Simulation-based Learning of the Peg-in-Hole Process Using Robot-Skills

Arik Lämmle, Philipp Tenbrock, Balázs Bálint, Frank Nägele, Werner Kraus, József Váncza, Marco F. Huber

***Abstract*— Increasingly volatile markets challenge companies and demand flexible production systems that can be quickly adapted to new conditions. Machine Learning has proven to show significant potential in supporting the human operator during the time-consuming and complex task of robot programming by identifying relevant parameters of the underlying robot control program. We present a solution to learn these parameters for contact-rich, force-controlled assembly tasks from a simulation using hardware-independent robot skills. We show that successful learning and real-world execution are possible even under process deviation and tolerances utilizing the designed learning system. We present learning skill parameters as high-level robot control, evaluation and comparison of extensive simulations, and preliminary experiments on a physical robot test-bed. The developed solution approach is evaluated and discussed using the Peg-in-Hole process, a typical benchmark process in force-controlled assembly.**

***Index Terms*— Robot Skills, Deep Reinforcement Learning, Simulation**

I. Introduction

Many assembly operations pose a high potential for robot-based automation but are not realized in industrial applications. One reason is the complex and time-consuming programming of the required force control or vision systems to automatically adapt to process deviations and product tolerances. Thus, easy-to-use and efficient tools for robot programming are highly desirable. A promising approach to hide the complexity from the user is task-level programming employing predefined robot skills. Skills provide reusable position and force-controlled modules, encapsulating the relationship between sensor observations and robot control, and can easily be parameterized to fulfill the desired assembly task. In contrast to standard teach-in methods, robot skills can be transferred to other robots as they rely on commands in the task-space and not the robot-specific joint-space. Despite their advantages, it still takes some time to parameterize and tune the skills manually on the physical robot system, limiting their application for small lot sizes or high variances. Another solution is simulation-based offline programming. Compared to online methods, this does not require an actual robot cell; however, the control programs still have to be generated manually and adjusted afterwards due to the differences between the digital and real world.

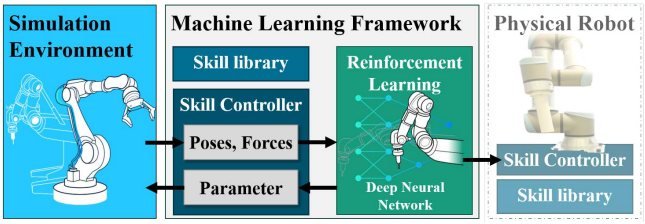

Fig. 1. Paper focus: Learning skill parameters from simulation.

In this paper, we combine the strengths of robot skills with the advantages of offline programming in a simulation environment. The time-consuming parameterization of the skills is compensated by a learning method that allows the situation-specific adaptation of the robot to industry-typical position tolerances. The robot agent uses Deep Reinforcement Learning (DRL) to learn appropriate skill parameters through continuous trial and error, similar to a human learning a manipulation skill by interacting with their environment. Thus, we employ DRL to train the complex interactions between the multidimensional sensor inputs and the situation-specific movements of the robot in order to achieve targeted sequential decision making. Using a digital simulation environment prevents damage to the robot or its environment during the learning process. Numerous current studies deal with the training of robots using DRL, either requiring a physical robot cell or using a simulation but training on the robot- and application-specific joint coordinates [1]–[3]. We demonstrate the effectiveness of the simulation-based training of robot skills using the force-controlled Peg-in-Hole benchmark process. For the training, three learning algorithms are compared: Soft-Actor-Critic (SAC), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Proximal Policy Optimization (PPO). To test the robustness of the trained robot controller, the position of the hole is varied over industry-typical uncertainties. In addition to the training in simulation, we present first experiments on a physical robot test-bed.

The contributions of applying DRL over the parameter space of a task-level force-based robot control policy are several-fold. First, doing so results in inherent policy-

All authors contributed equally.

The research presented in this paper has received funding from the *Bundesministerium für Bildung und Forschung* in the *Rob-aKademI* project (project number 01IS20009C) and from the European Union's *Horizon 2020* research and innovation programme in the *EPIC* research project (grant agreement No 739592).

A. Lämmle, P. Tenbrock, B. Bálint, F. Nägele and W. Kraus are with the department of Robot and Assistive Systems, Fraunhofer Institute for Manufacturing Engineering and Automation IPA. E-mails: {arik.laemmle, philipp.tenbrock, balazs.andras.balint, frank.naegele, werner.kraus}@ipa.fraunhofer.de

J. Váncza is with the Research Laboratory on Engineering and Management Intelligence, Institute for Computer Science and Control SZTAKI. E-mail: vancza.jozsef@sztaki.hu

M. F. Huber is with the Centre of Cyber Cognitive Intelligence, Fraunhofer Institute for Manufacturing Engineering and Automation IPA and with the Institute of Industrial Manufacturing and Management IFF, University of Stuttgart. E-mail: marco.huber@ieee.org

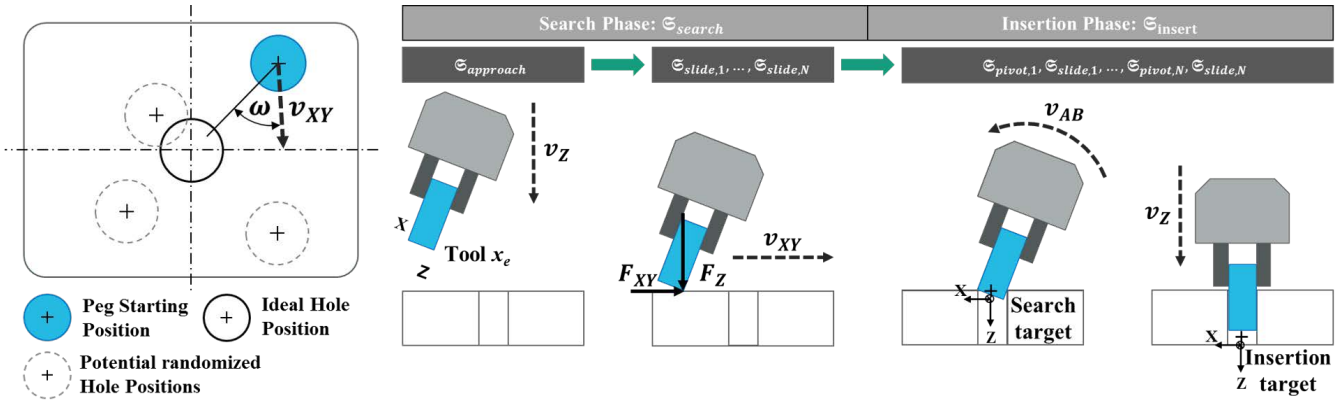

Fig. 2. Structure of the Peg-in-Hole task, used in the presented work consisting of a search and an insertion phase. The peg has a diameter of 39 mm, the hole 40 mm. Initially, the robot positions the peg above the hole body and 45 mm from the ideal hole position at an angle of 45° in the (xy) plane.

robustness against process uncertainties and noise, making the resulting program transfer more likely to the real world out-of-the-box. Second, defining such a skeleton for the resulting policy encapsulates a significant amount of the expert knowledge required for robot programming, enabling the straightforward generalization of the policy over robots used, part variants, and other cell elements and its easy adaptation to process uncertainties and tolerances.

The paper is structured as follows. Section II describes the considered skill parameterization problem. Details of the system design are described in Section III. Section IV focuses on simulation-based training, and presents first results from real-world experiments. Finally, we conclude the paper and provide an outlook on future work.

II. Considered Problem and Related Research

Instead of teaching the robot's movement directly, robot skills define relative motions and constraints between objects of the robotic system, mainly workpieces, tools, and fixtures. Our work aims to train a model that selects feasible relative motions, constraints, and controller parameters – in contrast to learning methods for assembly operations that train directly on the robot joint coordinates and implicitly learn the robot kinematics and the force control. In this field, most recent solutions [1]–[6] train policies that observe the robot joint states and transform them into end-effector movements or utilize fine-tuned trajectories. However, the learned controllers are specific to the robot system and the process and cannot be used without extensive retraining when the system changes. In contrast, the skills used in this work can be adapted to process changes, product variants, or even changes to the robot and the periphery by simply adjusting the parameters [7].

Regardless, these parameter adjustments are generally required when transferring such a skill-level robot control policy between platforms. While encapsulating the dependence on the controlled robot's dynamics, the configured skills still rely on them implicitly through the skill's parameter values. Consequently, while such a policy runs on any suitable platform, a performance drop is expected; parallel research has developed solutions for this problem with good results [8]–[10].

Nonetheless, similar recent frameworks or pipelines to the one this paper presents do exist. For example, Sharma et al. [11] use reinforcement learning to develop and compose task-axis robot controllers in place of skills, and Liang et al. [12] open up the skill sequence for the learning agent to solve force-controlled manipulation tasks. While the structure and sequence of skills remain fixed in our work, the framework we present incorporates a formal layer of skill composition that facilitates straightforward interchangeability and potentially allows non-expert human access, both of which are desirable for an industrial application. Nonetheless, automating our skills' composition and their sequencing is a part of our ongoing work.

The Peg-in-Hole task is a well-researched benchmark process and describes the basic procedure of multiple assembly processes in industry. The goal of the Peg-in-Hole is to insert an often cylindrical body (peg) into its counterpart (hole). The hole usually has a slightly larger diameter than the peg (clearance). The robotic automation of the task requires a force-controlled solution as the positioning errors in the process exceed the assembly clearance. Visuomotor solutions are also impractical for most applications, as occlusions can easily occur during the process. Generally, the Peg-in-Hole task consists of two subsequent phases building on each other, 1) the search and 2) the insertion phases (Fig. 2). Recent solutions achieve success rates of up to 100% in executing the Peg-in-Hole task. For example, Inoue et al. [1] first train their policy with 1 mm positional error of the hole, then further with 3 mm, learning to handle even sub-millimeter clearances. The solution of Fan et al. [3] can handle clearance of 0.2 mm, a positional error of 0.5 mm on the hole with a bounded exploration space of 3mm around it. The trajectory-based approach of Park et al. [6] can handle a clearance of 10 μm. Compared to the existing work of the Peg-in-Hole tasks, we train the used robot controllers entirely in a safe simulation environment. For the evaluation, we consider industry-typical uncertainties of the actual hole position. In general, to the best of our knowledge, no solution

exists for the offline training of position- and force-controlled skills in robot-based assembly.

III. System Design

We adopt the skill model of Nägele et al. [7], where each skill

$$\mathfrak{S} = (\mathcal{N}, \mathcal{KE}, \mathcal{T}, \mathcal{SC}, \mathcal{M}, \mathcal{TR}, \mathfrak{S}_{sub}) \tag{1}$$

is a 7-tuple; $\mathcal{N}$ is the unique name of the skill, $\mathcal{KE}$ is a set of kinematic elements that describe the robot's and the task's kinematic model based on the iTaSC formalism [13], $\mathcal{T}$ is a list of tasks that include the control variables and parameters, $\mathcal{SC}$ is a collection of scripts for additional support functions, the monitor $\mathcal{M}$ describes the stop conditions for terminating the skill, $\mathcal{TR}$ defines the transition between the skill and its subsequent skill, and lastly, $\mathfrak{S}_{sub}$ is a list of sub-skills.

A. Skill Formalism for the Peg-in-Hole Task

The two phases of the Peg-in-Hole task (Fig. 2) can be defined as the high-level skills $\mathfrak{S}_{search}$ and $\mathfrak{S}_{insert}$. The skill for the search phase

$$\mathfrak{S}_{search} = (\mathfrak{S}_{approach}, \mathfrak{S}_{slide,1}, ..., \mathfrak{S}_{slide,N}) \tag{2}$$

starts with approaching the hole body's surface and continues with up to $N = 15$ consecutive slides, which apply a constant contact force on the reached surface while moving along it. Early simulation studies showed that this provides the agent with a sufficient number of skills to explore its environment, especially at the beginning of the training. At the same time, the agent is trained to use as few skills as possible through higher rewards. The reinforcement learning agent learns a subset of the control variables and parameters $\mathcal{T}_{slide}$, which will be explained in Section III-B; the skill $\mathfrak{S}_{approach}$ is not learned. Thus, the trained policies can adjust the robot's behavior in each consecutive slide skill adapting to the sensor observations. Since the peg is directly mounted on the robot flange, no additional scripts $\mathcal{SC}_{slide}$ are needed to control a gripper or other end-effector. The parameters for the transition $\mathcal{TR}_{slide}$ were hand-crafted through experiments both in simulation and on the real-world robotic test-bed (see Section IV) to make sure that the criterion for finding the hole in the search phase allows joining the peg in the subsequent joining phase. The condition evaluates whether the tip of the peg (i) is within the confidence interval of 2.5 mm in the (xy) plane around the center of the true hole position and (ii) if it is dipped at least 2.0 mm into the hole. The stop conditions $\mathcal{M}_{slide}$ consist of (i) a timeout of 3.5 s per skill, (ii) the agent reaching the predefined number N of skill executions without fulfilling $\mathcal{TR}_{slide}$, and (iii) the forces measured on the robot's flange exceeding a predefined force threshold to protect the hardware.

The skill for the insertion phase

$$\mathfrak{S}_{insert} = (\mathfrak{S}_{pivot,1}, \mathfrak{S}_{slide,1}, ..., \mathfrak{S}_{pivot,N}, \mathfrak{S}_{slide,N}) \tag{3}$$

is defined similarly to that of the search phase. The agent can choose to pivot the peg around the axes x or y or both combined with a subsequent pushing movement along the peg's longitudinal axis. This alternation enables the agent to learn a continuous threading movement and adapt to even tight tolerances. The transition $\mathcal{TR}_{insertion}$ is whether the agent could insert the peg 20 mm deep into the hole, thus solving the Peg-in-Hole task. Again, early simulation and real world experiments showed that inserting the peg even further into the hole with the previously set movements is not a challenge for the robot.

B. Reinforcement Learning Framework

The reinforcement learning objective can be formulated as

$$\begin{aligned} \pi^* &= \arg\max_{\pi} G(\pi) \\ &= \arg\max_{\pi} \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=1}^{\mathrm{T}} \gamma^{t-1} r(o_t, a_t) \right], \end{aligned} \tag{4}$$

finding the optimal policy π^*, which selects the agent's actions a_t to maximize the expected, discounted return over a T-step trajectory τ and its probability distribution $p(\tau|\pi)$. At each time step t, the agent receives an observation

$$\begin{aligned} o_t &= [p_{ideal,t}, \hat{o}_{t-i} : \forall i \in \{0, 4, 8, 12, 16\}] \in \mathbb{R}^{28} \\ \hat{o}_{t-i} &= [F_{e,t-i}, p_{e,t-i}] \in \mathbb{R}^5 \end{aligned} \tag{5}$$

of its environment that consists of the forces $F_{e,t} = (F_{e,xy,t}, F_{e,z,t}) \in \mathbb{R}^2$ acting on the tip of the peg, the peg tip's position $p_{e,t} \in \mathbb{R}^3$, and the ideal hole position $p_{ideal,t} \in \mathbb{R}^3$. A history of four previous observations is also provided to the agent. The type of the observed data is the same during training and evaluation in the simulation and policy execution on the real-world robotic system to facilitate the policy transfer.

Two learnable action configurations $\hat{\mathcal{T}}_i \subset \mathcal{T}_{slide}$ are tested for $\mathfrak{S}_{search}$. In the first configuration

$$\hat{\mathcal{T}}_1 = [\omega_t, v_{e,XY,t}] \ , \tag{6}$$

the agent can set the angle $\omega_t \in [-45°, 45°]$ between its linear movement and the ideal hole position, and the magnitude of the peg tip's velocity $v_{e,XY,t} \in [2.5 \ \frac{mm}{s}, 10 \ \frac{mm}{s}]$ in the world frame's (xy) plane (see Fig. 2). In the second configuration

$$\hat{\mathcal{T}}_2 = [\omega_t, v_{e,XY,t}, k, F_{e,Z,t}] \ , \tag{7}$$

on top of the previous values, the agent can also adapt the PD controller's $k \in [0.0001, 0.001]$ factor and reference force $F_{e,Z,t} \in [1\,\mathrm{N}, 20\,\mathrm{N}]$ that regulates the contact force between the hole body and the peg along the latter's longitudinal axis.

Preliminary simulated training runs showed that enabling the agent to set ω_t and $v_{e,XY,t}$ for sliding instead of constraining its movements to either the axes x or y resulted in better training results. The peg is also tilted from its upright position in the direction of sliding, reducing the contact area between the peg and the body with the hole from 3D to 2D. Experiments conducted on the real-world test-bed (see Section IV) showed that tilting the peg by $15°$ is highly favorable for applying a constant contact force on the body's surface with the hole compared to the angles $0°$, $30°$, or $45°$.

Furthermore, tilting the peg also resulted in smoother contact forces in the simulation.

During training, the agent receives a scalar reward

$$r(o_t, a_t) = e^{\left(\frac{-2 \cdot d_{xy}}{K}\right)} - 1 \tag{8}$$

after each skill execution based on the distance d_{xy} of the peg's tip and the hole's center point. $K = 70$ mm is the radius of an early termination zone around the hole's center point. If the agent leaves this zone, the current training epoch is terminated and a new one is started. The optimization problem defined in (4) is addressed by employing model-free reinforcement learning approaches, the two off-policy algorithms Soft-Actor Critic (SAC) [14] and Twin Delayed Deep Deterministic Policy Gradient (TD3) [15] and the on-policy algorithm Proximal Policy Optimization (PPO) [16]. We selected a negative reward and a discount rate of $\gamma = 0.99$ for the agent to not only find the hole but to do so using as few skills as possible.

One of the training's primary goals is to develop control policies that can handle the uncertainties resulting from process-specific tolerances and inaccuracies. We made these tolerances and inaccuracies present through perturbing the hole's ideal position according to a uniform distribution in the range of $[-10\text{ mm}, 10\text{ mm}]$ before every attempt at task execution both on our real-world test-bed and in our digital model. Typical deviations of the hole position in industrial Peg-in-Hole tasks are ± 5 mm. However, the modeled randomization is deliberately chosen to be larger in order to show the limits of simulation-based skill training. To prepare the trained policies for their transfer from simulation to reality, we applied additional randomizations in simulation. The starting position of the peg is varied uniformly by $[-5\text{ mm}, 5\text{ mm}]$ in the (xy) plane around its initial position (see Fig. 2). Also, the joint position signals that the robot controller receives are randomized by adding a component to them from the normal distribution $\mathcal{N}\left(0\text{ rad}, 0.02^2\text{ rad}^2\right)$ to model the positioning inaccuracies and the sensor noise of the robot.

IV. Simulation-based Training

The in-simulation training of robot skills is the focus of the work presented. In the following section, the modeling of the Peg-in-Hole task will be discussed first in the employed physics engine MuJoCo [17]. Then, a physical robot test-bed is presented to evaluate the modeling and the subsequent execution of the trained policies.

A. Setup of Simulation and Robotic Test-Bed

The general parameters of the employed physics engine MuJoCo (simulation time-step, contact stiffness, contact damping) were tuned before deploying the training, apparently tempering the claimed generalizability of the presented approach. However, this tuning was only qualitative to ensure the general physical plausibility of the contact forces from the simulation and their rough correspondence to the values recorded from the physical test-bed. We find that such an initial configuration is usually necessary due to the engine's nature; an auxiliary aspect of our ongoing complementary research [18] is to develop a more formalized approach to aid the user in doing so.

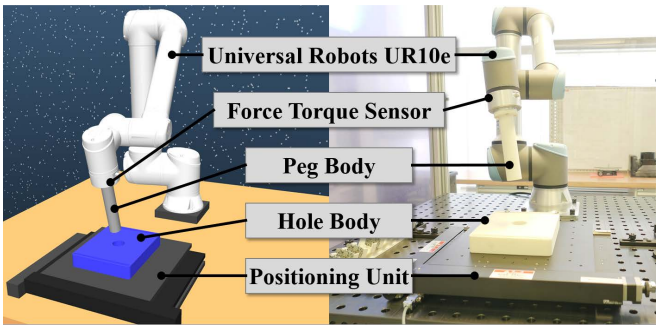

Fig. 3. Simulated digital twin and physical robot test-bed

Our physical test-bed (Fig. 3) consists of a Universal Robots UR10e, a cross table for positioning the hole body, and the surrounding robot cell, including safety. The test-bed is, thus, a replica of the digital model in the simulation. The peg is made of plastic, was produced by selective laser sintering (SLS), and is screwed directly onto the robot's force-torque sensor. The hole body was also manufactured using the SLS process and is screwed to the mounting plate of the cross table. The cross table is used for the hole's high-precision positioning and mapping the randomized hole positions for the control policy evaluation. Before starting each experiment, the cross table is calibrated to ensure the internal digital angle encoders' accuracy. The robot is controlled through skills and a ROS interface. A randomized component was added to the simulated force values from the normal distribution $\mathcal{N}\left(0\text{ N}, 0.2^2\text{ N}^2\right)$ to reproduce the force-torque sensor's noise.

MuJoCo relies on smooth dynamics in the model simulated to produce stable, physically plausible data. Therefore, directly writing the desired robot joint velocity signals provided by the controller into the simulation as joint velocities of the digital robot proved insufficient by itself. It resulted in "jumpy" robot movements and unrealistically high forces, so a model of the low-level robot controller had to be considered for simulation. The physics engine enabled the smooth control of joints through per-joint PID controllers. While it had long been possible to synthesize such a controller for an industrial robot arm [19], doing so generally required knowing parameters of the robot that are rarely available from robot manufacturers. Hence, we decided to employ a constraint-based controlling approach instead.

We defined soft equality constraints along the translational Cartesian axes between the robot flange and the peg attached to it, practically resulting in a compliant member in the kinematic tree. MuJoCo allowed us to directly tune the second-order dynamical error decay of such constraints, which we could set so that the error disappeared fast enough not to impact the controller's observations. Consequently, we assumed that the dynamical behavior of the controlled robot arm was practically rigid while solving the Peg-in-

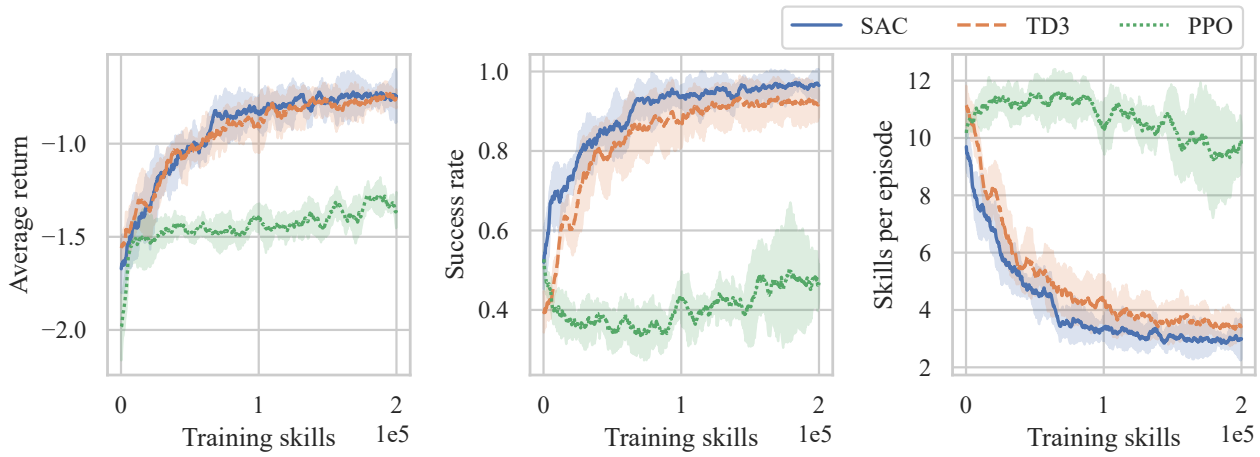

Fig. 4. Average return, success rate and number of used skills from the simulation-based training for $\hat{\mathcal{T}}_1$

Hole task. Using this approach, we could apply the target joint velocity signals directly in the simulation; their effect on the peg's movement was realized through a "compliance" that implemented the smoothness that MuJoCo required for physical plausibility.

B. Training Results in Simulation

For the learning agent, Soft-Actor Critic (SAC) [14], Twin Delayed Deep Deterministic Policy Gradient (TD3) [15], and Proximal Policy Optimization (PPO) [16] from Stable Baseline3 [20] were used. Five short and five long training runs were performed using five varying seeds for each of the three algorithms; resulting in 150 training runs total, each with up to 50.000 skill executions in the short and up to 200.000 in the long runs. The policies in training were evaluated after every 200 skill executions, leading to 250 policy evaluations for the short and 1.000 for the long runs. Each policy evaluation itself consisted of five policy rollouts, in which the hole position was randomly varied at the beginning. We used the average return, success rate, and the number of skills executions until successful task completion as evaluation metrics. The results visualized in Fig. 4 show the in-simulation evaluations of the long training runs with SAC and TD3 in the action configuration $\hat{\mathcal{T}}_1$. Both algorithms converge towards comparable bounds, and the results of the short training runs show lower returns and success rates and, on average, used more skills to find the hole across the board, hinting that the agent could leverage the additional time for learning that the long runs provided. Using the second action configuration $\hat{\mathcal{T}}_2$, in which the agent could also train the parameters of the force controller, did generally not lead to significant changes in returns or success rates. An overview of the training results is provided in Table I. In both configurations, the average skill count converged towards 3. The training with PPO did not yield any significant success, which can be attributed in particular to the design of the learning only after each individual skill execution.

TABLE I

SUCCESS RATES OF THE SIMULATION EXPERIMENTS

		Maximum randomization radius r_{rnd}			
		$\leq$ 2.5 mm	$\leq$ 5 mm	$\leq$ 10 mm	total
SAC, $\hat{\mathcal{T}}_1$	short	100.0%	100.0%	98.36%	92.83%
SAC, $\hat{\mathcal{T}}_2$	short	100.0%	98.77%	89.09%	78.28%
TD3, $\hat{\mathcal{T}}_1$	short	100.0%	96.30%	78.68%	67.62%
TD3, $\hat{\mathcal{T}}_2$	short	100.0%	99.51%	91.36%	83.95%
SAC, $\hat{\mathcal{T}}_1$	long	100.0%	99.75%	99.43%	96.37%
SAC, $\hat{\mathcal{T}}_2$	long	98.10%	99.01%	97.98%	94.33%
TD3, $\hat{\mathcal{T}}_1$	long	100.0%	99.75%	98.68%	97.32%
TD3, $\hat{\mathcal{T}}_2$	long	100.0%	100.0%	96.09%	92.83%

C. Discussion of Simulation Experiments

The policies trained with SAC achieve success rates up to 96.37%, while TD3 reaches 97.32%, even for a randomization of the hole above 10 mm. Only minor differences can be observed between the two action configurations. Generally, the policies trained with SAC outperform those trained with TD3. A possible explanation for the differences in the two algorithms' performance is the stronger dependence of TD3 on the chosen training hyper-parameters. Similar results are also reflected in the average number of skills used. The success rates of the simulation-based skill-trainings are, thus, less than 3% below the results of state-of-art solutions for the Peg-in-Hole task. At the same time, the trained policies are outstanding in particular for their higher generalization capability and robustness to deviations in the hole position. Our trained policies also perform with success rates of 100% for a randomization radius $r_{rnd} \leq 5$ mm of the hole position around its ideal position. If the hole position is randomized even over industry-typical values with more than 10 mm (total), success rates up to 97.32% are still achievable. In contrast, the remarkably small diameter differences between peg and hole from the work of Inoue et al. [1] were not verified, as they exceed the limits of the simulation resolution.

In the short training runs, the success rates of the policies with either action configuration are slightly apart, compared

to the long runs. Again, a reason behind this could be the hyper-parameter-sensitivity of TD3, as no extensive tuning is performed over them. The longer training runs generally give better results, even with the extended action space $\hat{\mathcal{T}}_2$. The agent appears to leverage more interactions to better learn the situation-specific adaptation of the skills. Overall, both off-policy algorithms perform very well, which speaks for their use for training skill parameters. Policy updates occur after and not during the execution of a skill, which explains the significantly poorer performance of the on-policy algorithm PPO, as it can obtain states only at the end of skill execution, which presents a non-smooth volatile behavior of the robot. A possible solution could be to reduce the skills' duration or examine the possibility of learning while the skill is still being executed, but this was not further investigated.

D. Execution Results in Reality

The following part presents the results from a first policy transfer to the physical robotic test-bed. The policies trained with different algorithms and action configurations (see (6) and (7)) that perform best in the simulation are selected for the evaluation experiments. For each policy, 43 experiments are performed by varying the position of the hole. 33 positions are chosen on a 2.5 mm grid and 10 randomly around the ideal hole position. Each experiment consists of 5 episodes or policy rollouts, in which the agent has up to $N = 15$ skills at its disposal to solve the Peg-in-Hole task. The rollout is considered successful if the agent reaches the hole within these N skills. In total, six different policies were tested on the physical test-bed, resulting in 1290 executed episodes. Fig. 5 shows the results using the policy trained with SAC with action configuration $\hat{\mathcal{T}}_1$ and in a short training run. The position of each circle marks the hole position in the experiment. Larger circles represent on average fewer required skills, while the color scale represents the success rate.

E. Discussion of Real-World Experiments

For our first experiments, we focused on executing the policies from the short training runs. The results are presented in Table II. Remarkably, the success rates of 84% are close to the simulated results for a hole randomization of $r_{rnd} \leq 5\,\text{mm}$ in the real world for executing policies trained with SAC in action configuration $\hat{\mathcal{T}}_1$. However, if we raise the maximum distance to the total range of more than $10\,\text{mm}$, the success rates are with 57% significantly lower.

Admittedly, an adaptation of the slide phases' success criterion was needed: we had to increase the peg's target immersion depth to $5\,\text{mm}$ instead because the physical robot would often stop "early" when passing the hole while using the simulation's $2\,\text{mm}$ criterion. I.e., the policy would trigger the insertion phase even if the hole was only "partially" found; the peg was not appropriately aligned for the insertion. In light of this observation, we adapted and executed a few training runs. However, the target immersion depth of $5\,\text{mm}$ did not lead to any training progress in the simulation, which is potentially due to the introduced "compliance" between the robot arm and the peg being too soft; the robot's dynamical behavior was made rigid with respect to its controller, but not from a general mechanical perspective to save as much computational power as possible.

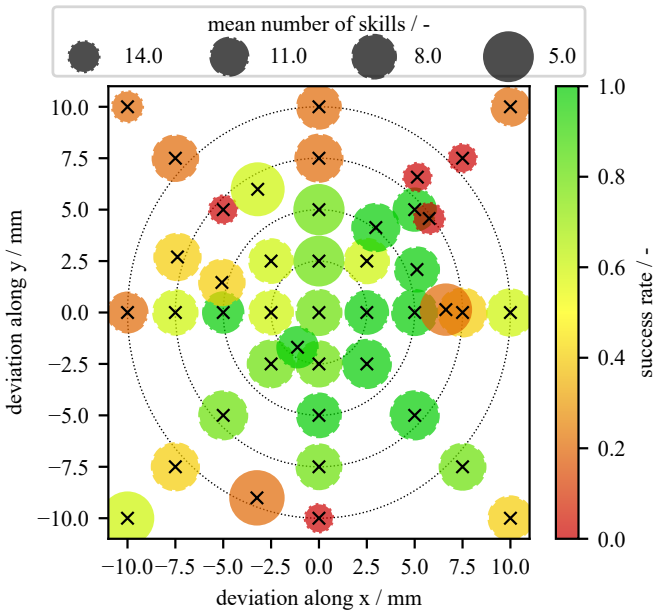

Fig. 5. Average success rate and mean number of skills in the real world

TABLE II
SUCCESS RATES OF THE PHYSICAL EXPERIMENTS

		Maximum randomization radius r_{rnd}			
		≤ 2.5 mm	≤ 5 mm	≤ 10 mm	total
SAC, $\hat{\mathcal{T}}_1$	short	83.33%	84.29%	61.49%	56.81%
SAC, $\hat{\mathcal{T}}_2$	short	60.00%	64.29%	58.86%	51.16%
TD3, $\hat{\mathcal{T}}_1$	short	63.33%	55.71%	46.86%	39.07%
TD3, $\hat{\mathcal{T}}_2$	short	86.67%	80.00%	59.43%	52.56%

Furthermore, the robot is observed to pass over the hole's side frequently while in policy execution. In such scenarios, the peg submerges into the hole but not deeply enough to trigger the transition to the insertion phase; then, it reemerges, stops a distance away, and the robot continues the searching motion in a direction away from the hole. This behavior might indicate that the agent could benefit from observing its intermediate states while executing a skill more. The history of observations it gets upon skill completion represents only the latter part of the execution; it is generally not long enough to record the first part of a skill. A more extended history could mitigate this issue, but it would not scale well; instead, observing the intermediate states in the frequency domain, aggregating them, or adapting the deep neural network's architecture would be worth investigating.

V. Conclusion and Future Prospects

Our work presents an easy-to-use solution approach for the simulation-based learning of reusable skills for high-level robot control. We selected the Peg-in-Hole task as a benchmark process for force-controlled assembly and

demonstrated the training in simulation and first results from the subsequent transfer to a physical robot system. The trained control policies showed up to 100% success rates and high robustness towards tolerances of the hole position exceeding even industrial-typical deviations.

Ongoing and future work includes adapting the success conditions in simulation and reality, as these represent the only current difference in the system's design. For this purpose, both manually adapted conditions and trained models are used, which observe and evaluate the robot's progress even during a skill execution to adapt the behavior of the robot agent. In addition to applying and evaluating further methods of sim-to-real transfer, this should lead to an increase in the success rates in reality and a closer alignment with the simulation results. Current research also focuses on learning the skill sequence or even new skill compositions and the task and controller parameters of the individual skills. In parallel, simulation-based training and transfer of further assembly processes, such as mounting terminal blocks in electrical control cabinets, is also taking place.

REFERENCES

[1] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 819–825.

[2] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *CoRR*, vol. abs/1710.06537, 2017. [Online]. Available: http://arxiv.org/abs/1710.06537

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

[4] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5844–5850.

[5] M. Kaspar, J. D. Muñoz Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2020, pp. 4383–4388.

[6] H. Park, J. Park, D.-H. Lee, J.-H. Park, and J.-H. Bae, "Compliant peg-in-hole assembly using partial spiral force trajectory with tilted peg posture," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4447–4454, July 2020.

[7] F. Nägele, L. Halt, P. Tenbrock, and A. Pott, "A prototype-based skill model for specifying robotic assembly tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 558–565.

[8] L. Halt, P. Tenbrock, F. Naegele, and A. Pott, "On the implementation of transferable assembly applications for industrial robots," in *50th International Symposium on Robotics*, 2018, pp. 1–7.

[9] L. Halt, F. Pan, P. Tenbrock, A. Pott, and T. Seel, "A transferable force controller based on prescribed performance for contact establishment in robotic assembly tasks," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 830–835.

[10] L. Halt, "Reglersynthese für aufgabenraumgesteuerte industrieroboter," Dissertation, University of Stuttgart, Germany, 2022. [Online]. Available: http://elib.uni-stuttgart.de/handle/11682/12272

[11] M. Sharma, J. Liang, J. Zhao, A. LaGrassa, and O. Kroemer, "Learning to compose hierarchical object-centric controllers for robotic manipulation," *CoRR*, vol. abs/2011.04627, 2020. [Online]. Available: https://arxiv.org/abs/2011.04627

[12] J. Liang, M. Sharma, A. LaGrassa, S. Vats, S. Saxena, and O. Kroemer, "Search-based task planning with learned skill effect models for lifelong robotic manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6351–6357.

[13] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018. [Online]. Available: http://arxiv.org/abs/1801.01290

[15] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018. [Online]. Available: http://arxiv.org/abs/1802.09477

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[17] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[18] A. Lämmle, Z. Xiang, and B. A. Bálint, "Extension of established modern physics simulation for the training of robotic electrical cabinet assembly," *Procedia CIRP*, vol. 107, pp. 1317–1322, 2022.

[19] P. Rocco, "Stability of pid control for industrial robot arms," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 606–614, 8 1996.

[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html