Journal of Machine Engineering, 2026, Vol. 26
ISSN 1895-7595 (Print) ISSN 2391-8071 (Online)

Received*: 20 April 2026* / Accepted: *28 May 2026* / Published online: *16 July 2026*

reinforcement learning,
industrial robots,
inverse kinematics

Karle NUTONEN[1*],
Vladimir KUTS[2,3],
Tauno OTTO[2]

GENERALIZED PROCESS SIMULATION OF HETEROGENEOUS INDUSTRIAL ROBOTS USING A SHARED MACHINE LEARNING MODEL

Industrial robot process simulation plays an important role in production planning, optimization, and digital twin development. However, many existing simulation approaches rely on robot-specific and manually parametrized models, which limits scalability in heterogeneous robot environments. This paper proposes a unified machine learning–based simulation framework for modelling and simulating process behaviour across different industrial robot platforms using a shared task-space learning policy. The proposed approach combines reinforcement learning and inverse kinematics with clearly separated roles. The reinforcement learning policy generates end-effector motion in task space, while inverse kinematics converts the desired motion into robot-specific joint configurations and supports geometric feasibility. This separation reduces the dependence of the learning model on the joint structure of a particular robot. Experimental evaluation on heterogeneous robot platforms showed that the proposed framework can learn accurate target-reaching behaviour in simulation. The best-performing policy achieved a success rate close to 1.0, a mean final distance error of 0.0082 m, and a mean final orientation error of 5.55°. These results indicate that a shared task-space learning policy combined with robot-specific inverse kinematics can support scalable robot process simulation.

1. INTRODUCTION

Industrial robots play a central role in modern manufacturing because they enable the automation of tasks requiring high precision, repeatability, and reliability, such as welding, assembly, material handling, and precision manipulation. Their use improves production efficiency, enhances quality, and reduces the risks associated with hazardous and repetitive work operations [1].

With the development of Industry 4.0 and smart manufacturing, the need for digital modelling, simulation, and virtual validation of production systems has increased. The concept of the digital twin enables a virtual representation of a physical system that can be

[1] Virumaa innovation centre of digitalisation and green technologies (ViruTech): Virumaa College, Tallinn University of Technology, Estonia
[2] Department of Mechanical and Industrial Engineering, Tallinn University of Technology, Estonia
[3] Faculty of Engineering, Universidad Autonoma de Bucaramanga, Colombia
[*] E-mail: karle.nutonen@taltech.ee
https://doi.org/10.36897/jme/222557

used to analyse, optimize, and predict its behaviour [2, 3]. In this context, robot simulation is important because it makes it possible to evaluate trajectories, detect collisions, and test control strategies before their deployment in a real system.

However, many existing simulation solutions are based on robot-specific models that depend on the kinematics, geometry, and control logic of a particular manipulator [3-5]. Although such an approach may ensure high accuracy, it limits scalability in heterogeneous manufacturing environments.

In recent years, machine learning methods, especially reinforcement learning, have become an important research direction in robot motion planning and control [6–8]. At the same time, RL-based approaches often depend on the joint-space representation and configuration space of a specific robot, which limits their ability to generalize across different robot platforms [7].

This work proposes a hybrid simulation framework in which a machine learning agent plans the robot tool trajectory in task space, while inverse kinematics solves the corresponding joint configurations for a specific robot. The aim of this approach is to reduce robot specificity, improve generalization across different robot platforms, and support the development of more scalable robot simulation solutions. Figure 1 illustrates the conceptual difference between traditional robot-specific simulation pipelines and the proposed unified task-space learning framework.

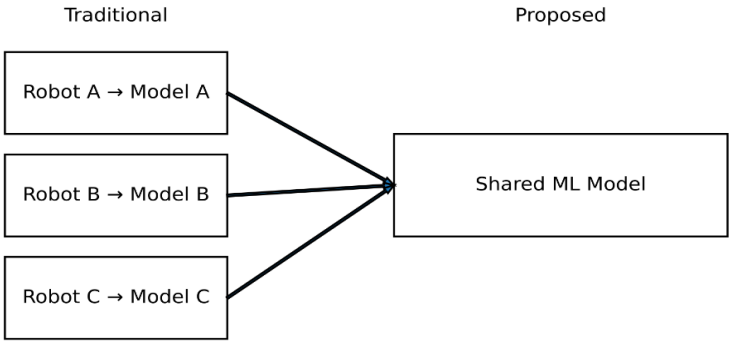

Fig. 1. Comparison between traditional robot-specific modeling and the proposed shared machine learning framework for multi-robot simulation

The main contributions of this work are threefold:

(1) a robot-agnostic task-space reinforcement learning formulation for industrial robot trajectory generation;

(2) an integrated simulation framework combining reinforcement learning and inverse kinematics; and

(3) an experimental evaluation demonstrating learning stability and transferability across multiple industrial robot platforms.

2. RELATED WORKS

Robot trajectory planning, simulation, and the application of machine learning in robotics have been long-standing research areas. From the perspective of this work, four

directions are particularly relevant: classical trajectory planning, reinforcement learning in robot control, hybrid machine learning and model-based approaches, and simulation combined with the digital twin concept.

Classical trajectory planning methods are commonly used to generate feasible robot motion paths in configuration spaces. Rapidly-exploring Random Tree (RRT) is a sampling-based planning method that incrementally builds a tree of possible robot configurations and is especially useful in high-dimensional spaces. RRT* is an asymptotically optimal extension of RRT, meaning that the quality of the generated path can improve as more samples are added. The Probabilistic Roadmap (PRM) method constructs a graph of collision-free configurations and then searches this graph for a feasible path between the start and goal states [9–11]. These methods are effective for finding collision-free trajectories in high-dimensional configuration spaces, but they are typically robot-specific because they operate directly in the configuration space of a particular manipulator.

Optimization-based trajectory planning methods further improve the quality of robot motion. Stochastic Trajectory Optimization for Motion Planning (STOMP) optimizes a trajectory by using sampled noisy trajectory variations and evaluating their costs. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) formulates motion planning as trajectory optimization and improves path smoothness and obstacle avoidance by minimizing a cost function [12, 13]. These approaches are useful for generating smooth and feasible robot trajectories, but they still mainly address geometric feasibility and usually do not provide a robot-independent learned policy.

Reinforcement learning (RL) has emerged as an important approach for learning robot control policies [7, 8]. *n* continuous action-space problems, algorithms such as Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) have been widely used. DDPG is an actor–critic reinforcement learning algorithm designed for continuous control tasks, where the policy directly outputs continuous actions. PPO is a policy-gradient method that improves training stability by limiting overly large policy updates during optimization [14, 15]. In this work, PPO was selected because it is well suited for continuous task-space control and provides stable learning behaviour in simulation-based robot training. At the same time, many RL-based methods operate at the joint-space or torque-space level, which often makes the learned policies robot-specific and difficult to transfer to other robot platforms [7].

Hybrid approaches combine machine learning and classical robotics methods in such a way that the learning component focuses on high-level decision-making or trajectory generation, while model-based components ensure motion feasibility [16–19]. Their main strength lies in reducing the complexity of the learning problem and making better use of structural knowledge. However, many existing solutions still remain centred on a specific robot or task.

Simulation environments play a central role in robot training, validation, and virtual testing. MuJoCo is a physics-based simulation environment widely used for model-based control, reinforcement learning, and the analysis of robotic systems with contact-rich dynamics. Gazebo is an open-source robotics simulator commonly used together with the Robot Operating System (ROS) for testing robot models, sensors, controllers, and multi-robot systems in realistic virtual environments. Unity is a general-purpose real-time 3D engine that

can be extended for robotics and reinforcement learning through the ML-Agents toolkit. In this context, Unity provides flexible scene construction, visual simulation, parallel training environments, and integration with learning-based control methods [20–22].

In this work, Unity was selected as the simulation platform because it allows the creation of randomized training scenes, supports parallel reinforcement learning environments, and integrates directly with the ML-Agents framework used for PPO-based policy training. This makes it suitable for evaluating the proposed task-space learning approach under varying target poses, obstacle placements, and robot configurations. To reduce the gap between simulation and real-world systems, domain randomization strategies are often used [23, 24]. In an industrial context, this is closely related to the concept of the digital twin, whose goal is to support system analysis, optimization, and prediction [25, 26]. Nevertheless, many existing solutions still rely on robot-specific models.

2.1. RESEARCH GAP AND MOTIVATION

Previous literature shows that classical methods provide strong geometric feasibility but do not directly support learning or transferability across different robots [9–13]. RL-based approaches offer greater flexibility, but they often remain dependent on the robot configuration [7, 14, 15]. Hybrid methods are promising, yet they mostly focus on specific robots or tasks [16-19].

Therefore, the number of solutions that enable the use of a single machine learning model for multiple different industrial robots remains limited. This work addresses this gap by combining a task-space RL policy and inverse kinematics into a unified simulation framework. Table 1 summarizes the key differences between representative classical, learning-based, and hybrid approaches, highlighting the gap addressed by the proposed framework.

Abbreviations: RRT – Rapidly-exploring Random Tree; RRT* – asymptotically optimal RRT; PRM – Probabilistic Roadmap; STOMP – Stochastic Trajectory Optimization for Motion Planning; CHOMP – Covariant Hamiltonian Optimization for Motion Planning; RL – reinforcement learning.

3. METHODOLOGY

This work proposes a hybrid robot simulation framework that combines reinforcement learning and inverse kinematics to enable the use of a single machine learning model across multiple different industrial robots. The central idea of the approach is to separate trajectory planning from the robot's kinematic realization: the agent plans the tool motion in task space, while inverse kinematics determines the joint configurations for a specific robot.

The proposed system consists of four main components: the simulation environment, the machine learning agent, the inverse kinematics solver, and the robot model. The agent generates the desired incremental tool motion in task space, while inverse kinematics converts

this motion into robot-specific joint configurations. The robot model then executes the resulting configuration in simulation. After the inverse kinematics step, the full robot configuration is evaluated for feasibility, including workspace limits, obstacle collisions, robot-body collisions, end-effector collisions, and self-collision cases.

Table 1. Comparison of representative motion planning and learning-based approaches

Method	Planning space	Learning-based	Robot-specific	Collision-aware	Cross-robot transfer
RRT [9]	Configuration space	No	Yes	Yes	No
RRT* [11]	Configuration space	No	Yes	Yes	No
PRM [10]	Configuration space	No	Yes	Yes	No
STOMP [12]	Trajectory/configuration space	No	Yes	Yes	No
CHOMP [13]	Trajectory/configuration space	No	Yes	Yes	No
Deep RL control [7, 14]	Joint space/action space	Yes	Yes	Limited	No
Visuomotor RL [5, 6]	Task-dependent control space	Yes	Yes	Limited	Limited
Sim-to-real RL [23, 24]	Usually joint or task space	Yes	Usually yes	Limited	Limited
Hybrid learning + model-based methods [16–19]	Mixed/hierarchical	Yes	Usually yes	Yes	Limited
Proposed framework	Task space + inverse kinematics	Yes	No	Yes	Yes

This division of responsibilities makes it possible to keep the learning policy as robot-independent as possible, while robot-specific feasibility is still checked at the simulation level (Fig. 2).

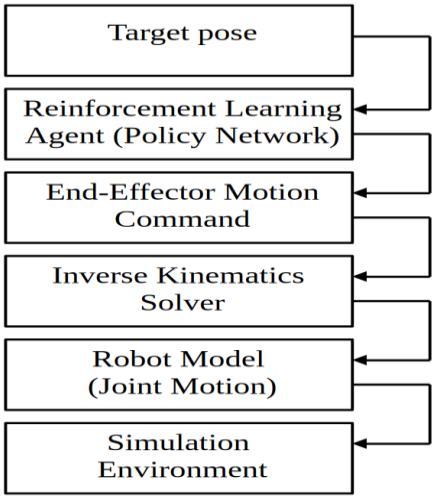

Fig. 2. System architecture of the proposed hybrid simulation framework integrating task-space reinforcement learning, inverse kinematics, and robot-specific execution

Training is conducted in a Unity-based physics simulation environment that includes the robot model, the target object, workspace boundaries, and obstacles. The coordinate axes shown in Fig. 3 define the task-space reference frame used for the end-effector motion. In Unity, the X and Z axes describe the horizontal workspace directions, while the Y axis describes the vertical direction. At the beginning of each episode, the position and orientation of the target object, the placement of obstacles, and the robot's initial configuration are randomized. This setup improves the generalization capability of the policy and supports the learning of a more robust strategy. Fig. 3 illustrates the main elements of the training environment, including the robot, target pose, workspace boundaries, and obstacles.

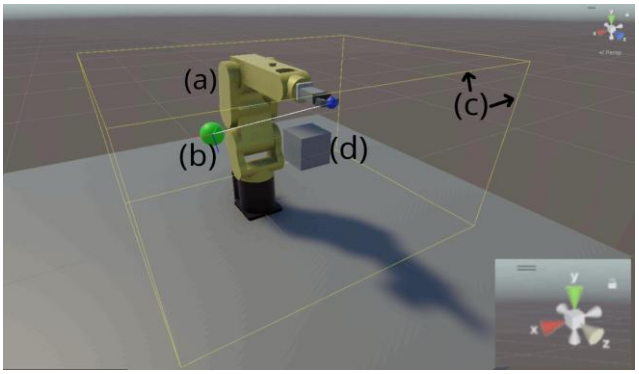

Fig. 3. Unity-based simulation environment used for task-space reinforcement learning: (a) industrial robot model, (b) target object or target pose, (c) workspace boundaries, and (d) randomly placed obstacles. The *X*, *Y*, and *Z* axes indicate the Unity task-space coordinate frame used for defining end-effector motion, where X and Z represent the horizontal workspace directions and *Y* represents the vertical direction

Robot trajectory planning is formulated as a Markov decision process. The state space includes the relative geometry of the tool and the target object, as well as environmental features such as obstacle locations. The action space is defined in task space and consists of incremental translational and orientation changes of the tool:

$$a_t=(\Delta x,\Delta y,\Delta z,\Delta roll,\Delta pitch,\Delta yaw) \tag{1}$$

The reward function encourages approaching the target, aligning the orientation, avoiding collisions, and successfully completing the task. Figure 4 summarizes the observation space, action representation, and reward structure.

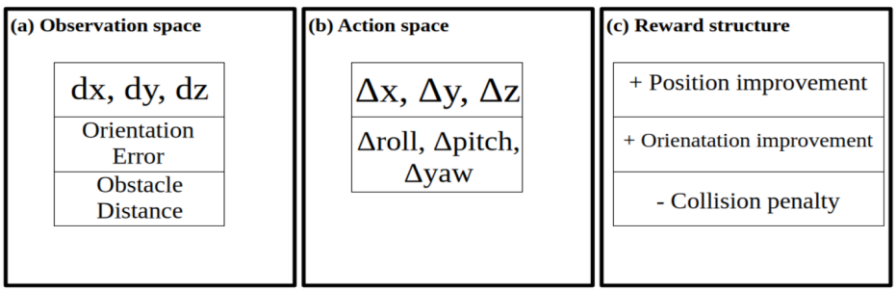

Fig. 4. Reinforcement learning formulation of the proposed framework: (a) observation space based on relative target and environment features, (b) task-space action representation for incremental end-effector translation and rotation, and (c) reward structure combining positional progress, orientation alignment, collision penalties, and success reward

The output of the machine learning model is converted into robot joint configurations using an inverse kinematics solver. In this work, the BioIK framework is used, as it makes it possible to account for tool position, orientation, and joint constraints. It serves as a bridge between the robot-independent task-space policy and the joint-level control of a specific robot.

3.1. COLLISION CHECKING AND ROBOT-BODY FEASIBILITY

This work presented a hybrid simulation framework that combines task-space reinforcement learning and inverse kinematics to support the use of a shared learning policy across heterogeneous robot platforms. As summarized in Fig. 5, the RL agent generates end-effector motion in task space, while inverse kinematics converts this motion into robot-specific joint configurations. Full-body feasibility is then checked in simulation using the available robot, gripper, workspace, and obstacle collision geometry.

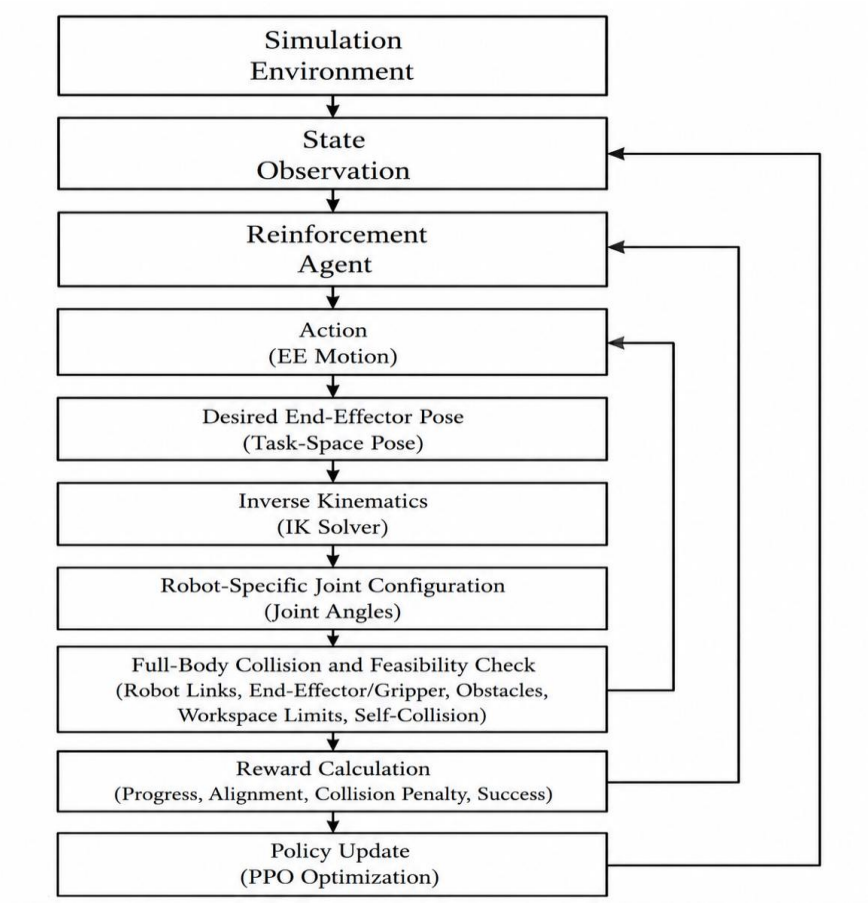

Fig. 5. Reinforcement learning pipeline of the proposed framework, including state observation, task-space end-effector action generation, inverse kinematics, robot-specific joint configuration, full-body collision and feasibility checking, reward calculation, and PPO policy update

The experimental results showed that the proposed framework can learn accurate target-reaching behaviour in simulation. In the best-performing configuration, the policy achieved a success rate close to 1.0, a mean final distance error of 0.0082 m, and a mean final orientation

error of 5.55°. A second evaluated model achieved a success rate of 0.9995, a mean final distance error of 0.0285 m, and a mean final orientation error of 12.87°. These results indicate that a shared task-space policy combined with robot-specific inverse kinematics can support scalable simulation across different robot platforms.

The main limitations are that the experiments were conducted in simulation, detailed collision-rate metrics were not separately logged, and real robot dynamics, sensor noise, delays, and process-specific gripper geometries were not experimentally evaluated. Future work should focus on sim-to-real transfer, more detailed dynamic constraints, quantitative feasibility metrics, and validation on additional industrial scenarios.

4. EXPERIMENTAL SETUP

The objective of the experiments in this study was to evaluate whether the proposed hybrid simulation framework can learn the robot tool trajectory in task space in such a way that the target position and orientation are reached while avoiding undesirable collisions. It was also evaluated separately whether the same control policy can be applied to multiple different industrial robots without using an action space that depends on the number of robot joints.

The experimental environment was implemented in the Unity game engine using the ML-Agents framework and the PPO algorithm. Unity was used to construct the 3D training scene, including the robot model, target object, workspace boundaries, and obstacles. The ML-Agents framework provided the interface between the Unity simulation and the reinforcement learning algorithm. It was used to define observations, task-space actions, rewards, episode termination conditions, and parallel training environments. PPO was used as the policy optimization algorithm.

Each training episode took place in a bounded workspace containing the robot base, the initial tool position, the target object, and obstacles. The position and orientation of the target object, as well as the placement of obstacles, were randomized between episodes in order to improve the generalization capability of the policy. Fig. 3 illustrates the training environment.

The agent's action vector consisted of six components describing the incremental translational and orientation changes of the tool in task space:

$$a_t=(\Delta x,\Delta y,\Delta z,\Delta\phi,\Delta\theta,\Delta\psi) \tag{2}$$

All actions were normalized to the range [−1, 1] and scaled according to the maximum translational and rotational step sizes. The action-scaling parameters were adjusted during training in order to support a coarse-to-fine learning strategy. In the initial training phase, one action step corresponded to a maximum translational change of 0.3 m along each Unity task-space axis and a maximum rotational change of 80° around each orientation axis. In the later training phase, the scaling was changed to a maximum translational change of 0.6 m and a maximum rotational change of 20° per action step. This adjustment allowed the policy to maintain sufficient translational motion range while reducing the maximum rotational change for more stable final orientation alignment. The observation vector included the relative

geometry of the tool and the target, as well as environmental features, but not the robot joint angles, in order to keep the policy as robot-independent as possible.

The PPO algorithm was used for training for up to 2,000,000 steps. The simulation used 25 parallel training environments to increase data collection efficiency and improve convergence. The main hyperparameters, including the learning rate, batch size, buffer size, and discount factor, are presented in Table 2.

At the beginning of each episode, the initial tool position was reset and a new target pose was generated while avoiding overly simple initial conditions. In addition, the target orientation and obstacle placement were randomized. This strategy reduced the risk of overfitting and supported the development of a more robust policy.

The reward function combined positional progress, reduction of orientation error, collision penalties, and rewards for successfully completed episodes. An additional fine-shaping component was used near the target to improve final alignment accuracy. Fig. 4(c) summarizes the reward structure.

Collision handling was included in the simulation loop after the inverse kinematics step. The computed joint configuration was checked against the robot body collision geometry, obstacles, workspace limits, and the end-effector collision model. Collision events were penalized in the reward function, and severe collisions or invalid configurations terminated the episode. This allowed the policy to learn task-space motions that were not only directed toward the target pose but also feasible for the robot model used in the simulation.

The generalization capability of the framework was evaluated on three heterogeneous robot platforms: the Fanuc LR Mate 200iD/4S, the Fanuc CRX-10iA, and the ABB CRB 15000 GoFa 5 kg. These robots were selected because of their different geometries, workspaces, and application profiles. Their main specifications are presented in Table 3.

Table 2. PPO hyperparameters and training settings used in the experiments

Parameter	Value
Trainer	PPO
Learning rate	3e-4
Batch size	2048
Buffer size	40960
Gamma	0.99
Lambda	0.95
Clipping epsilon	0.2
Hidden units	256
Number of hidden layers	2
Max training steps	2,000,000
Time horizon	256
Parallel environments	25

Table 3. Evaluated heterogeneous robot platforms used for cross-platform transferability assessment

Robot platform	DOF	Reach	Payload	Repeatability	Role in evaluation
Fanuc LR Mate 200iD/4S	6	550 mm	4 kg	+/-0.01mm	compact industrial manipulator
Fanuc CRX-10iA	6	1249 mm	10 kg	+/-0.04 mm	collaborative industrial robot
ABB CRB 15000 GoFa 5kg	6	950 mm	5 kg	+/-0.02 mm	collaborative robot for transfer evaluation

This heterogeneous selection supports the main objective of the study: to investigate whether a shared task-space learning model can be applied across different robot platforms without defining a separate joint-space action representation for each robot. The comparison should therefore be interpreted as a cross-platform transferability evaluation, not as a performance comparison between industrial robots and cobots.

5. RESULTS AND DISCUSSION

This section analyses the performance of the proposed hybrid machine learning and inverse kinematics–based simulation framework on the basis of training logs and final evaluation metrics. The evaluation is not intended to compare the selected robots and cobots as machines designed for the same industrial task. Instead, the selected platforms are used to test whether the proposed task-space policy can be executed across heterogeneous robot geometries through robot-specific inverse kinematics.

The training process showed stable convergence for both best-performing training runs. As shown in Fig. 6(a), the cumulative reward increased during training, indicating that the policy gradually learned actions that improved task completion. At the same time, Fig. 6(b) shows that the episode length decreased from the maximum episode length toward shorter episodes, which indicates that the target pose was reached with fewer simulation steps as training progressed.

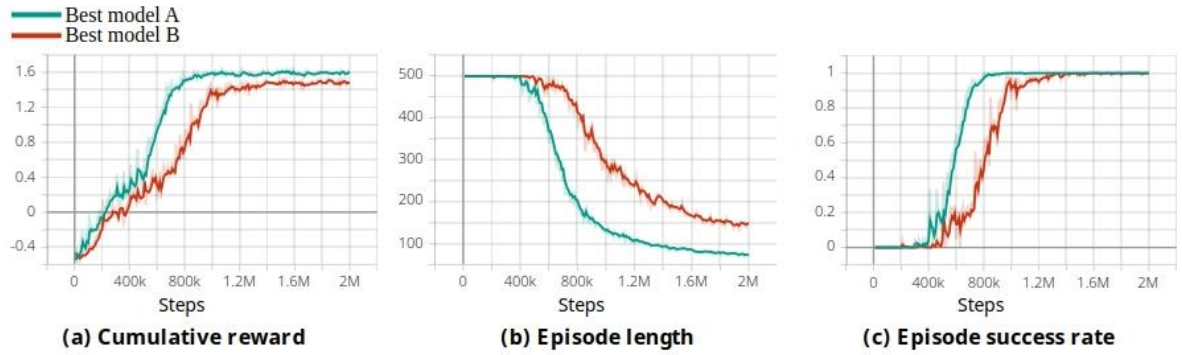

Fig. 6. Training convergence of the proposed framework during PPO optimization for the two best-performing training runs: (a) cumulative reward, (b) episode length, and (c) episode success rate over training steps. The cumulative reward and success rate increase during training, while the episode length decreases, indicating that the policy learns to reach the target pose more efficiently and reliably

Fig. 6(c) shows that the episode success rate increased and approached 1.0 in the final phase of training. These trends support the conclusion that the policy learned a more efficient and reliable task-space control strategy.

The PPO learning dynamics supported this trend: policy entropy decreased during training, indicating that initially exploratory behaviour became more stable, while the extrinsic reward increased consistently with the task-performance improvements shown in Fig. 6.

The development of final pose accuracy is shown in Fig. 7, while the corresponding quantitative evaluation results of the best-performing policies are summarized in Table 4. As shown in Fig. 7(a), the final distance error decreased during training for both best-performing runs. Fig. 7(b) shows a similar decreasing trend in the final orientation error. These trends indicate that the policy did not only learn to complete the task more frequently, but also improved the accuracy of the final end-effector pose. This is further confirmed by the quantitative results in Table 4, where Best model A achieved a mean final distance error of 0.0082 m and a mean final orientation error of 5.55°, while Best model B achieved a mean final distance error of 0.0285 m and a mean final orientation error of 12.87°.

Table 4. Quantitative final evaluation results of the best-performing policies, including success rate, final pose error, and episode length

Model	Success rate	Mean end distance (m)	Mean end angle (deg)	Mean episode length
Best model A	~1	0.0082	5.5473	74.9877
Best model B	0.9995	0.0285	12.8671	148.6766

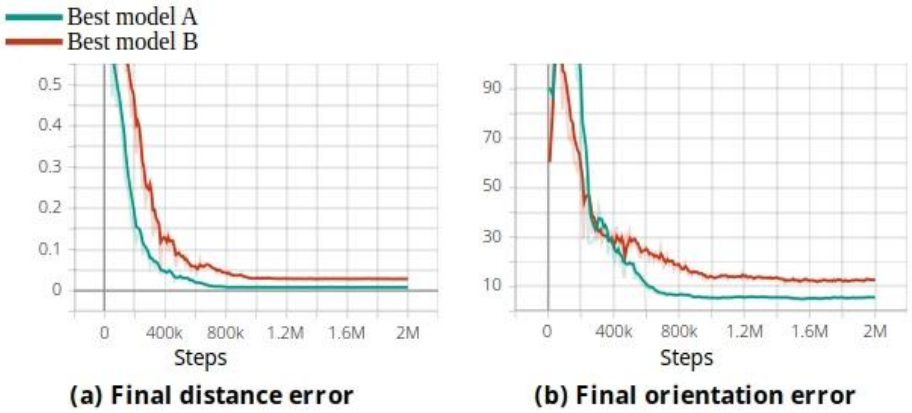

Fig. 7. Final pose-error convergence of the two best-performing training runs: (a) final distance error and (b) final orientation error over training steps. Both error measures decrease during training, showing that the policy improves not only target-reaching success but also final pose accuracy

As shown in Fig. 7(a), the final distance error decreased during training and stabilized at 0.0082 m for Best model A and 0.0285 m for Best model B, as reported in Table 4. Figure 7(b) shows that the final orientation error also decreased during training and reached 5.55° for Best model A and 12.87° for Best model B. These values indicate that the policy improved both positional and orientational alignment, rather than only increasing the episode success rate. Such behaviour is expected in tasks where reaching the target pose requires both

rapid translational approach and more precise rotational adjustment. From a practical perspective, this is important because in industrial robot applications it is not sufficient merely to reach the vicinity of the target position; the correct orientation of the tool is also required.

The final evaluation results are summarized in Table 4. Best model A achieved a success rate close to 1.0, a mean final distance error of 0.0082 m, a mean final orientation error of 5.55°, and a mean episode length of 74.99 steps. Best model B achieved a success rate of 0.9995, a mean final distance error of 0.0285 m, a mean final orientation error of 12.87°, and a mean episode length of 148.68 steps. These results show that both policies learned successful target-reaching behaviour, while Best model A achieved lower final pose error and shorter episodes.

Obstacle avoidance and self-collision penalties were included during training, but detailed feasibility metrics such as collision rate, self-collision rate, IK failure rate, and workspace violation rate were not separately logged. This is a limitation of the current evaluation and should be addressed in future work. Nevertheless, collision feedback was included after the inverse kinematics step, allowing obstacle contacts, invalid configurations, and robot-body collisions to influence policy learning through the reward function.

6. CONCLUSION AND FUTURE WORK

This work presented a hybrid simulation framework that combines task-space reinforcement learning and inverse kinematics to support the use of a shared learning policy across heterogeneous robot platforms. As summarized in Fig. 5, the RL agent generates end-effector motion in task space, while inverse kinematics converts this motion into robot-specific joint configurations. Full-body feasibility is then checked in simulation using the available robot, gripper, workspace, and obstacle collision geometry.

The experimental results showed that the proposed framework can learn accurate target-reaching behaviour in simulation. In the best-performing configuration, the policy achieved a success rate close to 1.0, a mean final distance error of 0.0082 m, and a mean final orientation error of 5.55°. A second evaluated model achieved a success rate of 0.9995, a mean final distance error of 0.0285 m, and a mean final orientation error of 12.87°. These results indicate that a shared task-space policy combined with robot-specific inverse kinematics can support scalable simulation across different robot platforms.

The main limitations are that the experiments were conducted in simulation, detailed collision-rate metrics were not separately logged, and real robot dynamics, sensor noise, delays, and process-specific gripper geometries were not experimentally evaluated. Future work should focus on sim-to-real transfer, more detailed dynamic constraints, quantitative feasibility metrics, and validation on additional industrial scenarios.

ACKNOWLEDGEMENTS

This article thesis was supported by the project "Increasing the knowledge intensity of Ida-Viru entrepreneurship" co-funded by the European Union (IKRA-T5.0 – Development of process-adaptable robot platforms in the Industry 5.0 concept (incl. digital twin)).

REFERENCES

[1] ZHANG Z., ZENG J., 2022, *A Review on Development and Application of Industrial Robot*, Acad. J. Sci. Technol., 2/2, 78–81, https://doi.org/10.54097/ajst.v2i2.1165.

[2] ONAJI I., TIWARI D., SOULATIANTORK P., SONG B., TIWARI A., 2022, *Digital Twin in Manufacturing: Conceptual Framework and Case Studies*, Int. J. Comput. Integr. Manuf., 35/8, 831–858, https://doi.org/10.1080/0951192X.2022.2027014.

[3] UGARTE M., ETXEBERRIA L., UNAMUNO G., BELLANCO J.L., UGALDE E., 2022, *Implementation of Digital Twin-Based Virtual Commissioning in Machine Tool Manufacturing*, Procedia Comput. Sci., 200/1–2, 527–536, https://doi.org/10.1016/j.procs.2022.01.250.

[4] GALAMBOS P., et al., 2015, *Design, Programming and Orchestration of Heterogeneous Manufacturing Systems Through VR-Powered Remote Collaboration*, Robot. Comput. Integr. Manuf., 33, 68–77, https://doi.org/10.1016/j.rcim.2014.08.012.

[5] LEVINE S., PASTOR P., KRIZHEVSKY A., IBARZ J., QUILLEN D., 2018, *Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection*, Int. J. Rob. Res., 37/4–5, 421–436, https://doi.org/10.1177/0278364917710318.

[6] LEVINE S., FINN C., DARRELL T., ABBEEL P., 2016, *End-to-end Training of Deep Visuomotor Policies*, J. Mach. Learn. Res., 17, 1–40.

[7] KOBER J., BAGNELL J.A., PETERS J., 2013, *Reinforcement Learning in Robotics: a Survey*, Int. J. Rob. Res., 32/11, 1238–1274, https://doi.org/10.1177/0278364913495721.

[8] VU M.N., BECK F., SCHWEGEL M., HARTL-NESIC C., NGUYEN A., KUGI A., 2023, *Machine Learning-Based Framework for Optimally Solving the Analytical Inverse Kinematics for Redundant Manipulators*, Mechatronics, 91, 102970, https://doi.org/10.1016/j.mechatronics.2023.102970.

[9] LAVALLE S.M., KUFFNER J.J., 2001, *Randomized Kinodynamic Planning*, Int. J. Rob. Res., 20/5, 378–400, https://doi.org/10.1177/02783640122067453.

[10] KAVRAKI L.E., SVESTKA P., LATOMBE J.C., OVERMARS M.H., 1996, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE Trans. Robot. Autom., 12/4, 566–580, 1996, https://doi.org/10.1109/70.508439.

[11] KARAMAN S., FRAZZOLI E., 2011, *Sampling-Based Algorithms for Optimal Motion Planning*, Int. J. Rob. Res., 30/7, 846–894, https://doi.org/10.1177/0278364911406761.

[12] KALAKRISHNAN M., CHITTA S., THEODOROU E., PASTOR P., SCHAAL S., 2011, *STOMP*: *Stochastic Trajectory Optimization for Motion Planning*, Proc. - IEEE Int. Conf. Robot. Autom., 4569–4574, https://doi.org/10.1109/ICRA.2011.5980280.

[13] RATLIFF N., ZUCKER M., ANDREW BAGNELL J., SRINIVASA S., 2009, *CHOMP: Gradient Optimization Techniques for Efficient Motion Planning*, Proc. - IEEE Int. Conf. Robot. Autom., 489–494, https://doi.org/10.1109/ROBOT.2009.5152817.

[14] LILLICRAP T.P., et al., 2015, *Continuous Control with Deep Reinforcement Learning*, 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., https://arxiv.org/pdf/1509.02971.

[15] SCHULMAN J., WOLSKI F., DHARIWAL P., RADFORD A., OPENAI O.K., 2026, *Proximal Policy Optimization Algorithms*, https://arxiv.org/pdf/1707.06347.

[16] TOUSSAINT M., 2009, *Robot Trajectory Optimization Using Approximate Inference*.

[17] TASSA Y., EREZ T., TODOROV E., 2012, *Synthesis and Stabilization of Complex Behaviors Through Online Trajectory Optimization*, IEEE Int. Conf. Intell. Robot. Syst., 4906–4913, https://doi.org/10.1109/IROS.2012.6386025.

[18] KALAKRISHNAN M., BUCHLI J., PASTOR P., MISTRY M., SCHAAL S., 2011, *Learning, Planning, and Control for Quadruped Locomotion Over Challenging Terrain*, Int. J. Rob. Res., 30/2, 236–258, https://doi.org/10.1177/0278364910388677.

[19] KUMAR A., FU Z., PATHAK D., MALIK J., 2026, *RMA: Rapid Motor Adaptation For Legged Robots*, Robot. Sci. Syst., http://arxiv.org/abs/2107.04034.

[20] TODOROV E., EREZ T., TASSA Y., 2012, *Mujoco: A Physics Engine for Model-Based Control*, IEEE Int. Conf. Intell. Robot. Syst., 5026–5033, 2012, https://doi.org/10.1109/IROS.2012.6386109.

[21] KOENIG N. HOWARD A., 2004, *Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator*, 2004 IEEE/RSJ Int. Conf. Intell. Robot. Syst., 3, 2149–2154, https://doi.org/10.1109/iros.2004.1389727.

[22] Unity-Technologies/ml-agents: Available: https://github.com/Unity-Technologies/ml-agents, Accessed: Jan. 12, 2022.

[23] TOBIN J., FONG R., RAY A., SCHNEIDER J., ZAREMBA W., ABBEEL P., 2017, *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*, IEEE Int. Conf. Intell. Robot. Syst., 23–30, https://doi.org/10.1109/IROS.2017.8202133.
[24] JAMES S., DAVISON A.J., JOHNS E., 2026, *Transferring end-to-end Visuomotor Control from Simulation to Real World for a Multi-Stage Task*, http://arxiv.org/abs/1707.02267.
[25] GRIEVES M., VICKERS J., 2017, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, Transdiscipl. Perspect. Complex Syst. New Find. Approaches, 85–113, https://doi.org/10.1007/978-3-319-38756-7_4.
[26] Tao F., Xiao B., Qi Q., Cheng J., Ji P., 2022, *Digital Twin Modeling*, J. Manuf. Syst., 64/7775, 372–389, https://doi.org/10.1016/j.jmsy.2022.06.015.