The 13th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
4-6 September, 2025, Gliwice, Poland

Sim-to-Sim Control of a UR3e Robot Using Soft Actor-Critic and Real-Time MATLAB–ROS–URSim Integration

Ahmed Iqdymat, Grigore Stamatescu
Email: ahmed.iqdymat@stud.acs.upb.ro; grigore.stamatescu@upb.ro
Department of Automation and Industrial Informatics, Faculty of Automatic Control and Computers,
University Politehnica of Bucharest, Bucharest, Romania

***Abstract*— This paper introduces a real-time control framework based on Soft Actor-Critic (SAC) reinforcement learning, tailored for a 6-DOF UR3e collaborative robotic manipulator performing a simulated pick-and-place task. The training process is conducted in MATLAB/Simulink, Reinforcement Learning Toolbox and deployed via the Robot Operating System (ROS) to the URSim environment. The control system integrates MATLAB (Windows host) with URSim (Linux virtual machine) through ROS topics for action and feedback exchange. After 395 training episodes, the learned policy achieved smooth Cartesian motion and stable joint coordination throughout the pick-and-place cycle. The resulting end-effector trajectory confirms task segmentation and motion smoothness across approach, pick, transport, and place phases. These results demonstrate the effectiveness of SAC for precise manipulator control in simulated industrial environments.**

Keywords*— *Reinforcement Learning, Soft Actor-Critic, UR3e Robot, MATLAB, ROS, URSim, Pick and Place, trajectory tracking.

I. INTRODUCTION

A. Background and Motivation

Robotic manipulators such as the 6-DOF UR3e arm are increasingly adopted in automation scenarios that demand precision, repeatability, and adaptability. In pick and place applications, where the robot must operate under dynamic object positions or uncertain execution timing, conventional control methods like PID or LQR often fall short in terms of flexibility [1],[2]. This challenge motivates the integration of data-driven approaches, such a reinforcement learning, which enable learning-based adaptation directly from environmental feedback.

B. Problem Statement

Deploying reinforcement learning in a functional control system is not just an algorithmic challenge. The practical difficulty lies in the integration layers communication, middleware, and timing that surround the learning agent. These aspects are often overlooked in RL studies. In our case, the control architecture includes MATLAB for training and policy execution, ROS as the middleware, and URSim as the robot simulator. Synchronization issues, latency, and message formatting between these components can pose significant barriers to real-time control.

With the UR3e manipulator, the objective is not limited to offline policy evaluation. The SAC agent must generate joint-space trajectories in real time, transmit them over ROS, and ensure they are executed by the simulated robot without delay or drift. This work focuses precisely on that challenge: achieving reliable closed-loop control from policy inference to motion execution in a fully integrated simulation pipeline.

C. Proposed Framework and Implementation Scope.

The proposed system builds upon previous reinforcement learning research and introduces a real-time SAC-based control architecture tailored to the UR3e robotic manipulator. Unlike earlier work that focused on joint tracking with the ABB IRB120, this framework addresses new challenges introduced by UR3e's expanded kinematic capabilities and real-time communication requirements. The trained policy operates within a synchronized control pipeline that combines MATLAB, ROS, and URSim, enabling live trajectory updates through ROS topics. This configuration allows the agent to learn and execute coordinated pick-and-place motions under realistic simulation conditions. The study focuses on evaluating the technical feasibility, stability, and future deployment potential of the trained policy on physical UR3e hardware.

II. RELATED WORK

A. Classical and Learning-Based Control Methods.

Industrial robots are commonly controlled using predefined strategies such as PID or LQR, which offer consistent performance when system dynamics are well understood [1],[2]. However, these methods tend to degrade when applied to tasks involving variable contact, nonlinearities, or uncertain object positions scenarios frequently encountered in collaborative settings.

Reinforcement learning (RL) offers a model-free alternative by allowing the system to improve through interaction. While approaches like DDPG and PPO have shown potential in simulated manipulation tasks [3]– [5], their use in real-time continuous control remains constrained by sample inefficiency and sensitivity to design

choices. Unlike many prior works that rely on idealized or task-specific assumptions, this study investigates whether Soft Actor-Critic (SAC) can produce stable joint-space motion on the UR3e under realistic simulation constraints using ROS and URSim.

B. Soft Actor-Critic and Interaction Control.

Soft Actor-Critic (SAC) has become a state-of-the-art approach for continuous control problems due to its entropy-driven policy updates and training stability [6]. It has been extended to offline learning [7], replay-based memory efficiency [8], and robust constrained optimization [9]. SAC has also been benchmarked for precision control using real-world robotic platforms [10] and applied in force-sensitive tasks involving real UR3e manipulators [11]. In parallel, researchers have explored integrating reinforcement learning with impedance control to enable safe and adaptive interaction. These hybrid approaches adjust stiffness and damping in response to external feedback or object properties, as seen in work on object-aware impedance adaptation [12]. Such strategies mark a shift toward compliant and interaction-aware policies, particularly relevant in collaborative robotics.

C. Integrated Control Pipelines and Gap Identification

While ROS and Gazebo are widely adopted for robotic learning pipelines, the integration of MATLAB and URSim remains underrepresented in the literature. Although MATLAB supports ROS communication and reinforcement learning tools, most implementations are offline or unsuitable for real-time control [13], [14]. Recent studies have explored RL-based motion planning for industrial manipulators [15], control through MATLAB–ROS–URSim pipelines remains largely unexplored. This gap is especially relevant for industrial tasks like pick and place, where execution timing and feedback responsiveness are critical. Additional relevant contributions include the use of actor-critic reinforcement learning for energy-efficient control in industrial environments [16], and sensor data correction techniques for improved robot perception [17]. The present study addresses this gap by introducing a complete SAC-based framework for the UR3e, trained in MATLAB and deployed using a synchronized MATLAB–ROS–URSim setup.

A preliminary version of this system was presented in [18], where the SAC-based policy was applied to the UR3e in a real-time ROS–Simulink setup. This study extends that work with a complete training framework and additional evaluation layers under the MATLAB–ROS–URSim environment [19].

III. System Architecture

The proposed control architecture establishes a real-time communication loop between a Soft Actor-Critic (SAC) agent trained in MATLAB/Simulink and a simulated UR3e robotic arm operating in URSim. The SAC agent runs on a Windows host, where it generates continuous joint-space trajectories. Meanwhile, URSim executes these commands within a Linux-based virtual machine.

The Robot Operating System (ROS) middleware layer connects the two platforms; The system allows messages to be sent and received in both directions using topics and services. This closed-loop configuration facilitates synchronized action observation cycles during simulation. allowing the agent to adapt its behavior based on feedback.

The overall setup is aligned with official integration practices from MathWorks [20] and Universal Robots [21], providing a robust foundation for testing learning-based controllers in simulation. A structural overview is shown in Figure 1.

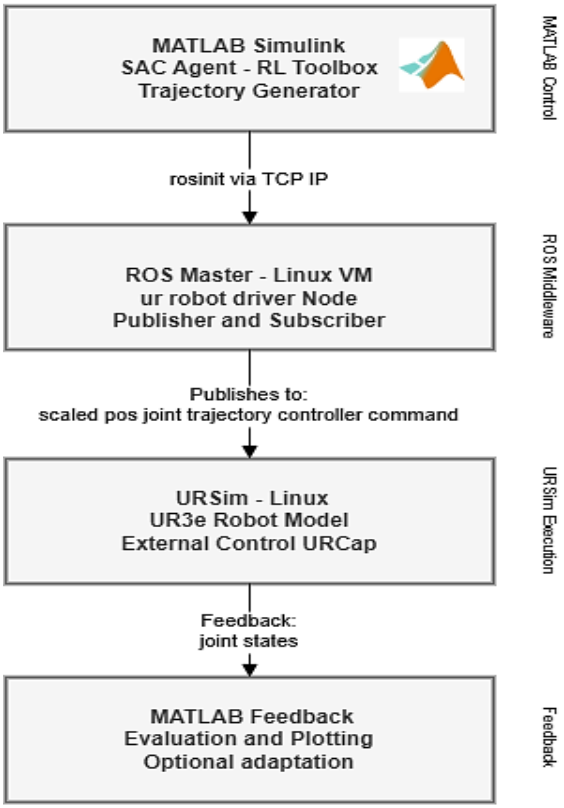

Figure 1. MATLAB–ROS–URSim Control Architecture.

A. Control Agent Layer (MATLAB/Simulink).

The SAC agent is trained using MATLAB's Reinforcement Learning Toolbox on a Windows host machine. It generates continuous joint-space trajectories for the UR3e manipulator during the pick-and-place task. A trajectory generator module formats the agent's output into a ROS-compatible structure, which is then published over the network using TCP/IP.

Communication Layer (ROS Core on Linux VM).

A virtual machine running Ubuntu hosts the ROS core, including the ur_robot_driver node that serves as a middleware interface. MATLAB establishes a TCP/IP connection to the ROS master via *rosinit() Trajectory commands are published to the /scaled_pos_joint_trajectory_controller/comman.*, topic,

while joint feedback is retrieved from /joint states. This layer handles synchronization and message transfer between the control policy and the simulator.

B. Execution Layer (URSim with UR3e Model).

URSim, also running on the Linux VM, provides a realistic simulation of the UR3e manipulator. The External Control URCap was installed to enable the robot model to subscribe to motion commands via ROS and execute them in real time. This integration allows the simulated robot to follow joint trajectory inputs published from MATLAB through ROS, enabling synchronized interaction between the policy and the robot within the virtual environment.

C. Feedback and Monitoring Layer

Joint state feedback from the URSim environment is streamed to MATLAB for evaluation and visualization. This closed loop structure allows for real time performance analysis and policy adaptation, supporting both qualitative and quantitative assessment of the agent's behavior.

IV. Methodology.

This section describes the procedure used to design, train, and deploy a Soft Actor-Critic (SAC) agent for real-time control of a UR3e robotic manipulator in a simulated pick-and-place scenario. The implementation integrates MATLAB for agent training, ROS middleware for communication, and URSim as the execution environment.

A. Training Environment.

The SAC agent was trained using MATLAB R2023b and the RL Toolbox. The training environment was modeled in Simulink as a closed-loop control system, where the agent receives continuous observations and generates joint position actions. The UR3e robot model was configured with 6-DOF and constrained within a fixed workspace suitable for tabletop manipulation.

The agent's observation space included the end-effector position, the relative position of the target object, and the current joint angles. The action space consisted of continuous joint position deltas for all 6 joints. The simulation was executed with a fixed step time of 0.005 seconds. Each episode lasted for 3 seconds, resulting in 600 steps per episode.

To encourage policy generalization, the initial positions of the robot and the object were randomized across episodes. Training was performed for 1000 episodes, and the agent's performance was monitored using two key metrics: cumulative reward and task success metrics These metrics help evaluate how well the agent improves and learns over time throughout the training process.

B. SAC Agent Configuration.

The SAC agent utilized fully connected neural networks for the actor and twin critics, each consisting of two hidden layers with 256 units and ReLU activation functions. The policy was stochastic and optimized under the maximum entropy framework. Automatic entropy coefficient tuning and gradient clipping were enabled to stabilize training.

TABLE I. SAC Agent Hyperparameters and Configuration.

Parameter	Value	Description
Actor/Critic Hidden Layers	[256, 256]	Two fully connected layers per network
Activation Function	ReLU	Applied to all hidden layers
Discount Factor (γ)	0.98	Future reward discount
Target Smoothing (τ)	0.001	For target network update
Entropy Coefficient (α)	Auto-tuned	Adjusted during training
Learning Rate	1×10^{-5}	For both actor and critic
Mini-Batch Size	128	Number of samples per training step
Experience Buffer Length	1×10^{6}	Capacity of replay memory
Warm-Up Steps	100	Exploration steps before training starts

C. . Reward Function and Termination Criteria

The reward function was designed to promote accurate reaching and smooth motion. At each time step, the agent received a scalar reward defined as:

$$r_t = -\lambda_1 \|p_t - p_{goal}\|^2 - \lambda_2 \|\Delta q_t\|^2 + r_{success} \quad (1)$$

Equation (1) combines position accuracy, motion smoothness, and task success into a single scalar reward $\|p_t - p_{goal}\|^2$ penalizes deviations from the goal position, encouraging precise reaching. The joint movement term $\|\Delta q_t\|^2$ discourages abrupt actions and supports smoother transitions. $r_{success}$ adds a positive bonus when the object is placed successfully within tolerance. The weighting factors λ_1 and λ_2 were empirically selected to balance precision and energy efficiency

An episode was terminated either upon successful object placement within a 2 cm tolerance or upon reaching the predefined maximum number of steps.

D. Real-Time Simulation Setup using ROS–URSim.

Simulink and URSim were used to deploy the policy in a closed-loop configuration in order to assess the trained SAC agent's performance under realistic timing and communication constraints. The agent executed its control policy continuously in response to live feedback from ROS, enabling closed-loop testing under simulation constraints., generating joint position commands that were sent to the simulated robot. The URSim instance, equipped with the External Control URCap, received these commands and executed them without predefined trajectories.

ROS topics were used for all control and feedback signals, ensuring compatibility with future hardware implementation.

V. EXPERIMENTAL VALIDATION AND RESULTS

This section presents the experimental outcomes derived from a simulation-based reinforcement learning framework using a UR3e manipulator trained to perform a structured pick & place task. The SAC agent was trained in MATLAB/Simulink, with all robot modeling and trajectory planning executed through custom UR3e model. The control loop was tested within a closed-loop ROS–URSim, allowing synchronized feedback between the agent and the simulated UR3e platform. The reported results include training reward progression, reconstructed end-effector trajectories, and joint-space behavior all extracted directly from the trained policy after convergence.

1) Episode Reward Trend

The SAC-trained UR3e agent was evaluated over 395 training episodes using episodic rewards to assess policy development. Each episode reward reflects the total cumulative return obtained during one complete rollout, incorporating penalties for deviation and bonuses for task completion. This formulation captures the agent's performance in terms of trajectory accuracy, motion smoothness, and timely execution.

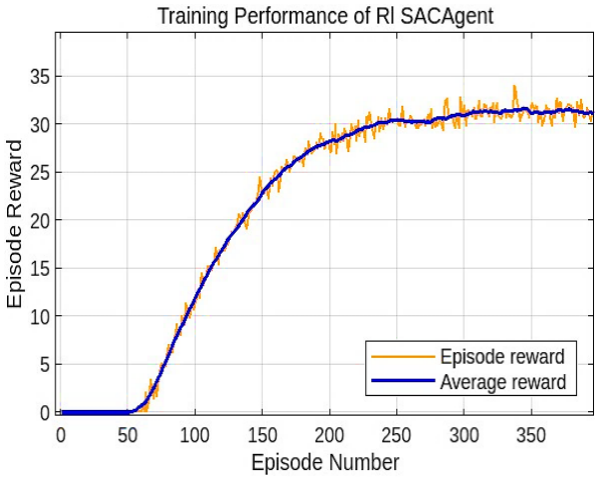

Figure 2. Reward progression during training.

As illustrated in Figure 2, the orange curve represents the raw episode rewards, which exhibit steady improvement up to episode 250. a 20-episode moving average (blue curve) was used to reduce noise and highlight long-term learning behavior. The final average reward converged near 31.89, and individual episode rewards reached 32.15, indicating reliable and repeatable performance.

This elaboration addresses reviewer feedback by clarifying that each reward corresponds to a single training episode and not an aggregate or maximum deviation. The combination of raw and smoothed data provides a transparent view of the agent's learning progression and convergence quality.

Although training was originally configured for 1000 episodes, the learning curve plateaued early, and the extracted policy at episode 395 already demonstrated consistent task completion. This choice optimized computational resources while preserving policy stability.

The agent achieved stable performance early, confirming efficient policy learning and reducing the need for extended training.

2) End-Effector Trajectory Analysis .

To evaluate the spatial behavior of the trained UR3e SAC agent, the 3D trajectory of the end-effector was reconstructed during a complete pick-and-place task. As shown in Figure 3, the trajectory begins at a predefined home position and proceeds through four main phases: approach, pick, transport, and place. Each segment is color-coded to reflect its functional role in the motion sequence. The motion of the end-effector is smooth and continuous across all transitions, with clear segmentation between task stages. Key points corresponding to the pick and place targets were consistently reached with stable positioning. The trajectory reached a total length of 1393.9 mm, and the maximum vertical displacement was 300 mm, ensuring sufficient clearance for the object handling cycle. Minor curvature in the trajectory reflects the natural characteristics of the learned policy under dynamic control conditions.

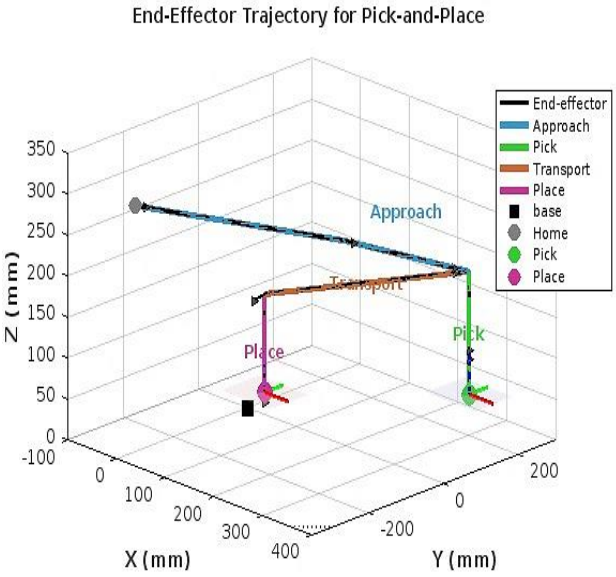

Figure 3 . end-effector trajectory during the learned pickand place task

Within the Simulink–ROS–URSim simulation environment, these findings validate the UR3e SAC agent's capacity to coordinate intricate spatial transitions and generate dependable, focused on objectives movements.

The segmented and smooth motion pattern across all task phases indicates that the SAC agent generalized task behavior without relying on manually scripted trajectories. The observed curvature highlights adaptive, learned responses rather than rigid planning, confirming task understanding and spatial consistency.

3) Joint-Level Motion Behavior.

To assess the feasibility of the learned control signals in joint space, the joint positions of all six revolute joints (θ_1 to θ_6) of the UR3e robot were analyzed during task execution. This examination aimed to verify whether the motion remains continuous, coordinated, and physically valid throughout the pick-and-place cycle. The curves reveal continuous and stable joint movements across all six axes. Each phase of the task approach, pick, transport, and place is clearly reflected in the joint transitions. Notably, no abrupt spikes or erratic behavior is observed, indicating that the policy maintains both mechanical feasibility and smooth temporal dynamics. Such behavior reinforces the agent's ability to generalize motion planning across task stages while remaining within actuator limits

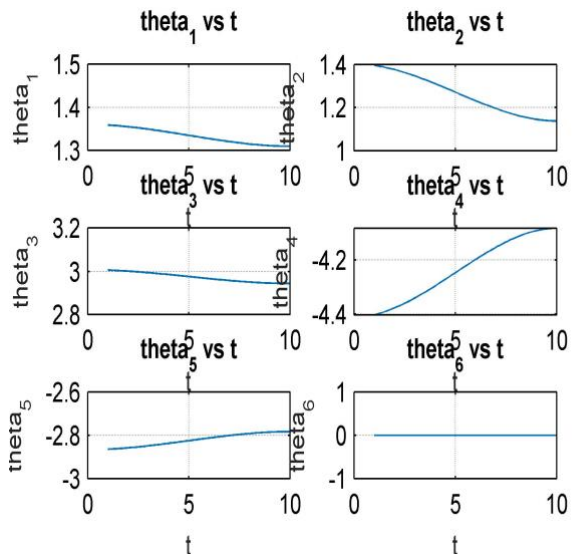

Figure 4. Joint angle profiles (θ_1–θ_6) during pick & place execution

The absence of abrupt transitions or erratic joint behavior validates the mechanical feasibility of the learned control policy. This consistency confirms that the SAC agent respects kinematic constraints and generates realistic motion commands suitable for deployment in a real-time robotic system.

VI. CONCLUSION.

The proposed SAC-based control framework demonstrated effective trajectory learning and execution for a 6-DOF UR3e robotic manipulator using a full MATLAB–ROS–URSim integration. The agent achieved reliable pick-and-place performance with smooth joint transitions and accurate positioning.

Based on reviewer feedback, the reward function explanation was refined to emphasize how position error, joint variation, and task success collectively guide learning. This clarified formulation reinforces the link between reward design and observed robot behavior.

The integration approach also showed that real-time control and feedback exchange between MATLAB and URSim can be achieved without requiring external ROS nodes or latency-heavy bridges. Future work will focus on deploying the trained policy to a physical UR3e platform and validating performance under hardware constraints.

ACKNOWLEDGMENT .

The author would like to thank the Department of Automation and Industrial Informatics at the University Politehnica of Bucharest for their support during the development of this work.

DECLARATION ON GENERATIVE AI .

During the preparation of this work, one of the authors (Ahmed Iqdymat) used ChatGPT to improve the language and clarity of specific sections, particularly the *Introduction* and *Related Work*. No AI-generated content was used to produce the original concepts, methodology, experimental results, or analyses. The final manuscript was reviewed and approved by both authors, who take full responsibility for the publication's content.

REFERENCES

[1] E. A. Alandoli, M. Z. A. Rashid, and M. Sulaiman, "A comparison of PID and LQR controllers for position tracking and vibration suppression of flexible link manipulator," J. Theor. Appl. Inf. Technol., vol. 95, pp. 2949–2955, 2017.

[2] M. Mohamed, F. Anayi, M. Packianather, B. A. Samad, and K. Yahya, "Simulating LQR and PID controllers to stabilise a three-link robotic system," in Proc. 2nd Int. Conf. on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, pp. 2033–2036, 2022. doi: 10.1109/ICACITE53722.2022.9823512.

[3] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in Proc. 33rd Int. Conf. on Machine Learning (ICML), 2016.

[4] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, and J. Pachocki et al., "Learning dexterous in-hand manipulation," Int. J. Robot. Res., vol. 39, no. 1, pp. 3–20, Jan. 2020.

[5] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, "A survey on deep reinforcement learning algorithms for robotic manipulation," Sensors, vol. 23, no. 7, Art. no. 3762, 2023. doi: 10.3390/s23073762.

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a

stochastic actor," in Proc. 35th Int. Conf. on Machine Learning (ICML), 2018.

[7]] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-learning for offline reinforcement learning,"" in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[8] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. 35th Int. Conf. on Machine Learning (ICML)*, 2018.

[9] C.-C. Wong, S.-Y. Chien, H.-M. Feng, and H. Aoyama, "Motion planning for dual-arm robot based on soft actor-critic," IEEE Access, vol. 9, pp. 26871–26885, 2021. doi: 10.1109/ACCESS.2021.3056903.

[10]] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," *CoRL*, 2018.

[11] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. on Machine Learning (ICML)*, 2017.

[12] J. Park, J.-J. Kim, and D.-Y. Koh, "Experimental evaluation of precise placement with pushing primitive based on Cartesian force control," Appl. Sci., vol. 15, no. 1, Art. no. 387, Jan. 2025. doi: 10.3390/app15010387.

[13] . A. Vivas and J. M. Sabater, "UR5 robot manipulation using MATLAB/Simulink and ROS," in Proc. IEEE Int. Conf. on Mechatronics and Automation (ICMA), Takamatsu, Japan, 2021, pp. 338–343. doi: 10.1109/ICMA52036.2021.9512650

[14] N. Rosillo, N. Montés, J. P. Alves, and N. M. F. Ferreira, "A generalized Matlab/ROS/robotic platform framework for teaching robotics," in Robotics in Education (RiE 2019), M. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh, and D. Obdržálek, Eds., Advances in Intelligent Systems and Computing, vol. 1023, Cham: Springer, 2019, pp. 159–169. doi: 10.1007/978-3-030-26945-6_15.

[15]] R. Meyes, H. Tercan, S. Roggendorf, T. Thiele, C. Büscher, and M. Obdenbusch et al., "Motion planning for industrial robots using reinforcement learning," in Proc. 50th CIRP Conf. on Manufacturing Systems, vol. 63, pp. 107–112, 2017. doi: 10.1016/j.procir.2017.03.095.

[16] D. Schwung, A. Schwung, and S. X. Ding, "Actor-Critic Reinforcement Learning for Energy Optimization in Hybrid Production Environments," International Journal of Computing, vol. 18, no. 4, pp. 360–371, 2019. doi: 10.47839/ijc.18.4.1607.

[17] V. Koval, V. Turchenko, A. Sachenko, J. A. Becerra, R. J. Duro, and V. Golovko, "Infrared Sensor Data Correction for Local Area Map Construction by a Mobile Robot," in Developments in Applied Artificial Intelligence, Springer, vol. 2718, 2003. doi: 10.1007/3-540-45034-3_31.

[18] A. Iqdymat and G. Stamatescu, "Real-Time Pick-and-Place Task Execution Using Reinforcement Learning with SAC Algorithm in Simulink–ROS–UR3e Setup," in Proc. 21st Int. Conf. on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2024.

[19] A. Iqdymat and G. Stamatescu, "Reinforcement Learning of a Six-DOF Industrial Manipulator for Pick-and-Place Application Using Efficient Control in Warehouse Management," Sustainability, vol. 17, no. 2, art. 432, 2025. doi: 10.3390/su17020432.

[20] . MathWorks, "ROS Toolbox User's Guide," 2023. [Online]. Available: https://www.mathworks.com/help/ros/.

[21] Universal Robots, "Setting up environment for use with MATLAB for UR development," 2024. [Online]. Available: Universal Robots - Setting up environment for use with MATLAB for UR development