Learning and Generalising Object Extraction Skill for Contact-rich Disassembly Tasks: An Introductory Study

Antonio Serrano Muñoz (aserrano@mondragon.edu)
Mondragon Unibertsitatea https://orcid.org/0000-0001-9938-673X
Nestor Arana-Arexolaleiba
Mondragon Unibertsitatea
Dimitrios Chrysostomou
Mondragon Unibertsitatea
Simon Bøgh
Mondragon Unibertsitatea

Research Article

Keywords: Circular economy, Remanufacturing, Disassembly, Robotics, Reinforcement Learning, Contact-rich Manipulation

Posted Date: March 24th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-331448/v1

Noname manuscript No.
(will be inserted by the editor)

Learning and generalising object extraction skill for contact-rich disassembly tasks: An introductory study

Antonio Serrano-Muñoz · Nestor Arana-Arexolaleiba · Dimitrios Chrysostomou · Simon Bøgh

Received: date / Accepted: date

Abstract Remanufacturing automation must be designed to be flexible and robust enough to overcome the uncertainties, conditions of the products, and complexities in the process's planning and operation. Machine learning, particularly reinforcement learning, methods are presented as techniques to learn, improve, and generalise the automation of many robotic manipulation tasks (most of them related to grasping, picking, or assembly). However, not much has been exploited in remanufacturing, in particular in disassembly tasks. This work presents the State-of-the-Art of contact-rich disassembly using reinforcement learning algorithms and a study about the object extraction skill's generalisation when applied to contact-rich disassembly tasks. The generalisation capabilities of two State-of-the-Art reinforcement learning agents (trained in simulation) are tested and evaluated in simulation and real-world while perform a disassembly task. Results shows that, at least, one of the agents can generalise the contact-rich extraction skill. Also, this work identifies key concepts and gaps for the reinforcement learning algorithms' research and application on disassembly tasks.

Keywords Circular economy · Remanufacturing · Disassembly · Robotics · Reinforcement Learning · Contact-rich Manipulation

Antonio Serrano-Muñoz
Mondragon Unibertsitatea. Loramendi 4., Arrasate, Spain
E-mail: aserrano@mondragon.edu

Nestor Arana-Arexolaleiba
Mondragon Unibertsitatea & Aalborg University, Denmark

Dimitrios Chrysostomos
Aalborg University, Denmark

Simon Bøgh
Aalborg University, Denmark

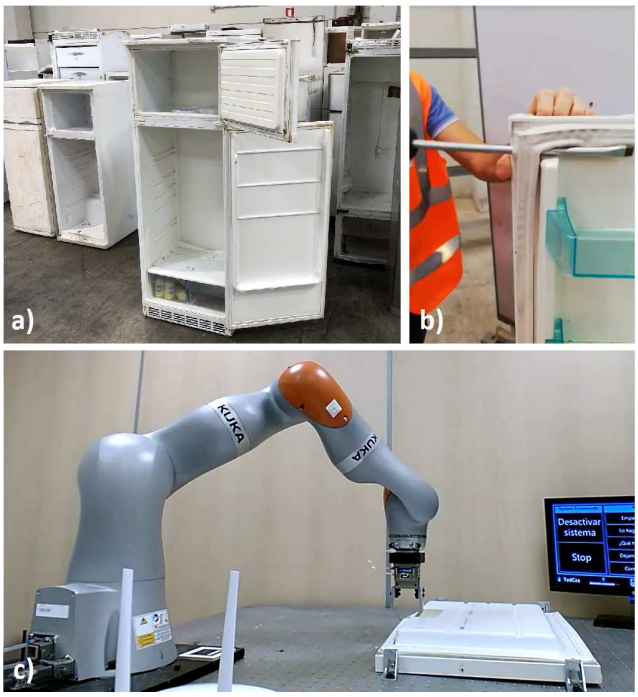

Fig. 1 Disassembly of end-of-life refrigerators. a) Stock of refrigerators in the warehouse. b) Manual removing of the door's gasket using a screwdriver. c) Experimental prototype to remove the door's gasket in our lab.

1 Introduction

As the world's population exponentially grows, consumption rates and the demand for new products also increase dramatically. Many end-of-life (EOL) products are continuously being disposed of, leading to several environmental problems. Responsible EOL treatment, which may include reusing, recycling, or remanufacturing [1] products or parts, is desirable in dealing with these products [2]. These processes can be beneficial

both environmentally [3] and economically [4][5]. Waste is minimised while valuable components and materials are recovered.

The disassembly of products is one of the EOL treatment processes' primary steps and involves the extraction and segregation of the desired components, parts, or materials from the product [6]. The disassembly does not only input towards EOL treatment but also allows the repair and maintenance of products.

Automation has been successfully implemented for decades in traditional manufacturing processes e.g. assembly, bin picking and material handling. However, disassembly processes, where manual labour is preferred (as shown in Figure 1 (b)), introduce great challenges in the handling of the takeback products. These products are often returned back from the customers in used condition with many uncertainties in their physical appearance and condition that renders them difficult to handle and eventually successfully disassemble, like the refrigerators shown in Figure 1 (a), for example.

Therefore, numerous (semi-)automated robotic disassembly cells have been introduced utilising more flexible approaches able to adapt to such uncertainties. Vongbunyong et al. [7] investigated a cognitive-based vision system that introduced reasoning, monitoring of execution and learning abilities so it can disassemble products without prior product information. Bdiwi et al. [8] studied the disassembly process of electric motors based on image processing techniques for screw detection and classification, while Schneider et al. [9] proposed an algorithm able to compute complex nonlinear disassembly paths for objects which collide with each other during the disassembly process.

The majority of these works proposed controllers based on classical control theory, which are widely used to control dynamic systems. However, they need a model of the system to compute the controller [10], which in our case, it is difficult due to the variety of conditions.

Intelligent control, particularly Machine Learning (ML), has been proposed as an alternative to control a dynamic system where the controllers will adapt to achieve the goal. ML, compared with humans, can identify more features in the signals [11]. Also, ML algorithms play an essential role in sustainable manufacturing in general, helping to optimise energy and material consumption [12].

As an ML area, Reinforcement Learning (RL) has been studied in several domains, including control theory. These algorithms can learn and generalise skills [13] through interactions with the environment detecting the features automatically, in contrast with supervised learning, where extensive human intervention is needed to label thousands or millions of data.

This work presents the State-of-the-Art of contact-rich disassembly using RL algorithms. Furthermore, as far as our knowledge, it presents the first attempt to study the generalisation capabilities of two State-of-the-Art RL algorithms to learn the object extraction skill when applied to the contact-rich disassembly task.

The rest of this article is organised as follows: Related works and critical analysis of the State-of-the-Art of contact-rich disassembly tasks in Section 2. The problem formulation and discussion of the RL approach and the selected algorithms in Section 3. The method and details of the experiments in Section 4 and Section 5, respectively. A deeper discussion and comparison of the obtained results in Section 6. And a few concluding remarks in Section 7.

2 Related works

This section presents the State-of-the-Art of contact-rich disassembly tasks using RL. Also, some related works using RL for contact-rich assembly are analysed to find key concepts that could be used to perform disassembly tasks.

2.1 Assembly

Several works have applied deep RL algorithms in contact-rich assembly manipulation tasks. They are classified based on the reinforcement learning taxonomy[1] described by the OpenAI.

Model-based algorithms learn a model of the environment that is used to predict the results of actions taken, allowing the agent to plan and act accordingly.

Luo et al. [14] used a model-based policy search incorporating a world dynamics model to learn the peg-in-hole task where the hole is made of deformable material. In another article, Luo et al. [15] combined a model-based algorithm with an operational space force controller for a high-precision peg-in-hole task. Despite the neural network controller produces slightly better results against the RL algorithm, it suffers from high variance related to the force/torque sensor's accuracy. Both systems work when the peg is close to the hole, but they were not evaluated from arbitrary starting positions in free space to generalise the learned skill.

Thomas et al. [16] proposed a method that combines motion planning to generate collision-free trajectories to learn assembly skills. However, their approach relies on high-quality geometric information from the

[1] https://spinningup.openai.com/en/latest/

object's CAD model to assemble, which is not generalisable. Ding et al. [17] propose a transferable force-torque dynamics model for the peg-in-hole task. They implement a multi-pose force-torque state representation to handle ambiguous feedback from sensors and an offline data generation method to reduce the number of real-world interactions. However, their approach relies on finetuning to be sample-efficient. Fan et al. [18] developed a framework that combines a model-based algorithm for computing optimal trajectories with positional and force/torque feedback and a model-free algorithm to learn manipulation skills for precision assembly tasks.

Value-based algorithm is one of the basic model-free RL algorithms, where the Q value is used as a reference to take the decision.

Li et al. [19] used deep Q-Learning to adjust the robotic end-effector's pose and orientation for the circuit breaker assembly. To learn the manipulation skill, they developed a reward system based on the support vector machine (SVM) classification model. Despite this work shown generalisation across various random initial positions, the average success rate is not good enough and learns a reward classifier is not feasible most times.

Oikawa et al. [20] used DQN to output the stiffness matrices as actions for admittance control. This approach maintains high control performance by outputting the position and force commands in short cycles for peg-in-hole and gear-insertion tasks. However, the admittance model requires higher sampling times to perform contact-rich tasks without unstable motion.

Policy Optimisation algorithms are also model-free algorithms that improve the policy parameters directly without considering the Q value.

Levine et al. [21] employ a hybrid approach that combines the flexibility of mode-free algorithms with the efficiency of model-based algorithms to optimise a linear-gaussian controller to learn a range of motion skills. To generalise the learning, they located the targets in various initial positions.

Lee et al. [22] used TRPO to perform the peg-in-hole task. To enable efficient real robot training, they trained a model to encode heterogeneous sensory inputs, such as RGB images, force/torque data, and robot proprioceptive data, into a compact multimodal representation.

Actor-Critic algorithms integrate Q-Learning and Policy Gradient algorithms.

Some authors combine RL algorithms with conventional feedback control (residual RL) to reduce the exploratory behaviour due to safety concerns and solve contact-rich manipulation tasks. Johannink et al. [23] use TD3 combined with a human-designed controller to insert a block in a hole. Although the proposed algorithm can learn a feedback controller that adapts to variations in the orientations, it requires a carefully crafted vision setup to infer the target blocks' positions and angles in real-world scenarios. Schoettler et al. [24] used a P-controller to inject prior information into the RL algorithm to speed-up the training process and minimise unsafe exploration behaviour. They learned a policy only from images (using those as state representation and goal specification) with Soft Actor-Critic (SAC) and TD3. Beltran et al. [25] used a PD controller to speed up the control policy's learning for the peg-in-hole task given an uncertain goal position. Despise they improved generalisation capabilities, the framework is very sensitive to the force-control parameters which may cause undesired behaviour during manipulation. Beltran et al. [26] combined SAC with a traditional force control algorithm to perform the peg insertion task. However, their implementation is highly dependent on the controller's hyperparameters.

Li et al. [27] used DDPG, as distinct from previous work [19] where they used Q-Learning, to perform circuit breaker assembly. In this case, they limit the maximum commanded velocity, the joint position limits, the range of the end-effector, and the range of each joint's torque parameters to guarantee safety constraints.

Luo and Li [28] developed a technique that allows a recurrent distributed DDPG algorithm to perform peg-in-hole and lap joint tasks by augmenting human demonstrations with successful episodes generated by the agents from replay buffers. Also, Luo and Li [29] introduced recurrency in a distributed DDPG to study partially observable assembly tasks, using as observation only the force/torque measurements from the sensor mounted on the robot end-effector.

Wu et al. [30] trained an encoder-decoder network to learn dense rewards from images and force/torque feedback. The SAC algorithm uses the learned dense reward function to perform peg-in-hole and object insertion tasks. The proposed algorithm reached better results on the evaluated tasks against other reward functions (sparse reward, handcrafted continuous reward), but it needs to be trained on the specific task.

2.2 Disassembly

Few works exploit RL to perform contact-rich disassembly tasks. Kristensen et al. [31] used Q-learning algorithm to train and test agents into their own deployed framework for robotic unscrewing tasks.

Simonivc et al. [32] used the information obtained during disassembly tasks to perform assembly tasks. In

this case, they implemented a hierarchical RL algorithm and a graph representation under the criteria that an assembly task is just a reverse execution of the corresponding multiple-stage disassembly task. The disassembly is complete when the motion is unconstrained in the desired degrees of freedom, and the built graph is used to perform the corresponding assembly task.

Strategies to allow the separation of a fixed component into a slot are presented by Herold et al. [33]. The authors identify that adjusting the robot's position end-effector proportionally to the measured forces and including oscillating motion could be a good solution for the task. However, they execute the task performing predefined actions (the robot does not learn anything from the process), making it difficult to generalise to other scenarios.

2.3 Critic analysis

Unlike the assembly process, the information about the state of the product (physical and mechanical properties, for example) is often inaccurate or incomplete at the beginning of the process but is revealed during it. Then, even when the disassembly is not necessarily the reverse of assembly, it is possible to use the assembly's advances in our favour to identify trends and relevant concepts. Table 1 shows how RL algorithms have been successfully applied to learn robotic manipulation skills, mostly related to the assembly of products.

The State-of-the-Art shows that actor-critic (that combine the best from policy optimisation algorithms and value-based algorithms) is the most widely used algorithm, followed by model-based algorithms, to learn manipulation skills. The most frequently learned skill is object insertion (widely applied to peg-in-hole like tasks). Those publications consider that the object to be manipulated is already grasped by the robot's end-effector.

Force/torque, vision, and pose are the primary information used to observe the system's state and act according to it. Vision seems to be not extensively used. However, some of those publications suggest including the vision as a promising direction in future works.

Concerning disassembly, few works apply RL to perform tasks, particularly those that require contact-rich manipulation skills (and, as far as our knowledge, no-one has exploited the extraction skill). This gap opens the door to a vast field in the research and application of RL algorithms for flexible remanufacturing processes in general and disassembly in particular.

3 Problem formulation

The nature of the disassembly process requires flexible and robust enough autonomous systems to overcome some issues such as the large variety and physical uncertainties associated with the EOL product condition and operation complexities. Driven by those issues, this work explores and evaluates two State-of-the-Art RL algorithms' generalisation capabilities to learn the extraction skill to perform a contact-rich disassembly task.

In this case, the problem was formulated as a Markov Decision Process (MDP), as shown in Figure 2, modelled with a finite-horizon discounted return.

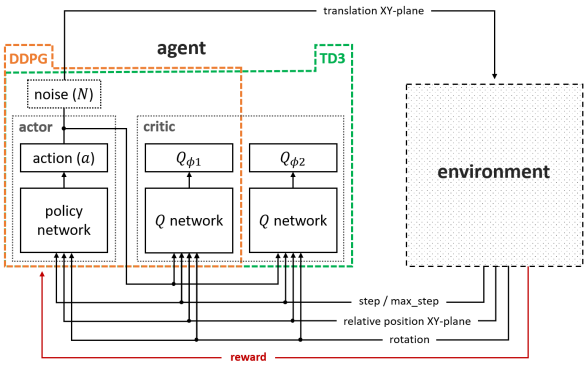

Fig. 2 Reinforcement learning environment

At every timestep of interaction with the environment, the agent sees an observation o of the complete description of the state $s \in S$ of the environment. Then, it decides which action $a \in A$ to take from the action space using, in our case, a parameterised policy π_θ. The environment, which changes by itself and/or by the agent's action, gives a reward signal $r_t = R(s_t, a_t, s_{t+1})$ to the last one to measure how good or bad is the new state. The gent's goal is to maximise the cumulative reward discounted by a factor $\gamma \in (0, 1]$ by adjusting the policy's behaviour via some optimisation algorithm.

3.1 Reinforcement learning algorithms

As it is indicated in subsection 2.1, model-based RL algorithms need a model of the system. Since the disassembly process can be dealing with a large variety of products, this is not possible. Then, only model-free algorithms will be considered. Actor-critic algorithms were selected because they combine the best from both policy optimisation algorithms and value-based algorithms. They use the learned value function as a baseline to update the actor's policy. These al-

Taxonomy	Algorithm(s)	Publication	Observation (primary sensors)			Skill(s)
			Force/Torque	Vision	Pose*	
Model-based	GPS	Luo et al. [14]	x			insertion
	iLQG	Luo et al. [15]	x			insertion
	GPS	Thomas et al. [16]			x	insertion
	M-based/A3C	Ding et al. [17]	x		x	insertion
	GPS/DDPG	Fan et al. [18]	x		x	insertion, shape joint
Value-based	Q-Learning	Li et al. [19]	x		x	shape joint
	DQN	Oikawa et al. [20]	x		x	insertion
	Q-Learning	Kristensen et al. [31]	x		x	unscrewing
	SARSA	Simonivc et al. [32]	x		x	insertion
Policy Optimisation	GPS	Levine et al. [21]	x		x	insertion, screwing
	TRPO	Lee et al. [22]	x	x	x	insertion
Actor-Critic	TD3	Johannink et al. [23]	x		x	insertion
	TD3, SAC	Schoettler et al. [24]		x		insertion
	SAC	Beltran et al. [25]	x		x	insertion
	SAC	Beltran et al. [26]	x		x	insertion
	DDPG	Li et al. [27]	x		x	shape joint
	DDPG	Luo and Li [28]	x		x	shape joint
	DDPG	Luo and Li [29]	x			shape joint
	SAC	Wu et al. [30]	x	x		insertion

Table 1 Summary of related works according to the RL algorithms' taxonomy. The primary sensors used for the observations and the robotic skills involved in the tasks are listed. *The field *pose* groups information from one or many of the next sources: the end-effector's position/orientation, manipulated object's position/orientation, pose error. The highlighted rows are publications about disassembly or publications that are very close to this area.

gorithms hold the promise of delivering faster convergence [34] while allowing us to learn from experience (off-policy). To learn the extraction skill for disassembly tasks two model-free off-policy actor-critic RL algorithm for the continuous domain were selected: DDPG and TD3.

Deep Deterministic Policy Gradients (DDPG) is a model-free and deterministic off-policy actor-critic algorithm. It uses deep function approximators to learn the policy (and to estimate the action-value function) in high-dimensional, continuous action spaces [35].

Twin Delayed Deep Deterministic policy gradient algorithm (TD3) is an actor-critic algorithm based on DDPG. This algorithm relies on double Q-learning, target policy smoothing and delayed policy updates to address the problems introduced by overestimation bias, leading to estimates and suboptimal policies, in actor-critic algorithms [36].

4 Method

This initial research, about the extraction skill to perform contact-rich disassembly, uses a single-arm robotic manipulator and a target object to be disassembled on a table, as shown in Figure3. These elements form the prototype's simplified version, as shown in Figure 1 (c).

The agents will learn how to move the robotic manipulator's end-effector, on the cartesian plane parallel to the table, to perform the contact-rich extraction/separation of two rigid objects (object extraction skill). The action taken, the received observation, and perceived reward shown in Figure 2, together with other concepts related to the RL problem are discussed below.

4.1 Setup to learn object extraction skill

Observations: To evaluate the agents's generalisation capabilities an observation based on the object or manipulator pose was designed. The observation space is composed by the normalised length of the episode, represented as the ratio between the current timestep and the maximum allowed timestep (max_steps); the relative position in the XY-plane of the TCP with respect to the initial grasping pose (pos_{XY}), and the rotation of the object (between 0 and 180 degrees scaled to the $[-1, 1]$ interval).

Actions: The action space is a 2-dimensional vector. Each component maps to the respective translation (in centimetres) of the robot's end-effector in the $[-1, 1]$ continuous interval on the XY-plane.

Reward: A deterministic sparse-reward function provides the reward at the end of the episodes. The environment gives the agent a positive reward equal to the maximum timestep (max_steps) for successful extractions. If the extraction fails because a critic force is detected or the maximum timestep is reached, the agent perceives a reward from the continuous function

described by the Equation 1. This function heavily penalises short-displacement extractions. The instantaneous reward is empty (zero) during the execution of the task.

$$r = -max\_steps/2 + e^{ln(max\_steps+1)||pos_{XY}/0.1||} - 1 (1)$$

A sparse-reward function is used to identify how the agents learn and generalise the extraction skill. However, instead of guide the agents to the goal, the sparse-reward makes the problem more challenging to solve.

Episode termination: The episodes end when the agent performs the corresponding disassembly task: extract the fixed object from the slotted one. Also, the episodes end when a critic force is detected or when the maximum timestep (max_steps) is reached regardless of whether the task was successfully executed or not.

Agents architectures: Both, the policy and the $Q_{\phi 1}$ network for DDPG (also, the $Q_{\phi 2}$ network for TD3) have the same architecture. They receive as input the observation space's components, concatenated as flatten vector, followed by two hidden dense layers of 32 neurons each one. The hidden layers use $ReLU$ as the activation function. The policy has two output neurons that use the hyperbolic tangent function ($tanh$) to fit the action to the expected interval, and the Q networks use a linear activation function for their output neuron.

5 Experiments

The proposed RL experiments were trained and tested in simulation and validated in the real world. All components were connected and operated through a custom framework, developed in our lab, using Gym and Robot Operating System (ROS) Melodic. The simulated environment was developed on the novel Omniverse Isaac Sim 2020.2.2[2] robotics specific simulation platform from NVIDIA. In simulation and reality, the experiments were conducted using the highly sensitive, compliant and lightweight KUKA LBR Iiwa 14 R820 collaborative robot with a payload capacity of 14Kg.

The target to be disassembled contains two solid objects, as shown in Figure 3: a slotted fixed base attached to the table and an embedded object (that fits inside the base) gripped by the robot's end-effector. The dimensions are 0.1 x 0.1 x 0.03 metres for the fixed base, 0.1 x 0.02 x 0.01 metres for the slot, and 0.1 x 0.02 x 0.04 metres for the embedded object. The slot is centred by 0.01 metres deep with respect to the base's upper face.

The RL was implemented using the open-source library RLlib v1.0.0. The architectures for both agents were implemented using the Keras API form TensorFlow.

The neural network parameters were learned using the Adam optimiser with a learning rate of 10^{-3} for both, actor and critic. The training was done using an experience buffer replay of size 10^5 and batches of size 256. The discount factor was 0.99. The agents used the same exploration noise functions described in their original papers. Ornstein-Uhlenbeck noise with $\theta = 0.15$, $\sigma = 0.2$ and a base scale of 0.1 for DDPG. Mean-zero Gaussian noise with a standard deviation of 0.1 for TD3. Both explorations were reduced linearly, from an initial scale of 1.0 to a final scale of 0.001, throughout training during 150 thousand timesteps. Also, for TD3 a mean-zero Gaussian noise with $\sigma = 0.1$ was used for target policy smoothing clipped to $(-0.5, 0.5)$.

Ten training sessions were done for each algorithm to try to identify repetitive behaviours. They were performed using a computer with an Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, 125GiB of RAM and a GPU GeForce RTX 2080 Ti. Each session was stopped at 375 thousand timesteps.

For testing in simulation and real-world, the best policies were selected, attending to the mean reward obtained during training. Figure 3 shows a typical extraction sequence, in simulation and real-world, for DDPG and TD3.

6 Results and discussion

6.1 Training in simulation

Figure 4 shows the mean and standard deviation of the perceived reward (top plot) for both agents in the 10 training sessions. Also, it shows the mean and standard deviation of the estimated Q-value returned by the Q-network (bottom plot) for both agents. In the case of the TD3 agent, the Q-value corresponds to the $Q_{\phi 1}$-network used to optimise the policy. This plot reveals that DDPG can better learn the extraction skill than TD3, for the setup described in previous sections.

DDPG achieved a great performance at the end of its exploration stage, then it improved the performance during the exploitation stage very slowly. Even when its Q-function overestimated the Q-value during the exploitation stage, this value remained practically constant around the maximum expectation. On the other side, TD3 began to learn the skill but at a certain point (at the end of its exploration stage), its learning performance decayed without recovering. This behaviour results from the poor Q-function estimation used as a baseline to update its actor's policy, as shown in Figure 4 (bottom plot).

[2] https://developer.nvidia.com/isaac-sim

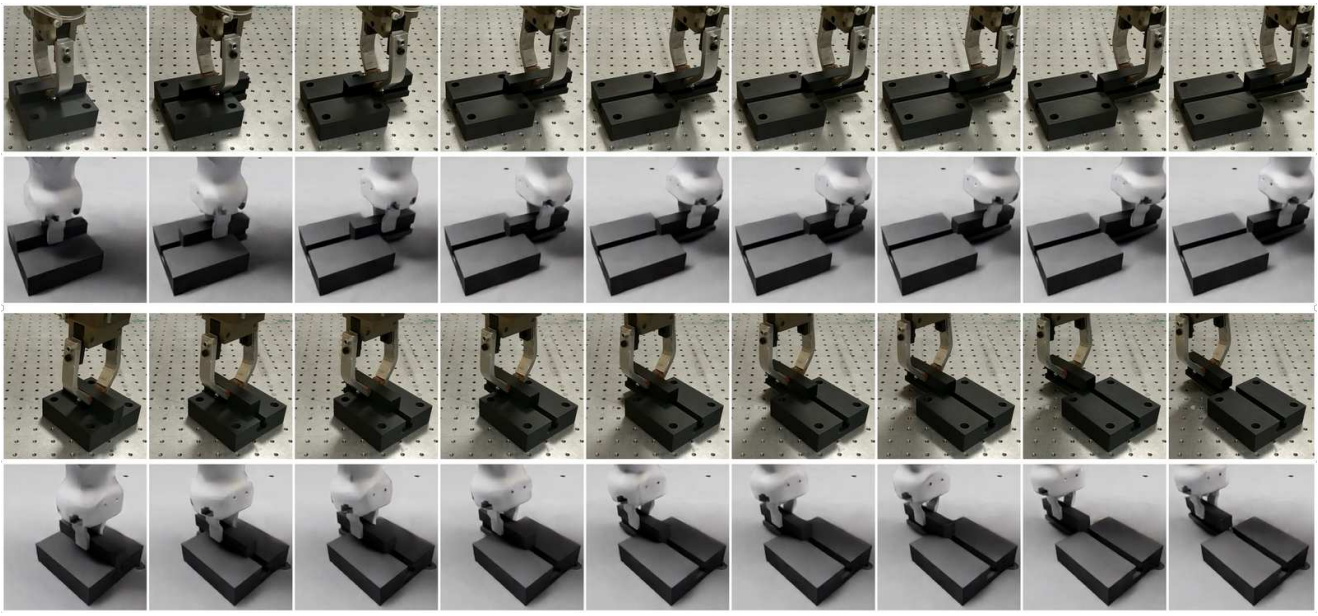

Fig. 3 Extraction task's sequence using the best-learned policies for DDPG in real-world and simulation (first and second rows respectively) and TD3 in real-world and simulation (third and fourth rows respectively). Frames are taken every 3 timesteps for DDPG (object rotated at 70 degrees) and every timestep for TD3 (object rotated at 130 degrees).

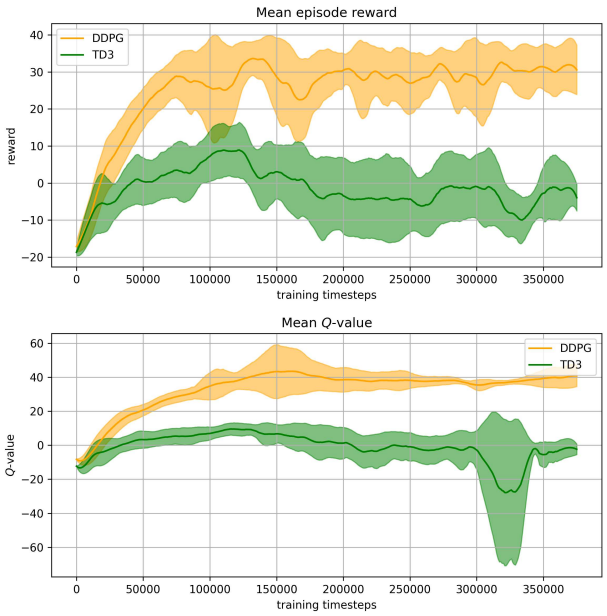

Fig. 4 Training results in simulation: Mean reward and standard deviation of the perceived reward during training (top plot). Mean estimated Q-value and standard deviation returned by the Q-networks (bottom plot)

Unexpectedly, there is a clear difference between how both policies act. The action space domain allows the manipulator to execute a maximum displacement of 0.1 metres in any direction. The minimum amount of interaction with the environment (minimum episode length) required to perform a successful extraction is about 10 timesteps, considering the initial object's position and its physical dimension.

TD3 learned the quickest way to perform the disassembly task as long as it can complete an episode successfully. This behaviour is supported by the mean episode length, as shown in Figure 5. DDPG exhibits a trend to execute more timesteps than TD3 (in its better training's performance interval around 100 thousand timesteps).

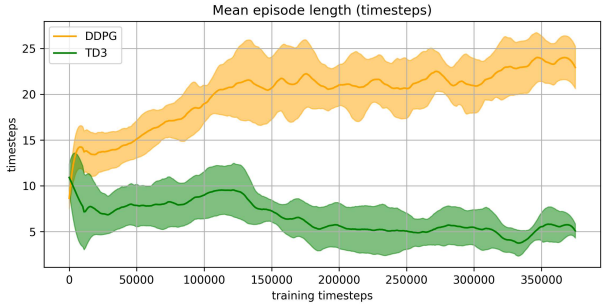

Fig. 5 Mean and standard deviation of the episode length during training.

This behaviour is reflected in the mapping of the actions taken by both policies shown in Figure 6. The graphical point cloud of the DDPG's actions (left chart) is clustered around a median circumference. Because DDPG performs shorter displacements, its extraction speed (understood as the effective length of the episodes) is slower. On the other side, the TD3's actions (right

chart) are majorly mapped around the limits of the action space (extreme actions). This implies a maximum extraction speed but, at the same time, less robustness.

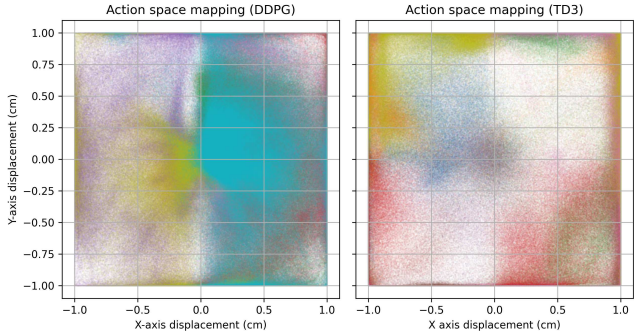

Fig. 6 Mapping of the action $a \in [-1, 1]$ performed during training by DDPG (left) and TD3 (right).

6.2 Testing in simulation

The learned skill's generalisation capabilities were assessed with two tests: 1) different initial object's rotations about its centre and 2) different initial object's locations on the table. The agents ran 30 episodes for each particular case. Those tests were done with the exploration behaviour disabled.

In the first experiments, the target object's rotation is sampled discretely between 0 and 180 degrees, spaced by one degree. Figure 7 shows the mean and the standard deviation of the reward perceived by the best-trained policies and their mean episode length.

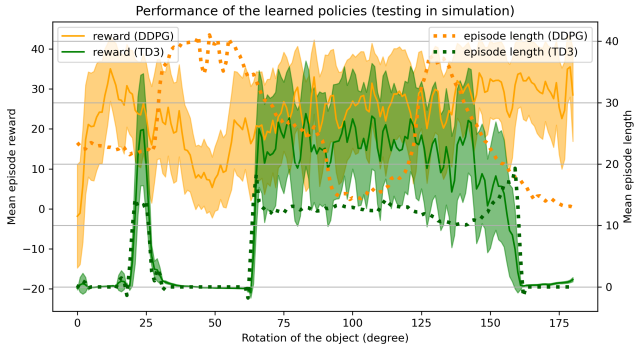

Fig. 7 Performance of the learned policies in the simulated environment (without exploration) for the extraction of the object at different initial rotations. Mean perceived reward and its standard deviation (left axis scale), and mean episode length (right axis scale).

The DDPG policy can execute the task successfully (more than 15 successful episodes per angle) 87.29% of the sampled rotations. The proportion of completed-task for DDPG could reach a higher value if the episode runs more timesteps, than the maximum allowed, for the rotation interval between 35 and 55 degrees. TD3 executed the task successfully (more than 15 successful episodes per angle) 35.35% of the sampled rotations. Consistent with the training process, DDPG has a much better performance than TD3 in the execution of the disassembly task, but TD3 is faster than DDPG.

In the second experiment, the initial position of the target object on the table was sampled to cover a rectangular region of 0.1 x 0.25 metres with a step of 2.5 millimetres while keeping a fixed rotation. The region of interest is centred, in front of the robot, at 0.6 metres. Two fixed rotations (50 and 130 degrees) were selected from the regions where the policies' performance is quite diverse according to Figure 7. The mean reward perceived by both agents at the end of the task is shown in Figure 8.

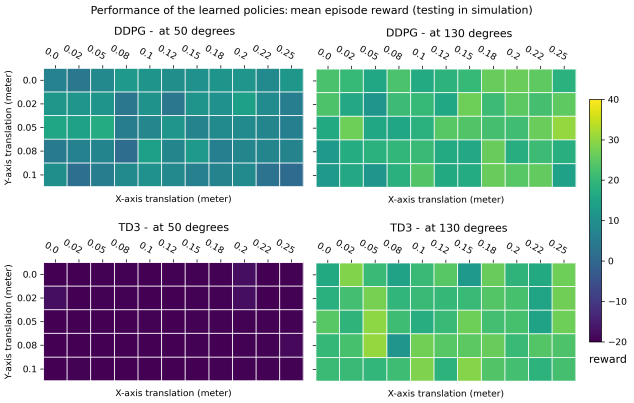

Fig. 8 Performance of the learned policies in the simulated environment (without exploration) for the disassembly task at different initial positions. DDPG (top row) and TD3 (bottom row).

The results show that the learned skill is generalisable for an extensive range of different initial positions. The minor variations are produced by the dynamic and precision of the manipulator's motion during the simulation.

6.3 Transference to the real world

Also, the learned skill's generalisation capabilities were evaluated in the real world. Different initial object's rotations were introduced. In this case, the target object's rotation was sampled discretely between 0 and 180 degrees, spaced by 10 degrees to reduce the number of performed episodes. The agents ran 5 episodes

for each particular case. Figure 9 shows the mean reward perceived by the best-trained policies and their mean episode length.

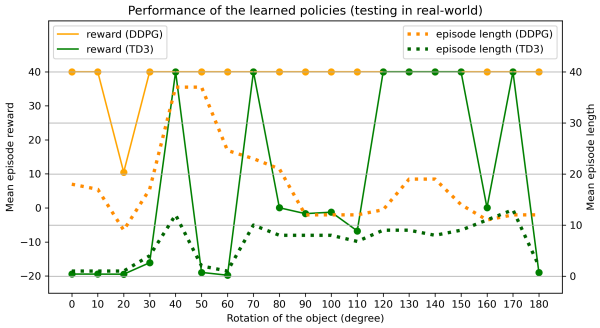

Fig. 9 Performance of the learned policies in the real world (without exploration) for the disassembly task at different initial rotations. Mean perceived reward (left axis scale), and mean episode length (right axis scale).

With a very similar behaviour to the simulation, the DDPG policy can execute the task successfully for at least all the sampled angles. TD3 maintains a poor performance, as seen in the simulation. In this case, the higher success rate on the task's execution in the real world whit respect to the simulation is because the critic force threshold's minor differences used in both environments. The trainings and tests carried out in the simulated environment were performed using a lower critic force threshold than the value used in reality (60 Newtons). Figure 10 shows the force measured during the execution of the episodes in the real world.

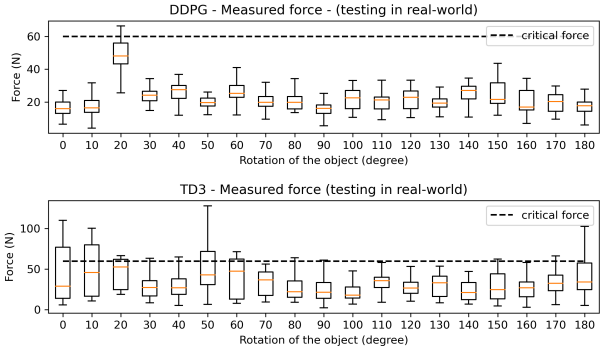

Fig. 10 Force measured during the validations performed in real-world for DDPG (upper plot) and TD3 (bottom plot). The dashed line is the threshold used as the condition for the episode termination.

The simulated environment used a lower critical force value to ensure more safety execution of the task in the real world.

7 Conclusion and future work

Even when reinforcement learning algorithms have been successfully applied to learn robotics skills to perform many manipulation tasks, such as assembly, there is no sufficiently studied on disassembly tasks (which is not necessarily the assembly tasks' reverse process).

Reinforcement learning algorithms can learn the object extraction skill interacting with the environment in an off-policy way. Also, they can generalise the learning through multiples initial conditions such as translations and rotations.

Force/toque and vision sensors are exposed, from their utilisation on assembly tasks, as a relevant information source to identify the environment's state and act according to it. In future works, more sensors will be tested.

Additional Information

Funding: This study was partially financed by European Union's SMART EUREKA programme under grant agreement S0218-chARmER. Also, it is partially financed by H2020-WIDESPREAD project no. 857061 "Networking for Research and Development of Human Interactive and Sensitive Robotics Taking Advantage of Additive Manufacturing – R2P2"

Author contribution: All the related authors make a contribution during the conceptualization, data curation, investigation, methodology, writing–original draft, writing–review and editing of the manuscript.

Availability of data and materials: Data used in this work have been properly cited within the article.

Declarations

Conflict of interest: The authors declare that they have no conflict of interest.

Ethical approval: The authors understand and approve the ethical responsibilities of the authors.

Consent to participate: The authors consent to participate.

Consent to publish: The authors consent to transfer copyright of the article to publish.

References

1. M. Matsumoto, W. Ijomah, in *Handbook of sustainable engineering* (Springer Netherlands, 2013), pp. 389–408
2. J. Fellner, J. Lederer, C. Scharff, D. Laner, et al., Journal of Industrial Ecology **21**(3), 494 (2017)
3. I. D'Adamo, P. Rosa, The International Journal of Advanced Manufacturing Technology **86**(9), 2575 (2016)

4. F.J. Ramírez, J.A. Aledo, J.A. Gamez, D.T. Pham, Computers & Industrial Engineering **142**, 106339 (2020)
5. X. Xia, H. Zhu, Z. Zhang, X. Liu, L. Wang, J. Cao, The International Journal of Advanced Manufacturing Technology **106**(9), 4611 (2020)
6. S. Parsa, M. Saadat, The International Journal of Advanced Manufacturing Technology **104**(5), 1769 (2019)
7. S. Vongbunyong, S. Kara, M. Pagnucco, Assembly Automation (2013)
8. M. Bdiwi, A. Rashid, M. Putz, in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2016), pp. 2500–2505
9. D. Schneider, E. Schömer, N. Wolpert, in *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)* (IEEE, 2015), pp. 35–40
10. J. Kober, J.A. Bagnell, J. Peters, The International Journal of Robotics Research **32**(11), 1238 (2013)
11. D. Ni, Z. Xiao, M.K. Lim, Soft Computing pp. 1–21 (2021)
12. R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, F. De Felice, Sustainability **12**(2), 492 (2020)
13. M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, W. Zaremba, arXiv preprint arXiv:1707.01495 (2017)
14. J. Luo, E. Solowjow, C. Wen, J.A. Ojea, A.M. Agogino, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 2062–2069
15. J. Luo, E. Solowjow, C. Wen, J.A. Ojea, A.M. Agogino, A. Tamar, P. Abbeel, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 3080–3087
16. G. Thomas, M. Chien, A. Tamar, J.A. Ojea, P. Abbeel, in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2018), pp. 3524–3531
17. J. Ding, C. Wang, C. Lu, arXiv preprint arXiv:1912.00260 (2019)
18. Y. Fan, J. Luo, M. Tomizuka, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 811–817
19. F. Li, Q. Jiang, S. Zhang, M. Wei, R. Song, Neurocomputing **345**, 92 (2019)
20. M. Oikawa, K. Kutsuzawa, S. Sakaino, T. Tsuji, arXiv preprint arXiv:2002.12207 (2020)
21. L. Sergey, N. Wagener, P. Abbeel, in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA* (2015), pp. 26–30
22. M.A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, J. Bohg, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 8943–8950
23. T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J.A. Ojea, E. Solowjow, S. Levine, in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 6023–6029
24. G. Schoettler, A. Nair, J. Luo, S. Bahl, J.A. Ojea, E. Solowjow, S. Levine, arXiv preprint arXiv:1906.05841 (2019)
25. C.C. Beltran-Hernandez, D. Petit, I.G. Ramirez-Alpizar, K. Harada, Applied Sciences **10**(19), 6923 (2020)
26. C.C. Beltran-Hernandez, D. Petit, I.G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, K. Harada, IEEE Robotics and Automation Letters **5**(4), 5709 (2020)
27. F. Li, Q. Jiang, W. Quan, R. Song, Y. Li, in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (IEEE, 2019), pp. 13–18
28. J. Luo, H. Li, in *Conference on Robot Learning* (PMLR, 2020), pp. 1191–1200
29. J. Luo, H. Li, arXiv preprint arXiv:2010.08052 (2020)
30. Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, S. Schaal, arXiv preprint arXiv:2011.08458 (2020)
31. C.B. Kristensen, F.A. Sørensen, H.B. Nielsen, M.S. Andersen, S.P. Bendtsen, S. Bøgh, Procedia Manufacturing **38**, 225 (2019)
32. M. Simonič, L. Žlajpah, A. Ude, B. Nemec, in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)* (IEEE, 2019), pp. 230–236
33. R. Herold, Y. Wang, D. Pham, J. Huang, C. Ji, S. Su, in *Industry 4.0–Shaping The Future of The Digital World: Proceedings of the 2nd International Conference on Sustainable Smart Manufacturing (S2M 2019), 9–11 April 2019, Manchester, UK* (CRC Press, 2020), p. 101
34. V.R. Konda, J.N. Tsitsiklis, in *Advances in neural information processing systems* (Citeseer, 2000), pp. 1008–1014
35. T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, arXiv preprint arXiv:1509.02971 (2015)
36. S. Fujimoto, H. Hoof, D. Meger, in *International Conference on Machine Learning* (PMLR, 2018), pp. 1587–1596

Figures

Figure 1

Disassembly of end-of-life refrigerators. a) Stock of refrigerators in the warehouse. b) Manual removing of the door's gasket using a screwdriver. c) Experimental prototype to remove the door's gasket in our lab.

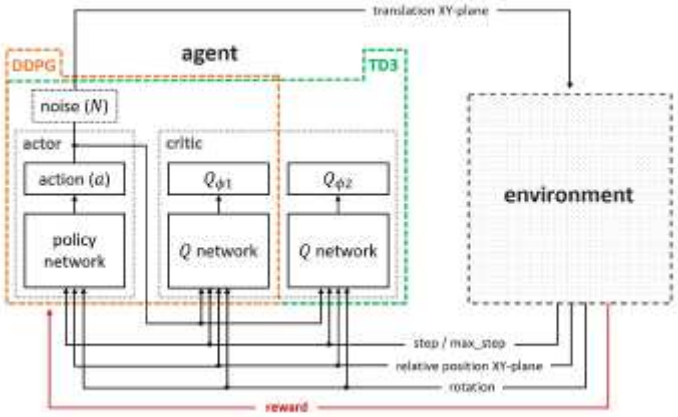

Figure 2

Reinforcement learning environment

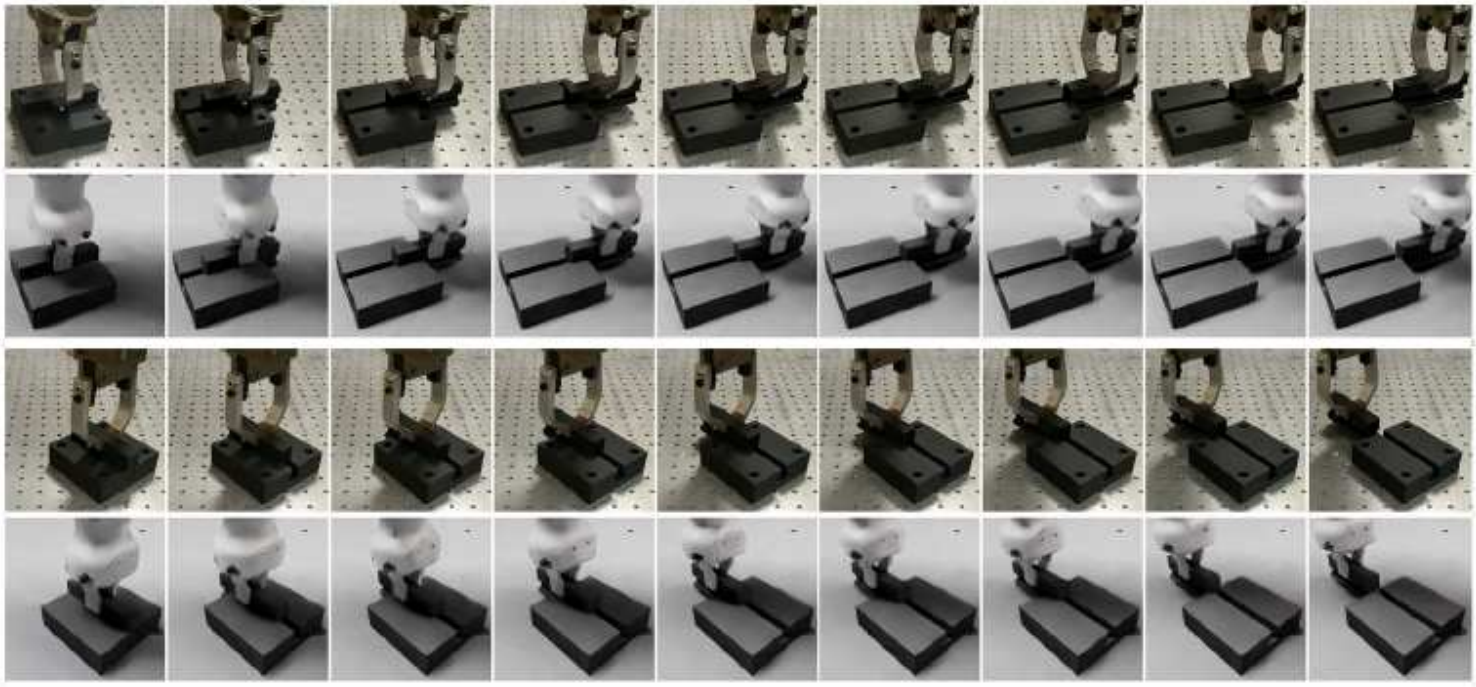

Figure 3

Extraction task's sequence using the best-learned policies for DDPG in real-world and simulation (rst and second rows respectively) and TD3 in real-world and simulation (third and fourth rows respectively). Frames are taken every 3 timesteps for DDPG (object rotated at 70 degrees) and every timestep for TD3 (object rotated at 130 degrees).

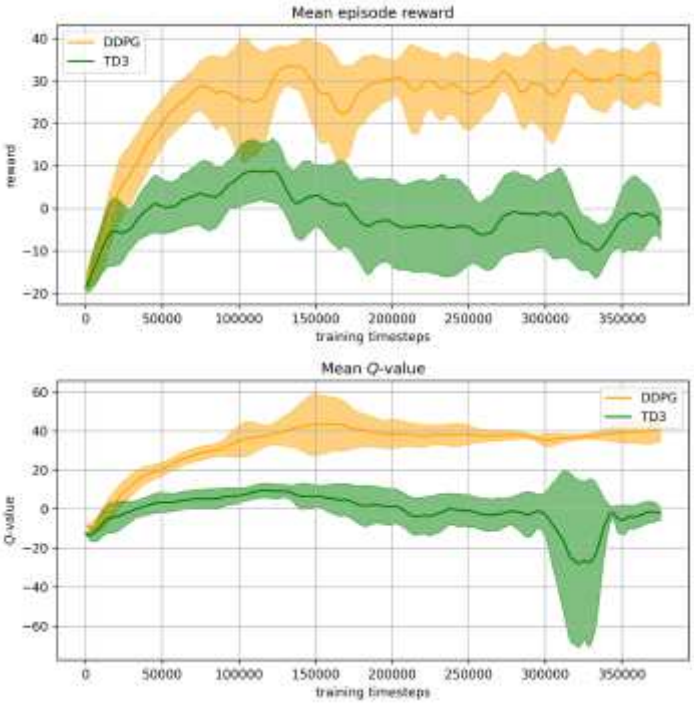

Figure 4

Training results in simulation: Mean reward and standard deviation of the perceived reward during training (top plot). Mean estimated Q-value and standard deviation returned by the Q-networks (bottom plot)

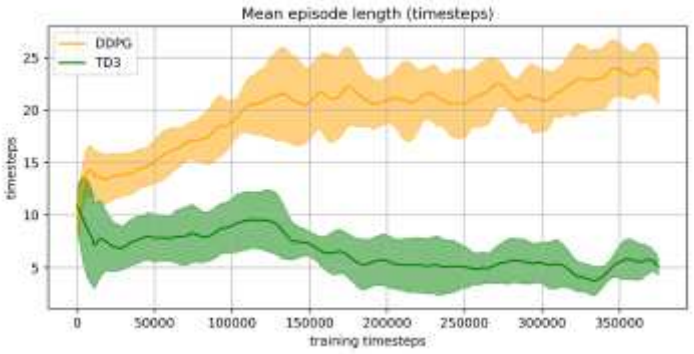

Figure 5

Mean and standard deviation of the episode length during training.

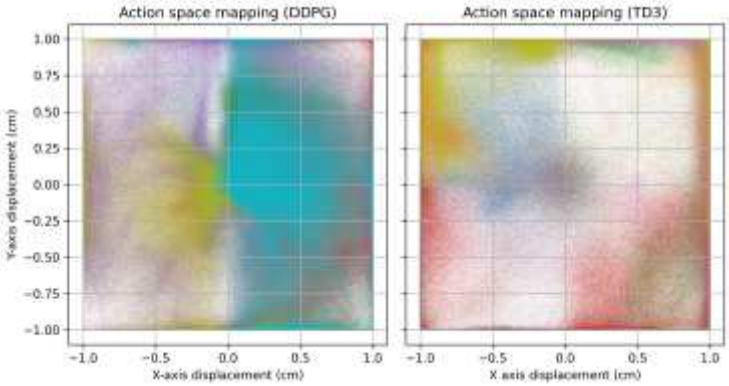

Figure 6

Mapping of the action a E [-1; 1] performed during training by DDPG (left) and TD3 (right).

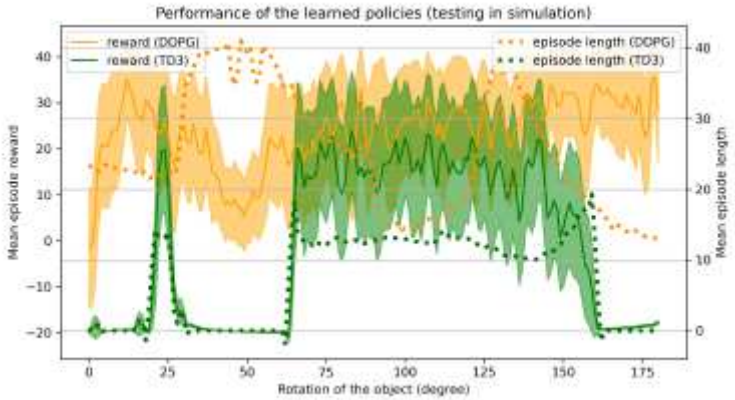

Figure 7

Performance of the learned policies in the simulated environment (without exploration) for the extraction of the object at different initial rotations. Mean perceived reward and its standard deviation (left axis scale), and mean episode length (right axis scale).

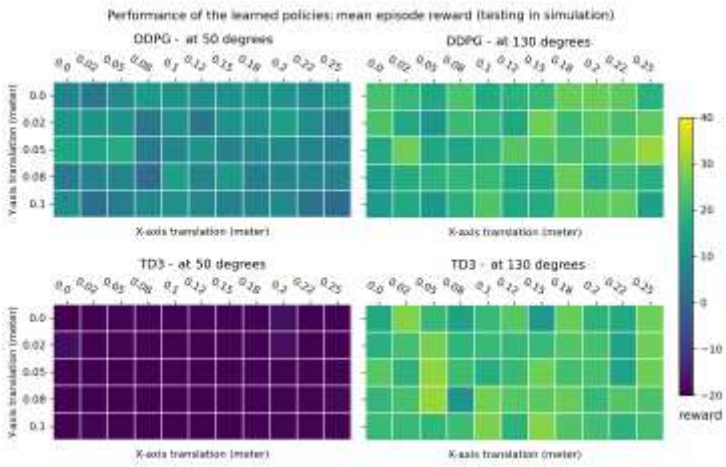

Figure 8

Performance of the learned policies in the simulated environment (without exploration) for the disassembly task at different initial positions. DDPG (top row) and TD3 (bottom row).

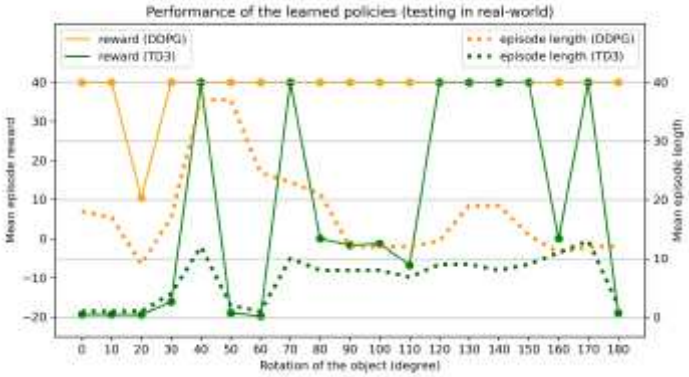

Figure 9

Performance of the learned policies in the real world (without exploration) for the disassembly task at different initial rotations. Mean perceived reward (left axis scale), and mean episode length (right axis scale).

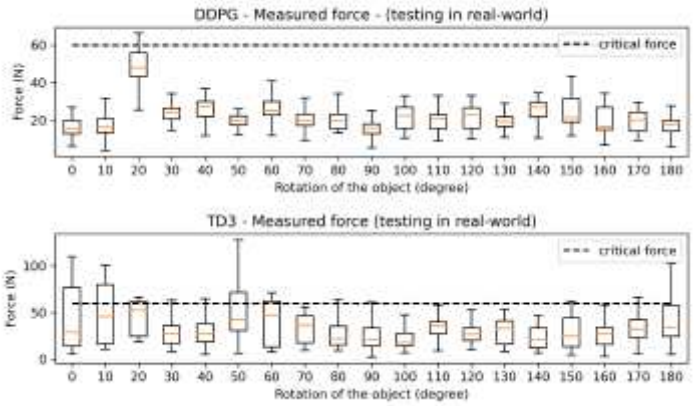

Figure 10

Force measured during the validations performed in real-world for DDPG (upper plot) and TD3 (bottom plot). The dashed line is the threshold used as the condition for the episode termination.