the Adam optimiser (Kingma & Ba, 2015) with the hyper-parameters described in Table 3.1. The Adam optimiser is widely used in deep RL studies (Fujimoto et al., 2018; Lillicrap et al., 2016; Ren et al., 2018). The neural networks are developed based on the TensorFlow platform (version 1.15.3) in the Python programming language (version 3.7).

Algorithm 3.1: DDPG with Delayed Update.

1 Initialise actor network, $\pi(s|\theta^{\pi})$, and critic network, $Q(s,a|\theta^{Q})$;

2 Initialise target actor network, $\pi'(s|\theta^{\pi'})$, with parameters $\theta^{\pi'} \leftarrow \theta^{\pi}$, and target critic network, $Q'(s,a|\theta^{Q'})$, with parameters $\theta^{Q'} \leftarrow \theta^{Q}$;

3 Initialise replay buffer, $\mathcal{R}$;

4 **for** episode, e **do**

5 **for** time step, t **do**

6 Sample an action, a_t, from actor network $\pi(s_t|\theta^{\pi}) + \mathcal{N}(0,\epsilon)$, where $\mathcal{N}(0,\epsilon)$ is a Gaussian noise for exploration;

7 Clip a_t with action boundaries, (a_{low}, a_{high}), for safety;

8 Execute a_t, receive next state, s_{t+1}, and corresponding reward, r_{t+1};

9 Store transition tuple $(s_t, a_t, s_{t+1}, r_{t+1}, T)$ into $\mathcal{R}$, where T indicates the termination of the episode;

10 **until** any termination condition is met;

11 **for** $N = length\ of\ the\ episode$ **do**

12 Sample a batch of transitions, $\mathcal{B}(s_t, a_t, s_{t+1}, r_{t+1}, T)$, from $\mathcal{R}$;

13 Compute update target, $y_t = r_t + \gamma Q'(s_{t+1}, \pi'(s_{t+1}|\theta^{\pi'})|\theta^{Q'})$;

14 Update critic network by minimizing loss function, $L = N^{-1}\sum_N (y_t - Q(s_t, a_t|\theta^{Q}))^2$;

15 Update actor network by gradient ascent, $\nabla_{\theta^{\pi}}\mathcal{J} \approx N^{-1}\sum_N Q(s_t, \pi(s_t|\theta^{\pi})|\theta^{Q})$;

16 Soft update the target networks, $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$, $\theta^{\pi'} \leftarrow \tau\theta^{\pi} + (1-\tau)\theta^{\pi'}$;

17 Decrease the exploration by a decay constant, $\epsilon \leftarrow c * \epsilon$;

18 **until** maximum number of episodes is reached;

3.3.3. Robot with Decreased Precision

3.3.3.1. Robot Precision Modification

A method to modify the robot's precision is proposed to study the effects of robot precision on the performance of a learned controller, as shown in Algorithm 3.2. The proposed method simply adds an uncorrelated zero-mean Gaussian noise, $\mathcal{N}(0,\psi)$, to the robot actions, where ψ represents the reduction in precision in millimetres (step 6 in Algorithm 3.2).

Table 3.1: Hyper-Parameters Used in the Experiment

Neural Networks Parameters		DDPG Parameters	
Hidden layer size	50*50	Replay buffer size	1.00E+05
Hidden layer Neuron	ReLU	Batch size	30
Output layer Neuron	Tanh	Discount rate, γ	0.9
Optimiser	Adam	Soft update rate, τ	0.9
Adam, alpha	0.001	Starting exploration, ϵ	3
Adam, beta1	0.9	Exploration decay rate, c	0.995
Adam, beta2	0.999		
Adam, epsilon	1.00E-07		
Adam, AMSGrad	FALSE		

For clarity, the term *imprecision* in the following sections will be used to indicate the level of reduction in precision. For example, $\psi = 0.5mm$, or an *imprecision level* of $0.5mm$, means a $\pm 0.5mm$ error being added to a robot action. In other words, it is equivalent to a reduction of the robot's precision by $0.5mm$. In this experiment, the following imprecision levels are tested: $\psi = 0.0mm$, $0.1mm$, $0.2mm$, $0.3mm$, $0.4mm$, and $0.5mm$. It is also worth noting that the documented intrinsic repeatability of the robot used in this experiment is $\pm 0.1mm$ (KUKA, 2022). However, the robot's intrinsic repeatability does not affect each controller's relative performance since it is applied to all the controllers. Thus, the intrinsic repeatability of the robot is not involved in the following discussions.

The key difference between adding noise to the actions (i.e., the proposed method) and various action exploration methods is that in action exploration methods, the learning agent utilises noisy actions as training data. In contrast, despite the noisy actions being executed in the proposed method, the learning agent still updates by the original actions. In other words, in the

Algorithm 3.2: DDPG Training with Increased Imprecision Level.

1 Initialisation;
2 **for** episode, e **do**
3 **for** time step, t **do**
4 $a_t = \pi(s_t|\theta^\pi) + \mathcal{N}(0, \epsilon)$;
5 Clip a_t with action boundaries, (a_{low}, a_{high});
6 $a_t^\psi = a_t + \mathcal{N}(0, \psi)$;
7 Clip a_t^ψ with new action boundaries, $(a_{low} - \psi, a_{high} + \psi)$;
8 Execute a_t^ψ and receive feedback;
9 Store transition tuple $(s_t, a_t, s_{t+1}, r_{t+1}, T)$ into $\mathcal{R}$;
10 **until** any termination condition is met;
11 Update parameters;
12 **until** maximum number of episodes is reached;

proposed method, the learning agent is unaware that noisy actions are executed (step 9 in Algorithm 3.2).

3.3.3.2. Experimental Procedures

The training is repeated ten times for each imprecision level to test the success rate. For each training, if the robot can reach the destination, i.e., x-displacement greater than $55mm$, the training is deemed a success. Training that does not have any successful episodes within 200 episodes is labelled failure.

After each training, a validation is followed up in which the exploration coefficient, ϵ, is set to zero to validate the performance of the trained controller. Then, the trained controllers operate ten times consecutively while the robot remains at the same imprecision level. The experimental results described in later sections are based on the results from the validation phases unless specified.

3.4. Experimental Results and Discussions

This section gives the results of the training. Figure 3.3 shows the photographs from the training process of one specific training. The robot takes random actions at the beginning of the training, and these actions can easily trigger the termination of the episode, which leads to a reset. It also shows that the robot progresses in learning the task. Each episode lasts approximately 5s to 40s in real time, depending on the triggering of the termination signal.

The success rate for each imprecision level is summarised in Table 3.2. The first observation is that even when the repeatability is modified down to as low as $\pm 0.5mm$, which is five times the robot's intrinsic designed precision ($\pm 0.1mm$), the system is robust enough to overcome the low repeatability of the robot and results in a 40% success rate.

Figure 3.3: Photographs taken from the footage of one training session: (a) at the beginning of the episode, (b) after several episodes of training. It can be seen that the robot learned the sliding task by trial and error. (Note: t in here represents the real time of the video, not the time step in MDP.)

Table 3.2: Training Success Rate for Robots with Different Imprecision levels

Imprecision Level, ψ (mm)	Training Success Rate [a]
0.0	70%
0.1	60%
0.2	30%
0.3	30%
0.4	60%
0.5	40%

[a] The success rates are calculated based on ten training sessions.

The second observation is that the success rate does not necessarily decrease when a robot with less repeatability is deployed. This shows the robustness of the training system despite noisy actions being executed. Additionally, there are a few random factors that are associated with the training, such as the initialisation of the neural network parameters and the exploration parameter, ϵ. A fairer comparison could be made with a controlled random seed. However, experiments with controlled random seeds are less meaningful in a real-world setting.

3.4.1. Disassembly Task Training Results

First, the training performance of the robot without any modified repeatability is analysed. In other words, the following analysis concerns the cases where the imprecision level is at $0.0mm$.

The results are from a typical successful of training (training round number 5). Figure 3.4(a) plots the learning curve of the agent gaining more episodic rewards with an increased number of episodes. As shown in the graph, the learning becomes stable after convergence. The validation results are plotted in Figure 3.4(b), showing that the learned controller can stably repeat the operation.

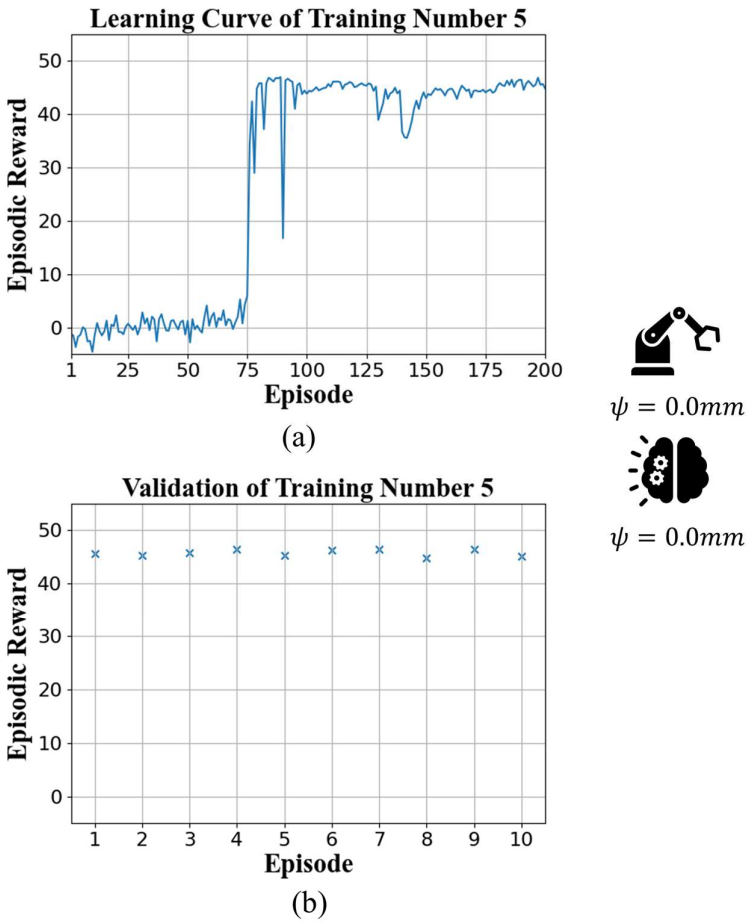


Figure 3.4: Episodic reward against episodes for a specific successful training without modification of robot precision: (a) learning phase, (b) validation phase. The learning curve in (a) shows that the agent successfully learned the task at approximately episode 80. The validation curve in (b) shows that the learned controller can stably repeat the learned skill.

Figure 3.5 shows the state information of a typical successful episode (learning 5, validation phase, episode 8). Figure 3.5(a), (b) and (c) show the displacement, average axial torque and average axial force, respectively. It can be observed that the robot is able to slide the bolt along the groove to the destination point with an average force of $2.0N$ and an average torque of $0.91\ Nm$ experienced by the robot flange across the episode, which is measured without an external F/T sensor but by the internal joint-level torque sensors alone. These force and torque

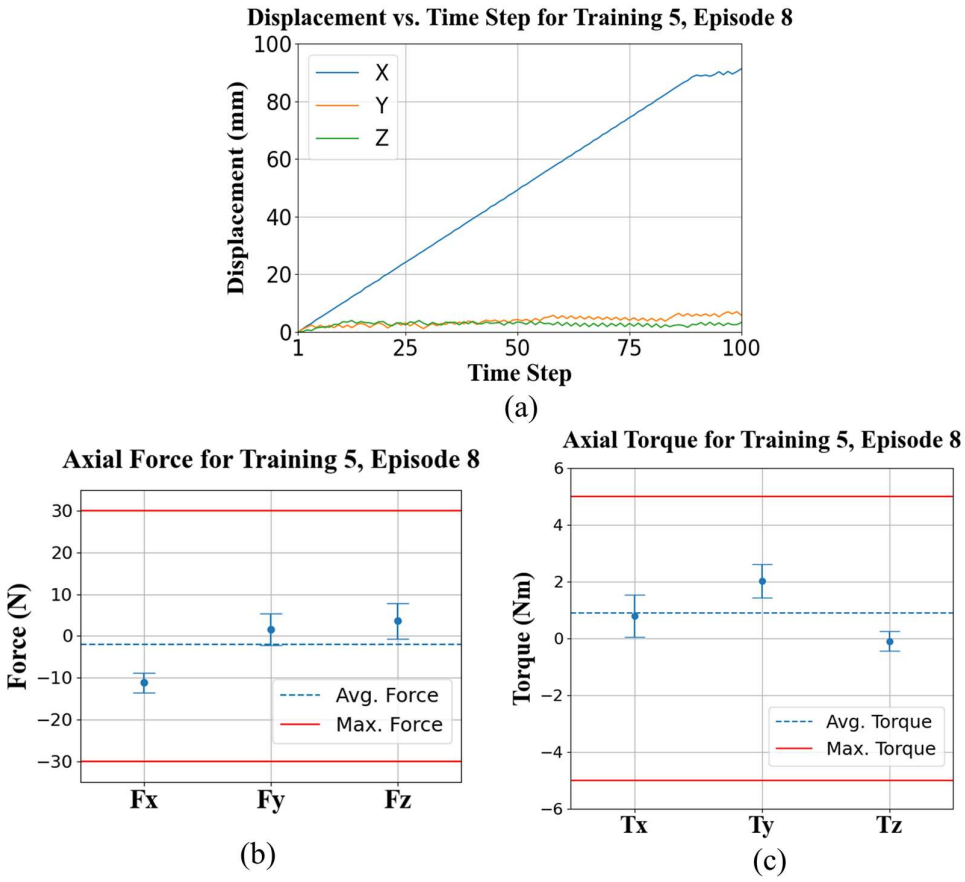


Figure 3.5: State information from a typical successful episode: (a) displacement of robot, (b) average axial force, (c) average axial torque. These results demonstrate that the robot can slide the bolt to the destination and maintain the force and torque within safety limits.

values are far from the maximum values set for the task. By comparison, the payload of the manipulator is roughly $140N$, and the maximum torque for the weakest joint is $40Nm$ (KUKA, 2022). Therefore, the goal of training the robot to experience low forces and torques is successful.

3.4.2. Learning Directly from Dataset

As mentioned in Section 3.3.2, one of the motivations for selecting the DDPG algorithm is that it is an off-policy algorithm, meaning it can learn policies directly from a previously generated dataset (Sutton & Barto, 2018). In other words, the algorithm can learn not only from the

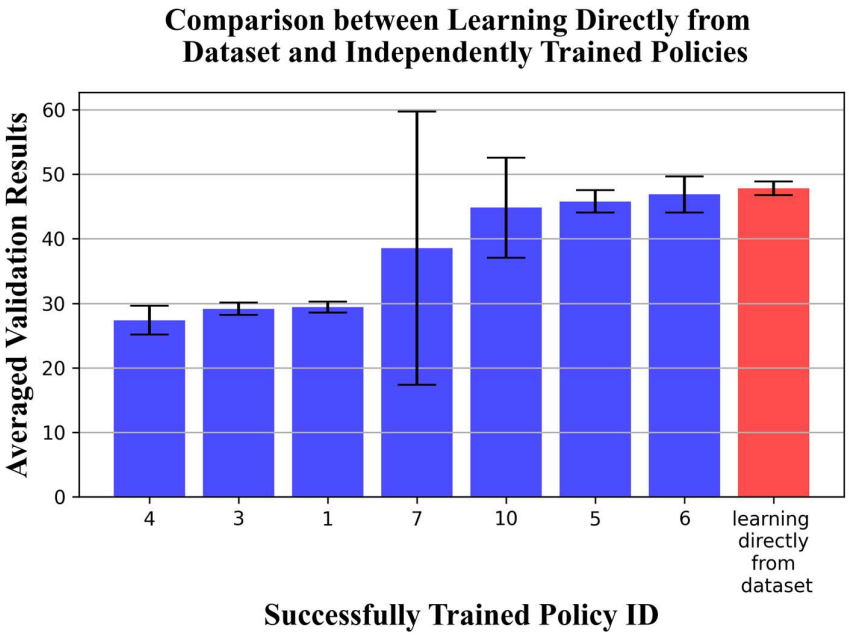


Figure 3.6: Comparison between policy that learns directly from a dataset and all the other independently trained policies. The results are obtained by consecutively repeating the learned controller ten times. It demonstrates the ability of the RL method to learn directly from a dataset without taking any real-world actions. It also shows that if a robot can learn from a dataset larger than the dataset generated by individual training, the learning performance can outperform every individual. .

transition data of the current training but also from data generated by other training. This feature is particularly useful if a dataset is available before the training.

An experiment is designed to confirm this aspect of the algorithm. After the training is repeated ten times in $\psi = 0.0mm$, all the transition data are gathered into one dataset. Then, the robot is required to learn only from this dataset without taking any real-world actions.

Figure 3.6 compares the performance of this policy that only learns from the dataset with that of all the other independently trained successful policies. The result shows the ability of the training platform to learn skills directly from a dataset without taking any real-world actions. It also shows that the policy that learns directly from the whole dataset outperforms all the

other policies in terms of averaged rewards and stability. This is because a larger dataset contributes to a better learning performance as long as all the data have been gathered by the same procedure (Sutton & Barto, 2018).

3.4.3. Robots with Different Degrees of Repeatability

The results of the validation phases for robots with different imprecision levels are shown in Figure 3.7.

An instability index, I_s, defined as

$$I_s = \sum_i \frac{\sigma_i}{N_i} \tag{3.5}$$

is proposed to compare the stability of the learned policies, where σ_i is the standard deviation of the episodic reward for a single validation episode and N_i is the number of successfully learned episodes. It can be seen that although the task can be learned successfully, the stability of the learned policies decreases, as shown in Figure 3.9. The relationship between the imprecision level and instability index is roughly linear.

The reason is that the decreased repeatability increases the probability of taking actions that cause the maximum force and torque to be exceeded. Therefore, for the same learning task, if only one manipulator is used, in other words, without considering the transfer of the learned skills, the precision of the robot is a factor to consider. An intuitive conclusion that can be drawn is that the better the precision is, the more stable the learned skills.

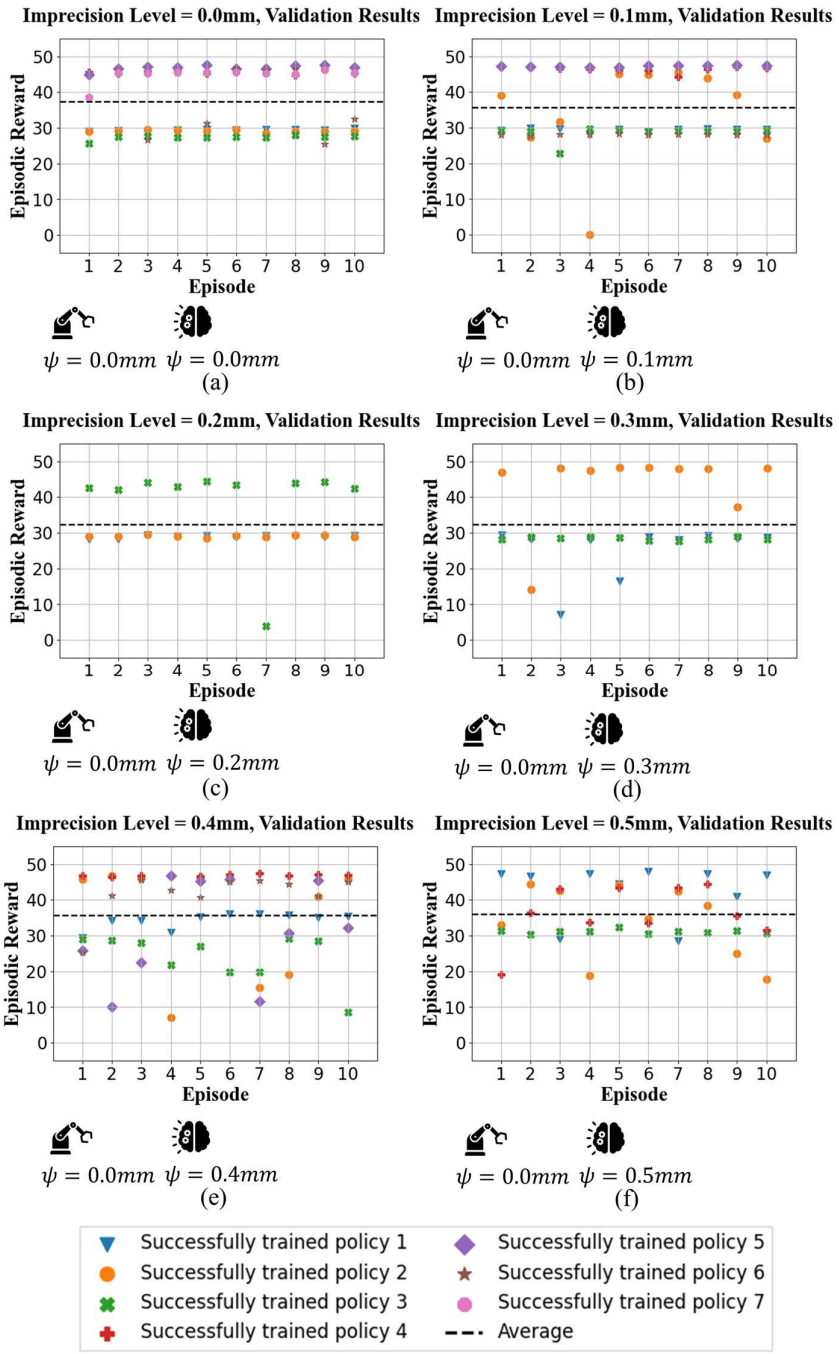


Figure 3.7: Validation results of the successfully trained policies for imprecision levels of (a) $0.0mm$, (b) $0.1mm$, (c) $0.2mm$, (d) $0.3mm$, (e) $0.4mm$, and (f) $0.5mm$. The plots show that although some training is successful even when the imprecision is increased to $0.5mm$, some of the learned controllers are not able to repeat the learned motions stably for consecutive times. This observation becomes more obvious with increased imprecision levels.

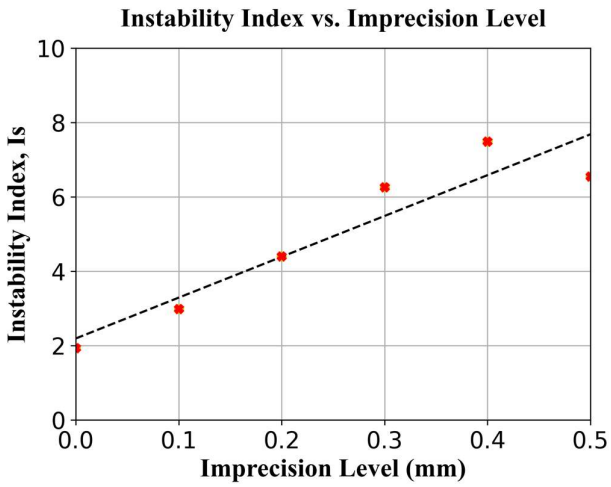


Figure 3.8: Instability index plot against imprecision level. This plot demonstrates that the stability of the learned skills will decrease if the precision of the robot is low.

3.4.4. Transferability of Learned Policies

Another main motivation of this research is to examine the transferability of the learned policies among robots with different precision. In this subsection, the transferability of the learned skills is empirically analysed.

A series of experiments are designed to transfer the successfully learned policies from one imprecision level to the others. Three levels are selected in these experiments: $\psi = 0.0mm$, $0.3mm$, and $0.5mm$. For example, the skills that are successfully trained when $\psi = 0.0mm$ are repeated on robots with imprecision levels of $0.3mm$ and $0.5mm$.

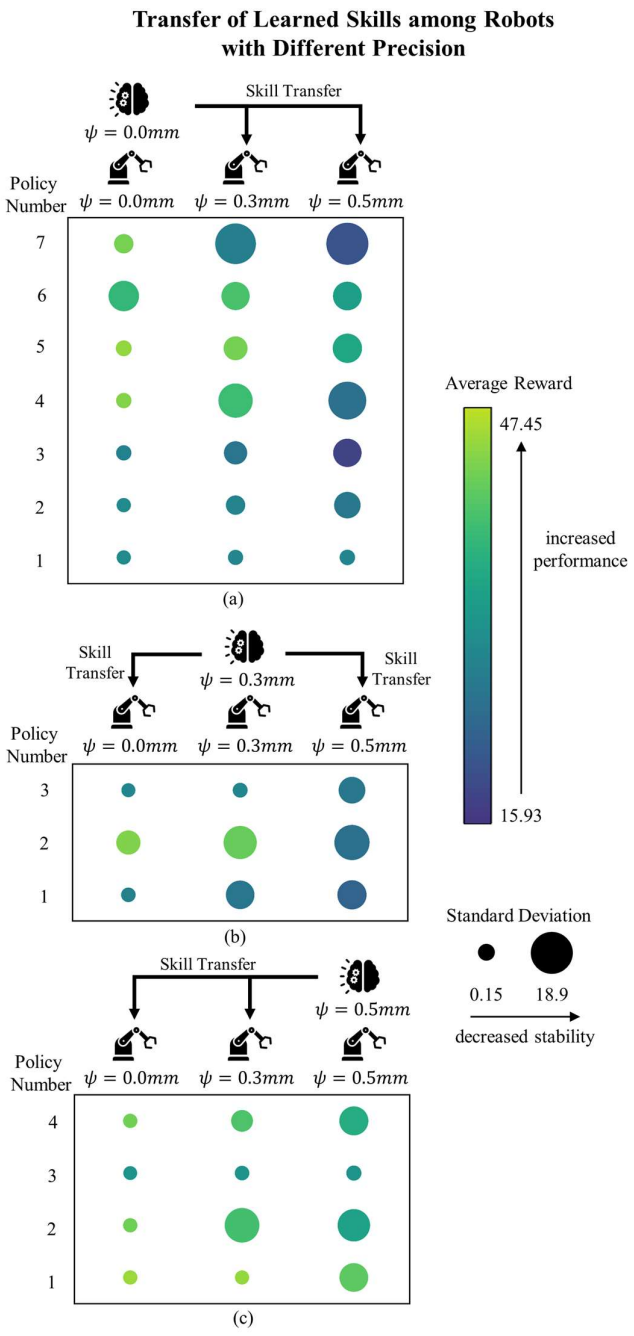


Figure 3.9: Qualitative visualisation of transferring the skills learned from (a) $\psi = 0.0mm$, b) $\psi = 0.3mm$, (c) $\psi = 0.5mm$, to the others. The results are obtained by consecutively repeating the learned controllers by a robot with other imprecision levels for ten times. From left to right, most of the circles representing the performance of the controllers become darker and larger, meaning that the skills received less average reward and became less stable. This result does not depend on whether the skills are transferred from a low-precision robot to a high-precision robot or from a high-precision robot to a low-precision robot.

The results are shown in Figure 3.9. It is interesting to note that, on the one hand, skills learned by a robot with low precision can transfer to a robot with high precision and become better at performance and stability. On the other hand, policies learned by a robot with high precision perform worse on a robot with low precision.

Moreover, this transferability study is further tested on a simulation platform using other RL algorithms from the actor-critic family: Soft Actor-Critic (SAC) (Haarnoja et al., 2018) and Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al., 2018). Since DDPG, SAC and TD3 are off-policy actor-critic methods, they are expected to share similar results to those described for physical experiments.

The simulation model is developed on a commercial multibody dynamics simulation software, MSC Adams (Student Edition 2022.1), based on the Euler-Lagrange method to create equations of motion. Although a simulation model can only approximate the task dynamics, the model is tested to produce similar behaviours of the real-life task. Furthermore, error signals are added to all dimensions of the states, described in Equation (3.2), to mimic the measuring errors from the sensors.

Each policy is trained with the RL algorithm for 500 episodes and validated on the other imprecision levels for 200 episodes. The results of the experiments are shown in Table 3.3, and they are consistent with the physical experiment results: out of 36 times skill transfer, the skills learned from a low-precision robot will show better performance and stability on a high-precision robot; the skills learned from a high-precision robot will show reduced performance and stability on a low-precision robot.

This finding can be explained by the degree of difficulty that the system would need to overcome to learn the task successfully. The lower the precision during training, the greater the

Table 3.3: Skill Transfer for Three Different Actor-Critic RL Algorithms

RL algorithms	DDPG			SAC [c]			TD3 [d]		
imprecision level	**0.0mm** [a]	0.3mm	0.5mm	**0.0mm**	0.3mm	0.5mm	**0.0mm**	0.3mm	0.5mm
performance [b]	**34.5**	29.7 (-13.9%)	22.6 (-34.6%)	**54.8**	45.0 (-17.8%)	26.3 (-52.0%)	**65.8**	42.1 (-36.0%)	22.9 (-65.3%)
instability	**5.6**	6.4 (+12.8%)	8.2 (+46.3%)	**7.8**	11.0 (+41.3%)	12.7 (+63.7%)	**1.5**	14.6 (+865.1%)	11.1 (+636.9%)
imprecision level	0.0mm	**0.3mm**	0.5mm	0.0mm	**0.3mm**	0.5mm	0.0mm	**0.3mm**	0.5mm
performance	35.8 (+9.4%)	**32.7**	18.0 (-44.9%)	53.3 (+10.7%)	**48.1**	32.1 (-33.3%)	68.4 (+11.1%)	**61.6**	44.2 (-28.3%)
instability	0.7 (-84.2%)	**4.6**	14.0 (+205.6%)	3.5 (-37.9%)	**5.6**	12.6 (+122.2%)	2.0 (-77.4%)	**8.8**	15.9 (+81.4%)
imprecision level	0.0mm	0.3mm	**0.5mm**	0.0mm	0.3mm	**0.5mm**	0.0mm	0.3mm	**0.5mm**
performance	49.8 (+58.4%)	41.9 (+33.4%)	**31.4**	57.12 (+79.8%)	49.97 (+57.3%)	**31.77**	57.8 (+32.9%)	53.9 (+23.9%)	**43.5**
instability	3.0 (-74.7%)	7.2 (-38.4%)	**11.8**	3.85 (-78.5%)	7.43 (-58.6%)	**17.95**	1.0 (-85.8%)	3.1 (-54.5%)	**6.9**

[a] The results in bold represent the original skill that is transferred to the other imprecision levels, and the numbers in brackets represent the relative increment or reduction of the performance and instability after the transfer comparing against the original skill (the numbers in bold).

[b] Performance and instability is measured by taking the average and standard deviation of the results of repeating the learned skill on each imprecision level for 200 times.

[c, d] The hyper-parameters of SAC and TD3 are kept the same the ones for DDPG where possible. The other hyper-parameters of SAC are: target noise=0.01, target noise limit=0.01, policy delay=2. The other hyper-parameters of TD3 are: entropy regularization coefficient=0.2. Since the aim is to study skill transferability, the hyper-parameters are not optimised for the task.

difficulty and effort for learning, thus, the more generalisation capability of the learned skills. From this perspective, the mechanism is similar to domain randomisation (Chebotar et al., 2019): adding randomness to the observations to improve the robustness of the learned policies. In the case of this work, a low-precision robot is used to avoid overfitting the skills to high-precision robots. However, to the best of the authors' knowledge, whether randomness is added to the system from the states or the actions, the underlying mechanism of randomness improving the learned policies has not been theoretically proven.

This knowledge of the relationship between the precision of the machine and the RL algorithm can be exploited to improve RL-based training systems. A possibility could be to develop a "training robot" with a lower level of repeatability than the other robots within a factory and exploit that robot to learn policies that can be transferred to all the robots with higher repeatability.

3.5. Conclusion

This chapter has presented an RL-based training platform for robotic disassembly operations. The results show that the control skill of a robot performing a contact-rich manipulation task can be learned by a data-driven approach.

As the authors did not have access to robots with different precision levels, to study the robustness of the proposed learning method, a technique to change the precision artificially was devised to enable a single robot to be used. In this study, the robot's precision was modified to as low as $\pm 0.5mm$, and the robot can still learn successfully. Additionally, the results empirically demonstrated that the stability of the learned controllers depended on the precision of the machine. The study has also shown that the robot can learn directly from a dataset previously generated by other training without taking any actual actions. Finally, the transferability of the learned skills was empirically studied by applying the proposed precision modification technique in real and simulated environments. It has been found that the skills learned from a low-precision robot can be transferred to a high-precision robot, and they have shown increased performance and stability after the transfer. On the other hand, policies learned from a high-precision robot will obtain less performance and stability when implemented on robots with lower levels of precision.

In the future, the following aspects of the work could be investigated. First, this chapter observes a pattern in the transferability of the skills, but the underlying principles and mechanisms require further investigation. Second, the application of the training platform could be extended to more complex disassembly tasks, for example, those involving a combination of movements such as twisting and pulling. Third, the findings shown in this chapter (in particular about skill transfer) can be tested on other robotic operations and RL

algorithms. Third, the results about the transferability of robots with different precision levels should be further validated on actual different robots and in large-scale applications.

4. Digital-twin-assisted robotic disassembly task training with system identification by the Bees Algorithm

4.1. Preliminaries

As discussed in the previous chapters, applying reinforcement learning (RL) in robotic disassembly skills generation can be useful for the rapid deployment of industrial robots in disassembly automation scenarios. However, applying RL in robotics exhibits a low sample efficiency problem associated with its exploration process (Chatzilygeroudis et al., 2020), which is also accompanied by risky exploring actions and excess wear on the robots and the objects.

An intuitive and promising solution is to train the robot using a simulation-to-reality (sim-to-real) transfer approach. In sim-to-real transfer, the decision-making agent is first trained in a simulation environment and then deployed to a real-world environment for operation or subsequent training (Salvato et al., 2021). Since data can be generated safely and cheaply in a simulated environment, robots can freely explore the dynamics of the environment. Furthermore, tasks that are too complex to be trained in reality can now be trained in simulated environments, such as manipulating Rubik's Cube (Akkaya et al., 2019) and locomotion for quadruped robots (Tan et al., 2018).

However, the sim-to-real transfer of the robot skills has been known to have a critical issue known as the "reality gap", which describes a mismatch in performance when directly implementing the learned policy into the real-world setting, as simulators can only approximate the real process to a certain degree. Also, some phenomena are too costly to be modelled in a

simulated environment. Furthermore, building a high-quality simulation model might introduce additional expert knowledge and computational hardware, which might contradict the aim of reducing the cost of the data collection process. Thus, researchers have proposed various methods to reduce the reality gap with the current state of the simulators.

A few researchers have started to address the reality gap issue by utilising the data collected from the real world. For example, by comparing with the real-world data, the simulation models can be optimised to produce responses as close to the real-world observations as possible. In this way, a "digital twin" of the environment can be built for RL agents to train their decision-making skills (policies), and they can be subsequently transferred to the real world. Furthermore, this approach allows parameters that are difficult to measure, such as coefficient of friction and stiffness of the joints, to be identified.

However, the previous research in this direction does not include force/torque feedback in their MDP models, which are crucial in contact-rich manipulation tasks for minimising the load on the robots and the objects used in assembly/ disassembly scenarios (Tiboni et al., 2023; Tsai et al., 2021).

In this chapter, the process of building a digital-twin-assisted RL platform for robotic disassembly tasks, focusing on minimising the reaction load, is proposed and tested. Furthermore, the Bees Algorithm, a decentralised heuristic optimisation that is robust against local optima, has been used as a key method to improve the quality of the digital twin (Pham et al., 2006).

Following the Introduction, Section 4.2 describes the disassembly task training process, the physical training platform, and some practical methods to process the observations. Section 4.3 details the proposed method to increase the learning performance by the sim-to-real approach

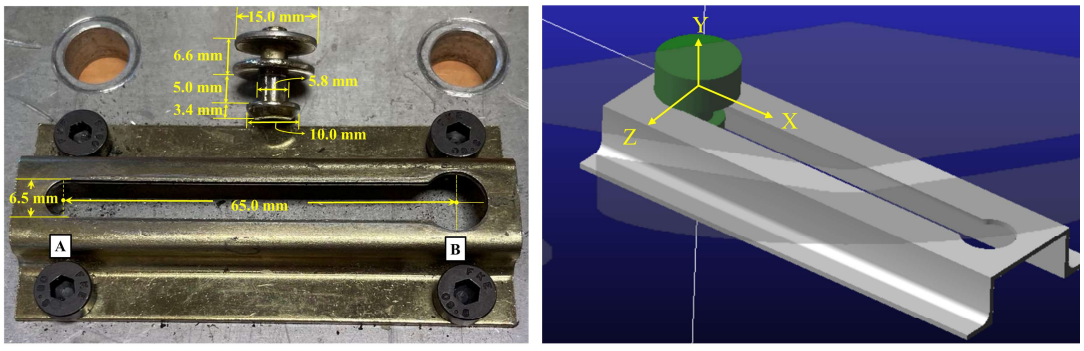
and identify the system parameters to build the digital twins of the training environment. The results of the experiments are presented and discussed in Section 4.4. Section 4.5 concludes the chapter and provides suggestions for future works.

4.2. Robotic disassembly task training

4.2.1. Disassembly task description

In robotic disassembly, an important type of manipulation task is to remove a component along a disassembling direction with minimised reaction force exerted on the robot, as the clearance between the components in this type of task is usually small enough to generate reaction force easily. To demonstrate the ability to find the correct disassembling direction and to adjust the pose based on the force and torque feedback, the task of removing a bolt from a door-chain groove, as shown in Figure 4.1, is selected as the case for the training.

Figure 4.1(a) shows the objects in the bolt sliding task with some critical dimensions. The aim is to slide the bolt from point A to point B and remove it from the groove. To demonstrate the ability of RL to generate skills with minimised expert knowledge, the coordinates of point B



(a) (b)

Figure 4.1: Illustration of the bolt sliding task the main dimensions in (a): real world and (b): simulator. The task starts with the pin inside the groove at point A. Then the robot will grasp the upper part of the bolt, slide it to point B, and take it out from point B.

are not given. Thus, the precise orientation and the distance to point B from point A are not known to the robot, and this information is crucial in the classical way of programming this type of task (Li et al., 2020; Zhang et al., 2019).

4.2.2. Physical training platform

Figure 4.2 shows the physical task training platform for the experiment. The platform deploys a lightweight robot with a 14kg payload. It is equipped with a wrist-mount 6-axis force/torque (F/T) sensor and a 2-finger parallel-jaw gripper. The tool centre point (TCP) of the operation

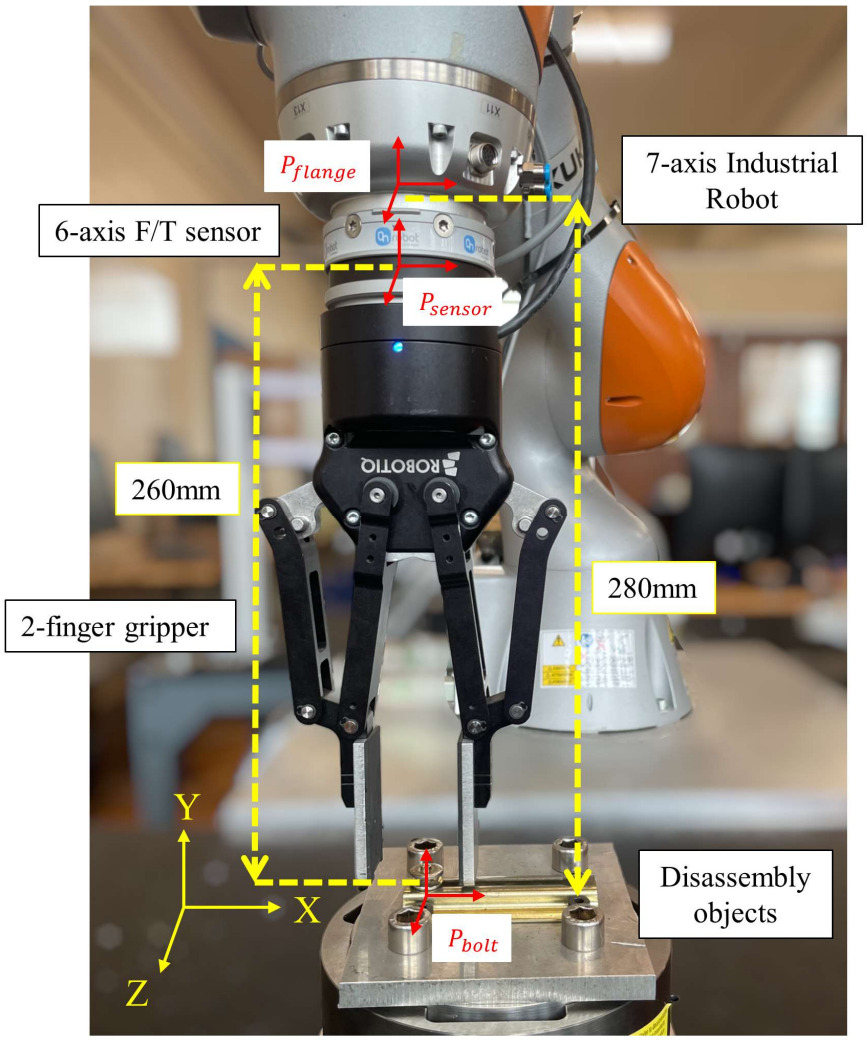


Figure 4.2: The experimental setup of the study. The training platform uses the Kuka IIWA LBR14 robot, the Robotiq 2F-140 gripper, the OnRobot Hex 6-axis F/T sensor, and an ordinary PC (not shown in the figure). The red coordinates denote the frames of the objects, which are 6-dimensional arrays of the positional information. (Note that the orientations of the red coordinates as shown in the figure do not represent the true orientation of the frames.)

is set at the P_{bolt} frame, as shown in Figure 4.2, for keeping the rotation point of the bolt and the F/T analysis point consistent with the simulation model.

4.2.3. Disassembly skill training with RL

4.2.3.1. RL problem formulation

The robotic disassembly task can be modelled as a Markov Decision Process (MDP) and solved by RL methods (Sutton & Barto, 2018). In each time step, t, an RL agent generates an action, $a_t \in \mathcal{A}$, based on the observed state, $s_t \in \mathcal{S}$, and the policy, $\pi(a_t|s_t, \theta_\pi)$, parameterised by θ. After executing the action, a_t, the next state, s_{t+1}, can be observed following a distribution, $p(s_{t+1}|s_t, a_t)$, and a reward, $r_t(s_{t+1})$, can be obtained. The aim of an RL algorithm is to maximise the total expected reward, $R = \sum_{t=0}^{T} \gamma^t r_t(s_{t+1})$, until reaching the termination time step, T, through updating the policy $\pi(a_t|s_t, \theta_\pi)$, where $\gamma \in [0,1]$ represents the discounting of rewards from future time steps.

A robotic force control problem can be represented by an MDP, with the states defined as,

$$s_t = [x, y, z, u, v, w, F_x, F_y, F_z, T_x, T_y, T_z] \tag{4.1}$$

where x, y and z denote the Cartesian coordinates of a point of reference (which is P_{bolt} in Figure 4.2 in this case), u, v and w denote the rotation angle, and F_x, F_y, F_z, T_x, T_y and T_z denote the force and torque acting along the x, y and z axes, respectively. The actions can be defined as,

$$a_t = [\Delta x, \Delta y, \Delta z, \Delta u, \Delta v, \Delta w] \tag{4.2}$$

where each element represents the movement command of the specified direction with respect to the point of reference of the manipulated object. The reward function is defined as

$$r_{t+1} = w_{dis}(x_{t+1} - x_t) - w_f \frac{F_{norm}}{F_{max}} - w_t \frac{T_{norm}}{T_{max}} + w_y \begin{cases} y_{t+1} - y_t & if\ (x_{t+1} - x_0) > 55\ and\ y_{t+1} - y_t > 0 \\ 0 & otherwise \end{cases} \quad (4.3)$$

where x_t denotes the coordinate of the robot flange along the x-axis at the current time step; ${F_{norm}}^2 = {F_x}^2 + {F_y}^2 + {F_z}^2$ and ${T_{norm}}^2 = {T_x}^2 + {T_y}^2 + {T_z}^2$ denote the squared normalised force and torque experienced around the three axes; F_{max} and T_{max} are the maximum allowed force and torque on the flange of the robot during the task; the last term is to encourage an upward movement to take the bolt out when the bolt is roughly near the hole; w_{dis}, w_f, w_t, and w_y are the coefficients of the reward terms for scaling each term to the same order of magnitude, and they are set as 1.6, 0.3, 0.3, and 5.0, respectively, in this case.

If F_{norm} or T_{norm} is greater than F_{max} or T_{max}, or the maximum time step has been reached, the episode will be terminated. In this experiment, the safety force is $20N$, the safety torque is $5Nm$, and the maximum time step is 100.

4.2.3.2. RL training procedure

The robot is trained for 100 episodes in each training. At each episode of the training, the bolt is reset to point A in Figure 4.1. Then, the robot moves the gripper to the location where the bolt can be securely gripped, zeros the sensor reading, and follows the procedures of the MDP until it meets the termination conditions.

In this work, the Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) is used as the RL algorithm (The details of the algorithm and its implementation can be found in Section 3.3.2.). The main hyper-parameters used in this training are listed in Table 4.1.

Table 4.1: Main Hyper-Parameters of the RL Used in the Physical Training.

Neural Networks Parameters		**DDPG Parameters**	
Hidden layer size	$128 * 128 * 128$	Replay buffer size	$1e5$
Hidden layer Neuron	ReLU	Batch size	2,000
Output layer Neuron	Tanh	Discount rate, γ	0.9
Optimiser	Adam	Soft update rate, τ	0.9
Adam, α	0.0005	Starting exploration, ϵ	5
Adam, β_1	0.9	Exploration decay rate, c	0.95
Adam, β_2	0.999		
Adam, ϵ	$1e-7$		
Adam, AMSGrad	FALSE		

4.2.4. Physical observation processing

As mentioned in Section 4.2.2, the reference frame for observing the state information in the physical setting should be consistent with that in the simulation models. Thus, the following three processes need to be sequentially performed: (1) aligning the orientations of the frames, (2) gravity compensation, and (3) transforming the F/T measurements from the 6-axis F/T sensor to the reference frame of interest (in this case, from P_{sensor} to P_{bolt}), as illustrated in Figure 4.3.

In this work, processing is applied to the physical readings. However, the same methods described in this subsection can also be applied to the digital observations to keep consistent with the physical reading.

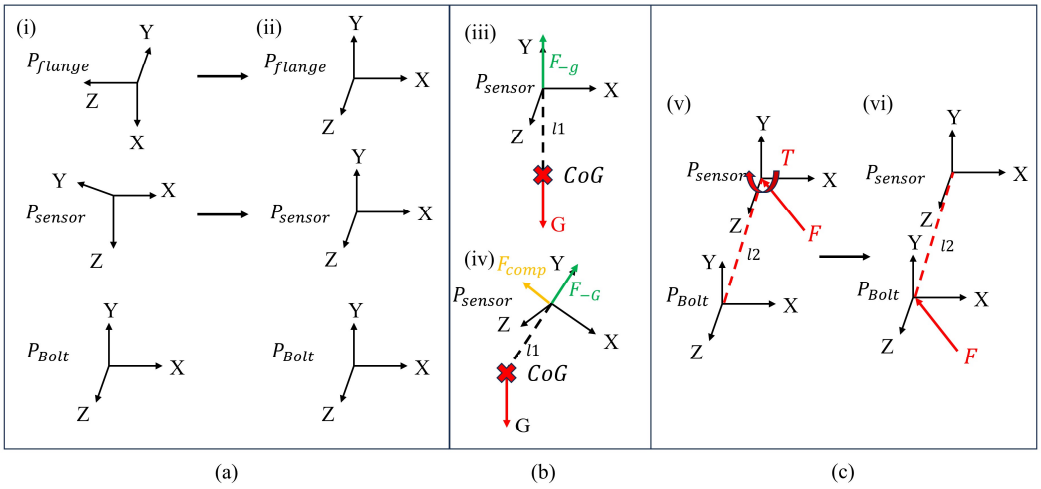


Figure 4.3: Illustration of the processing for the observations made from the physical experiment. (a) shows an example of aligning the orientations of the frames (i) before and (ii) after the processing. (b)(iii) illustrates the process of zeroing the reading of the sensor, which is to use an artificial force F_{-G} to offset the gravity. However, (iv) when the end-effector is rotated along the other two axes other than the vertical axis, an error on F/T reading will occur because the directions of F_{-G} and the gravity are not aligned. Thus, an artificial force F_{comp} needs to be computed to ensure a zero reading when no external force is presented. In (c), the F/T reading from the sensor needs to be converted to P_{bolt} to be the consistent with the simulation model. The $l1$ and $l2$ represent the distance from the measurement frame of the sensor to the centre of gravity (CoG) of the gripper and to the P_{bolt} frame, respectively.

4.2.4.1. Orientation alignment

To align the orientations of the measurements from the 6-axis F/T sensor and the TCP of the robot with the orientations used in the simulation model, rotational matrices are applied to the observations made at P_{flange} and P_{sensor},

$$s_t = \left[[x, y, z] \cdot \boldsymbol{M_A}, [u, v, w] \cdot \boldsymbol{M_A}, \left[F_x, f_y, F_z\right] \cdot \boldsymbol{M_B}, \left[T_x, T_y, T_z\right] \cdot \boldsymbol{M_B}\right] \tag{4.4}$$

where $\boldsymbol{M_A}$ and $\boldsymbol{M_B}$ represent the rotational matrices from P_{flange} and P_{sensor} to P_{bolt}, respectively,

$$\boldsymbol{M} = \begin{bmatrix} \mathrm{C}(\beta)\,\mathrm{C}(\gamma) & -\mathrm{C}(\beta)\,\mathrm{S}(\gamma) & \mathrm{S}(\beta) \\ \mathrm{S}(\alpha)\,\mathrm{S}(\beta)\,\mathrm{C}(\gamma) + \mathrm{C}(\alpha)\,\mathrm{S}(\gamma) & -\mathrm{S}(\alpha)\,\mathrm{S}(\beta)\,\mathrm{S}(\gamma) + \mathrm{C}(\alpha)\,\mathrm{C}(\gamma) & -\mathrm{S}(\alpha)\,\mathrm{C}(\beta) \\ -\mathrm{C}(\alpha)\,\mathrm{S}(\beta)\,\mathrm{C}(\gamma) + \mathrm{S}(\alpha)\,\mathrm{S}(\gamma) & \mathrm{C}(\alpha)\,\mathrm{S}(\beta)\,\mathrm{S}(\gamma) + \mathrm{S}(\alpha)\,\mathrm{C}(\gamma) & \mathrm{C}(\alpha)\,\mathrm{C}(\beta) \end{bmatrix} \tag{4.5}$$

where $S(\cdot) = \sin(\cdot), C(\cdot) = \cos(\cdot)$; α, β, and γ represent the rotation angles from the original orientations to the final orientations with respect to the original x, y, and z axes.

4.2.4.2. Gravity compensation

As illustrated in Figure 4.3(b), the error caused by the end-effector's rotation after zeroing must be compensated for keeping a zero reading when no external F/T is presented. By using the coordinate systems shown in Figure 4.3(b), given that the end-effector is rotated α and γ degrees along x and z axes, respectively, the error caused by the rotation can be computed as

$$-F_{comp} = \begin{bmatrix} -G * \sin(\gamma) \\ G * (2 - \cos(\alpha) - \cos(\gamma)) \\ G * \sin(\alpha) \\ -G * (\sin(\alpha)) * l_1 \\ 0 \\ G * (\sin(\gamma)) * l_1 \end{bmatrix} \tag{4.6}$$

where G is the gravity of the payload attached to the sensor, and l_1 is the distance between the sensor and the CoG of the payload (in this case, $G = 10.06N$, $l_1 = 0.07m$). Thus, the F/T reading can be computed by subtracting the error $F_{true} = F_{sensor} - F_{comp}$.

Note that this method only applies when (1) the zeroing operation is performed when the vertical direction of the sensor (Y axis) is aligned with the direction of gravity and (2) the centre of gravity lies only on the axis of the vertical direction of the sensor (Y axis).

4.2.4.3. F/T reading transformation

As illustrated in Figure 4.3(c), the F/T reading only reflects the F/T sensed at P_{sensor}. However, the point of contact is more interested in minimizing the damage to the components. Thus, to transform the reading from P_{sensor} to P_{bolt}, the following method is used,

$$F_{bolt} = \begin{bmatrix} F_x^{Sensor} \\ F_y^{Sensor} \\ F_z^{Sensor} \\ T_x^{Senso} + F_z^{Sensor} * l_2 \\ T_y^{Sensor} \\ T_z^{Sensor} - F_x^{Sensor} * l_2 \end{bmatrix} \tag{4.7}$$

where l_2 is the distance between P_{sensor} and P_{bolt} (in this case, $l_2 = 0.026m$). This method also assumes that P_{bolt} lies only on the axis of the vertical direction of the sensor (Y axis).

4.3. Digital-twin-assisted skill transfer

As discussed in Section 4.3, the proposed learning architecture involves (i) training a virtual agent in the digital twin of the real environment and (ii) using system identification to obtain the appropriate parameters for the digital twin, as illustrated in Figure 4.4. In this section, these two aspects are introduced respectively.

4.3.1. Sim-to-real skill transfer

In this work, the sim-to-real transfer is achieved by setting the same MDP for agents in both physical and digital spaces so that the policies trained in the simulators can be directly implemented on real robots.

The contact dynamics is simulated by a commercial software called MSC Adams (educational edition 2023.1). Since only the contact between the objects is interested, only the bolt and the groove are modelled instead of building the manipulator, gripper and sensor, as shown in Figure 4.1(b). Then, the movement commands work directly at the centre of the bolt. This approach assumes that the bolt is securely gripped during the operation so that no sliding of the bolt relative to the gripper has occurred.

At each time step, the bolt is moved according to the action command, a_t. Based on the simulation result, a state, s_t, which contains the displacement relative to the origin and the simulated reaction forces, is obtained. Also, noise is added to the F/T reading, with the same magnitude documented in the sensor's user manual, to improve the quality of the simulation ($noise = [0.2N,\ 0.8N, 0.2N, 0.01Nm, 0{,}02Nm, 0{,}01Nm]$).

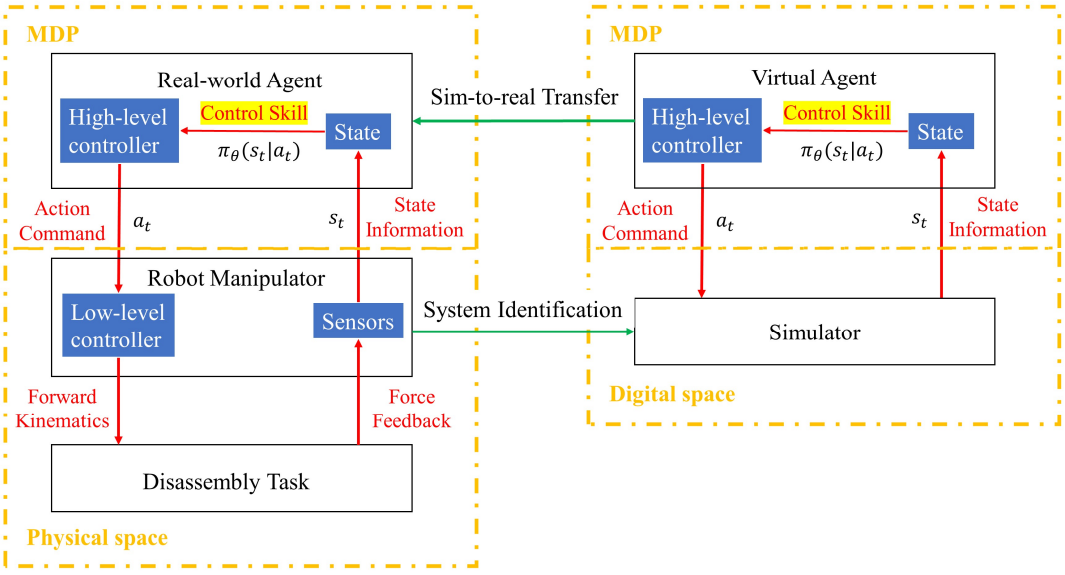


Figure 4.4: Overview architecture of the digital-twin assisted skill transfer. The virtual agent and the real-world agent share the same MDP formulation to ensure an easy transfer of skills. Also, system identification methods are used to improve the quality of the simulation.

4.3.2. System identification using the Bees Algorithm

The basic MSC Admas impact function includes four parameters, $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4]$, to compute the reaction forces: stiffness (θ_1), damping (θ_2), force exponent (θ_3), and penetration depth (θ_4). It is usually difficult to directly measure these parameters, but they can be identified by utilising the transition data, $T = [s_t, a_t, s_{t+1}]$, gathered from the physical training by minimising the response differences generated from simulation, $\boldsymbol{s_{t+1}^{sim}} = [s_{t+1,0}^{sim}, s_{t+1,1}^{sim}, \dots, s_{t+1,n}^{sim}]$, and the physical environment, $\boldsymbol{s_{t+1}^{real}} = [s_{t+1,0}^{real}, s_{t+1,1}^{real}, \dots, s_{t+1,n}^{real}]$ under the same actions, $\boldsymbol{a_t} = [a_{t,0}, a_{t,1}, \dots, a_{t,n}]$ and previous states $\boldsymbol{s_t^{real}} = [s_{t,0}^{real}, s_{t,1}^{real}, \dots, s_{t,n}^{real}]$, where n represents the size of the dataset, as illustrated in Figure 4.5.

Thus, the system identification process can be formulated as a minimisation problem and solved by the optimisation algorithms,

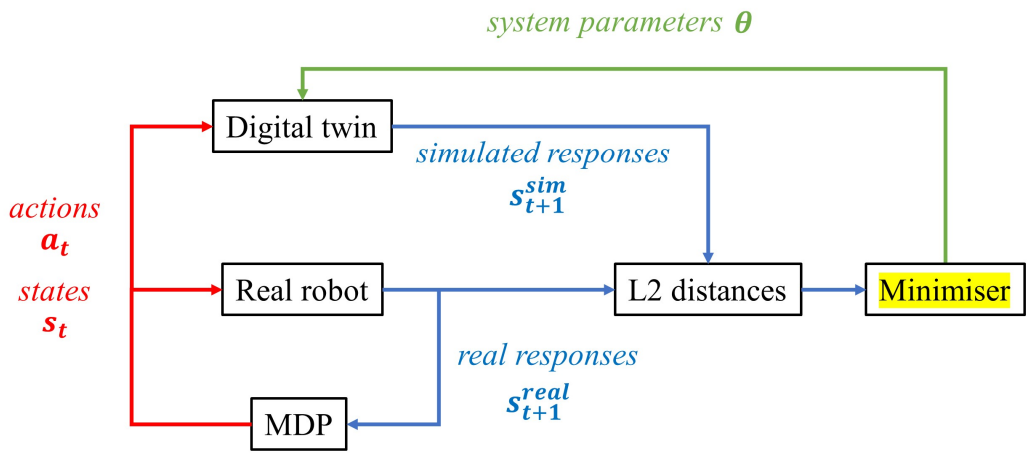


Figure 4.5: Illustration of the system identification process by formulating the process as a minimisation problem. The digital twin executes the same actions as have been executed on the real robots. Then, the optimiser minimises the response discrepancy by using different system parameters.

$$f\left(\boldsymbol{s_{t+1}^{sim}}|\boldsymbol{\theta}\right) = \frac{1}{n} L2\left(\boldsymbol{s_{t+1}^{real}}\left(\boldsymbol{a_t}, \boldsymbol{s_t^{real}}\right) - \boldsymbol{s_{t+1}^{sim}}\left(\boldsymbol{a_t}, \boldsymbol{s_t^{real}}|\boldsymbol{\theta}\right)\right) \tag{4.8}$$

$$\min f\left(\boldsymbol{s_{t+1}^{sim}}|\boldsymbol{\theta}\right) w.r.t.\ \boldsymbol{\theta} \tag{4.9}$$

The Bees Algorithm (BA) is selected as the optimisation algorithm for this task (Pham et al., 2006). BA is a decentralised heuristic optimisation algorithm that has been proven to perform well in overcoming local optima by iteratively performing local search and global search (Pham & Castellani, 2014, 2015). Also, BA does not rely on obtaining the derivatives of the objective function, which are difficult to obtain for this task. Furthermore, it can be naturally applied to tasks with continuous searching space, which is the case in this work.

The pseudo-code of the standard BA is shown in Algorithm 4.1 with the hyper-parameters listed in Table 4.2. Note that the number of bees selected in this experiment is kept at a minimum because it takes about 10 minutes to obtain one result from the fitness function, mainly due to

the time to initialise the simulation environment with different parameters. Also, although more precise searching boundaries can be provided, large searching boundaries across multiple orders of magnitude are used to demonstrate the independence of expert knowledge.

4.3.3. Experimental procedures

To validate the proposed approach, an experiment is designed and conducted. First, the physical training without any pre-training in the simulation has been performed as described in Section 4.2.3.2. Then, the system identification process is performed based on the transition data gathered from the physical experiment. Three digital twins are built based on the parameters obtained by running the BA (1) with $n = 883$ for 10 cycles, (2) with $n = 1776$ for 10 cycles, and (3) with $n = 883$ for 20 cycles (883 transitions are data generated from a complete training session). A virtual agent is trained for 150 episodes in each digital environment with the same training parameters shown in Table 4.1. Finally, the virtual agents are tested in the physical environment. Every RL training described above is replicated three times and represented as the average reward obtained among all replications at the same time step unless specified.

Algorithm 4.1: Standard Bees Algorithm (Baronti et al., 2020).

1 **(Initialisation)** Create an initial population, $\mathcal{P}$, with $ns + nr * nb$ numbers of solutions across the whole solution space. Each solution, s^g, marks the centre of the site (neighbourhood), s. The size of the neighbourhood is defined as $s^e = ngh * (M - m)$, where M and m denote the upper and lower boundaries of each variable. Together with the local search edge, s^e, s^e and s^g define the search scope, $C(s^g, s^e)$. For each site, s, a stagnation counter is also initialised: $s^{ttl} = stlim$.

2 (**Selection, Waggle Dance**) Based on s^g, the best nb sites are selected for local search, $\mathcal{S} = s^{(1)}, \dots, s^{(nb)}$, and the others are removed from $\mathcal{P}$.

3 (**Local Search**) For each $s \in \mathcal{S}$, the following steps are performed:

4 **(Site Abandonment)** If $s^{ttl} = 0$, then the site is abandoned.

5 (**Foraging**) Evenly sample nr solutions, $v_1, \dots, v_{nr}$, from $C(s^g, s^e)$. The best solution v is then selected for the following comparison:

6 If $f(v) > f(s^g)$, v replaces s^g as the new site centre. s^{ttl} and s^e are kept unchanged.

7 If s^g is not the best solution and f$(v) \leq f(s^g)$, the local search is said to be stagnate. (**Neighbourhood Shrinking**) $s^e = s^e * \alpha$ and $s^{ttl} = s^{ttl} - 1$.

8 (**Global Search**) Globally sample ns scout bees solutions, $v_1, \dots, v_{ns}$.

9 **(Population Update)** The new population, $\mathcal{P}$, now consists of the solutions from 3-Local Search and 4-Global Search.

10 Reset the stagnation counter for the best solution, $s^{tll} = stlim$.

11 **(Stopping)** Check with the stopping criteria and determine the stopping of the algorithm.

Table 4.2: Hyper-Parameters used in the BA.

Hyper-parameter	Description	Value
ns	Number of scout bees used ***only*** in the global search	2
nb	Number of sites where local search is performed	2
nr	Number of recruited forager bees for each nb site	2
$stlim$	Cycles of local stagnation before a site is abandoned	2
α	Neighbourhood shrinking scale, $(0 < \alpha < 1)$	0.5
M	Upper searching boundaries	$[1e5, 5, 1000, 1]$
m	Lower searching boundaries	$[0, 0, 0, 0]$

4.4. Results and discussions

4.4.1. Validation of the training systems

Figure 4.6 shows the optimisation progress for identifying the suitable parameters to build the digital twins after obtaining the transition data from the physical training. As can be observed from the graphs, there have been cycles of stagnation during the optimisation process, especially in Figure 4.6(b), due to the small number of bees assigned to the algorithm, as it takes about 10 minutes to evaluate the fitness function for a single time. For optimisation with larger scales, the use of surrogate models will be investigated in the future.

The results of the RL training are shown in Figure 4.7. It is demonstrated that the agents in all these environments have learned the task successfully. It takes approximately 15 minutes to complete the 100 episodes of training in the physical environment and about the same time to train each agent in the digital environments for 150 episodes.

4.4.2. Digital-twin-assisted training results

The results of the method proposed in this chapter are graphically demonstrated in Figure 4.8. To characterise the performance of each training, the following metric is used,

$$P = \frac{1}{E}\sum_{i=0}^{E} Epr_i \tag{4.4}$$

where Epr represents the episodic reward, and E denotes the total number of training episodes. This metric is used because it reflects the qualities of the control policies and the sample efficiency to reach those qualities. Finally, the performances of the training with the assistance of digital twins are compared with the one without the assistance, as shown in Table 4.3.

It can be observed from the second and third transfers that there have been remarkable 1164% and 673% learning performance increments, showing that the method proposed in this work can significantly increase the RL learning performance with the assistance of digital twins. However, a -509% decrement can also be observed from the first transfer, which can be explained by the inferior quality of the optimisation result obtained from the system process since digital twin 2 and digital twin 3 have been trained with doubled dataset size and optimisation cycles. It can also be concluded that increasing the dataset size and optimisation performance can build a more accurate digital twin.

It is also interesting to note that it is possible for the reality gap to cause worse performances than the training without the sim-to-real transfers, even though the same agent has successfully converged in its simulation environment.

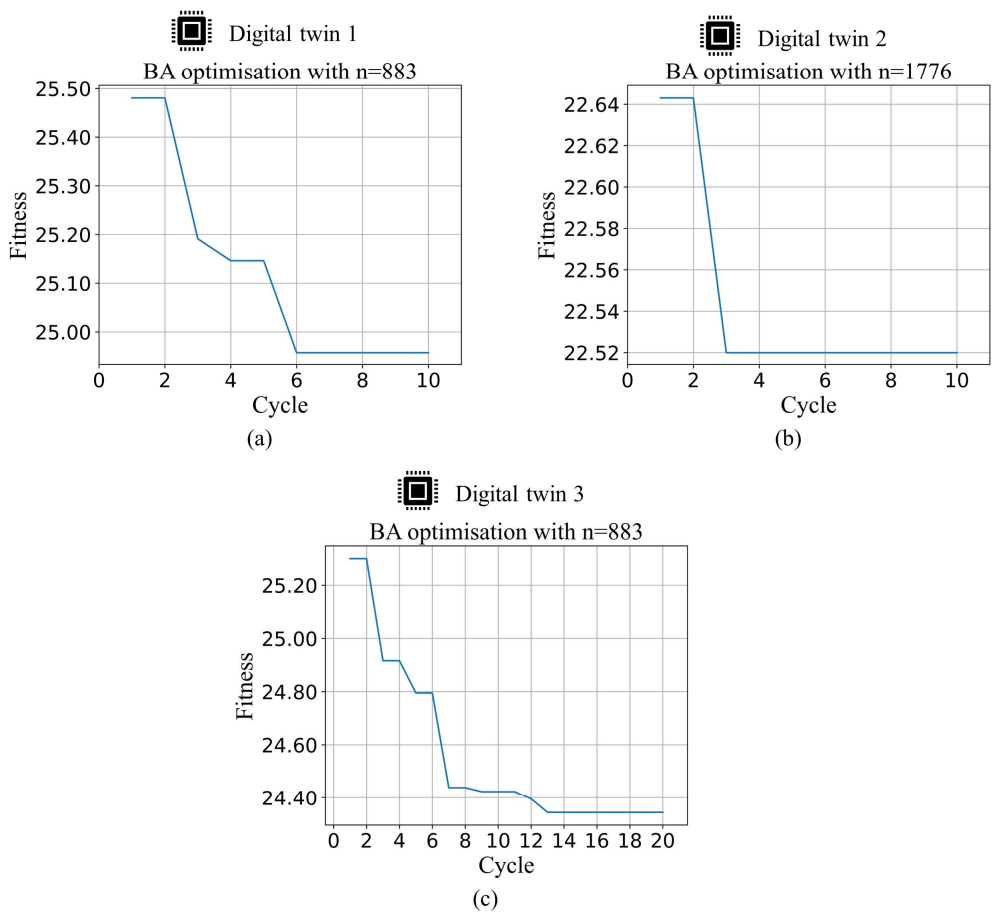


Figure 4.6: Progression plots of identifying the optimised parameters for building the digital twins by BA optimisation with (a) $n = 883$ for 10 cycles, (b) $n = 1776$ for 10 cycles, and (c) $n = 883$ for 20 cycles by BA. The downward trends show that the discrepancy between the digital twins and the real-life environment decreases with the optimisation cycles.

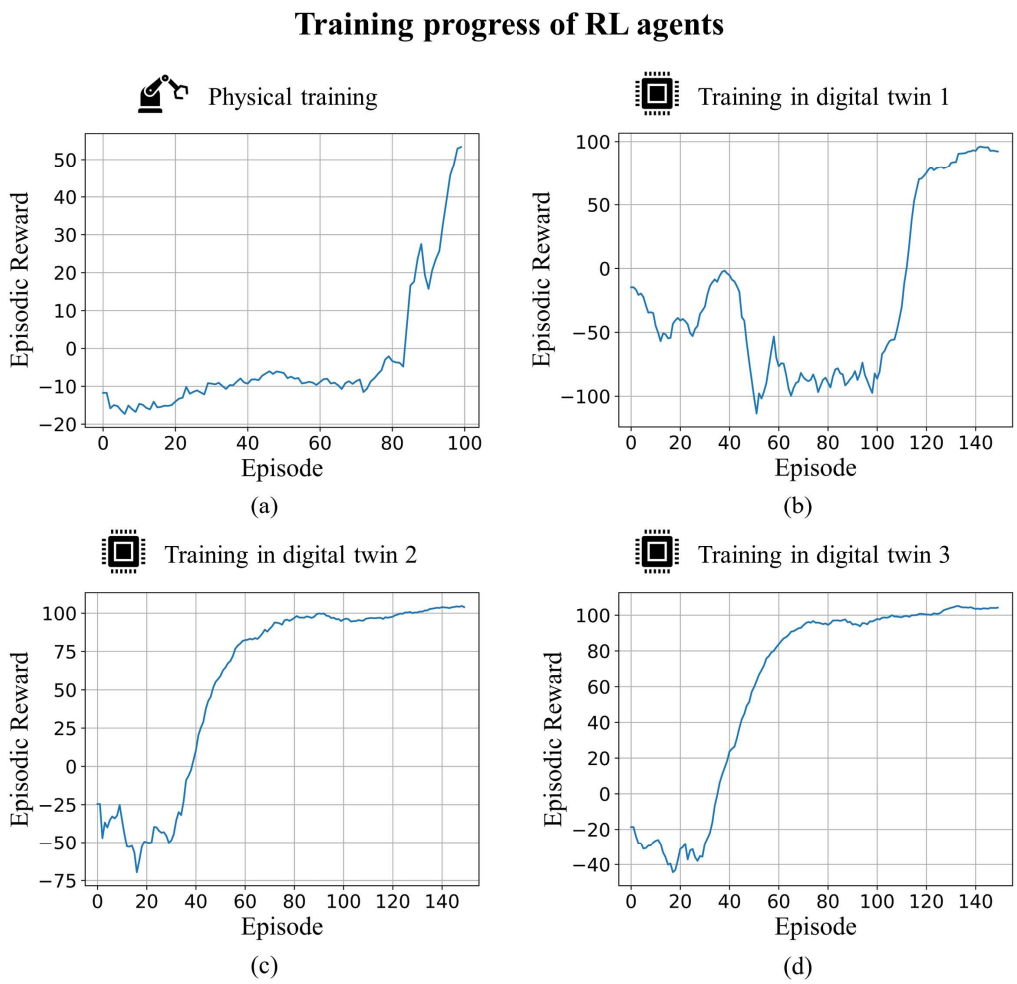


Figure 4.7: Learning curves of (a) the physical training and virtual trainings in the digital twins built based on the results after running the BA with (a) $n = 883$ for 10 cycles, (b) $n = 1776$ for 10 cycles, and (c) $n = 883$ for 20 cycles. The upward trends demonstrate that the training environments have been set up successfully for agents to progress on learning. (The learning curves are smoothed by a coefficient of 0.05).

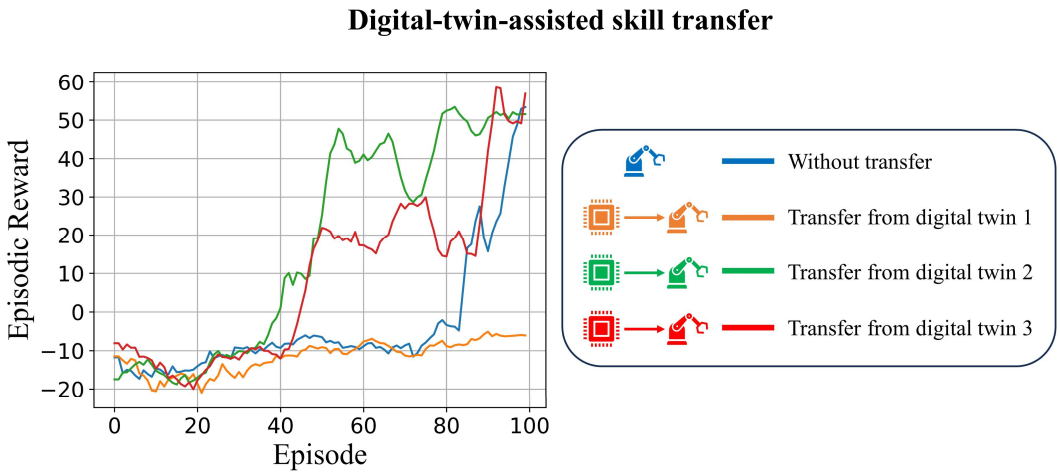


Figure 4.8: Learning curves of the physical training process with and without skill transfers from the digital twins. It can be observed that transferring skills from digital twins 2 and 3 is able to increase the overall learning performance in terms of overall rewards obtained from the same period of training. However, the skill transfer from digital twin 1 not only fails, but also worsen the overall performance compared the skills obtained from physical training only. (The learning curves are smoothed by a coefficient of 0.05).

Table 4.3: Comparison of the learning performances.

	Performance	**Comparison**
Without transfer	−1.86	
Skill transfer from digital twin 1	−11.34	−509%
Skill transfer from digital twin 2	19.80	1164%
Skill transfer from digital twin 3	10.66	673%

4.5. Conclusions and future works

This work investigates a method to increase the performance of the RL training in contact-rich robotic disassembly tasks by skill transfers from the digital twins of the training environments with their system parameters optimised by the Bees Algorithm. This work also described practical methods to process observation data for keeping the reference frames consistent between the simulated and the real environments. Finally, the sim-to-real transfers have been conducted, and significant performance increments have been achieved, providing that the system identification process has been effectively conducted. It has been found that it is possible to have worse performance with the sim-to-real transfer if the reality gap is not effectively addressed. Furthermore, with the proposed method, increasing the size of the dataset and optimisation cycles have been demonstrated to reduce the reality gap and lead to successful sim-to-real transfers.

The major issue of this research is the time cost to evaluate the fitness function at each time, which is a common limitation that should be addressed by developing novel methodologies or simulation environments that are more suitable for rapid initialisation. One of the promising research directions is to use surrogate models to increase the sample efficiency of the optimisation methods. Other suggested future works include large-scale testing with other optimisation and RL algorithms. Furthermore, optimisation with more dimensions of system parameters can be tested to further reduce the reality gap.

5. Hyper-parameter Study of Deep Deterministic Policy Gradient Algorithm in Robotic Disassembly Tasks

5.1. Preliminaries

In both Chapters 3 and 4, the implementation of reinforcement learning (RL) as a method to acquire robotic skills has been discussed. Although many RL algorithms have been proposed, the selection of their hyper-parameters has been heavily reliant on the expert knowledge and intuition of the practitioners (Fujimoto et al., 2018; Haarnoja et al., 2018; Lillicrap et al., 2016). For RL algorithms implemented to solve problems in simulation environments, faster data generation can reduce this problem to a certain degree, as the cost to obtain some preliminary results is low before large-scale implementation (Chatzilygeroudis et al., 2020; Salvato et al., 2021).

However, to implement RL in physical robot training, such as the ones discussed in previous chapters, having some guidance on choosing the appropriate hyper-parameters will shorten the time of setting up the training platform. One approach to obtain such guidance is to conduct a hyper-parameter study on a simulation environment that approximates real-world task training.

In this chapter, based on the simulated training task of removing a bolt from a door-chain groove described in Chapter 4, a full factorial study with four main hyper-parameters of the Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) has been conducted to (1) identify the hyper-parameters for the best learning performance, and (2) examine the sensitivity of the learning performance to the change of each hyper-parameter.

Following the Introduction, Section 5.2 introduces the method of the study, and the results are presented and discussed in Section 5.3. Section 5.4 concludes this chapter and provides suggestions for future research directions.

5.2. Full factorial hyper-parameter study

5.2.1. DDPG hyper-parameters

Following the discussions of Chapters 3 and 4, the robotic disassembly task of removing a bolt from a door-chain groove is used as the case in this hyper-parameter study. (More details of the task training environment, training procedures, and the Markov Decision Process (MDP) formulation can be found in Section 4.2).

For solving the MDP, an RL algorithm named Deep Deterministic Policy Gradient (DDPG) is selected for its capability of (1) selecting actions from continuous actions space and (2) learning directly from an offline dataset. (More details about the implementation of DDPG can be found in Section 3.3.2.) The pseudo-code of the algorithm is shown in Algorithm 5.1.

Although using other function approximators and optimisers for the actors and critics is theoretically possible, the functions are usually modelled by Artificial Neural Networks (ANNs) and updated by the Adam optimiser (Kingma & Ba, 2015) following the implementation of the original paper. It can be observed from Algorithm 1 that the decisions made by the RL agent are outputs of the actor network, and its update is dependent directly on (1) the critic network (step 13), (2) the dataset that is used as the samples for the ANNs update (step 10), and (3) rate of the soft update (step 14). Thus, the following hyper-parameters are selected for the full factorial study: (1) actor ANN learning rate, (2) critic ANN learning rate, (3) batch size of the data for ANNs update, and (4) rate of the soft update, τ. (Note that selecting these hyper-parameters does not mean that the other hyper-parameters, e.g. the architecture of the ANN or

Algorithm 5.1: Deep Deterministic Policy Gradient.

Initialise actor network, $\pi(s|\theta^{\pi})$, and critic network, $Q(s,a|\theta^{Q})$;
Initialise target actor network, $\pi'(s|\theta^{\pi'})$, with parameters $\theta^{\pi'} \leftarrow \theta^{\pi}$, and target critic network, $Q'(s,a|\theta^{Q'})$, with parameters $\theta^{Q'} \leftarrow \theta^{Q}$;
Initialise replay buffer, $\mathcal{R}$;
for episode, e **do**
    for time step, t **do**
        Sample an action, a_t, from actor network $\pi(s_t|\theta^{\pi}) + \mathcal{N}(0,\epsilon)$, where $\mathcal{N}(0,\epsilon)$ is a Gaussian noise for exploration;
        Clip a_t with action boundaries, (a_{low}, a_{high}), for safety;
        Execute a_t, receive next state, s_{t+1}, and corresponding reward, r_{t+1};
        Store transition tuple $(s_t, a_t, s_{t+1}, r_{t+1}, T)$ into $\mathcal{R}$, where T indicates the termination of the episode;
        Sample a batch of transitions, $\mathcal{B}(s_t, a_t, s_{t+1}, r_{t+1}, T)$, from $\mathcal{R}$;
        Compute update target, $y_t = r_t + \gamma Q'(s_{t+1}, \pi'(s_{t+1}|\theta^{\pi'})|\theta^{Q'})$;
        Update critic network by minimizing loss function, $L = N^{-1}\sum_N (y_t - Q(s_t, a_t|\theta^{Q}))^2$;
        Update actor network by gradient ascent, $\nabla_{\theta^{\pi}} \mathcal{J} \approx N^{-1}\sum_N Q(s_t, \pi(s_t|\theta^{\pi})|\theta^{Q})$;
        Soft update the target networks, $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$, $\theta^{\pi'} \leftarrow \tau\theta^{\pi} + (1-\tau)\theta^{\pi'}$;
        Decrease the exploration by a decay constant, $\epsilon \leftarrow c * \epsilon$;
    until any termination condition is met;
until maximum number of episodes is reached;

the exploration constant, have no effect on the learning performance. However, these parameters are selected as they directly affect the action of updating the actor network, where the actor network is the most important output of an RL algorithm as the actor network itself is a complete decision-making policy.)

5.2.2. Full factorial study

Full and fractional designs are widely accepted as the most informatic experimental designs in manufacturing. They enable researchers to investigate the effects of each factor and the joint effects of the factors on a response. A factorial design can be either fractional or full factorial.

The former design is more practical in planning experiments, and it is frequently used for several factors at two or more levels. The latter design is more appropriate for investigating multiple levels of several factors, which is widely applied to study complex systems (Montgomery, 2017).

In this study, each hyper-parameter will be assigned with three levels, as shown in Table 5.2. The levels are selected as reasonably distant as possible so the results can be less dependent on the practitioners' experience and intuition. With four variables and three levels, there are 81 possible combinations of the parameters to test, and each combination is replicated three times. In each training, the agent is trained in the simulation for 100 episodes and a maximum of 100 time steps for each episode. The other fixed hyper-parameters are listed in Table 5.1.

To characterise the performance of each training, the following metric is used,

$$P = \frac{1}{E}\sum_{i=0}^{E} Epr_i \tag{5.1}$$

where Epr represents the episodic reward, and E denotes the total number of training episodes, and each combination's performance is characterised by the average P among three replications, termed as the *average training reward*. This metric is used because it reflects the qualities of the control policies and the sample efficiency to reach those qualities.

Table 5.2: List of hyper-parameters and their levels.

Hyper-parameters	**Level 1**	**Level 2**	**Level 3**
Batch size	100	1000	2000
Soft update rate, τ	0.1	0.5	0.9
Actor learning rate	0.005	0.05	0.5
Critic learning rate	0.005	0.05	0.5

Table 5.1: List of the fixed hyper-parameters used in this study.

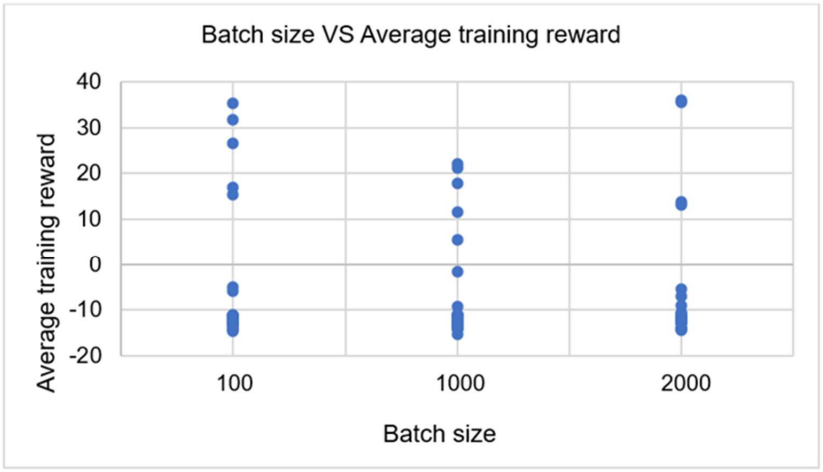
Neural Networks Parameters		**DDPG Parameters**	
Hidden layer size	$128 * 128$	Replay buffer size	$1e4$
Hidden layer Neuron	ReLU	Discount rate, γ	0.9
Output layer Neuron	Tanh	Starting exploration, ϵ	3
Optimiser	Adam	Exploration decay rate, c	0.95
Adam, β_1	0.9		
Adam, β_2	0.999		
Adam, ϵ	$1e-7$		
Adam, AMSGrad	FALSE		

5.3. Results and discussions

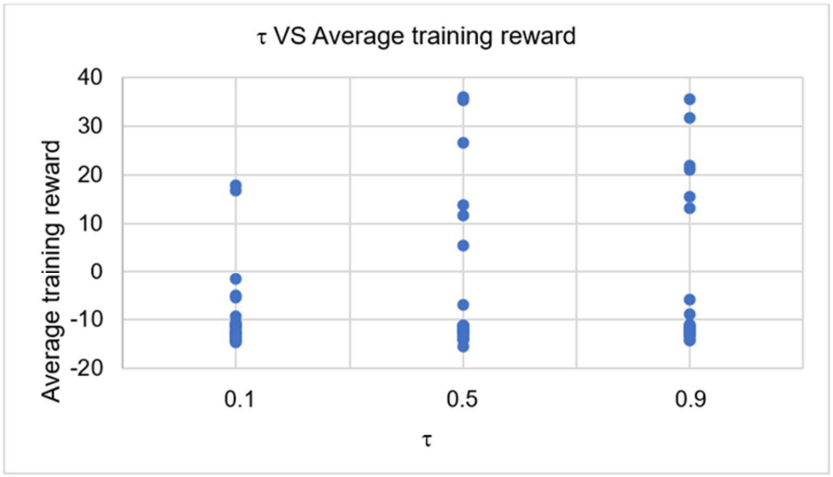
After obtaining the results from 243 trainings, the results are analysed and discussed in this section. The full results can be found in the Appendix.

Firstly, as shown in Figure 5.1, the average training reward of each combination is plotted by each level of the hyper-parameters. This analysis can be seen as changing only one hyper-parameter at a time. From the figure, we can observe that the distributions of batch size, soft update rate, the first level of actor network learning rate, and the first two levels of critic network learning rate are scattered, meaning that selecting hyper-parameters from these levels does not have a dominant effect on the learning performance. However, if the learning rate for either the actor or the critic network is too big, e.g. 0.5, it will result in a failure of all combinations that contain this level of the learning rate, regardless of the choice of other hyper-parameters.

The finding suggests that choosing the appropriate learning rates for both networks is important for successful training, as one inappropriate learning rate can result in a complete failure of the training. Furthermore, the results suggest that the learning rate with the lowest value should be tested first.

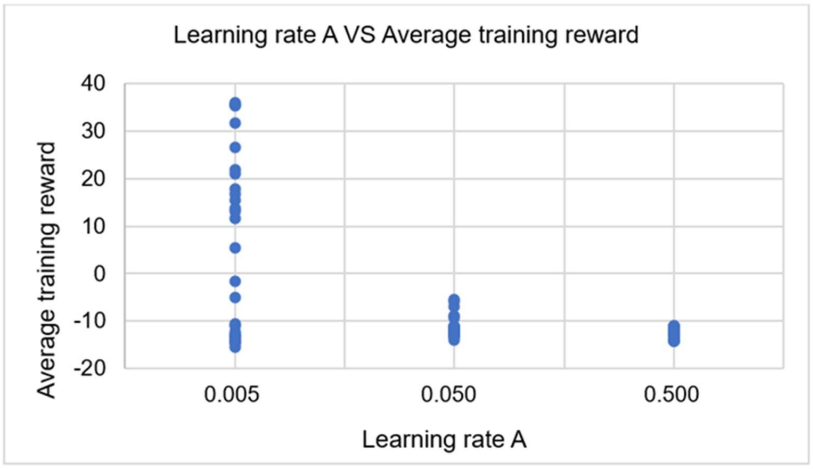


(a)

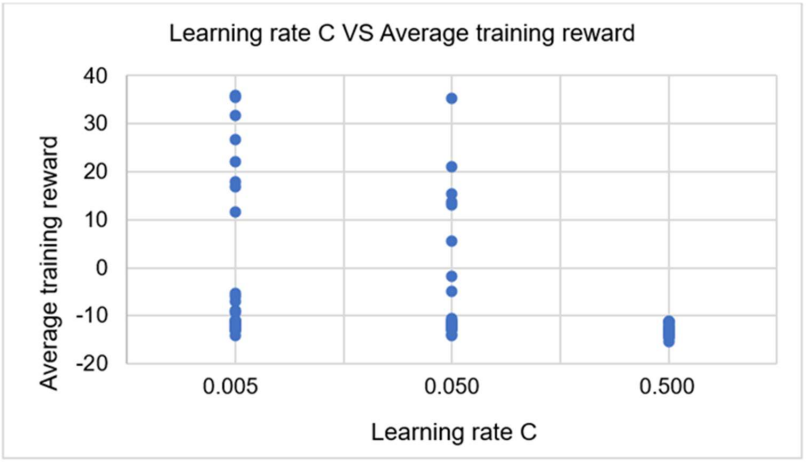


(b)

(Figure continues on the next page)



(c)



(d)

Figure 5.1: The average training reward distribution of different levels of the (a) batch size, (b) soft update rate, (c) actor network learning rate, and (d) critic network learning rate. Each dot represents the performance of each combination of the hyper-parameters. For batch size, soft update rate, the first level of actor network learning rate, and the first two levels of critic network learning rate, the distributions are scattered, meaning that selecting hyper-parameters from these levels does not have a dominant effect on the learning performance. However, if the learning rate for either of the actor or the critic network is too big, it will result in a failure of all combinations that contain this level of the learning rate, regardless the choice of other hyper-parameters.

As mentioned previously, the average training reward, P, is an average value among three replications. Table 5.3 summarises the range, standard deviation, and a 95% confidence interval (the probability that the average training reward would fall around the obtained average value reward).

The range of average value rewards is a direct indication to describe the influence of the hyper-parameters. From Table 5.3, the average value reward ranges of batch size, soft update rate, actor network learning rate and critic network learning rate are 1.88, 5.04, 18.09 and 12.72, respectively, as plotted in Figure 5.2. Thus, it can be concluded that the learning performance is much more sensitive to the learning rates of the actor and critic networks, and the actor network learning rate is the most influential hyper-parameter of the learning performance.

Table 5.3 Statistical results of the hyper-parameter study.

Hyper-parameter	Level	Average P (among three replications)	Range of P	Standard deviation	95% Cl
Batch size	100	-5.02	1.88	15.22	$-10.25, 0.22$
	1000	-6.88		11.51	$-12.12, -1.65$
	2000	-6.29		13.97	$-11.52, -1.05$
Soft update rate, τ	0.1	-9.23	5.04	8.22	$-14.39, -4.06$
	0.5	-4.77		15.45	$-9.94, 0.39$
	0.9	-4.19		15.47	$-9.35, 0.98$
Actor learning rate	0.005	5.57	18.09	15.84	$1.44, 9.71$
	0.05	-11.24		2.16	$-15.38, -7.11$
	0.5	-12.52		1.07	$-16.65, -8.38$
Critic learning rate	0.005	-0.29	12.72	17.35	$-5.12, 4.53$
	0.05	-4.88		13.15	$-9.70, -0.06$
	0.5	-13.01		1.13	$-17.83, -8.19$

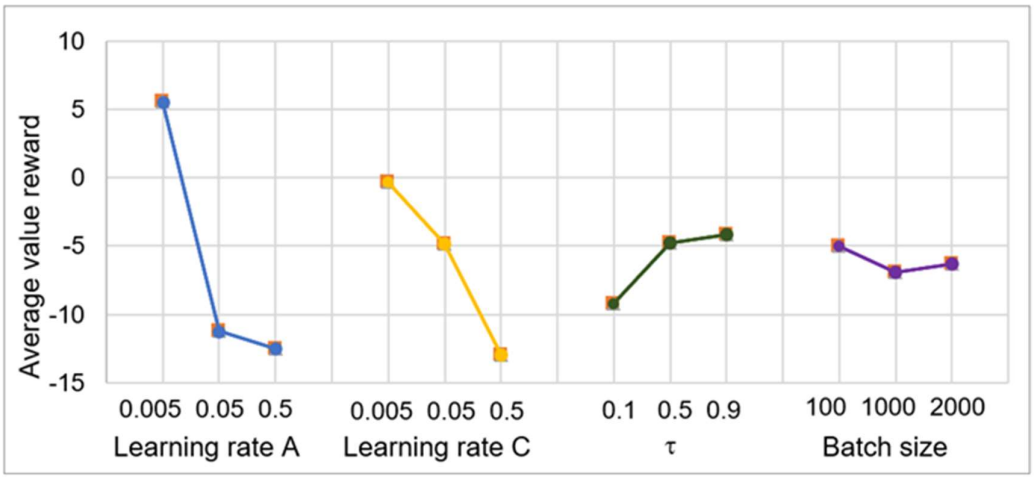


Figure 5.2: Comparison of the average training reward with respect to each level of all hyper-parameters. It can be observed that the learning performance is much more sensitive to the learning rates of the actor and critic networks, as they their ranges are larger than the ones of the soft update rate and the batch size..

5.4. Conclusions and future works

This chapter has conducted a full factorial study on implementing the DDPG algorithm in a simulated robotic disassembly task of removing a bolt from a door-lock groove. Four hyper-parameters that directly influence the updates of the policy ANN have been selected, with three levels of values assigned to each hyper-parameter. 241 simulated trainings have been performed, and their results have been presented. For this particular task, the learning rates of the actor and critic networks are the most influential hyper-parameters, while the batch size and soft update rate have relatively limited influence. Also, it was found that if either of the learning rates is set to an inappropriate value, the training will fail, regardless of the choice of other hyper-parameters.

However, the current work only examines performance in one environment. In the future, the findings should also be tested in physical environments or other tasks with similar features. Also, more information should be analysed from the full factorial study, such as the influence of the learning performance by multiple factors.

6. Summary of Conclusions and Future Works

This thesis investigates the implementation aspect of applying Reinforcement Learning (RL) algorithms for robots to obtain disassembly skills by trial and error.

In Chapter 2, a literature review on deploying industrial robots in disassembly automation and applying RL in industrial robotic operations has been conducted. A categorisation method for studies of developing a robotic disassembly system has been proposed, which includes four levels: (1) disassembly system design, (2) product sequence planning, (3) robotic motion planning, and (4) disassembly operation methods. Among them, disassembly operation is the most fundamental level because the decisions from every other level depend on the capability of robots to perform disassembly operations.

Enabling the robots to learn from trial-and-error and existing data on disassembly operations is one meaningful way to make the robots flexible and robust, which can be achieved by integrating with the advancement of RL. However, although RL has been implemented in virtual environments on control tasks, the specific studies on applying RL in actual robotic operations for both assembly and disassembly still require further investigation. The current studies on RL mainly focus on developing theories or designs from an algorithm perspective.

However, the application aspect, such as investigating the relationship between learning performance and the physical features of the robots and using simulations to accelerate trainings, requires further investigation.

In Chapter 3, an RL-based training platform for robotic disassembly operations has been presented. The results show that the control skill of a robot performing a contact-rich manipulation task can be learned by a data-driven approach.

As the authors did not have access to robots with different precision levels, to study the robustness of the proposed learning method, a technique to change the precision artificially was devised to enable a single robot to be used. In this study, the robot's precision was modified to as low as $\pm 0.5mm$, and the robot can still learn successfully. Additionally, the results empirically demonstrated that the stability of the learned controllers depended on the precision of the machine. The study has also shown that the robot can learn directly from a dataset previously generated by other training without taking any actual actions. Finally, the transferability of the learned skills was empirically studied by applying the proposed precision modification technique in real and simulated environments. It has been found that the skills learned from a low-precision robot can be transferred to a high-precision robot, and they have shown increased performance and stability after the transfer. On the other hand, policies learned from a high-precision robot will obtain less performance and stability when implemented on robots with lower levels of precision.

Regarding the research presented in Chapter 3, the following aspects of the work could be investigated. First, this chapter observes a pattern in the transferability of the skills, but the underlying principles and mechanisms require further investigation. Second, the application of the training platform could be extended to more complex disassembly tasks, for example, those involving a combination of movements such as twisting and pulling. Third, the findings shown in this chapter (in particular about skill transfer) can be tested on other robotic operations and RL algorithms. Third, the results about the transferability of robots with different precision levels should be further validated on actual different robots and in large-scale applications.

Chapter 4 investigates a method to increase the performance of the RL training in contact-rich robotic disassembly tasks by skill transfers from the digital twins of the training environments with their system parameters optimised by the Bees Algorithm. This chapter also described

practical methods to process observation data for keeping the reference frames consistent between the simulated and the real environments. Finally, the sim-to-real transfers have been conducted, and significant performance increments have been achieved, providing that the system identification process has been effectively conducted. It has been found that it is possible to have worse performance with the sim-to-real transfer if the reality gap is not effectively addressed. Furthermore, with the proposed method, increasing the size of the dataset and optimisation cycles have been demonstrated to reduce the reality gap and lead to successful sim-to-real transfers.

The major issue of this research is the time cost to evaluate the fitness function at each time, which is a common limitation that should be addressed by developing novel methodologies or simulation environments that are more suitable for rapid initialisation. One of the promising research directions is to use surrogate models to increase the sample efficiency of the optimisation methods. Other suggested future works include large-scale testing with other optimisation and RL algorithms. Furthermore, optimisation with more dimensions of system parameters can be tested to further reduce the reality gap.

Finally, in Chapter 5, a full factorial study has been conducted on implementing the DDPG algorithm in a simulated robotic disassembly task of removing a bolt from a door-lock groove. Four hyper-parameters that directly influence the updates of the policy ANN have been selected, with three levels of values assigned to each hyper-parameter. 241 simulated trainings have been performed, and their results have been presented. It was found that for this particular task, the learning rates of the actor and critic networks are the most influential hyper-parameters, while the batch size and soft update rate have relatively limited influence. Also, it was found that if either of the learning rates is set to an inappropriate value, the training will fail, regardless of the choice of other hyper-parameters.

However, the work described in Chapter 5 only examines performance in one environment. In the future, the findings should also be tested in physical environments or other tasks with similar features. Also, more information should be analysed from the full factorial study, such as the influence of the learning performance by multiple factors.

From the above summary of conclusions, it can be found that RL is a useful way to obtain robotic operational skills, especially in situations where precise measuring methods or prior knowledge to program the robots are unavailable. However, the standout problem is RL's tendency to overfit the learned policies to its training environment.

For example, the main finding from Chapter 3 about the transferability of the learned skills among robots with different repeatability can be explained by overfitting: the skills learned from high-precision robots have been overfitted to the environments where there is less randomness when progressing from one time step to the next one. The overfitting of the skills prevented the skills learned from a less randomised environment from being transferred to the one with more randomness – a less precise robot.

Similarly, the reality gap observed from Chapter 4 can also be explained by the overfitting issue: even though the agent has scored high rewards in a simulation environment, it still fails to learn in a real environment, which has a high structural similarity with the simulation environment and the same formulation of Markov Decision Process, because the skill has overfitted to the simulation environment.

Thus, the generalisation capability of the skills learned by the robots in different task configurations requires further investigation, and methods to improve on that matter should be developed.

Furthermore, the sample efficiency of the learning methods needs to be further improved if more complicated operations are to be learned by RL. The average time to finish a physical training for the task described in this thesis is about 30 minutes. However, this is a relatively simple task in terms of the low number of dimensions for states and actions. The training time is expected to be longer for tasks with larger numbers of dimensions, e.g., the ones that use images as observations or with more dimensions of actions. Therefore, methods to shorten the training time, e.g. simulation-to-reality skill transfer, require further investigation.

Finally, although using RL to generate robotic skills is a relatively new approach, and it provides some unique features, such as self-path-finding and self-tunning of parameters, the performance generated by RL needs to be compared with the one from the classic control methods or novel RL methods should be developed to integrate with the classic control methods.

Appendix

The following table presents the results obtained from 241 trainings, following the procedures described in Section 5.2.

Table A.1: Results obtained from the simulated trainings.

Exp	Batch size	τ	Learning rate A	Learning rate C	Result 1	Result 2	Result 3
1	100	0.1	0.005	0.005	9.17626	16.93363	24.48816
2	100	0.1	0.005	0.05	−11.3976	−0.08352	−3.4708
3	100	0.1	0.005	0.5	−15.8715	−12.5904	−15.1887
4	100	0.1	0.05	0.005	−11.9383	−12.0288	−12.7523
5	100	0.1	0.05	0.05	−11.897	−11.4616	−10.8061
6	100	0.1	0.05	0.5	−10.8009	−15.8515	−11.2518
7	100	0.1	0.5	0.005	−11.5274	−11.0365	−15.9025
8	100	0.1	0.5	0.05	−11.5065	−11.0129	−11.5619
9	100	0.1	0.5	0.5	−15.8369	−11.1858	−15.478
10	100	0.5	0.005	0.005	−0.8075	−4.25468	84.84028
11	100	0.5	0.005	0.05	16.50094	46.05646	43.55957
12	100	0.5	0.005	0.5	−12.4696	−11.4545	−15.7195
13	100	0.5	0.05	0.005	−11.501	−11.9962	−11.262
14	100	0.5	0.05	0.05	−15.1732	−10.7871	−10.7896
15	100	0.5	0.05	0.5	−10.8132	−10.7905	−15.3469
16	100	0.5	0.5	0.005	−10.965	−11.8685	−11.042
17	100	0.5	0.5	0.05	−11.4803	−15.1461	−15.6617
18	100	0.5	0.5	0.5	−11.229	−14.682	−12.0468
19	100	0.9	0.005	0.005	38.52126	47.70054	8.924718
20	100	0.9	0.005	0.05	11.56007	−5.33118	40.02306
21	100	0.9	0.005	0.5	−14.224	−11.4418	−11.2822
22	100	0.9	0.05	0.005	−11.8414	5.026472	−10.8226
23	100	0.9	0.05	0.05	−11.1814	−10.8087	−10.9757
24	100	0.9	0.05	0.5	−15.8598	−11.8901	−12.01
25	100	0.9	0.5	0.005	−15.6625	−11.6632	−11.4016
26	100	0.9	0.5	0.05	−11.0108	−12.0306	−11.4501
27	100	0.9	0.5	0.5	−15.9628	−11.6674	−11.4075

(Table continues on the next page.)

Exp	Batch size	τ	Learning rate A	Learning rate C	Result 1	Result 2	Result 3
28	1000	0.1	0.005	0.005	41.74139	−0.04781	11.7665
29	1000	0.1	0.005	0.05	−12.1637	−10.8676	17.9777
30	1000	0.1	0.005	0.5	−14.7528	−11.5924	−14.6871
31	1000	0.1	0.05	0.005	−11.2283	−10.532	−6.33732
32	1000	0.1	0.05	0.05	−11.6528	−11.0709	−15.9908
33	1000	0.1	0.05	0.5	−15.9284	−11.4884	−11.8796
34	1000	0.1	0.5	0.005	−11.0498	−11.4014	−11.2332
35	1000	0.1	0.5	0.05	−15.1029	−15.3884	−11.2129
36	1000	0.1	0.5	0.5	−11.2259	−11.2458	−10.8157
37	1000	0.5	0.005	0.005	−11.848	9.185039	37.30527
38	1000	0.5	0.005	0.05	14.46493	−3.51123	5.474035
39	1000	0.5	0.005	0.5	−15.2274	−15.771	−15.2814
40	1000	0.5	0.05	0.005	−11.5641	−11.2048	−11.375
41	1000	0.5	0.05	0.05	−10.7579	−11.2136	−11.4564
42	1000	0.5	0.05	0.5	−11.4411	−15.1693	−15.532
43	1000	0.5	0.5	0.005	−11.2482	−11.9125	−15.8834
44	1000	0.5	0.5	0.05	−12.0665	−11.345	−12.0122
45	1000	0.5	0.5	0.5	−15.9573	−10.7586	−10.7484
46	1000	0.9	0.005	0.005	10.61363	8.554989	46.83033
47	1000	0.9	0.005	0.05	25.99672	7.666488	29.67458
48	1000	0.9	0.005	0.5	−13.8785	−14.4529	−11.1343
49	1000	0.9	0.05	0.005	−11.3526	−12.7453	−11.4741
50	1000	0.9	0.05	0.05	−11.5608	−11.5983	−11.9312
51	1000	0.9	0.05	0.5	−12.0425	−14.7026	−11.1974
52	1000	0.9	0.5	0.005	−15.3142	−11.418	−15.8415
53	1000	0.9	0.5	0.05	−15.4548	−11.5997	−11.5163
54	1000	0.9	0.5	0.5	−11.4775	−11.2144	−10.7912

(Table continues on the next page.)

Exp	Batch size	τ	Learning rate A	Learning rate C	Result 1	Result 2	Result 3
55	2000	0.1	0.005	0.005	−5.69888	−11.3688	−15.6906
56	2000	0.1	0.005	0.05	−8.71551	−10.7951	−12.1149
57	2000	0.1	0.005	0.5	−12.8697	−14.7597	−15.6765
58	2000	0.1	0.05	0.005	−11.4661	−4.45361	−0.34589
59	2000	0.1	0.05	0.05	−10.7778	−10.7502	−15.7632
60	2000	0.1	0.05	0.5	−10.7344	−11.4447	−12.0646
61	2000	0.1	0.5	0.005	-10.7406	-10.7546	-11.4905
62	2000	0.1	0.5	0.05	-10.962	-15.2492	-11.942
63	2000	0.1	0.5	0.5	-14.7279	-15.6732	-11.1803
64	2000	0.5	0.005	0.005	3.505101	81.43456	22.95968
65	2000	0.5	0.005	0.05	7.835876	-2.6551	36.21246
66	2000	0.5	0.005	0.5	-15.8303	-14.3951	-11.8831
67	2000	0.5	0.05	0.005	-11.2238	0.812539	-10.3664
68	2000	0.5	0.05	0.05	-11.6398	-12.0428	-11.9123
69	2000	0.5	0.05	0.5	-11.5941	-10.9551	-11.3605
70	2000	0.5	0.5	0.005	-15.3015	-11.5669	-11.2123
71	2000	0.5	0.5	0.05	-15.5194	-11.6078	-10.7944
72	2000	0.5	0.5	0.5	-10.7606	-11.5724	-11.9258
73	2000	0.9	0.005	0.005	10.1667	66.04961	30.35536
74	2000	0.9	0.005	0.05	4.300311	32.64149	2.180627
75	2000	0.9	0.005	0.5	-15.2498	-10.7634	-12.5469
76	2000	0.9	0.05	0.005	-4.21451	-11.4038	-11.2191
77	2000	0.9	0.05	0.05	-11.8574	-11.2174	-12.0139
78	2000	0.9	0.05	0.5	-14.7739	-12.0241	-11.8801
79	2000	0.9	0.5	0.005	-14.7215	-11.1772	-10.9693
80	2000	0.9	0.5	0.05	-12.0656	-11.5186	-11.1792
81	2000	0.9	0.5	0.5	-15.8027	-14.7397	-11.9489

References

Abuzied, H., Abbas, A., Awad, M., & Senbel, H. (2020). Investigation of Active Disassembly in Large Force Applications. *Journal of Manufacturing Science and Engineering*, *143*(2). https://doi.org/10.1115/1.4048852

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., & Ribas, R. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Amezquita, T., Hammond, R., Salazar, M., & Bras, B. (1995, 1995/09/17-20). *Characterizing the Remanufacturability of Engineering Systems* https://doi.org/10.1115/DETC1995-0036

AUTOREMAN project. (2017). *http://autoreman.altervista.org/index.html*. Retrieved from

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv*, *abs/1803.01271*.

Bakker, B. (2001, 2001/01/03). *Reinforcement learning with long short-term memory* Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic, Vancouver, British Columbia, Canada.

Baronti, L., Castellani, M., & Pham, D. T. (2020). An analysis of the search mechanisms of the bees algorithm. *Swarm and evolutionary computation*, *59*, 100746.

Beltran-Hernandez, C., Petit, D., Ramirez-Alpizar, I. G., & Harada, K. (2020a). Variable Compliance Control for Robotic Peg-in-Hole Assembly: A Deep-Reinforcement-Learning Approach. *Applied Sciences*, *10*(19). https://doi.org/10.3390/app10196923

Beltran-Hernandez, C., Petit, D., Ramirez-Alpizar, I. G., Nishi, T., Kikuchi, S., Matsubara, T., & Harada, K. (2020b). Learning Contact-Rich Manipulation Tasks with Rigid Position-Controlled Robots: Learning to Force Control. *ArXiv*, *abs/2003.00628*.

Blankemeyer, S., Wiens, D., Wiese, T., Raatz, A., & Kara, S. (2021). Investigation of the potential for an automated disassembly process of BEV batteries. *Procedia CIRP*, *98*, 559-564. https://doi.org/https://doi.org/10.1016/j.procir.2021.01.151

Bogue, R. (2019). Robots in recycling and disassembly. *Industrial Robot: the international journal of robotics research and application*, *46*(4), 461-466. https://doi.org/10.1108/IR-03-2019-0053

British Standards Institution (2009). *BS 8887-2:2009 Design for manufacture, assembly, disassembly and end-of-life processing (MADE). Terms and definitions*. Retrieved from https://bsol-bsigroup-com.ezproxyd.bham.ac.uk/

British Standards Institution (2010). *BS 8887-220:2010 Design for manufacture, assembly, disassembly and end-of-life processing (MADE). Terms and definitions*. Retrieved from https://bsol-bsigroup-com.ezproxyd.bham.ac.uk/

Carrell, J., Zhang, H.-C., Tate, D., & Li, H. (2009). Review and future of active disassembly. *International Journal of Sustainable Engineering*, *2*(4), 252-264. https://doi.org/10.1080/19397030903267431

Chatzilygeroudis, K., Vassiliades, V., Stulp, F., Calinon, S., & Mouret, J. (2020). A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials. *IEEE Transactions on Robotics*, *36*(2), 328-347. https://doi.org/10.1109/TRO.2019.2958211

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., & Fox, D. (2019, 20-24 May 2019). Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience. 2019 International Conference on Robotics and Automation (ICRA),

Chiaverini, S., & Sciavicco, L. (1993). The parallel approach to force/position control of robotic manipulators. *IEEE Transactions on Robotics and Automation*, *9*(4), 361-373. https://doi.org/10.1109/70.246048

Chierici, E., & Copani, G. (2016). Remanufacturing with Upgrade PSS for New Sustainable Business Models. *Procedia CIRP*, *47*, 531-536. https://doi.org/https://doi.org/10.1016/j.procir.2016.03.055

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, *31*.

Daneshmand, M., Noroozi, F., Corneanu, C., Mafakheri, F., & Fiorini, P. (2022). Industry 4.0 and prospects of circular economy: a survey of robotic assembly and disassembly. *The International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-021-08389-1

Deisenroth, M. P., Neumann, G., & Peters, J. (2013). *A Survey on Policy Search for Robotics*. now. http://ieeexplore.ieee.org/document/8186816

DiFilippo, N. M., & Jouaneh, M. K. (2018). A System Combining Force and Vision Sensing for Automated Screw Removal on Laptops. *IEEE Transactions on Automation Science and Engineering*, *15*(2), 887-895. https://doi.org/10.1109/TASE.2017.2679720

Duan, Y., Chen, X., Houthooft, R., Schulman, J., & Abbeel, P. (2016, 2016/06/19-24). *Benchmarking Deep Reinforcement Learning for Continuous Control* 33rd Int. Conf. Mach. Learn. (ICML), New York City, NY, USA. https://arxiv.org/abs/1604.06778

Fan, Y., Luo, J., & Tomizuka, M. (2019, 2019/05/20-24). A Learning Framework for High Precision Industrial Assembly. 2019 International Conference on Robotics and Automation (ICRA),

Fang, H. C., Ong, S. K., & Nee, A. Y. C. (2014). Product Remanufacturability Assessment based on Design Information. *Procedia CIRP*, *15*, 195-200. https://doi.org/https://doi.org/10.1016/j.procir.2014.06.050

Fofou, R. F., Jiang, Z., & Wang, Y. (2021). A Review on the Lifecycle Strategies Enhancing Remanufacturing. *Applied Sciences*, *11*(13). https://doi.org/10.3390/app11135937

Foo, G., Kara, S., & Pagnucco, M. (2022). Challenges of robotic disassembly in practice. *Procedia CIRP*, *105*, 513-518. https://doi.org/https://doi.org/10.1016/j.procir.2022.02.085

Fujimoto, S., Hoof, H. v., & Meger, D. (2018, 2018/07/10-15). *Addressing Function Approximation Error in Actor-Critic Methods* 35th Int. Conf. Mach. Learn. (ICML), Stockholm, Sweden.

Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems*, 3-27.

Guo, X., Zhou, M., Abusorrah, A., Alsokhiry, F., & Sedraoui, K. (2021). Disassembly Sequence Planning: A Survey. *IEEE/CAA Journal of Automatica Sinica*, *8*(7), 1308-1324. https://doi.org/10.1109/JAS.2020.1003515

Gupta, A., Devin, C., Liu, Y., Abbeel, P., & Levine, S. (2017, 24-26 Apr.). *Learning invariant feature spaces to transfer skills with reinforcement learning* 2017 Int. Conf. Learn. Represent. (ICLR), Toulon, France. https://arxiv.org/abs/1703.02949

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018, 2018/08/08). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor* Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v80/haarnoja18b.html

Heess, N. M. O., Dhruva, T., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M. A., & Silver, D. (2017). Emergence of Locomotion Behaviours in Rich Environments. *ArXiv*, *abs/1707.02286*.

Hellmuth, J. F., DiFilippo, N. M., & Jouaneh, M. K. (2021). Assessment of the automation potential of electric vehicle battery disassembly. *Journal of Manufacturing Systems*, *59*, 398-412. https://doi.org/https://doi.org/10.1016/j.jmsy.2021.03.009

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1). https://ojs.aaai.org/index.php/AAAI/article/view/11694

Hjorth, S., & Chrysostomou, D. (2022). Human–robot collaboration in industrial environments: A literature review on non-destructive disassembly. *Robotics and Computer-Integrated Manufacturing*, *73*, 102208. https://doi.org/https://doi.org/10.1016/j.rcim.2021.102208

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Hasselt, H. V., & Silver, D. (2018). Distributed Prioritized Experience Replay. *ArXiv*, *abs/1803.00933*.

Huang, J., Pham, D. T., Li, R., Qu, M., Wang, Y., Kerin, M., Su, S., Ji, C., Mahomed, O., Khalil, R., Stockton, D., Xu, W., Liu, Q., & Zhou, Z. (2021). An experimental human-robot collaborative disassembly cell. *Computers & Industrial Engineering*, *155*, 107189. https://doi.org/https://doi.org/10.1016/j.cie.2021.107189

Huang, J., Pham, D. T., Wang, Y., Qu, M., Ji, C., Su, S., Xu, W., Liu, Q., & Zhou, Z. (2019). A case study in human–robot collaboration in the disassembly of press-fitted components. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *234*(3), 654-664. https://doi.org/10.1177/0954405419883060

Ijomah, W. L. (2009). Addressing decision making for remanufacturing operations and design-for-remanufacture. *International Journal of Sustainable Engineering*, *2*(2), 91-102. https://doi.org/10.1080/19397030902953080

Ijomah, W. L., McMahon, C. A., Hammond, G. P., & Newman, S. T. (2007). Development of robust design-for-remanufacturing guidelines to further the aims of sustainable development. *International Journal of Production Research*, *45*(18-19), 4513-4536. https://doi.org/10.1080/00207540701450138

Inoue, T., Magistris, G. D., Munawar, A., Yokoya, T., & Tachibana, R. (2017, 2017/09/24-28). Deep reinforcement learning for high precision assembly tasks. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),

Islam, R., Henderson, P., Gomrokchi, M., & Precup, D. (2017). Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *ArXiv*, *abs/1708.04133*.

Jansson, K. (2016). Circular economy in shipbuilding and marine networks–a focus on remanufacturing in ship repair. Collaboration in a Hyperconnected World: 17th IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2016, Porto, Portugal, October 3-5, 2016, Proceedings 17,

Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J. A., Solowjow, E., & Levine, S. (2019, 2019/05/20-24). Residual Reinforcement Learning for Robot Control. 2019 International Conference on Robotics and Automation (ICRA),

Kamran, M., Raugei, M., & Hutchinson, A. (2021). A dynamic material flow analysis of lithium-ion battery metals for electric vehicles and grid storage in the UK: Assessing the impact of shared mobility and end-of-life strategies. *Resources, Conservation and Recycling*, *167*, 105412. https://doi.org/https://doi.org/10.1016/j.resconrec.2021.105412

Kerin, M., & Pham, D. T. (2019). A review of emerging industry 4.0 technologies in remanufacturing. *Journal of Cleaner Production*, *237*, 117805. https://doi.org/https://doi.org/10.1016/j.jclepro.2019.117805

Kerin, M., & Pham, D. T. (2020). Smart remanufacturing: a review and research framework. *Journal of Manufacturing Technology Management*, *31*(6), 1205-1235. https://doi.org/10.1108/JMTM-06-2019-0205

Khamassi, M., Velentzas, G., Tsitsimis, T., & Tzafestas, C. (2017, 2017/04/10-12). Active Exploration and Parameterized Reinforcement Learning Applied to a Simulated Human-Robot Interaction Task. 2017 First IEEE International Conference on Robotic Computing (IRC),

Kingma, D. P., & Ba, J. (2015, 7-9 May). *Adam: A Method for Stochastic Optimization* 2015 Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA. http://arxiv.org/abs/1412.6980

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238-1274. https://doi.org/10.1177/0278364913495721

Kong, L., Li, C., Jiang, J., & Pecht, M. G. (2018). Li-Ion Battery Fire Hazards and Safety Strategies. *Energies*, *11*(9). https://doi.org/10.3390/en11092191

Kroemer, O., Niekum, S., & Konidaris, G. (2021). A review of robot learning for manipulation: Challenges, representations, and algorithms. *The Journal of Machine Learning Research*, *22*(1), 1395-1476.

KUKA. (2022). *LBR iiwa*. Retrieved from https://www.kuka.com/en-gb/products/robotics-systems/industrial-robots/lbr-iiwa

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., & Abbeel, P. (2018). Model-Ensemble Trust-Region Policy Optimization. *ArXiv*, *abs/1802.10592*.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2017). 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. https://doi.org/10.1177/0278364917710318

Li, R., Pham, D. T., Huang, J., Tan, Y., Qu, M., Wang, Y., Kerin, M., Jiang, K., Su, S., Ji, C., Liu, Q., & Zhou, Z. (2020). Unfastening of Hexagonal Headed Screws by a Collaborative Robot. *IEEE Transactions on Automation Science and Engineering*, *17*(3), 1455-1468. https://doi.org/10.1109/TASE.2019.2958712

Li, X., Liu, H., & Dong, M. (2022). A General Framework of Motion Planning for Redundant Robot Manipulator Based on Deep Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, *18*(8), 5253-5263. https://doi.org/10.1109/TII.2021.3125447

Liao, H., Deng, Q., Wang, Y., Guo, S., & Ren, Q. (2018). An environmental benefits and costs assessment model for remanufacturing process under quality uncertainty. *Journal of Cleaner Production*, *178*, 45-58.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016, 2016/05/02-04). *Continuous control with deep reinforcement learning* 2016 Int. Conf. Learn. Represent. (ICLR), San Juan, Puerto Rico.

Liu, Z., Li, T., Jiang, Q., & Zhang, H. (2014). Life Cycle Assessment–based Comparative Evaluation of Originally Manufactured and Remanufactured Diesel Engines [https://doi.org/10.1111/jiec.12137]. *Journal of Industrial Ecology*, *18*(4), 567-576. https://doi.org/https://doi.org/10.1111/jiec.12137

Luo, J., Solowjow, E., Wen, C., Ojea, J. A., & Agogino, A. M. (2018, 2018/10/01-05). Deep Reinforcement Learning for Robotic Assembly of Mixed Deformable and Rigid Objects. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),

Luo, J., Solowjow, E., Wen, C., Ojea, J. A., Agogino, A. M., Tamar, A., & Abbeel, P. (2019). Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic

Assembly. *2019 International Conference on Robotics and Automation (ICRA)*, 3080-3087.

Ma, Y., Xu, D., & Qin, F. (2021). Efficient Insertion Control for Precision Assembly Based on Demonstration Learning and Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, *17*(7), 4492-4502. https://doi.org/10.1109/TII.2020.3020065

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., & Bergstra, J. (2018, 2018/09/20). *Benchmarking Reinforcement Learning Algorithms on Real-World Robots* Proceedings of The 2nd Conference on Robot Learning, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v87/mahmood18a.html

Mania, H., Guy, A., & Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *ArXiv*, *abs/1803.07055*.

Mckerrow, P. (1991). *Introduction to robotics*. Addison-Wesley Longman Publishing Co., Inc.

Michalos, G., Makris, S., Tsarouchi, P., Guasch, T., Kontovrakis, D., & Chryssolouris, G. (2015). Design Considerations for Safe Human-robot Collaborative Workplaces. *Procedia CIRP*, *37*, 248-253. https://doi.org/https://doi.org/10.1016/j.procir.2015.08.014

Mitchell, P., & Morgan, J. (2015). *Employment and the circular economy Job creation in a more resource efficient Britain*. https://doi.org/10.13140/RG.2.1.1026.5049

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv*, *abs/1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533. https://doi.org/10.1038/nature14236

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Montgomery, W. H., & Levine, S. (2016, 2016/07/15). Guided Policy Search via Approximate Mirror Descent. NIPS,

Mouret, J.-B. (2016). Micro-Data Learning: The Other End of the Spectrum. *ArXiv*, *abs/1610.00946*.

Nedjalkov, A., Meyer, J., Köhring, M., Doering, A., Angelmahr, M., Dahle, S., Sander, A., Fischer, A., & Schade, W. (2016). Toxic Gas Emissions from Damaged Lithium Ion Batteries—Analysis and Safety Enhancement Solution. *Batteries*, *2*(1). https://doi.org/10.3390/batteries2010005

Nuttin, M., Brussel, H. V., Peirs, J., Soembagijo, A., & Sonck, S. (1997). Learning the peg-into-hole assembly operation with a connectionist reinforcement technique. *Comput. Ind.*, *33*(1), 101–109. https://doi.org/10.1016/s0166-3615(97)00015-8

Ogenyi, U. E., Liu, J., Yang, C., Ju, Z., & Liu, H. (2021). Physical Human–Robot Collaboration: Robotic Systems, Learning Methods, Collaborative Strategies, Sensors, and Actuators. *IEEE Transactions on Cybernetics*, *51*(4), 1888-1901. https://doi.org/10.1109/TCYB.2019.2947532

Ong, S. K., Chang, M. M. L., & Nee, A. Y. C. (2021). Product disassembly sequence planning: state-of-the-art, challenges, opportunities and future directions. *International Journal of Production Research*, *59*(11), 3493-3508. https://doi.org/10.1080/00207543.2020.1868598

Östlin, J. (2008). *On remanufacturing systems: analysing and managing material flows and remanufacturing processes* Institutionen för ekonomisk och industriell utveckling].

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009, May). Learning and generalization of motor skills by learning from demonstration. IEEE Int. Conf. Robot. Autom. (ICRA), Kobe, Japan.

Pham, D. T., & Castellani, M. (2014). Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms. *Soft Computing*, *18*, 871-903.

Pham, D. T., & Castellani, M. (2015). A comparative study of the Bees Algorithm as a tool for function optimisation. *Cogent Engineering*, *2*(1), 1091540.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm—a novel tool for complex optimisation problems. In *Intelligent production machines and systems* (pp. 454-459). Elsevier.

Polydoros, A. S., & Nalpantidis, L. (2017). Survey of Model-Based Reinforcement Learning: Applications on Robotics. *Journal of Intelligent & Robotic Systems*, *86*(2), 153-173. https://doi.org/10.1007/s10846-017-0468-y

Poschmann, H., Brüggemann, H., & Goldmann, D. (2020). Disassembly 4.0: A Review on Using Robotics in Disassembly Tasks as a Way of Automation [https://doi.org/10.1002/cite.201900107]. *Chemie Ingenieur Technik*, *92*(4), 341-359. https://doi.org/https://doi.org/10.1002/cite.201900107

Priyono, A., Ijomah, W., & Bititci, U. S. (2016). Disassembly for remanufacturing: A systematic literature review, new model development and future research needs. *Journal of Industrial Engineering and Management*, *9*(4), 899-932. https://doi.org/10.3926/jiem.2053

Ren, T., Dong, Y., Wu, D., & Chen, K. (2018). Learning-Based Variable Compliance Control for Robotic Assembly. *Journal of Mechanisms and Robotics*, *10*(6). https://doi.org/10.1115/1.4041331

Roveda, L., Pallucca, G., Pedrocchi, N., Braghin, F., & Tosatti, L. M. (2018). Iterative Learning Procedure With Reinforcement for High-Accuracy Force Tracking in Robotized Tasks. *IEEE Transactions on Industrial Informatics*, *14*(4), 1753-1763. https://doi.org/10.1109/TII.2017.2748236

Salvato, E., Fenu, G., Medvet, E., & Pellegrino, F. A. (2021). Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access*, *9*, 153171-153187. https://doi.org/10.1109/ACCESS.2021.3126658

Schoettler, G., Nair, A., Luo, J., Bahl, S., Ojea, J. A., Solowjow, E., & Levine, S. (2020, 2020/10/25-29). Deep Reinforcement Learning for Industrial Insertion Tasks with Visual Inputs and Natural Rewards. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),

Sergey, L., & Vladlen, K. (2013, 2013/05/26). *Guided Policy Search* https://proceedings.mlr.press/v28/levine13.html

Shahbazi, S., Johansen, K., & Sundin, E. (2021). Product Design for Automated Remanufacturing—A Case Study of Electric and Electronic Equipment in Sweden. *Sustainability*, *13*(16). https://doi.org/10.3390/su13169039

Sharma, K., Shirwalkar, V., & Pal, P. K. (2013, 2013/12/16-18). Intelligent and environment-independent Peg-In-Hole search strategies. 2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE),

Sigaud, O., & Stulp, F. (2019). Policy search in continuous action domains: An overview. *Neural Networks*, *113*, 28-40. https://doi.org/https://doi.org/10.1016/j.neunet.2019.01.011

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354-359. https://doi.org/10.1038/nature24270

Silver, T., Allen, K. R., Tenenbaum, J. B., & Kaelbling, L. P. (2019). Residual Policy Learning. *ArXiv*, *abs/1812.06298*.

Soh, S. L., Ong, S. K., & Nee, A. Y. C. (2014). Design for Disassembly for Remanufacturing: Methodology and Technology. *Procedia CIRP*, *15*, 407-412. https://doi.org/https://doi.org/10.1016/j.procir.2014.06.053

Sundin, E., & Lee, H. M. (2012). In what way is remanufacturing good for the environment? Design for innovative value towards a sustainable society: Proceedings of EcoDesign 2011: 7th International Symposium on Environmentally Conscious Design and Inverse Manufacturing,

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (second ed.). The MIT Press.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., & Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*.

Thomas, G., Chien, M., Tamar, A., Ojea, J. A., & Abbeel, P. (2018, 2018/05/21-25). Learning Robotic Assembly from CAD. 2018 IEEE International Conference on Robotics and Automation (ICRA),

Tiboni, G., Arndt, K., & Kyrki, V. (2023). DROPO: Sim-to-real transfer with offline domain randomization. *Robotics and Autonomous Systems*, *166*, 104432.

Tsai, Y. Y., Xu, H., Ding, Z., Zhang, C., Johns, E., & Huang, B. (2021). DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration. *IEEE Robotics and Automation Letters*, *6*(2), 3168-3175. https://doi.org/10.1109/LRA.2021.3062311

Villani, V., Pini, F., Leali, F., & Secchi, C. (2018). Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications. *Mechatronics*, *55*, 248-266. https://doi.org/https://doi.org/10.1016/j.mechatronics.2018.02.009

Vongbunyong, S., & Chen, W. H. (2015). *Disassembly Automation: Automated Systems with Cognitive Abilitie*. Springer, Cham.

Vongbunyong, S., Kara, S., & Pagnucco, M. (2013). Basic behaviour control of the vision-based cognitive robotic disassembly automation. *Assembly Automation*, *33*(1), 38-56. https://doi.org/10.1108/01445151311294694

Wan, A., Xu, J., Chen, H., Zhang, S., & Chen, K. (2017). Optimal Path Planning and Control of Assembly Robots for Hard-Measuring Easy-Deformation Assemblies. *IEEE/ASME Transactions on Mechatronics*, *22*(4), 1600-1609. https://doi.org/10.1109/TMECH.2017.2671342

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E. D., Zhang, M. S., Zhang, G., Abbeel, P., & Ba, J. (2019). Benchmarking Model-Based Reinforcement Learning. *ArXiv*, *abs/1907.02057*.

Wegener, K., Chen, W. H., Dietrich, F., Dröder, K., & Kara, S. (2015). Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries. *Procedia CIRP*, *29*, 716-721. https://doi.org/https://doi.org/10.1016/j.procir.2015.02.051

Whitney, D. E. (1982). Quasi-Static Assembly of Compliantly Supported Rigid Parts. *Journal of Dynamic Systems, Measurement, and Control*, *104*(1), 65-77. https://doi.org/10.1115/1.3149634

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3), 229-256. https://doi.org/10.1007/BF00992696

Xiao, L., Liu, W., Guo, Q., Gao, L., Zhang, G., & Chen, X. (2018). Comparative life cycle assessment of manufactured and remanufactured loading machines in China. *Resources, Conservation and Recycling*, *131*, 225-234. https://doi.org/https://doi.org/10.1016/j.resconrec.2017.12.021

Xu, J., Hou, Z., Liu, Z., & Qiao, H. (2019a). Compare Contact Model-based Control and Contact Model-free Learning: A Survey of Robotic Peg-in-hole Assembly Strategies. *ArXiv*, *abs/1904.05240*.

Xu, J., Hou, Z., Wang, W., Xu, B., Zhang, K., & Chen, K. (2019b). Feedback Deep Deterministic Policy Gradient With Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks. *IEEE Transactions on Industrial Informatics*, *15*(3), 1658-1667. https://doi.org/10.1109/TII.2018.2868859

Zhang, X., Wang, Y., Xiang, Q., Zhang, H., & Jiang, Z. (2020a). Remanufacturability evaluation method and application for used engineering machinery parts based on fuzzy-EAHP. *Journal of Manufacturing Systems*, *57*, 133-147. https://doi.org/https://doi.org/10.1016/j.jmsy.2020.08.016

Zhang, X., Zhang, M., Zhang, H., Jiang, Z., Liu, C., & Cai, W. (2020b). A review on energy, environment and economic assessment in remanufacturing based on life cycle assessment method. *Journal of Cleaner Production*, *255*, 120160. https://doi.org/https://doi.org/10.1016/j.jclepro.2020.120160

Zhang, Y., Lu, H., Pham, D. T., Wang, Y., Qu, M., Lim, J., & Su, S. (2019). Peg–hole disassembly using active compliance. *Royal Society Open Science*, *6*(8), 190476. https://doi.org/10.1098/rsos.190476

Zhou, Z., Liu, J., Pham, D. T., Xu, W., Ramirez, F. J., Ji, C., & Liu, Q. (2018). Disassembly sequence planning: Recent developments and future trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *233*(5), 1450-1471. https://doi.org/10.1177/0954405418789975