



Article

# Curriculum Design and Sim2Real Transfer for Reinforcement Learning in Robotic Dual-Arm Assembly <sup>†</sup>

Konstantin Wrede <sup>1,\*</sup>, Sebastian Zarnack <sup>1</sup>, Robert Lange <sup>1</sup>, Oliver Donath <sup>1</sup>, Tommy Wohlfahrt <sup>1</sup> and Ute Feldmann <sup>2</sup>

<sup>1</sup> Fraunhofer Institute for Integrated Circuits IIS, Division Engineering of Adaptive Systems EAS, Muenchner Strasse 16, 01187 Dresden, Germany

<sup>2</sup> Institute of Control Theory, TU Dresden, Georg-Schumann-Straße 7a, 01187 Dresden, Germany

\* Correspondence: konstantin.wrede@eas.iis.fraunhofer.de

<sup>†</sup> This paper is an extended version of our paper published in Proceedings of the 4th IFSA Winter Conference on Automation, Robotics and Communications for Industry 4.0/5.0 (ARCI 2024), 7–9 February 2024, Innsbruck, Austria.

**Abstract:** Robotic systems are crucial in modern manufacturing. Complex assembly tasks require the collaboration of multiple robots. Their orchestration is challenging due to tight tolerances and precision requirements. In this work, we set up two Franka Panda robots performing a peg-in-hole insertion task of 1 mm clearance. We structure the control system hierarchically, planning the robots' feedback-based trajectories with a central policy trained with reinforcement learning. These trajectories are executed by a low-level impedance controller on each robot. To enhance training convergence, we use reverse curriculum learning, novel for such a two-armed control task, iteratively structured with a minimum requirements and fine-tuning phase. We incorporate domain randomization, varying initial joint configurations of the task for generalization of the applicability. After training, we test the system in a simulation, discovering the impact of curriculum parameters on the emerging process time and its variance. Finally, we transfer the trained model to the real-world, resulting in a small decrease in task duration. Comparing our approach to classical path planning and control shows a decrease in process time, but higher robustness towards calibration errors.

**Keywords:** reinforcement learning; curriculum learning; simulation-to-reality transfer; dual-arm robots; robotic assembly; peg-in-hole assembly



**Citation:** Wrede, K.; Zarnack, S.; Lange, R.; Donath, O.; Wohlfahrt, T.; Feldmann, U. Curriculum Design and Sim2Real Transfer for Reinforcement Learning in Robotic Dual-Arm Assembly. *Machines* **2024**, *12*, 682. <https://doi.org/10.3390/machines12100682>

Academic Editor: Sergey Y. Yurish

Received: 29 August 2024

Revised: 19 September 2024

Accepted: 20 September 2024

Published: 29 September 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In this paper, we build upon the foundational work presented in our earlier publication “Curriculum-organized Reinforcement Learning for Robotic Dual-Arm Assembly” at the Automation, Robotics & Communications for Industry 4.0/5.0 (ARCI' 2024): 4th IFSA Winter Conference [1]. We aim to expand the original study's scope by performing a simulation-to-reality (Sim2Real) transfer for the proposed dual-arm robotic assembly task. This extension not only broadens the theoretical framework but also incorporates additional experimental results, enhancing the relevance and applicability of our findings.

In modern manufacturing, assembly consumes about 50% of total production time and accounts for 30% of costs [2]. Especially insertion tasks—critical in the final integration of assemblies in consumer electronics like mobile phones—account for 60% of manual human labor time [3]. Assembly tasks typically involve multiple sequential insertion operations, making the use of dual-arm robots advantageous as they can eliminate the need for reorientation and reclamping of workpieces, thus reducing overall process time. However, this solution adds layers of complexity to the control problem. Traditional robotic systems for such assembly are hindered by high costs associated with image processing systems and force/torque measurement sensors, as well as the involvement of human

programming. They are also limited regarding the adaptability to varying workpiece geometries, initial inaccuracies, and tight clearances.

In response, this work introduces a strategy utilizing reinforcement learning (RL) to train control algorithms for such tasks. Our approach not only aims at predicting the average process time but also models the variability and resulting distribution introduced by uncertainties in simulation.

We set up two Franka Panda robots to collaboratively perform a square peg-in-hole task. Training control systems in reality can be slow due to its confinement to real-time operations, costly because it necessitates multiple hardware units for parallelization, and risky due to potential damage during exploration phases. Consequently, simulations have become an essential element in the control design stage. We use the Python library *robosuite* [4], which interfaces with the *MuJoCo* physics engine [5]. Our control architecture adopts a hierarchical approach, with a low-level impedance control receiving desired trajectories from a central policy trained with RL. The RL training methods are implemented via *Stable Baselines3* (SB3) [6] and utilize a sparse reward function proposed by *robosuite*. The training process is designed as a reverse curriculum: we begin the training with the insertion task nearly completed. As the policy's performance improves, we progressively enhance the difficulty of the task until it reaches a configuration that is realistic for practical applications in terms of the initial peg-hole distance. To ensure the policy's adaptability to varying initial robot configurations, we incorporate domain randomization (DR).

Nevertheless, a significant challenge persists, known as the simulation–reality gap, which complicates the transfer of control systems from simulation to real-world applications [7]. This gap often results from discrepancies such as model mismatches (e.g., inaccurate dynamic parameters) and disturbances affecting the inputs and outputs in the actual system. Additionally, RL faces difficulties in terms of interpreting the trained policy's actions and adapting to changing system conditions without considerable retraining. These issues underscore the importance of exploring the robustness of Sim2Real transfer. We assess the efficacy of this policy in simulation compared to traditional control methods and compare its performance in real-world to the simulation environment.

In this study, we address the research question of how to train a dual-arm robotic assembly policy in simulation and transfer it to a real-world setup. How robust performs the resulting control system to varying tasks in reality, especially compared to a classical method? A more detailed insight into the particular task as problem formulation is provided by Section 3. To approach these research questions, we have structured our investigation into the following points:

- We establish a simulation and training environment for dual-arm robotic assembly using *robosuite*, coupled with the *MuJoCo* physics engine.
- We integrate the simulation with SB3 RL methods, employing the OpenAI Gym standard [8].
- We design a training methodology based on a reverse curriculum, enhancing both convergence rates and sample efficiency.
- We investigate how variations in curriculum parameters influence the process characteristics observed in simulations.
- We transfer the trained policy to a real-world setup, comparing its performance to simulation and traditional control strategies.

The remainder of this paper is organized as follows. The following Section 2 introduces related work in the fields of simulation environments, robotic assembly control strategies, and curriculum learning as well as Sim2Real transfer. We use Section 3 to concisely specify the task of this study. In Section 4, we detail the methodological foundation of our research, describing the interplay between low-level controllers and the policy, the reverse curriculum design for training and the eventual Sim2Real transfer. Section 5 first examines the setup and outcomes of experiments in simulation. It explores the transfer of the trained policy to a real-world setting and compares results to those achieved using traditional control

strategies. Finally, Section 6 discusses implications of these results, before Section 7 sums up our work, highlights limitations, and derives potential directions for future research.

## 2. Related Work

In this section, we first present previous work on relevant methods and techniques used in our study, before we provide more details on their implementations in the context of our specific assembly task in Section 4.

### 2.1. Reinforcement Learning

Reinforcement learning (RL) is a branch of Machine Learning that seeks to mimic the learning processes observed in both humans and animals. Unlike traditional Machine Learning approaches that rely on predefined datasets, RL involves the dynamic generation of data through interaction within a specific environment. In this framework, an agent engages with the environment by performing actions, guided by a reward system that informs the suitability of its decisions based on the currently observed system state. This continuous interaction facilitates the development of a policy, which decides on the agent's responses to different environmental conditions. Such policies may be deterministic, prescribing explicit actions, or stochastic, suggesting actions with certain probabilities. The overarching goal of the agent is to optimize its reward accumulation, thereby progressively refining its decision-making process by training and adapting to novel states within the environment. Consequently, the essence of an RL system is encapsulated in the interaction between the agent and its environment [9].

### 2.2. Robust Sim2Real Transfer

Transferring policies trained in simulation environments with RL to reality is a necessary step towards actual application of learning-based robotic systems. This process, known as Sim2Real transfer, faces the challenge of being robust to deviations, uncertainties, and unforeseen situations, known as Sim2Real gap, between the two domains. We address this challenge at different stages from decisions on the control system design until deployment.

First, such robustness by architecture can be achieved through a hierarchical control model. This structure integrates classical controllers for low-level hardware actuation with a high-level policy framework. The high-level policy is designed to process all measured values as observations, subsequently generating setpoints for the low-level control. By this architecture the action space is adapted for an accelerated exploration during training and improved safety at deployment. Furthermore, the robustness of the Sim2Real transfer with respect to model mismatch is increased due to decoupling of the policy and system dynamics [10]. In this paper, leveraging experience from past research, as outlined in Section 2.4, we specifically select a suitable low-level controller, as explained in Section 4.1.

Furthermore, it is possible to improve the robustness at the training stage in simulation by applying DR. This technique introduces variations in the simulation settings, such as initial conditions, to more accurately reflect the distribution of real-world data, bridging the gap created by biases between simulated and actual environments. Additionally, DR allows for the modeling of time-varying behaviors that result from system drifts or degradation [11]. This is achieved through dynamics randomization by altering the system's dynamic parameters. To prevent a distribution shift between simulation and reality, it is crucial to consider the diversity of randomization within a dataset—not merely the quantity of samples [12]. In our work, we apply DR described in Section 4.4 matching to a task-specific curriculum with its general concept explained in Section 2.5 and task-specific implementation as a reverse curriculum in Section 4.3.

During the deployment stage on real hardware, the robustness of Sim2Real transfer can be further enhanced by employing methods such as policy-level action integration (PLAI). While dynamics randomization is a common strategy to address model inaccuracies and improve robustness, it often requires extensive additional training and may compromise

precision eventually. Drawing inspiration from PID control, PLAI modifies incremental setpoints by integrating policy actions based on the previous desired state rather than the current state. This adaptation significantly improves the control system's ability to reject disturbances and minimize steady-state errors, thereby enhancing overall performance [13]. Our particular utilization of PLAI is detailed in Section 4.5.

### 2.3. Simulation Environment

For robotics simulations, the gym interface often serves as the cornerstone for connecting with RL toolboxes. A notable example is PyBullet [14], a lightweight yet robust simulation engine, which has been successfully integrated into various RL frameworks, including PyTorch [15]. This integration is used in several studies [16–18]. Additionally, NVIDIA's Isaac Sim has gained traction as a robotics simulator within the RL community, distinguished by its comprehensive and so far most advanced GPU support [19]. It commonly interfaces with PyTorch-based libraries such as skrl [20]. Furthermore, past research [21–23] has utilized MuJoCo within the robosuite framework for its efficient low-level control implementations, a lean framework compared to Isaac Sim. Our work leverages this particular setup, capitalizing on its streamlined integration and benefits in development.

### 2.4. Control of Robotic Assembly

For single-arm peg-in-hole assembly tasks, compliant control systems such as impedance control have been extensively applied, demonstrating their effectiveness across various studies [24–26]. This conventional approach has been successfully extended to dual-arm setups, with notable implementations detailed by [27]. In these configurations, while one robot actively manipulates the peg, the other typically plays a passive role by just holding the hole part with high compliance. However, engaging both robots actively could significantly enhance process time and reliability. This potential improvement has spurred interest in applying RL techniques. RL has been effectively used to generate reference trajectories for established low-level impedance controllers, as shown in recent research [16].

### 2.5. Curriculum Learning

Curriculum learning is a technique applicable to goal-oriented tasks that do not require prior domain knowledge, depending exclusively on achieving a state indicative of task completion. The method entails a progressive learning process where the robot starts from various initial states, each incrementally increasing in difficulty (easy2hard), often measured by the spatial distance from the goal. Previous research, such as [28], has explored the automatic generation of such learning curricula. In contrast, our approach involves manually defining the learning steps, tailored to our specific needs.

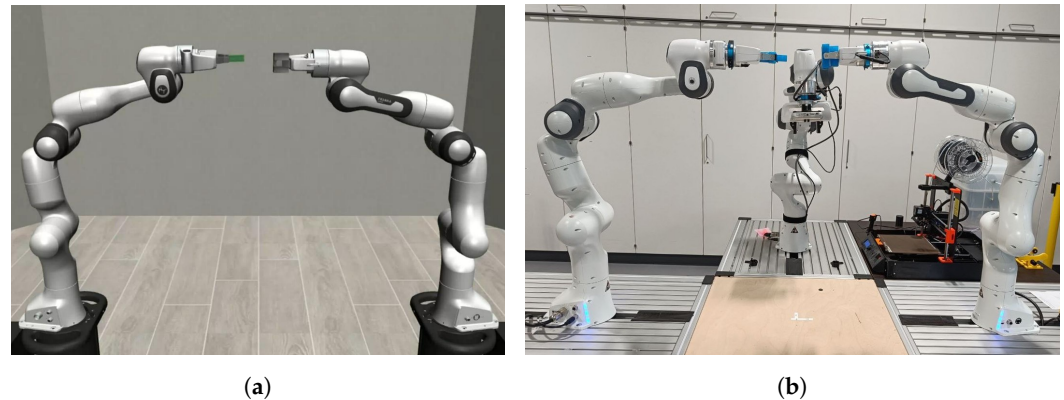
A recent advancement in this field is the introduction of a sampling-based curriculum [13], which involves gradually elevating the lower bound of a task difficulty. The initial system configurations are selected from a distribution determined by this bound. In scenarios involving dual robots, like in our study, setting these difficulty bounds is more complex. The condition for activating DR in the curriculum is crucial, as it is chosen based on the insertion state to prevent the system from initializing into a potential jamming situation. To our knowledge, this work is the first implementation of curriculum-organized RL specifically applied to a dual-arm robotic assembly task. Moreover, the two-phase structure of our curriculum introduces a novel approach to this field.

## 3. Problem Formulation

Our aim is to develop a control policy in simulation and successfully transfer it to a real-world setup for tactile assembly applications. Specifically we focus on a peg-in-hole problem with squared objects, implying the relative orientation of the parts to be crucial. Figure 1 illustrates our experimental setup in both simulation and reality, featuring squared



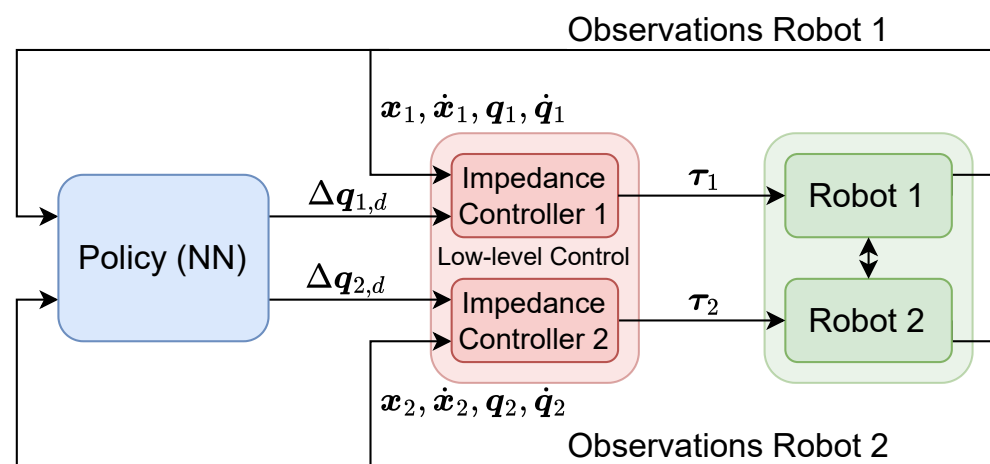
objects with a 1 mm clearance, and both Frankas' bases positioned 1.4 m apart. The parts used in the experiments were manufactured from PLA material using a 3D printer, with a printing resolution of 0.1 mm—ten times more precise than the clearance. The gripping process of both parts is out of scope of this work and assumed to be performed in advance.



**Figure 1.** Environment consisting of two robot arms, peg and hole workpieces. (a) Setup in simulation. (b) Setup in reality.

#### 4. Methodology

We propose a hierarchically structured control system consisting of two decentral low-level impedance controllers, one per robot, which directly deliver torque commands to the joint motors. A central policy in form of a neural network (NN), trained with RL in a curriculum-based framework, generates setpoint values for these impedance controllers feedbacking current measurements as observations. This architecture is illustrated in Figure 2. To enhance the robustness of the policy and facilitate a successful Sim2Real transfer, DR is applied during specific training sections.



**Figure 2.** Hierarchical control system architecture with a central policy in form of a NN generating setpoints for the decentral low-level impedance controllers.

##### 4.1. Low-Level Control

To optimize the action space for the policy network, on each robot, impedance control is implemented, as outlined by [29]. The recommendation to adapt the action space through Cartesian impedance control facilitates rapid policy exploration in simulation. Furthermore, for deploying policies in the real world, the use of joint velocity impedance control is advisable to enhance the efficacy of the Sim2Real transfer. However, to simplify implementation efforts and reduce the need for coordinate transformations, we opted for joint position impedance control. Reducing the amount of Franka Panda specific terms and retaining a mainly robot-agnostic implementation, we omit the moment of inertia

matrix, the Coriolis, and centripetal matrix. This leaves a PD-like controller with gravity compensation similar to the joint space position controller with fixed impedance provided by robosuite. To emphasize the compliant behavior of the robot controllers, necessary for small-tolerance insertions, we still refer to this as impedance control. Also, the controller's setpoints are expressed as delta positions. The control law is expressed as:

$$\begin{aligned}\tau &= K_p(q_d - q) - K_d\dot{q} + g(q), \\ q_d &= q + \Delta q_d.\end{aligned}\quad (1)$$

where  $\tau$  denotes the joint motor torques, the variable  $\Delta q_d$  refers to the desired delta joint positions, and  $q$  and  $\dot{q}$  denote the measured joint positions and velocities, respectively. Adding the configuration dependent gravity compensation  $g(q)$  to the control law will be accomplished by the Franka control implementation in the real-world setup internally. The matrix  $K_p$  represents the stiffness and  $K_d$  the damping as control gains, which are maintained at standard values of the robosuite implementation:

$$K_p = 50 \text{ Nm} \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}, K_d = 2\sqrt{50} \text{ Nms} \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}, K_p, K_d \in \mathbb{R}^{7 \times 7}. \quad (3)$$

The relationship between these gains is defined by  $K_{d,i} = 2\sqrt{K_{p,i}}$ , ensuring aperiodical damping and stability.

#### 4.2. Reinforcement Learning

We formulate the problem as a Markov decision process, explicitly defining the observation space, action space, and reward function. We utilize a state-of-the-art model-free, off-policy RL algorithm known as Soft Actor-Critic (SAC) [30], employing the implementation provided by SB3. To address the computational demands of the CPU-based physics simulation in MuJoCo and enhance sample efficiency, we integrate a replay buffer using Hindsight Experience Replay (HER) [31]. Table 1 presents relevant hyperparameters involved in the RL training as well as network parameters of the policy.

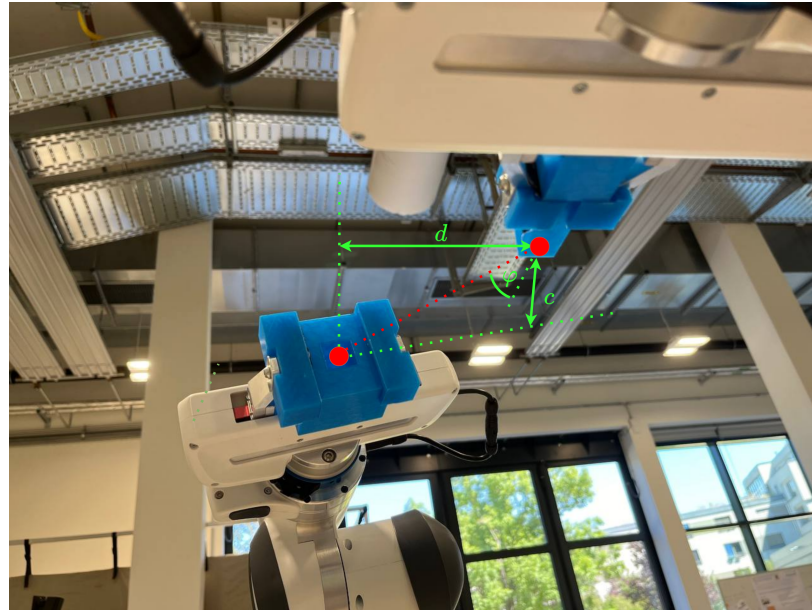
**Table 1.** Relevant training and network parameters of the actor-critic policy trained with SAC.

Parameter	Value
Optimizer	Adam
Learning Rate	$3 \times 10^{-4}$
Discount Factor	0.99
Entropy Coefficient	Auto (init 1.0)
Batch Size	256
Buffer Size	$1 \times 10^6$
Replay Buffer Size	6
Architecture Actor	$1 \times 256$ neurons as hidden layer
Architecture Critic	$2 \times 256$ neurons as hidden layers
Activation Function	ReLU with bias

The observation space  $\mathcal{O} \ni \mathbf{o}$  for our RL setup is based on the default configuration provided by robosuite for the *TwoArmPegInHole* environment:

- Positions and quaternions of the peg and the hole;
- A difference vector and angle  $\varphi$  between the peg and the hole;
- Parallel distance  $c$  and perpendicular projected distance  $d$  between the peg and the hole;
- The state of each robot, which encompasses the sine and cosine of the joint angles  $q$ , joint velocities  $\dot{q}$ , as well as positions and quaternions of the end-effectors (EE).

It is important to note that this observation space does not include any force or torque measurements from additional sensors. The geometric quantities described are illustrated in Figure 3.



**Figure 3.** Quantities describing the geometric relationship between the peg and the hole.

The action space is defined by the choice of joint position impedance control, specifically targeting the desired deltas of joint coordinates for each robot. The range of these actions is symmetrically scaled by the parameters *output\_min* and *output\_max*, with the maximum delta values set at:

$$\Delta q_{d,max} = -\Delta q_{d,min} = (0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05)^T \text{ rad.} \quad (4)$$

Limiting the actions in form of delta joint positions given a fixed control period results in a rate limit of the joint velocities. Furthermore, this specification determines the maximum achievable speed for task completion during training and if not scaled during deployment. The numerical limits were empirically established through simulation trials and are subject to further refinement based on the absolute maximum ratings of robot components, work pieces, and the specific requirements of the assembly task.

The reward function utilizes the default sparse reward setup of the *TwoArmPegInHole* environment, structured as follows:

$$r = \begin{cases} 1, & d < d_{max}; |c| < c_{max}; \cos \varphi > \cos \varphi_{max} \\ 0, & \text{else.} \end{cases} \quad (5)$$

where  $d_{max} = 1$  mm is determined by the part clearance,  $c_{max} = 2$  mm, and  $\cos \varphi_{max} = 0.95$  rad is set based on intuitive experimentation. This goal-oriented and sparse reward structure is particularly effective for facilitating the application of HER.

#### 4.3. Reverse Curriculum Generation

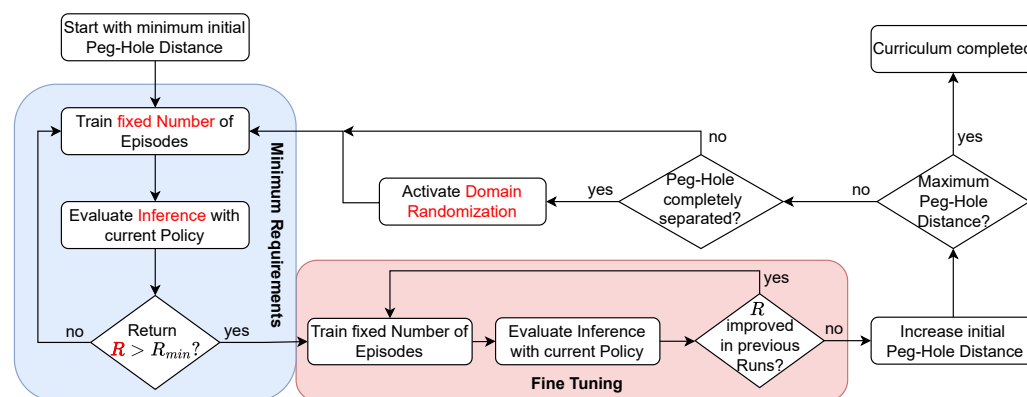
A reverse curriculum arranges a sequence of related tasks in reverse order, beginning with an initialization close to the goal state and progressing towards more complex initial conditions. This approach leverages the reversible nature of most assembly tasks, aiming to optimize learning efficiency in environments characterized by sparse rewards concentrated in a limited portion of the state space. Under such a curriculum, a policy trained with RL, exploring with randomized actions, is able to quickly accumulate rewards and expedite its learning progression. The policy applies the capabilities acquired at simpler stages to

increasingly difficult scenarios. As its competence grows, the initial peg–hole distance or task difficulty is systematically increased to maintain alignment with the policy’s improving performance. Specifically, in goal-oriented tasks, the initial system state  $s_0$  can be adjusted to more challenging configurations based on the cumulative reward (return)  $R(s_0)$  gathered during inference, thus tailoring the difficulty to the policy’s evolving capabilities.

In this study, we structure the curriculum into two distinct phases within each iteration:

1. **Minimum Requirements:** After a predetermined number of training episodes, an inference test determines whether the policy achieves a return  $R$  that surpasses a set threshold  $R_{min}$ . If successful, the curriculum progresses to the fine-tuning phase. Else, this phase is repeated.
2. **Fine Tuning:** Subsequent to additional training over a fixed number of episodes, another inference is evaluated. If there is a relative improvement  $R$  across a specified number of inferences, the fine-tuning phase is extended. If improvement stalls, the difficulty of the initial state is increased, advancing the curriculum to the next iteration, which starts again at the Minimum Requirements phase.

The curriculum is completed successfully by finishing the Fine Tuning step with an initial configuration equivalent to the original task formulation. Figure 4 illustrates this curriculum.



**Figure 4.** Iterative training procedure of the dual-robot peg-in-hole task as reverse curriculum. Quantities represented by positions marked red will change if DR is activated.

To specify the initial system state, we manually configured the joint angles of both Franka robots as:

$$q_{init} = (0, 0.20, 0, -1.20, 0, 2.90, 0.79)^T \text{ rad.} \quad (6)$$

This setup places the peg tip with  $c = 80$  mm away from the hole, defining the baseline for the original task. The curriculum begins with the peg nearly fully inserted into the hole by 24 mm. Initially, during the deterministic phase of the curriculum, which spans 40 iterations and excludes DR, the peg is gradually extracted by 1 mm per iteration at initialization. Following this, starting from a peg–hole distance of 5 mm, DR is applied as white noise to all initial joint angles over the next 15 iterations. During this phase, the peg–hole distance is increased by an average of 3 mm per iteration until the system reaches to the original configuration.

The threshold parameter is set to  $R_{min} = 50$  mm, which requires the insertion to be completed for at least a third of the episode, consisting of 150 steps. Choosing the parameter this way strikes a balance between training efficiency and the reliable continuation of the insertion task. In the Fine Tuning phase, the curriculum is designed to allow for three inference evaluations before the reward is considered to have stagnated, indicating proceeding into the next iteration. Each phase of the curriculum involves 100 training episodes in one step. However, an exception is made during the Minimum Requirements phase following the activation of DR. Due to the increased stochastic nature of the environment with DR, the first phase involves not just one but 30 inference evaluations, with returns averaged to

account for variability. To ensure an appropriate ratio of training and inference episodes, the number of training episodes per step is increased to 300 during this as long as DR is applied.

#### 4.4. Domain Randomization

In this work, DR plays a critical role in enhancing the policy's ability to adapt to a broad range of initial system configurations. We implement DR by introducing normally distributed white noise with a standard deviation of  $\sigma_q = 1.15^\circ$  to all joint angles during initialization. However, to prevent the emergence of undesirable states—such as the peg and hole starting in an condition of both objects penetrating each other—DR is deactivated at first. DR is applied when the initial state involves a complete separation between the peg and the hole. We do not use DR in form of dynamic parameters or sensor measurements, but similar to [13], we choose to implement PLAI instead.

#### 4.5. Deployment on Real-World Hardware

To successfully transfer the trained policy from a simulated environment to the real-world setup, we developed a toolchain that embeds the NN into a control system structured according to Section 4.1. The toolchain for Sim2Real transfer includes the following steps and components:

- **Training the Policy:** The first step involves training the policy within the introduced simulated environment. After finishing the training, the policy is exported as a PyTorch file. This format is chosen for its flexibility and compatibility with various deployment environments.
- **Low-Level Control System:** The low-level joint space impedance controllers are implemented in C++ to interface with the libfranka library on each robot. This ensures real-time control of the robot's joint motors with a frequency of 1 kHz.
- **Connecting both Robots:** Both robots are connected to a central computer that runs the Robot Operating System 2 (ROS 2) [32]. ROS 2 facilitates communication between the robot's specific low-level controllers and the central policy, enabling coordinated actions and reliable low latency data exchange. The motion planning framework *MoveIt 2* [33] is used in tandem with *ros2\_control* and RViz to provide an interface for managing and visualizing the robot state as well as an option for classical trajectory planning and execution with interchangeable low-level controllers on the robot hardware. The policy is embedded within ROS 2, providing setpoints for the low-level controllers. This integration allows the high-level policy to guide the robot's actions based on sensory inputs while still meeting real-time constraints. To further streamline the process of integrating a policy into the real-world control system, we developed several ROS 2-based tools. This control utils package provides a simplified python interface for trajectory planning and interfacing MoveIt 2 by wrapping its API utilizing *pymoveit2*.
- **Data Recording:** To evaluate the performance of the policy, we set up a recording and logging mechanisms to capture the joint efforts, joint velocities, and EE forces of every robot arm as well as the total time needed for every episode.

To enhance the robustness of the policy during real-world execution, we apply PLAI to minimize steady-state errors. In simulation, incremental setpoints are based on the current measured joint positions, as stated in Equation (2). In contrast, in the real world setup, we derive these desired joint positions from the previous desired values (last setpoints) and the output of the policy  $\Delta q_d(t) = \pi(o(t))$ :

$$q_d(t) = q_d(t-1) + \pi(o(t)). \quad (7)$$

This approach enhances the stability and robustness of the policy during real-world execution by reducing the dependency on immediate sensor feedback and instead relying on the consistent application of the policy's outputs over time. The result is a more resilient



control strategy that better accommodates the uncertainties and variances inherent in real-world robotic systems (e.g., sensor noise).

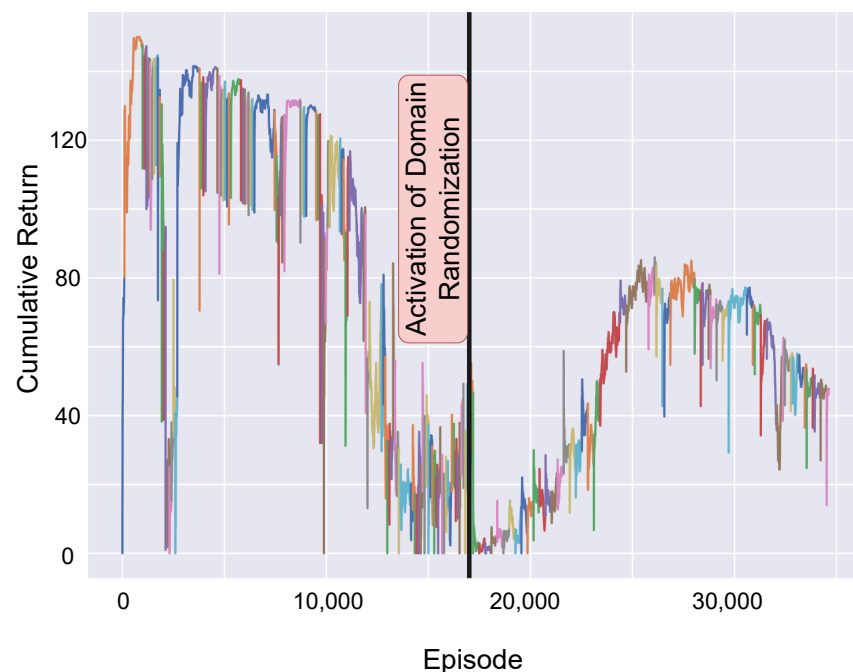
## 5. Results

### 5.1. Training in Simulation

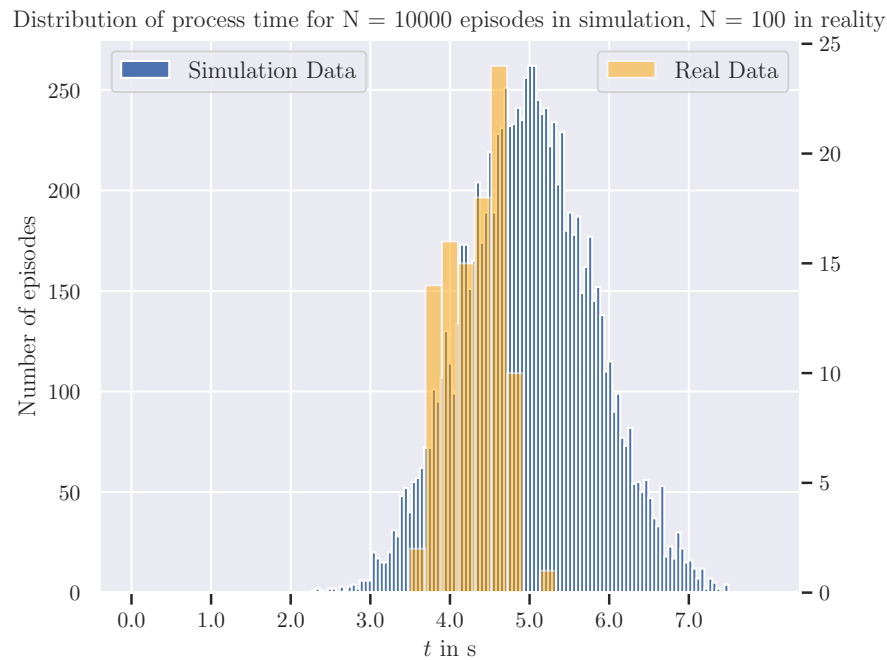
The primary objectives of the simulation is to train a robust policy for a subsequent Sim2Real transfer and investigate the impact of curriculum parameters on the characteristics and performance of the trained control system. By modifying the distance between both robots and object geometries of robosuite's *TwoArmPegInHole* environment, we establish a simulation consistent with the task description outlined in Section 3.

The moving average of the return  $R$  sampled during training is illustrated in Figure 5. The curriculum spans approximately 35,000 episodes to complete. Notably, the reward curve exhibits distinct discontinuities, corresponding to adaptations in the initial joint configuration as the curriculum progresses by iterations. The deterministic learning phase concludes around 17,000 episodes, marked by a significant increase in difficulty due to the activation of DR. This change leads to a slower ascent in the learning curve. With noise introduced at initialization, changes to the joint configuration caused by curriculum progression begin to exert less influence on the task difficulty. After 28,000 episodes, the reward curve plateaus and eventually begins to decline, reflecting the challenges posed by increased initial distances and the speed limitations imposed by actuation constraints. The specified maximum episode length is just sufficient to achieve the threshold  $R_{min} = 50$  mm.

Following the training phase, the control policy was evaluated across 10,000 inference episodes using DR at the original initial task configuration. The histogram shown in Figure 6 illustrates the distribution of the process cycle time derived from the simulation step, which posed the first positive sparse reward. On average, the task is accomplished within 4.99 s with a standard deviation of 0.80 s. Notably, the insertion task succeeds in 99.8% of all inference episodes in less than 7.5 s, demonstrating a high reliability and effectiveness of the trained control system in simulation.



**Figure 5.** Return per training episode. Each color highlights a new iteration of the curriculum.



**Figure 6.** Comparison of process time for the proposed RL-based control system in simulation and reality.

### 5.2. Transfer to Reality and Evaluation

To evaluate the effectiveness of the trained control policy beyond simulation, we deployed it on the physical robots leveraging the methods explained in Section 4.5. During 100 inference episodes initiated from the original joint configuration stated in Equation (6), the policy achieved a success rate of 100%. Figure 6 compares the process times for both the simulated and real-world setup. In reality, the average process time was measured as 4.31 s, with a standard deviation of 0.34 s, both smaller than in simulation. This could be caused by overestimating uncertainties in the initial configuration due to measurement noise and limited encoder resolution by a high amount of DR in simulation, while having a very narrow distribution of actual initial states close to the nominal original setup in reality. Additionally, 48% of all real-world inferences completed the insertion within this average time. The high success rate of 100% in reality proves the robustness of the trained policy, confirming its effectiveness in real-world settings for the original task configuration.

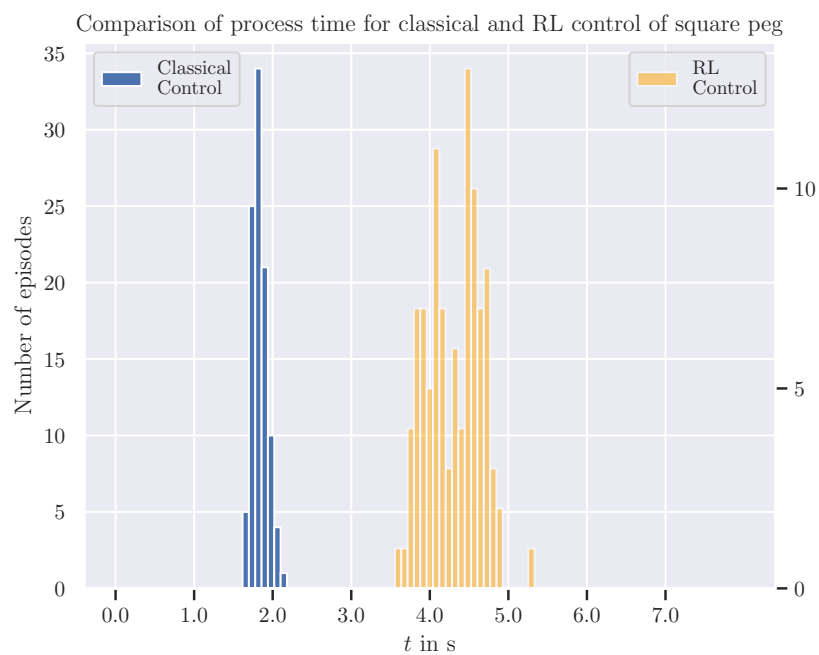
Additionally, we compared a classical trajectory planning approach using MoveIt 2 to our RL-trained policy. The classical control method, which employs low-level PD joint controllers, used a pre-defined trajectory with parameters noted in Table 2.

**Table 2.** Planning parameters for a classical joint trajectory controller.

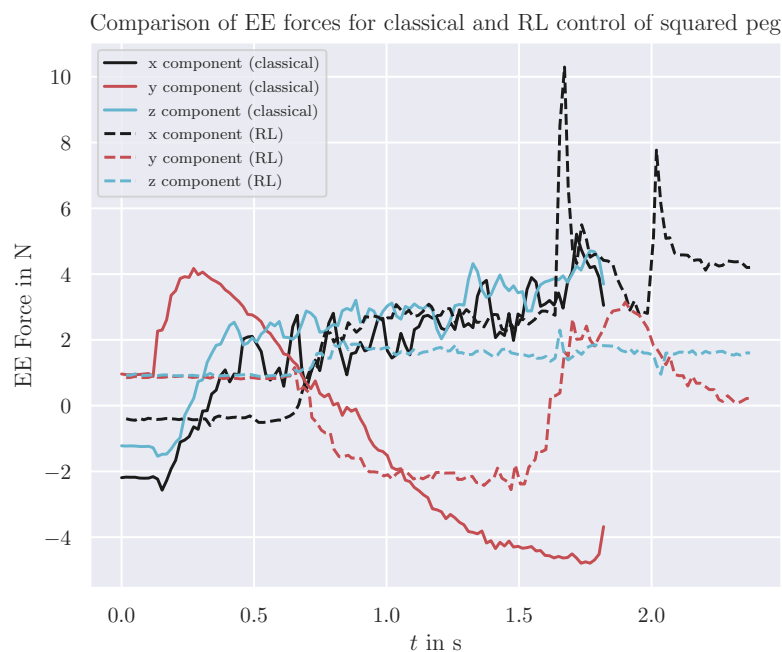
Parameter	Value
Planning Time	1 s
Planner	RRTConnect
Max. Velocity	20% of max. values
Max. Acceleration	20% of max. values
Cartesian planning	True

It also achieved a success rate of 100%. Moreover, it significantly outperformed the RL-based system in terms of insertion speed: the average process time for classical control was only 1.84 s, with a standard deviation of 0.10 s. The histogram in Figure 7 depicts the distribution of process times for both control strategies, clearly showing that the classical control completes the task in less than half the time required by the RL-based control on average.

At the deployment stage, the libfranka library calculates EE forces from measured joint torques. While not highly precise, these calculations are adequate for rough estimations of the applied forces. A comparison of the 3D EE forces exerted by the robot holding the peg during one representative insertion attempt are presented in Figure 8. They reveal the classical control approach to apply significantly less contact force, particularly in the pushing direction  $x$ . This method is carefully calibrated to insert the peg without contacting the surface of the hole, thus avoiding jamming. In contrast, the RL control exerts more than double the maximum force—up to 10 N—compared to approximately 5 N for classical control. This indicates that RL control maintains the ability to manage insertion tasks even when not precisely calibrated, making it more adaptable to variations in the environment and minor changes in the setup.



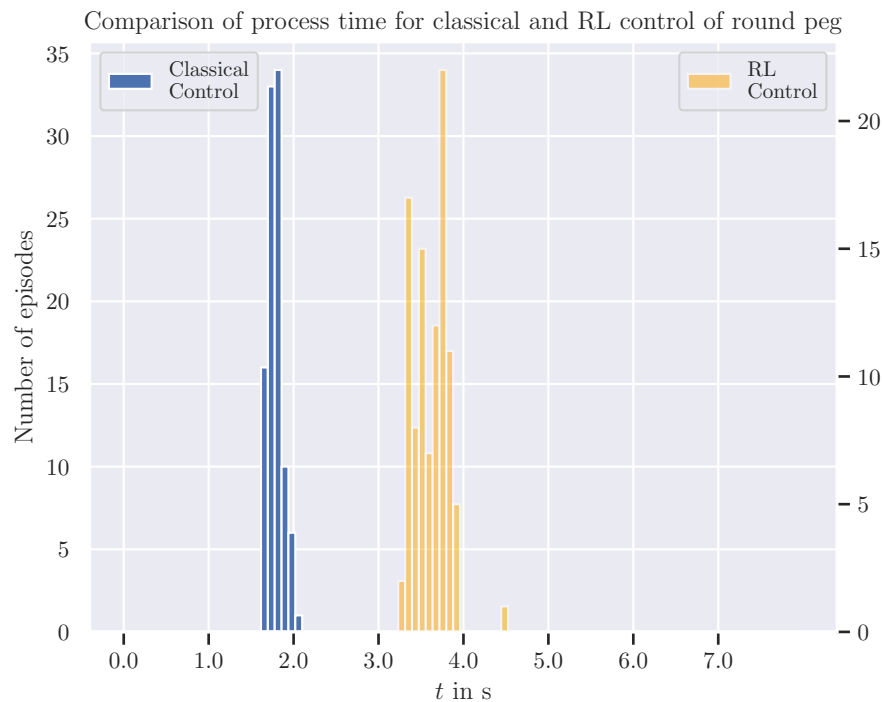
**Figure 7.** Comparison of process time for the classical and the proposed RL-based control in reality.



**Figure 8.** Comparison of EE forces during insertion of a squared peg for the classical and RL control strategy.

### 5.3. Adapted Peg Shape

We further modified the peg object to have a round shape, with its diameter matching the previous square length. For this variant as well, both control strategies achieved a 100% success rate. The average process time for inserting this round peg applying RL-based control was 3.62 s, with a standard deviation of 0.20 s. The classical control approach achieved an average cycle time of 1.78 s with a standard deviation of 0.09 s. The histogram in Figure 9 shows the distribution of the process times for both control strategies. Once again, the classical control significantly outperformed the RL-based system in terms of cycle time, requiring less than half as long on average.

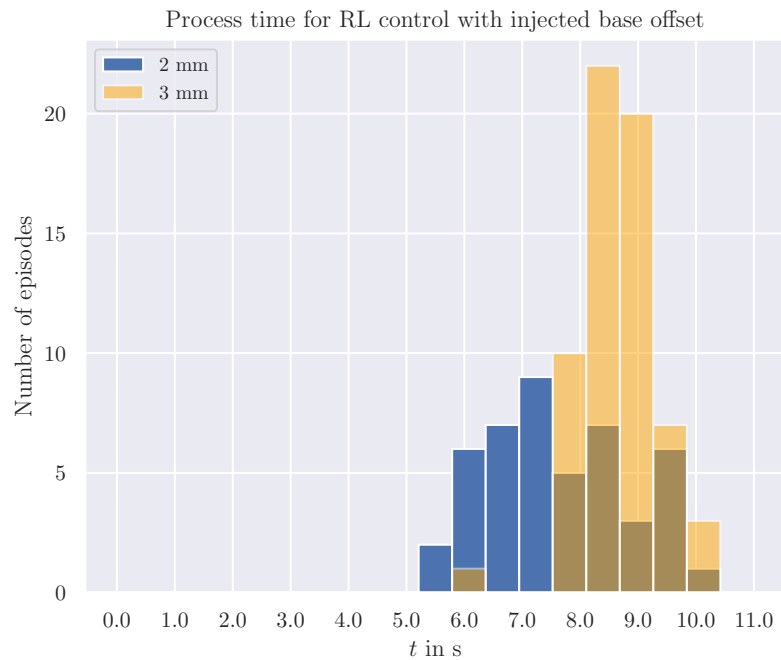


**Figure 9.** Adaptation of the peg to a circular base area. Comparison of process time for the classical and the proposed RL-based control system in reality.

### 5.4. Injected Calibration Offset

We evaluated the robustness of the control system towards calibration offsets of both arms, i.e., a perpendicular deviation of the right robot's basis from the original axis introduced in software. The classical trajectory control failed to complete the task in 100% of the cases for an offset bigger than 1.5 mm. This motivates to examine the RL-based control system's performance under similar conditions for a maximum episode duration of 10 s. The histogram in Figure 10 illustrates the distribution of process cycle times for both an offset of 2 mm and 3 mm.

At a shift of 2 mm, the RL approach still achieves a success rate of 47%, while for an offset of 3 mm, the success rate even increases to 63%. However, the average process time increases from 7.64 s to 8.64 s between both setups.



**Figure 10.** Distribution of process cycle times for RL control with an injected base offset of 2 mm and 3 mm.

## 6. Discussion

The selection of  $R_{min}$  for the curriculum implicitly predetermines the average process time by influencing the convergence of the return  $R$  toward this threshold. The maximum episode length parameter, here set to 150 steps, establishes a maximum process duration equivalent to 7.5 s. The simulation reveals a substantial variance in the normally distributed insertion duration, determined by the difference between  $R_{min}$  influencing the time needed to accumulate this reward and the maximum episode length. This variance can be reduced, albeit at the expense of a more complex learning process, by narrowing the gap between these two values. This finding is pivotal, as it indicates that for cyclic assembly processes, crucial characteristics can be statistically determined a priori by the curriculum parameters.

The classical control reaches a success rate of 100% in reality and so does our RL-based approach. We expected this, given that the clearance of both peg and hole are 1 mm and thereby one order of magnitude above the Franka's repeatability of pose and path of 0.1 mm. To rank this result in the state of research, we refer to [16]. They achieved a success rate of less than 70% in reality for a similar task. As we also utilized RL with a hierarchical control system, we identified the curriculum-based training strategy and application of PLAI as the key to our success. Even if classical control performed twice as fast as the RL-based control system regarding the process time, the latter offers several advantages. The RL-based control system is more flexible and could be adapted to strongly differing workpiece geometries with minimal effort of retraining given updated CAD files. On the other hand, the classical control system is more sensitive to variations in the environment that may lead to jamming and not finishing the insertion, requiring careful (re)calibrations. Figure 8 shows that the RL-based control actually has a lot of contact at the beginning of the insertion process and manages to prevent jamming by correcting its forces from feedback of the observations. An experiment introducing an error for one of the robot's base positions proved the classical control to fail at an offset of more than 1.5 mm, while the RL-based control system still achieved a success rate of 63% at an offset of 3 mm.

Experiments with a round peg in the real-world setup confirm the classical control to maintain the same process cycle time compared to the squared peg, as minimal contact and no dynamic behavior changes are observed. Meanwhile, the RL-based policy achieves a shorter cycle time compared to the squared peg scenario. This improvement is due to the elimination of the need for precise orientation, which reduces contacts and the likelihood



of jamming situations. It also confirms the ability of the RL approach to solve errors in orientation without jamming.

## 7. Conclusions

In this study, we build upon our previous work by successfully transferring a curriculum-based RL approach for robotic dual-arm assembly from simulation to reality. We focus on a peg-hole task with a 1.0 mm clearance, employing well-established software libraries such as robosuite for impedance control, MuJoCo for physics simulation, SB3 for RL, and gym for seamless interfacing. ROS 2 connects the robots to a workstation that runs the supervisory control policy. The control system architecture employs a low-level impedance controller on each robot, receiving joint position setpoints from the policy.

The training in simulation follows a reverse curriculum strategy, starting each iteration with fulfilling basic return thresholds and progressing to fine-tuning afterwards. We introduce DR in the form of noisy joint angles at initialization to enhance the system's adaptability to various initial states. Post-training, we analyze inferences from the controlled system to assess reliability and average process time, confirming that the curriculum parameters can effectively predict these outcomes, even after deployment on the real-world setup. Traditional control algorithms show less variance and shorter process times compared to our RL approach. Nevertheless, training a control system in simulation enables the system to be confronted with a wide distribution of situations and thereby increases adaptability and flexibility particularly demonstrated for calibration errors.

Currently, high training durations of more than 10 hours on a state-of-the-art desktop PC are limiting the deployment for fast changing requirements. We also require human expertise to manually construct a specific curriculum. This procedure has to be automated. As we first explored capabilities of such Sim2Real transfers, we decided to employ a large clearance of 1 mm. The range of relevant assembly applications reaches 0.5 mm at the upper bound of ISO 286.

Our future work will investigate the effects of bootstrapping a policy leveraging imitation learning from a small number of expert demonstrations to reduce the training time. We plan to explore tighter clearances and different geometries to enhance applicability to relevant real-world assembly tasks. Additionally, integrating low-cost cameras into the setup will allow us to increase adaptivity to task variations, thereby expanding the system's flexibility.

**Author Contributions:** Conceptualization, K.W., O.D., T.W. and U.F.; methodology, K.W., O.D. and U.F.; software, K.W., R.L. and T.W.; validation, K.W., S.Z. and R.L.; investigation, K.W., S.Z. and R.L.; data curation, S.Z. and R.L.; writing—original draft preparation, K.W.; writing—review and editing, S.Z.; visualization, K.W.; supervision, T.W. and U.F. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by Project “Rob4Print-Digital robot training to minimize waste in 3D printing”, Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV) Germany through the program DigiRes—Digital applications to increase resource efficiency in circular production processes.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

EE	end-effector
DR	domain randomization

HER	Hindsight Experience Replay
MDPI	Multidisciplinary Digital Publishing Institute
NN	neural network
PLAI	policy-level action integration
RL	reinforcement learning
Sim2Real	simulation-to-reality
SAC	Soft Actor-Critic
SB3	Stable Baselines3

## References

1. Wrede, K.; Donath, O.; Wohlfahrt, T.; Feldmann, U. Curriculum-Organized Reinforcement Learning for Robotic Dual-arm Assembly. In Proceedings of the Automation, Robotics & Communications for Industry 4.0/5.0, Innsbruck, Austria, 7–9 February 2024; pp. 8–14. [\[CrossRef\]](#)
2. Jiang, J.; Huang, Z.; Bi, Z.; Ma, X.; Yu, G. State-of-the-Art Control Strategies for Robotic PiH Assembly. *Robot. Comput.-Integr. Manuf.* **2020**, *65*, 101894. [\[CrossRef\]](#)
3. Yuan, F.; Shen, X.; Wu, J.; Wang, L. Design of Mobile Phone Automatic Assembly System Based on Machine Vision. *J. Phys. Conf. Ser.* **2022**, *2284*, 012012. [\[CrossRef\]](#)
4. Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; Zhu, Y. Robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. *arXiv* **2022**, arXiv:2009.12293. [\[CrossRef\]](#)
5. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A Physics Engine for Model-Based Control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 5026–5033. [\[CrossRef\]](#)
6. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
7. Salvato, E.; Fenu, G.; Medvet, E.; Pellegrino, F.A. Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access* **2021**, *9*, 153171–153187. [\[CrossRef\]](#)
8. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540. [\[CrossRef\]](#)
9. Sutton, R.S.; Barto, A. *Reinforcement Learning: An Introduction*, nachdruck ed.; Adaptive Computation and Machine Learning; The MIT Press: Cambridge, MA, USA, 2014.
10. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 737–744. [\[CrossRef\]](#)
11. Scheiderer, C.; Dorndorf, N.; Meisen, T. Effects of Domain Randomization on Simulation-to-Reality Transfer of Reinforcement Learning Policies for Industrial Robots. In *Advances in Artificial Intelligence and Applied Cognitive Computing*; Transactions on Computational Science and Computational Intelligence; Arabnia, H.R., Ferens, K., Kozerenko, E.B., Olivás Varela, J.A., Tinetti, F.G., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 157–169. [\[CrossRef\]](#)
12. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30. [\[CrossRef\]](#)
13. Tang, B.; Lin, M.A.; Akinola, I.A.; Handa, A.; Sukhatme, G.S.; Ramos, F.; Fox, D.; Narang, Y.S. IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality. In Proceedings of the Robotics: Science and Systems XIX, Daegu, Republic of Korea, 10–14 July 2023; Volume 19.
14. Coumans, E.; Bai, Y. PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016. Available online: <http://pybullet.org> (accessed on 19 September 2024).
15. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
16. Alles, M.; Aljalbout, E. Learning to Centralize Dual-Arm Assembly. *Front. Robot. AI* **2022**, *9*, 830007. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Yang, X.; Ji, Z.; Wu, J.; Lai, Y.K. An Open-Source Multi-goal Reinforcement Learning Environment for Robotic Manipulation with Pybullet. In *Towards Autonomous Robotic Systems*; Fox, C., Gao, J., Ghalamzan Esfahani, A., Saaj, M., Hanheide, M., Parsons, S., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 14–24. [\[CrossRef\]](#)
18. Panerati, J.; Zheng, H.; Zhou, S.; Xu, J.; Prorok, A.; Schoellig, A.P. Learning to Fly—A Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 7512–7519. [\[CrossRef\]](#)
19. Mittal, M.; Yu, C.; Yu, Q.; Liu, J.; Rudin, N.; Hoeller, D.; Yuan, J.L.; Singh, R.; Guo, Y.; Mazhar, H.; et al. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robot. Autom. Lett.* **2023**, *8*, 3740–3747. [\[CrossRef\]](#)
20. Serrano-Muñoz, A.; Arana-Arexolaleiba, N.; Chrysostomou, D.; Boegh, S. Skrl: Modular and Flexible Library for Reinforcement Learning. *J. Mach. Learn. Res.* **2022**, *24*, 1–9.

21. Nasiriany, S.; Liu, H.; Zhu, Y. Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 7477–7484. [\[CrossRef\]](#)
22. Wilcox, A.; Balakrishna, A.; Dedieu, J.; Benslimane, W.; Brown, D.; Goldberg, K. Monte Carlo Augmented Actor-Critic for Sparse Reward Deep Reinforcement Learning from Suboptimal Demonstrations. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 2254–2267.
23. Xu, M.; Veloso, M.; Song, S. ASPiRe: Adaptive Skill Priors for Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 38600–38613.
24. Park, H.; Bae, J.H.; Park, J.H.; Baeg, M.H.; Park, J. Intuitive Peg-in-Hole Assembly Strategy with a Compliant Manipulator. In Proceedings of the IEEE ISR 2013, Seoul, Republic of Korea, 24–26 October 2013; pp. 1–5. [\[CrossRef\]](#)
25. Broenink, J.F.; Tiernego, M.L.J. Peg-in-Hole Assembly Using Impedance Control with a 6 DOF Robot. In Proceedings of the 8th European Simulation Symposium, Genoa, Italy, 24–26 October 1996; pp. 504–508.
26. Nottensteiner, K.; Stulp, F.; Albu-Schäffer, A. Robust, Locally Guided Peg-in-Hole Using Impedance-Controlled Robots. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 5771–5777. [\[CrossRef\]](#)
27. Park, H.; Kim, P.K.; Bae, J.H.; Park, J.H.; Baeg, M.H.; Park, J. Dual Arm Peg-in-Hole Assembly with a Programmed Compliant System. In Proceedings of the 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Kuala Lumpur, Malaysia, 12–15 November 2014; pp. 431–433. [\[CrossRef\]](#)
28. Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; Abbeel, P. Reverse Curriculum Generation for Reinforcement Learning. In Proceedings of the 1st Annual Conference on Robot Learning, PMLR, Mountain View, CA, USA, 13–15 November 2017; pp. 482–495.
29. Aljalbout, E.; Frank, F.; Karl, M. On the Role of the Action Space in Robot Manipulation Learning and Sim-to-Real Transfer. *IEEE Robot. Autom. Lett.* **2024**, *9*, 5895–5902. [\[CrossRef\]](#)
30. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
31. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
32. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot Operating System 2: Design, Architecture, and Uses in the Wild. *Sci. Robot.* **2022**, *7*, eabm6074. [\[CrossRef\]](#) [\[PubMed\]](#)
33. Coleman, D.T.; Sucas, I.A.; Chitta, S.; Correll, N. Reducing the Barrier to Entry of Complex Robotic Software: A MoveIt! Case Study. *arXiv* **2014**, arXiv:1404.3785. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.