# Reinforcement Learning for Adaptive Grasping of Objects by a Collaborative Robot: from Simulation to a Real System

Polina Lazareva[a)] and Nikolay Gurko[b)]

*Kazan National Research Technical University named after A. N. Tupolev – KAI (KNRTU-KAI), 10 Karl Marx Street, Kazan, 420111 Russia*
[a)] *Corresponding author: palazareva@kai.ru*
[b)] *nickolay73@hotmail.com*

**Abstract.** This study presents a method for training a collaborative robot to perform adaptive object grasping using reinforcement learning and a curriculum learning strategy, followed by transferring the resulting high-level policy from simulation to a real robotic system. A digital model of the Rozum Pulse 75 robot was created in Unity with ML-Agents, where the grasping task was decomposed into a sequence of progressively more complex subtasks involving coarse positioning, orientation alignment, and grasp execution. To bridge the gap between simulation and reality, the high-level policy operates on target poses of the end-effector, while low-level dynamics are handled by the robot's built-in controller. The trained policy achieved an 87% success rate in simulation and an 82% success rate on the physical robot, without additional fine-tuning. The results confirm that the proposed training pipeline provides stable learning in simulation and enables effective Sim2Real transfer for robotic manipulation tasks.

## INTRODUCTION

Modern robotic systems are increasingly used to solve problems that go beyond deterministic programming, where trajectories, interaction logic, and stopping conditions are predefined. In such scenarios – e.g., when manipulating objects of uncertain shape, working in unstructured environments, or interacting with humans – traditional control algorithms lack the required flexibility [1]. Reinforcement learning (RL) provides the ability to shape behavior based on the experience accumulated by the agent and allows the autonomous formation of a control strategy without rigid programming. However, this approach typically requires millions of training episodes, which, when executed on real equipment, increases the risk of damaging the robot and demands a significant amount of time. Therefore, in practice, the Sim2Real paradigm is widely used, wherein the control policy is trained in a virtual environment and then transferred to a real robot [2] – [4]. This approach reduces the cost of experimentation, accelerates training iterations, and enables large-scale variation of environment parameters that is difficult to achieve in real-world settings.

The key challenge in simulation-based training is the so-called Sim2Real gap [5], which arises due to discrepancies in dynamics, lighting, sensor noise, and object interactions between virtual and physical environments. Various methods have been proposed to mitigate this gap. The most common is domain randomization, which introduces random perturbations to simulation parameters according to a specified probability distribution at each training episode [6]. Characteristics such as length, mass, center of gravity, friction coefficients, and sensor noise are subject to randomization, forcing the agent to develop a robust policy capable of operating reliably under changing conditions. Another approach is system identification, aimed at accurately calibrating the physical parameters of the simulation for a specific robot. The use of experimental data and physically correct models can significantly increase the correspondence between the simulated and real systems [7]. In parallel, architectural solutions are being explored, including the modular separation of perception and control, test-time adaptation, and residual learning methods that correct simulation errors on the real robot. According to the survey in [8], combining these techniques—especially when using realistic simulators with sensor noise modeling – yields the most stable policy transfer.

Despite noticeable progress in reducing the Sim2Real gap, most existing solutions still require significant computational resources and extensive tuning. Furthermore, end-to-end RL approaches, which integrate machine vision systems directly into the control loop and train from raw images, tend to be difficult to interpret and challenging to correct when systematic perception errors occur. This motivates the development of modular systems in which the perception module provides the controller with structured features (e.g., 6D object pose), and reinforcement learning is used solely to develop motion strategies [9], [10].

The combination of computer vision and reinforcement learning methods within a unified control architecture is an actively developing research direction. Several studies propose modular schemes in which the perception module

outputs a compact representation – such as an object's pose or a set of key features – while the RL control module learns actions based on this information [11]. This approach results in faster convergence and improves the robustness of policy transfer from simulation to real hardware.

However, the modular principle introduces its own risks: errors in the perception module are directly propagated to the controller and influence the final behavior. When transferred to real environments, models encounter distribution mismatches in sensor data (particularly depth maps) and contact characteristics. To address this, domain adaptation and fine-tuning on a small amount of real data are commonly applied, but achieving consistently stable transfer remains an open problem.

The goal of this work is to develop and experimentally validate a method for teaching a collaborative 6-axis robot to grasp objects using RL and a curriculum learning approach in the Unity ML-Agents environment, followed by transferring the learned policy to a real system.

The robot control system employs a hierarchical structure: a high-level (strategic) layer responsible for synthesizing trajectories or sequences of target poses of the end-effector, and a low-level (control) layer, where motion is executed using the built-in controller of the real manipulator, with dynamic effects compensated through joint-level forces and torques.

This separation of layers allows the use of a purely kinematic description of the robot during training, without simulating full physical dynamics, including friction forces, link inertia, and contact interactions. This is justified by the fact that the reinforcement learning policy, which generates target states (positions and orientations) in task space, does not directly interact with link dynamics; instead, it issues target poses that are subsequently executed by the low-level controller, which performs trajectory smoothing, velocity limiting, and compensation for dynamic and frictional effects.

Thus, the gap between simulation and reality – stemming from discrepancies in dynamic characteristics of the simulated and physical robot – is significantly reduced, since the strategy is learned at the level of abstract target states rather than physical interactions. As a result, the primary requirement for the simulation model is accurate reproduction of the robot's geometry, kinematics, and mobility constraints, along with reliable representation of surrounding objects to ensure correct trajectory planning and collision avoidance.

The use of a simplified 3D model also improves the computational efficiency of training: it accelerates inverse kinematics computation, reduces training episode duration, and simplifies the implementation of domain randomization procedures that improve policy robustness.

It should be noted, however, that such a simplified physical model is applicable only when the agent does not directly control actuator torques or velocities. Tasks involving contact control, grasp dynamics, or force interactions require a complete physical model of the environment. In this work, the control strategy is trained at the level of kinematic trajectory formation, while the physical execution of the trajectory is handled by the hardware-level controller of the collaborative robot. This makes it possible to use only geometric and kinematic components in the simulation without a significant loss in policy generalizability during transfer to the real environment.

## FORMULATION OF THE REINFORCEMENT LEARNING TASK FOR A COLLABORATIVE ROBOT IN A SIMULATED OBJECT GRASPING SCENARIO

Training high-level behavioral strategies is formalized as a control problem in a Markov decision process with environment-dependent parameters. Let $\mathcal{S}$ denote the state space, $\mathcal{A}$ the high-level action space (target poses), and $\mathcal{P}_\phi\left(s' \mid s, a\right)$ the transition probability kernel parameterized by the domain parameter vector $\phi$ (simulation parameters such as the positions and orientations of the robot and the object). Let $r : \mathcal{S} \times \mathcal{A} \to R$ be the reward function, and $\gamma \in (0,1)$ the discount factor. The objective is to find the parameterization of a stochastic policy $\pi_\theta\left(a \mid s\right)$, with parameters $\theta$, that maximizes the expected discounted return:

$$J(\theta) = E_{\phi \sim \mathcal{D}_\phi} E_{\tau \sim \mathcal{P}_\phi, \pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r\left(s_t, a_t\right)\right], \tag{1}$$

where $\mathcal{D}_\phi$ is the distribution of domain parameters, and $\tau = \left(s_0, a_0, s_1, a_1, \ldots\right)$ is the trajectory generated by the dynamics $\mathcal{P}_\phi$ and policy $\pi_\theta$. The expectation over $\phi$ formalizes the idea of learning a robust policy capable of tolerating variability in physical conditions during sim-to-real transfer.

An observation $o_t$ is produced by the perceptual module as a combination of the object pose estimates and additional features:

$$o_t = \left( \hat{p}_t^{\text{obj}}, \hat{R}_t^{\text{obj}}, s_t^{\text{robot}} \right), \tag{2}$$

where $\hat{p}_t^{\text{obj}} \in R^3$ is the estimate of the object position in the robot base frame, $\hat{R}_t^{\text{obj}} \in SO(3)$ is the estimated orientation (rotation matrix), and $s_t^{\text{robot}}$ is the robot state: position and orientation of the tool center point (TCP) in the base frame.

A high-level action is defined as an increment of the TCP 6D pose:

$$a_t = (\Delta p_t, \Delta \psi_t), \tag{3}$$

where $\Delta p_t \in R^3$ is the position increment, and $\Delta \psi_t$ is the orientation increment in $SO(3)$. This decomposition delegates the implementation of dynamics and force compensation to the robot's low-level controller, while the high level is responsible for strategic trajectory planning.

Let the true object pose in the base frame be $\left( p_t^{obj}, R_t^{obj} \right)$ and its estimate be $\left( \hat{p}_t^{\text{obj}}, \hat{R}_t^{\text{obj}} \right)$. The position error is defined as:

$$e_p = p_t^{obj} - \hat{p}_t^{\text{obj}}. \tag{4}$$

The orientation error, expressed through rotation matrices, is given by the geodesic distance on $SO(3)$:

$$e_R = \arccos\left( \frac{tr\left( (R_t^{obj})^T \hat{R}_t^{obj} \right) - 1}{2} \right). \tag{5}$$

Such metrics are used in the formulation of the reward function and the success criteria for grasping.

## CURRICULUM LEARNING STRATEGY FOR ROBOTIC GRASPING VIA HIERARCHICAL REWARD SHAPING

The choice of a reward function is one of the most critical components of reinforcement learning, as it determines both the success and the stability of the agent's emerging behavior. The reward structure defines the direction of optimization and effectively specifies which strategies the agent perceives as "optimal." Even minor adjustments to the weighting of individual components or the form of penalties can lead to radically different learning dynamics – ranging from rapid convergence to stagnation in local optima, or even the development of undesirable behaviors that maximize the reward while failing to solve the intended task.

In robotic manipulation tasks, particularly when transferring a policy from simulation to the real world, the reward function must provide informative feedback during the early stages of training (to guide exploration), remain robust to perception noise, and promote physically plausible and reliable movements. Thus, the reward function acts as a formal encoding of the task objectives and constraints, directly influencing the quality and generalizability of the resulting policy.

The problem of generating an optimal motion trajectory and gripper orientation for object grasping involves several hierarchical subtasks: coarse positioning of the manipulator to approach the object, orienting the gripper at an angle suitable for grasping, and fine alignment required for a stable grasp. These subtasks differ in scale and in their influence on the final outcome, and they become relevant at different stages of the approach. This multi-level coupling of motion parameters makes classical single-stage reward engineering ineffective, even with regularization, as a single reward specification cannot adequately reflect the shifting priorities across different phases of interaction.

In such cases, a curriculum learning (CL) strategy is more appropriate, where the overall task is decomposed into sequential stages ordered by increasing complexity. For the object-grasping scenario, these stages may include: approaching the target without considering orientation, selecting an appropriate gripper orientation for the grasp,

performing precise positional and orientation alignment, and finally executing the grasp. This staged learning structure enables the agent to develop a hierarchy of sub-goals that matches the logic of the real manipulation process, resulting in more stable and focused policy formation – an aspect that becomes particularly important when transferring the trained model from a simulated environment to a physical robotic system.

To train an agent to execute precise end-effector TCP positioning and orientation alignment for grasping, we adopt a six-stage curriculum learning strategy. The curriculum is designed to decompose the end-to-end grasping problem into a sequence of progressively more demanding subtasks, each characterized by a tailored reward function and environmental configuration. The agent controls only the tool center point (TCP) pose (position and z-axis orientation) of the manipulator, while low-level joint control is abstracted. All stages are trained in simulation, with the success criterion defined as TCP–object proximity below a threshold coupled, in later stages, with angular alignment.

Let

- $\mathbf{p}_t \in \mathbb{R}^3$ be the TCP position at time step $t$,
- $\mathbf{q}_t \in \mathbb{R}^3$ the target object position,
- $d_t = \|\mathbf{p}_t - \mathbf{q}_t\|_2$ the Euclidean distance,
- $\hat{\mathbf{f}}_t = \mathrm{proj}_{xz}(\mathbf{R}_t \mathbf{e}_z) \in \mathbb{R}^2$ the TCP forward direction projected onto the horizontal $xz$-plane, normalized,
- $\hat{\mathbf{u}}_t = \mathrm{proj}_{xz}(\mathbf{q}_t - \mathbf{p}_t) / \|\mathrm{proj}_{xz}(\mathbf{q}_t - \mathbf{p}_t)\|_2 \in \mathbb{R}^2$ the unit vector from TCP to target in the $xz$-plane,
- $\phi_t = \left|\mathrm{angdiff}\left(\theta^z_{\mathrm{tcp},t}, \theta^z_{\mathrm{tgt},t}\right)\right| \in [0^\circ, 180^\circ]$ the absolute Z-axis Euler angle difference (in degrees),
- $\phi_{t-1}$ its value at the previous step,
- $\Delta t$ fixed timestep,
- $\lambda_p$, $\lambda_t$, $\lambda_d$, $\lambda_a$, $\lambda_l$ – positional, time, directional, angular, and object lift reward weight, respectively.

The step-wise reward at stage $k \in \{1,\ldots,5\}$ is:

$$r_t^{(k)} = r_{\mathrm{time}} + r_{\mathrm{dist}}^{(k)} + r_{\mathrm{dir}}^{(k)} + r_{\mathrm{angle}}^{(k)} \tag{6}$$

with terminal reward:

$$r_{\mathrm{term}}^{(k)} = \begin{cases} +5000, & \text{if success condition } \mathcal{S}^{(k)} \text{ is met,} \\ -500, & \text{if episode timed out,} \\ -10, & \text{if TCP collides with wall or workspace boundary,} \\ -5, & \text{if TCP collides with robot base.} \end{cases} \tag{7}$$

Below we detail each component per stage.

*Stage 1: Position-Only Convergence*

The goal is to learn coarse navigation to target, ignoring orientation. Success condition is defined as

$$\mathcal{S}^{(1)} := \{d_t < 0.10 \text{ m}\}. \tag{8}$$

Reward components:

$$\begin{aligned} r_{\mathrm{time}} &= \lambda_t \\ r_{\mathrm{dist}}^{(1)} &= \lambda_p^{(1)} \left(1 - \frac{d_t}{d_{\max}}\right)_+ \Delta t, \\ r_{\mathrm{dir}}^{(1)} &= 0, \\ r_{\mathrm{angle}}^{(1)} &= 0. \end{aligned} \tag{9}$$

Here $(x)_+ = \max(0, x)$. This inverse-distance scaling avoids vanishing gradients far from the target.

Randomization is limited: the object position is sampled uniformly within a bounded workspace, while TCP and object orientations are fixed. This stage establishes basic spatial navigation competence and mitigates early exploration collapse.

*Stage 2: Coarse Position–Orientation Alignment*

Upon mastering Stage 1, orientation is introduced through a directional reward term proportional to the cosine similarity between the TCP's forward direction (projected onto the horizontal XZ-plane) and the vector toward the target. The success distance threshold is tightened to $d < 0.06$ m, and a proximity-triggered (by $d_{\text{ang}}^{\text{thr}}$) angular bonus is granted when $d < 0.01$ and the angular deviation in z satisfies $|\Delta\phi_z| < 40°$. Both TCP and object initial z-rotations are randomized within $\pm 90°$. This forces the agent to begin coordinating translation and rotation without overwhelming policy capacity.

New reward components:

$$\begin{aligned} r_{\text{dir}}^{(2)} &= \lambda_d \cdot \left(\hat{\mathbf{f}}_t^{\top} \hat{\mathbf{u}}_t\right) \cdot \Delta t, \\ r_{\text{angle}}^{(2)} &= \mathbb{I}\{d_t < d_{\text{ang}}^{\text{thr}}\} \cdot \lambda_a \cdot \left(1 - \frac{\phi_t}{\phi_{\text{ang}}^{\text{thr}}}\right)_+ . \end{aligned} \tag{10}$$

*Stage 3: High-Precision Positioning under Full Rotational Randomization*

The distance success criterion is further tightened to $d < 0.01$ m, requiring near-contact positioning. Crucially, rotational randomization is extended to the full circle ($\pm 180°$), eliminating any exploitable bias in object or robot orientation. All reward coefficients from Stage 2 are retained, preserving the balance between positional and orientational objectives. This stage tests generalization under maximal rotational variability Success condition is defined as:

$$\mathcal{S}^{(3)} \coloneqq \{d_t < 0.01 \text{ m}\}. \tag{11}$$

This stage tests generalization under maximal orientation uncertainty, but does not yet require angular alignment for success.

*Stage 4: Tight Angular Tolerance for Pre-Grasp Pose*

The angular tolerance for intermediate and terminal success is drastically reduced: the angular bonus now activates only for $|\Delta\phi_z| < 10°$, and the success distance is set to $d < 0.003$ m (sub-millimeter precision). This reflects the physical requirements of successful grasp initiation – precise alignment being necessary to avoid collision or slippage. The increased angular specificity encourages the emergence of fine motor corrections during approach. Success condition is given as:

$$\mathcal{S}^{(4)} \coloneqq \{d_t < 0.003 \text{ m}\}. \tag{12}$$

Only the angular threshold is refined:

$$\phi_{\text{ang}}^{\text{thr}} = 10^{\circ}, \quad r_{\text{angle}}^{(4)} = \mathbb{I}\{d_t < 0.01\text{m}\} \cdot \lambda_a \cdot \left(1 - \frac{\phi_t}{10^{\circ}}\right)_+ . \tag{13}$$

This forces the agent to achieve near-perfect Z-axis alignment in the final approach phase.

*Stage 5: Joint Position–Orientation Success with Differential Angular Feedback*

This stage aims at unifying position and orientation into a conjunction success criterion and improve credit assignment via derivative-based angular reward. Success condition is defined as:

$$\mathcal{S}^{(5)} \coloneqq \left\{ d_t < 0.003 \text{ m} \wedge \phi_t < 10^{\circ} \right\}. \tag{14}$$

Notably, the intermediate angular reward is reformulated as a differential improvement signal:

$$r_{\text{angle}}^{(5)} = \mathbb{I}\{d_t < 0.001 \text{ m}\} \cdot \lambda_{\Delta\phi} \cdot (\phi_{t-1} - \phi_t). \tag{15}$$

It encourages reduction in angular error step-to-step – more robust to noise and sparse absolute alignment signals. Positional reward weight $\lambda_p^{(5)}$ is reduced to prevent dominance. The proximity gate for angular reward is narrowed to 1 mm, ensuring it only activates in the final pre-grasp phase.

*Stage 6: Grasping and Lifting (Full Task Execution)*

Final goal is to close the gripper at correct pose and lift the object to a target height – completing the full manipulation pipeline. New state variables are introduced:

$$h_t = \mathbb{I}\{g_t = 1 \wedge d_t < d_{\text{succ}} \wedge \phi_t < \phi_{\text{succ}} \wedge \text{no prior } h\}, \tag{16}$$

$$\tau_t = (t - t_{\text{hold-start}}) \cdot \mathbb{I}\{h_t = 1\}. \tag{17}$$

where $h_t \in \{0,1\}$ is hold flag, $g_t \in \{0,1\}$ – gripper state, $\tau_t$ – hold duration in seconds.

Gripper state reward is introduced:

$$r_{\text{grip}} = \begin{cases} +100.0, & \text{if } h_t \text{ transitions } 0 \to 1 \quad \text{(first successful grasp)}, \\ +5.0 \cdot \left(1 - \dfrac{\tau_t}{\tau_{\max}}\right), & \text{if } h_t = 1 \wedge \tau_t < \tau_{\max} = 2.0\text{s} \quad \text{(hold incentive)}, \\ +10.0, & \text{if } h_t = 1 \wedge \tau_t \geq \tau_{\max} \quad \text{(ready to lift)}, \\ -5.0, & \text{if } g_t = 1 \wedge \neg \mathcal{S}^{(5)} \quad \text{(premature close)}. \end{cases} \tag{18}$$

Lift reward is given only if $h_t = 1$ : let $z_t = p_t^z$, $z_{\text{tgt}} = q_t^z + \Delta z_{\text{lift}}$, $\Delta z_{\text{lift}} = 0.1$ m. Then:

$$r_{\text{lift}} = \begin{cases} 5000.0, & \text{if } |z_t - z_{\text{tgt}}| < 0.01\text{m} \quad \text{(terminal lift success)} \\ \left(1 - \dfrac{|z_t - z_{\text{tgt}}|}{0.1}\right) \cdot \lambda_l \cdot \Delta t, & \text{otherwise.} \end{cases} \tag{19}$$

Proximity/orientation rewards remain as in Stage 5, with angular multiplier increased (stronger focus on fine tuning).

Full success condition is given as:

$$\mathcal{S}^{(6)} : h_t = 1 \wedge \tau_t \geq 2.0\text{s} \wedge |z_t - (q_t^z + 0.1)| < 0.01 \text{ m}. \tag{20}$$

The 2-second hold requirement simulates grasp stability verification, preventing "grasp-and-drop" overfitting. The lift subtask tests dynamic consistency – the agent must maintain balance during upward acceleration.

This staged curriculum embodies the scaffolding principle [12]: early stages provide high-information-density rewards to bootstrap learning, while later stages incrementally introduce physical constraints (precision, alignment) reflective of real-world grasping. The transition from absolute to differential angular reward (Stage 5) constitutes a

reward shaping refinement specifically aimed at stabilizing fine-tuning – a strategy supported by recent work on curriculum-induced credit assignment (e.g., [13], [14]).

## CREATING A DIGITAL ROBOT MODEL IN UNITY

The Unity platform, combined with the ML-Agents framework, provides a flexible environment for visually realistic simulation and reinforcement learning of robotic agents. It supports simulation of physical interactions as well as the construction of complex visual scenes with configurable cameras, textures, and lighting. Unlike specialized physics-oriented simulators such as Isaac Sim or Gazebo, Unity offers strong advantages in visual data synthesis and sensor emulation, making it particularly suitable for tasks where perception plays a central role.

A digital robot model is created in Unity using an imported CAD representation in OBJ or FBX format, pre-segmented into individual links with correctly defined articulation points. In Unity, a hierarchy of *GameObjects* is constructed to reflect the manipulator's kinematic structure. Each link is assigned an *ArticulationBody* component, which specifies the joint type, rotation axis, and physical properties. Colliders are added to approximate link geometry to ensure accurate interaction with the physics engine. Kinematic relationships are refined using Denavit–Hartenberg parameters, after which an inverse kinematics module is implemented to compute joint angles corresponding to target TCP poses. This allows realistic reproduction of robot motion and enables the model to be used for reinforcement learning.

For the experimental evaluation of the proposed high-level policy learning and Sim2Real transfer approach, the Rozum Pulse 75 collaborative robot and its corresponding digital model shown in Fig. 1 were used. It is a 6-axis articulated robot with a pneumatic gripper attached. The task considered in this work is grasping bolts located within the robot's workspace. The perceptual module provides a 6-dimensional vector representing the bolt's position and orientation in the robot's base coordinate frame.

The values of the hyperparameters used are presented in Table 1, while the randomization parameters are listed in Table 2. Randomization of TCP angle and object orientation was performed according to the curriculum description given above. The values of the randomized initial parameters were selected considering the robot's workspace and the presence of spatial constraints.

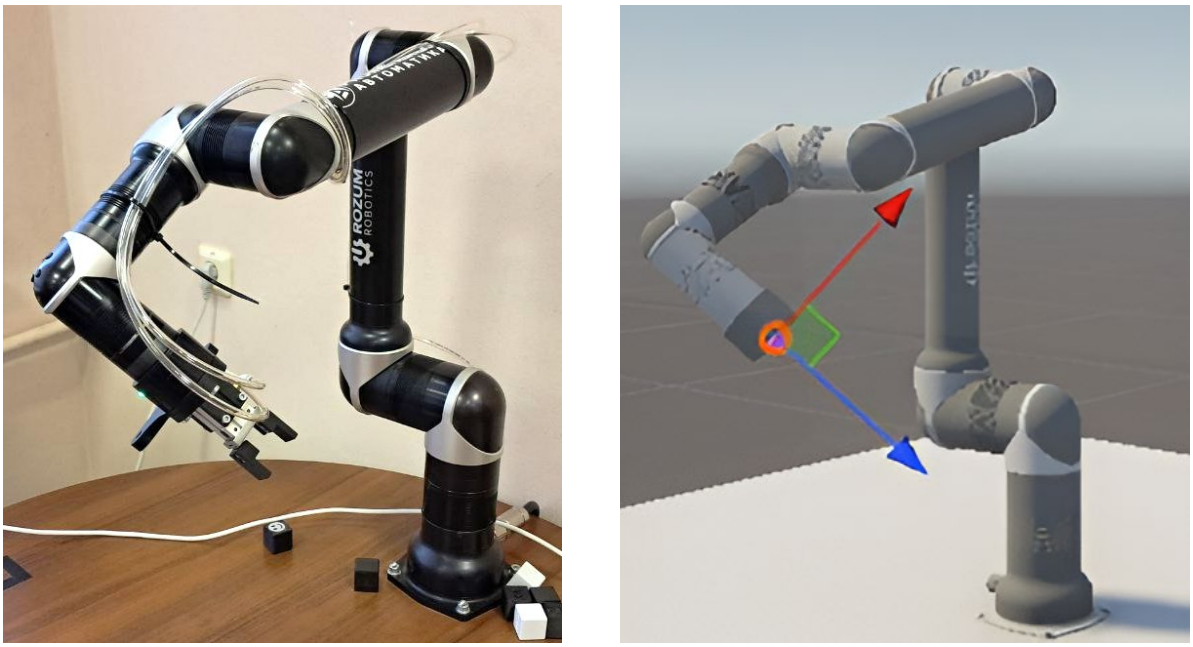
**TABLE 1.** Hyperparameter values

| Hyperparameter | Value | Purpose |
|---|---|---|
| $N_b$ | 10 | Number of samples per gradient update |
| $N_{buf}$ | 102,400 | Rollout buffer size (amount of collected experience before training) |
| $\alpha$ | 0.0003 | Learning rate |
| $\beta$ | 0.005 | Entropy regularization coefficient |
| $\varepsilon$ | 0.2 | PPO clipping parameter |
| $\lambda$ | 0.95 | GAE (Generalized Advantage Estimation lambda) |
| $\gamma$ | 0.99 | Discount factor |
| $H$ | 64 | Rollout horizon |

**TABLE 2.** Randomization parameters

| Parameter | Description | Range of values |
|---|---|---|
| TCP x-coordinate | Horizontal coordinate | [-0.5, 0.05] m |
| TCP y-coordinate | Lateral coordinate | [-0.5, 0.5] m |
| TCP z-coordinate | TCP height above the table | [0.06, 0.5] m |
| Target x-coordinate | Target horizontal coordinate | [-0.5, 0.05] m |
| Target y-coordinate | Target lateral coordinate | [-0.5, 0.5] m |
| Target z-coordinate | Target height above the table | [0.0001, 0.05] m |

PPO hyperparameters ($\epsilon$=0.2 $\lambda$=0.95, $\gamma$=0.99) are chosen as typical values for stable reinforcement learning with moderate exploration. A large batch size (10,240) and buffer size (102,400) help reduce stochastic gradient noise and improve update stability. A linear learning-rate schedule ensures a gradual reduction in update aggressiveness as training converges. The parameter *time_scale=20* accelerates training by increasing the simulation rate while preserving the physical consistency of the environment. The setting *max_steps=*$1 \cdot 10^8$ defines a sufficiently long training horizon, which is common for complex manipulation tasks.




(a) (b)

**FIGURE 1.** Collaborative robot Rozum Pulse 75: (a) real robot, (b) digital model in Unity

## EXPERIMENTAL RESULTS

According to the proposed curriculum learning strategy, the digital model of the Rozum Pulse 75 robot was trained in simulation using Unity and the ML-Agents framework. The target task consisted of reliably grasping bolts placed on the workspace surface. The curriculum was organized as a sequence of progressively more complex scenarios, which included increasing variability in bolt positions, orientations, and required approach trajectories.

During simulated training, the agent achieved a success rate of 87% after 67 million of episodes, demonstrating stable performance under the final stage of the curriculum. A trained policy was then transferred to the high-level Python control program of the physical robot, implementing a Sim2Real transfer without additional fine-tuning.

To evaluate real-world performance, 50 grasp attempts were conducted under conditions matching the simulated setup as closely as possible. The resulting success rate was 82%, indicating moderate performance degradation caused by unmodeled physical effects, but still demonstrating effective transfer of the learned policy.

A plot of the cumulative reward over the course of training (Fig. X) illustrates characteristic “drops” at the boundaries between curriculum stages. These discontinuities arise from changes in the reward structure and penalty coefficients introduced to reflect increased task difficulty. Nevertheless, the agent quickly adapted to each new stage, with the cumulative reward consistently recovering to its prior growth trajectory.

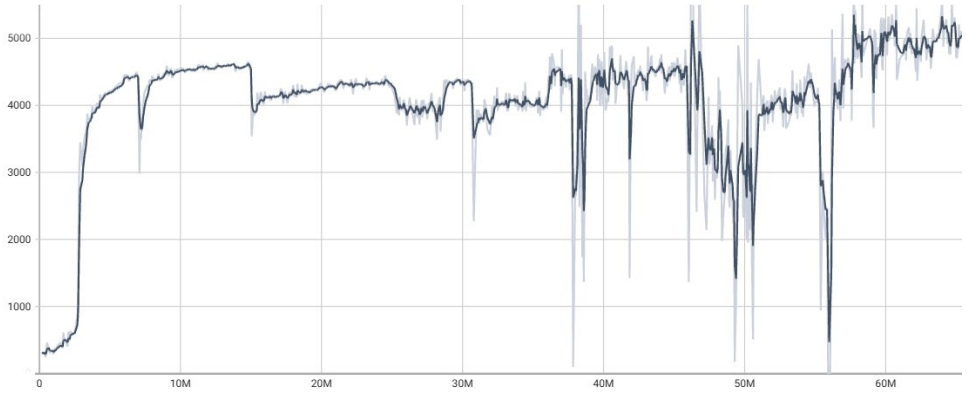



**FIGURE 2.** Cumulative reward function over millions of episodes for curriculum learning of a robot in the grasping scenario

Overall, the results confirm that the proposed curriculum-based training pipeline, combined with Sim2Real transfer, is effective for the robotic grasping task. The achieved performance indicates that the method provides both stable learning in simulation and robust transfer to the physical system.

## CONCLUSION

In this work, a reinforcement learning framework for adaptive grasping with a collaborative robot was developed and experimentally evaluated. The approach combines a hierarchical control architecture with a curriculum learning strategy, allowing the policy to be trained entirely at the kinematic level while delegating dynamic compensation to the robot's built-in controller. This design avoids the need for a fully accurate physical simulation and reduces the sensitivity of learning to modeling errors, which is particularly important for Sim2Real transfer.

A digital model of the Rozum Pulse 75 robot was implemented in Unity, and a six-stage curriculum was constructed to guide the agent from coarse positioning to full grasp execution. The staged structure proved essential for stabilizing learning: transitions between curriculum levels introduced temporary drops in cumulative reward, yet the agent consistently adapted to the new reward landscape and regained performance within a short number of iterations. The final policy demonstrated an 87% success rate in simulation.

The transfer of the learned policy to the real robot was carried out without additional tuning, relying on the abstraction of actions through target TCP poses. Testing on 50 real-world grasp attempts resulted in an 82% success rate, indicating that the policy retained its essential behavioral structure despite differences in perception, contact dynamics, and mechanical tolerances. The reduction in performance relative to simulation is consistent with known challenges of Sim2Real transfer, yet the overall level of success shows that the proposed training pipeline generalizes reliably to the physical system.

These results demonstrate that curriculum-based reinforcement learning, combined with a high-level control representation, offers a practical and robust solution for grasping tasks in collaborative robotics. The method provides a balance between simulation efficiency and real-world applicability, and can be extended to more complex manipulation scenarios, including multi-object handling, dynamic environments, and integration with more advanced perception modules.

## ACKNOWLEDGMENTS

The work was funded by the Strategic Academic Leadership Program of the Kazan National Research Technical University named after A.N. Tupolev ("PRIORITY-2030").

## REFERENCES

1. K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, IEEE Trans. Robot. **36**, 328–347 (2020).
2. Y. Chebotar et al., "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," in *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, Montreal, 2019), pp. 8973–8979.
3. A. Handa et al., "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," in *2023 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, London, 2023), pp. 5977–5984.
4. A. Allshire et al., "Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, Kyoto, 2022), pp. 11802–11809.
5. E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, IEEE Access **9**, 153171–153187 (2021).
6. J. Josifovski, M. Malmir, N. Klarmann, B. L. Žagar, N. Navarro-Guerrero, and A. Knoll, "Analysis of Randomization Effects on Sim2Real Transfer in Reinforcement Learning for Robotic Manipulation Tasks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, Kyoto, 2022), pp. 10193–10200.
7. J. Huber, F. Hélénon, H. Watrelot, F. B. Amar, and S. Doncieux, "Domain Randomization for Sim2real Transfer of Automatically Generated Grasping Datasets," in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, Yokohama, 2024), pp. 4112–4118.

8. W. Zhao, J. P. Queralta, and T. Westerlund, “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey,” in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, Canberra, 2020), pp. 737–744.
9. C. Wang et al., “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Long Beach, 2019), pp. 3338–3347.
10. Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun, “PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Seattle, 2020), pp. 11629–11638.
11. M. Yan, I. Frosio, S. Tyree, and J. Kautz, “Sim-to-Real Transfer of Accurate Grasping with Eye-In-Hand Observations and Continuous Control,” arXiv:1712.03303 (2017).
12. J. L. Elman, “Learning and development in neural networks: The importance of starting small,” Cognition **48**, 71–99 (1993).
13. C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, “Reverse Curriculum Generation for Reinforcement Learning,” in *Proceedings of the 1st Annual Conference on Robot Learning* (PMLR, 2017), pp. 482–495.
14. T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, “Teacher-Student Curriculum Learning,” IEEE Trans. Neural Netw. Learn. Syst. **31**, 3732–3740 (2020).