eISSN 2783-6851

VILNIUS TECH
Vilnius Gediminas Technical University

NEW TRENDS in COMPUTER SCIENCES

2025
Volume 3
Issue 1
Pages 49–63
https://doi.org/10.3846/ntcs.2025.23668

FRAMEWORK FOR DEEP REINFORCEMENT LEARNING IN WEBOTS VIRTUAL ENVIRONMENTS

Andrej ŠAREIKO ✉, Dalius MAŽEIKA, Algirdas LAUKAITIS

Department of Information Systems, Faculty of Fundamental Sciences, Vilnius Gediminas Technical University, Vilnius, Lithuania

Article History:
- received 05 April 2025
- accepted 28 May 2025

Abstract. Reinforcement learning (RL) algorithms, particularly deep reinforcement learning (DRL), have shown transformative potential in robotics by enabling adaptive behaviour in virtual environments. However, a comprehensive framework for efficiently testing, training, and deploying robots in these environments remains underexplored. This study introduces a standardized, open-source framework designed specifically for the Webots simulation environment. Supported by a robust methodology, the framework integrates innovative design patterns and the digital twin (DT) concept with three distinct design patterns for structuring agent-environment interaction, notably including a novel pattern aimed at improving sim-to-real transferability, to enhance RL workflows. The proposed framework is validated through experimental studies on both a model the inverted pendulum and a production-grade Pioneer 3-AT robotic platform. The experiments highlight the framework's ability to bridge the gap between virtual training and real-world implementation. All resources, including the framework, methodology, and experimental configurations, are openly accessible on GitHub.

Keywords: reinforcement learning, virtual environments, robot testing, Webots, Stable-Baselines3, simulation tools, digital twin, robotic training.

✉Corresponding author. E-mail: ***andrej.sareiko@stud.vilniustech.lt***

1. Introduction

Reinforcement learning (RL) is changing how robots learn and behave. It's a type of machine learning where robots improve their actions by interacting with their surroundings and learning from their experiences. RL uses rewards to guide the robots in choosing better actions over time. Virtual environments make this process safer and cheaper because robots can practice and learn without risks. These virtual spaces also speed up the development of smarter robots by giving them a flexible and controlled place to test and train (Ayala et al., 2020).

The utilization of virtual environments in robotics provides a crucial advantage: the ability to simulate complex real-world scenarios without the risks and constraints associated with physical experimentation. From dynamic navigation in cluttered spaces to precise manipulation of objects, virtual environments can replicate a diverse array of challenges that robots may encounter in the real world. These settings are particularly beneficial for training tasks that involve high levels of uncertainty or potential hazards, such as disaster response robotics or autonomous driving systems. Moreover, advancements in simulation technologies now enable the creation of hyper-realistic environments, bridging the gap between synthetic training and practical deployment.

This study investigates methodologies, development paradigms, and persistent challenges inherent in applying RL within virtual environments. We present a comprehensive analysis of fundamental principles, recent technological advancements, and examples that show the practical benefits of this approach. Furthermore, we carefully look at the limitations and suggest future research to improve the accuracy and reliability of simulated training environments. By studying these topics, we aim to show how RL can significantly change robotics, helping to build more adaptable, efficient, and practical machines.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive review of related work, outlining existing methodologies and tools relevant to RL in virtual environments. Section 3.1 introduces the framework, detailing its design patterns and comparing it to current RL approaches. Section 3.2 describes the methodology, focusing on the implementation of RL algorithms within the Webots simulation platform and the digital twin (DT) creation process. Section 4 presents the experimental results obtained using the proposed framework and methodology, demonstrating their effectiveness through the inverted pendulum task. Finally, the Conclusion summarizes the key findings, discusses the implications of the work, and outlines potential directions for future research and development.

2. Related work

RL (Tang et al., 2024) is a branch of machine learning focused on enabling agents to learn optimal behaviors by interacting with their environment through trial and error. RL leverages a system of rewards and penalties to encourage learning through feedback, allowing agents to make decisions that maximize cumulative rewards (Kilinc & Montana, 2022). A cornerstone of this methodology is the Markov Decision Process (MDP), which formalizes the interaction between an agent and its environment in terms of states, actions, transition probabilities, and rewards (Jonban et al., 2024).

A key feature of RL is the balance between exploration and exploitation. Exploration refers to testing new actions to discover potentially better strategies, while exploitation focuses on leveraging known strategies to maximize immediate rewards. Modern RL employs advanced algorithms like Proximal Policy Optimization (PPO) (Zhang et al., 2022), Deep Q-Networks (DQN), and Advantage Actor-Critic (A2C) (Talaat, 2020), which allow agents to tackle complex, high-dimensional tasks effectively.

RL applications in virtual environments have already demonstrated significant potential across industries. Robots trained via RL in simulation have been successfully deployed in manufacturing for precision assembly tasks, in healthcare for automated assistance, and even in space exploration for autonomous navigation. However, the transition from virtual to real-world scenarios remains a formidable challenge, primarily due to discrepancies in dynamics, noise levels, and environmental variability. Addressing these challenges necessitates robust techniques for domain adaptation and transfer learning to ensure the seamless applicability of policies learned in simulation.

The OpenAI Gym (now Gymnasium) framework has emerged as a vital tool in the development and testing of RL algorithms (Brockman et al., 2016). It provides a standardized interface for a diverse range of simulated environments, from simple tasks like CartPole

to intricate robotic control simulations. Gym supports both discrete and continuous action spaces, enabling researchers and practitioners to benchmark their RL models under consistent conditions. By offering ready-to-use environments, it accelerates experimentation, promotes reproducibility, and fosters innovation in the RL community (Towers et al., 2024).

The Gym framework's modularity extends its usability, allowing developers to create custom environments tailored to specific applications. For instance, robotic RL, tools like Gym's robotics suite, which integrate simulated tasks with physics engines like MuJoCo (Todorov et al., 2012), enable training of manipulation or navigation policies in virtual settings before deploying them in real-world scenarios. This ability to simulate environments mitigates risks and reduces costs during the training phase, making Gym a pivotal resource for RL research.

Deep reinforcement learning (DRL) (Ladosz et al., 2022), which combines RL with deep neural networks, has further enhanced the scope of Gym applications. Frameworks like Stable-Baselines3 (Raffin et al., 2021) and Ray's RLlib (Liang et al., 2017) build on Gym to provide robust RL algorithm implementations, simplifying the process of training agents in Gym-compatible environments. These integrations underscore Gym's flexibility and its role in advancing state-of-the-art RL techniques. In summary, RL principles, bolstered by frameworks like OpenAI Gym, continue to drive progress in artificial intelligence and robotics. By providing accessible, standardized environments, Gym empowers researchers and practitioners to push the boundaries of what autonomous agents can achieve in simulated and real-world scenarios.

As RL problems increase in complexity, simulation environments become essential for development and testing. Simulation reduces the need for direct experimentation with physical systems, which is particularly critical for tasks with high degrees of freedom, such as autonomous driving. Consequently, simulation platforms like Webots (Michel, 2004) are increasingly important, especially those that are open-source.

Alternatives to Webots include Gazebo (Uslu et al., 2017), RoboDK (Garbev & Atanassov, 2020), CoppeliaSim, OpenRave, and Unity (Tseeva et al., 2024), all of which support the integration of DRL algorithms. This paper focuses on Webots due to its capability to create robots from scratch, its realistic graphics, and its compatibility with the Robot Operating System (ROS). However, many aspects of the proposed framework can be adapted to other simulation environments.

A key distinction between environments like Gymnasium and simulation platforms like Webots is the fidelity of actuator and sensor modeling. In Webots, actuators and sensors, such as robotic arms and LiDAR, are designed to closely resemble their physical counterparts. Webots allows robot programming via controllers in multiple languages, including C, Python, and MATLAB. For DRL implementations, Python is preferred due to its ease of use and the straightforward translation of C examples.

Several projects aim to facilitate RL in robotic simulators. Gym-Ignition (Ferigo et al., 2020) provides an OpenAI Gym interface for Gazebo, supporting reproducible robot environments, external software integration, multiple physics and rendering engines, and ROS compatibility. Zamora et al. (2016) extends the Gym interface with ROS compatibility for Gazebo. Lopez et al. (2019) offers ROS 2 compatibility and is applied in real-world scenarios. NVIDIA Isaac ROS (NVIDIA, 2025) provides a comprehensive framework for DRL and robotics, featuring photorealistic rendering and parallelization. While Deepbots (Kirtas et al., 2020) aimed to

facilitate DRL in Webots, its current implementation suffers from bugs and compatibility issues due to its reliance on the older OpenAI Gym. Beyond the choice of specific libraries or simulator interfaces, the architectural design patterns governing agent-environment interaction within the simulation significantly impact RL workflow efficiency and, crucially, the potential for sim-to-real transfer. Many conventional setups in robotic simulation environments, including some earlier approaches for Webots, tend to rely heavily on centralized supervisor entities that possess global knowledge and control capabilities unavailable in physical robots. While such configurations can simplify certain aspects of simulation management (e.g., environment resets or complex reward calculations), this reliance often leads to policies that are difficult to deploy on physical systems constrained by their onboard sensors and actuators. Recognizing this critical gap, the present study systematically investigates three distinct design patterns for RL in Webots. These patterns, detailed in Section 3.1, range from simpler integrated approaches reflective of common practices to a novel decoupled architecture specifically engineered to minimize simulator-specific dependencies and thereby enhance the real-world applicability of trained agents (Behrens et al., 2012). This research presents the first generic interface for Webots using the updated OpenAI Gymnasium, standardizing and simplifying the application of modern RL algorithms.

3. Method

This section outlines the framework and methodology developed to facilitate RL within the Webots simulation environment. We first introduce the core design patterns that underpin the proposed framework, highlighting their capabilities and limitations in bridging the gap between virtual training and real-world deployment. Next, we detail the DT creation process, demonstrating how it enables safe and efficient testing of RL algorithms.

3.1. Design patterns of reinforcement learning in virtual environments

RL within virtual environments is facilitated through a dynamic interplay of simulation tools, RL frameworks, and machine learning algorithms. At the core of presented architecture lies the need for a simulated environment capable of accurately modeling real-world dynamics, providing a controlled and risk-free setting for robotic training. Tools like Webots, Gazebo, and Unity play a pivotal role by offering physics engines that replicate the physical world, including gravity, friction, and collisions, alongside providing sensory feedback such as visual, tactile, or distance-based data.

The WebotsRL system presented in this paper incorporates three design patterns used for experimentation. The first two patterns follow the RL loop commonly found in frameworks like Gymnasium or Deepbots. In these frameworks, the environment receives an action from the robot at each step and returns a pair of observations and rewards. This process is repeated until the robot either achieves its goal or an exception terminates the loop. While this approach is effective for testing RL in virtual environments, we identified a need for a more complex strategy when transferring RL-trained models and processes to real-world robots. To address this, we developed a third design pattern that builds upon the first two, eliminating the Supervisor concept to better facilitate the transition to physical robots.

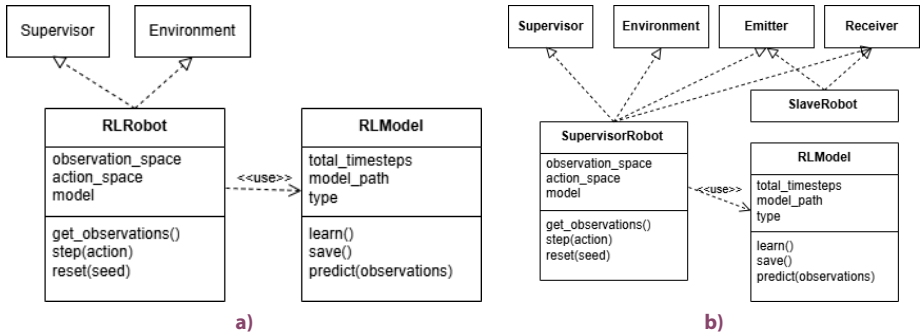

Figure 1. a – Simple design pattern with integrated Supervisor for robot RL implementation; b – Design pattern with separated Supervisor using Emitter-Receiver interface

Figure 1 illustrates two design patterns comparable to those introduced for Webots in Kirtas et al. (2020). The pattern shown on the left side of Figure 1 represents the simplest approach for testing RL algorithms in the Webots virtual robot simulation environment. This pattern is composed of two key classes: *RLRobot* and *RLModel*. The *RLRobot* class defines the robots to be trained and is responsible for gathering observations from the environment, executing actions, and resetting the environment to its initial state once the robot either achieves its goal or encounters a failure during the learning process.

A significant drawback of this design pattern arises when attempting to transfer RL models from simulated environments to real-world applications. This limitation stems from its reliance on inheriting functionality from the Supervisor class. In many RL implementations, the Supervisor class is used to provide the robot with a comprehensive understanding of the simulated environment. This includes access to information well beyond the capabilities of the robot's sensors, such as repositioning objects, calculating precise distances to targets, or retrieving detailed speed values for multiple objects by Webots built-in libraries.

While these advanced capabilities streamline simulation tasks, they are entirely impractical in real-world scenarios. Physical robots are inherently constrained, relying solely on data from their onboard sensors or external information supplied by collaborating robots or external computational systems, such as servers. The reliance on the Supervisor class introduces an unrealistic dependency that undermines the applicability and reliability of RL-trained models when transitioning from virtual simulations to real-world deployments. Addressing this discrepancy is critical to ensuring that RL systems are not only effective in simulation but also viable and robust in practical use cases.

To address these limitations, the design pattern illustrated on the right side of Figure 1 is proposed. This pattern adheres more closely to real-world constraints and aligns with the recommendations of Webots systems and previous work by Kirtas et al. (2020).

The second design pattern introduces two robot classes: *SupervisorRobot* and *SlaveRobot*. Additionally, it incorporates the *Emitter* and *Receiver* classes provided by the Webots system to simulate communication between robots using string messages over virtual radio waves. In the proposed *WebotsRL* system, the *Emitter* and *Receiver* are used for communication

between the *SupervisorRobot* and *SlaveRobot*, with the *Emitter* broadcasting messages and the *Receiver* receiving them.

Instances of *SupervisorRobot* are typically not typical robots and usually lack mass or physical properties in the simulation. For example, they can represent a computational device, such as a PC, that transmits actions to robots without interacting with the scene. Similarly, the *Emitter* and *Receiver* components could be simulated wireless devices, like Wi-Fi or Bluetooth modules.

In this design pattern, the RL framework operates as follows: the virtual environment is first reset to its initial state. Then, one or more *SlaveRobot* instances collect initial observations from the environment and use their *Emitter* to send the data to the *SupervisorRobot*. The *SupervisorRobot*, through its *Receiver*, gathers these observations and passes them to the *RLModel* instance. Additionally, the *SupervisorRobot* can augment the observations with extra information from the Webots *Supervisor* class, such as distances, absolute positions, and object speeds, which are inaccessible to the *SlaveRobot* through its sensors.

The *RLModel* generates an action based on these observations, which is transmitted back to the *SlaveRobot* using the *Emitter*. The *SlaveRobot* then executes the action through its actuators. This process is repeated iteratively until the *SlaveRobot* achieves its objective or a termination condition, such as reaching the maximum episode count or violating predefined constraints, is met.

A significant drawback of the second RL design pattern is its reliance on the *SupervisorRobot* class, which inherits from the Webots *Supervisor* class. While this inheritance simplifies the implementation of methods such as *reset*() –intended to reset the environment to its initial state – it renders the design impractical for real-world applications. The functionality provided by the *Supervisor* class violates physical constraints inherent in real-world scenarios, making it infeasible to transfer these capabilities from simulation to reality. This limitation underscores the intermediate nature of virtual robot simulations, which ultimately serve as a stepping stone toward physical implementations of RL systems.

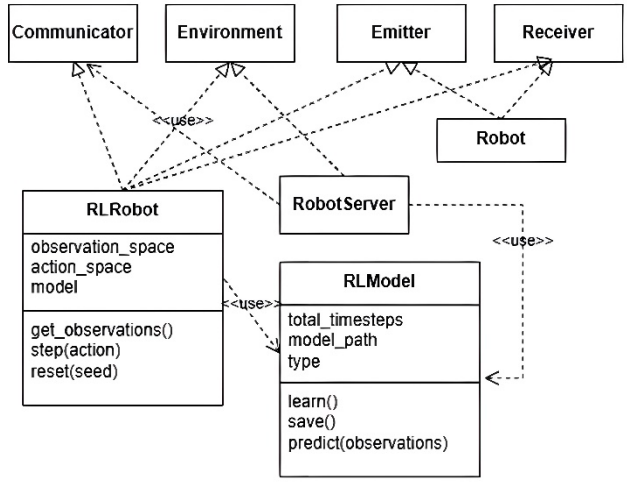

Figure 2. Proposed design pattern to facilitate the transfer of RL models from virtual simulations to real-world applications

To overcome this constraint, we propose a third RL design pattern that eliminates all references to the Webots *Supervisor* class within the simulation code. Instead of utilizing a *SupervisorRobot* class instance, this pattern introduces the concept of an external server, referred to as the *RobotServer*, which is accessed through internet-based communication, such as URL links. As shown in Figure 2, the *RobotServer* handles RL tasks for robots lacking the computational capacity to process complex tasks, such as object detection during visual processing.

Additionally, this design pattern introduces two types of robots: (1) RLRobot instances, which possess the computational power to perform RL tasks, and (2) Robot instances, which rely solely on their sensors and actuators. These Robot instances communicate with the *RobotServer* via the *Communicator* class over the internet or interact with other robots using the *Emitter* and *Receiver* classes. This approach ensures a more realistic alignment with physical constraints while maintaining the flexibility required for RL experimentation.

3.2. Digital twin development process

The DT concept has emerged as a transformative innovation, particularly in the field of industrial automation, where it is revolutionizing how robots are designed, tested, and optimized. This research introduces a generalized methodology aimed at guiding the development and application of DT to facilitate robot testing, especially in scenarios involving RL techniques.

At its core, the process emphasizes the creation of a precise and fully functional virtual representation of a physical robot, replicating its operational environment and interactions. The DT acts as a robust testing framework, allowing RL algorithms to be developed and refined in a safe, simulated setting. This approach empowers researchers and industry professionals to evaluate and optimize robotic systems while significantly reducing the risks, costs, and challenges associated with real-world experimentation.

To achieve this, the methodology leverages advanced software tools and cutting-edge frameworks to seamlessly integrate the robot's structural, functional, and sensory characteristics into a cohesive simulation. By incorporating these elements into a unified digital environment, the process ensures that the virtual twin mirrors the behavior and dynamics of the physical counterpart with high accuracy.

Consider the process of developing a DT for a robot as an example. Initially the robot's movements were programmed manually. However, adapting to changes in the environment required reprogramming, which often proved costly and error prone. To address this challenge and minimize the risk of damage to the robot or its surroundings, a digital representation of the robot and its environment was created. This DT environment could be modelled using any cad software like Fusion 360, with critical components subsequently transferred to the Webots simulation platform. A robot model was integrated into this simulation. The approach outlined in this study enabled the testing of various RL algorithms in a virtual setting, eliminating the potential hazards associated with using the physical robot. Once optimal movement strategies were identified, the trained deep learning models were transferred to the physical robot via ROS tools available within Webots.The advantages of digital twins are particularly evident in RL applications, where algorithms require iterative testing and fine-tuning. Through simulation, a wide range of algorithms can be explored under controlled conditions, free from the hazards and limitations of physical trials. Once optimal policies and

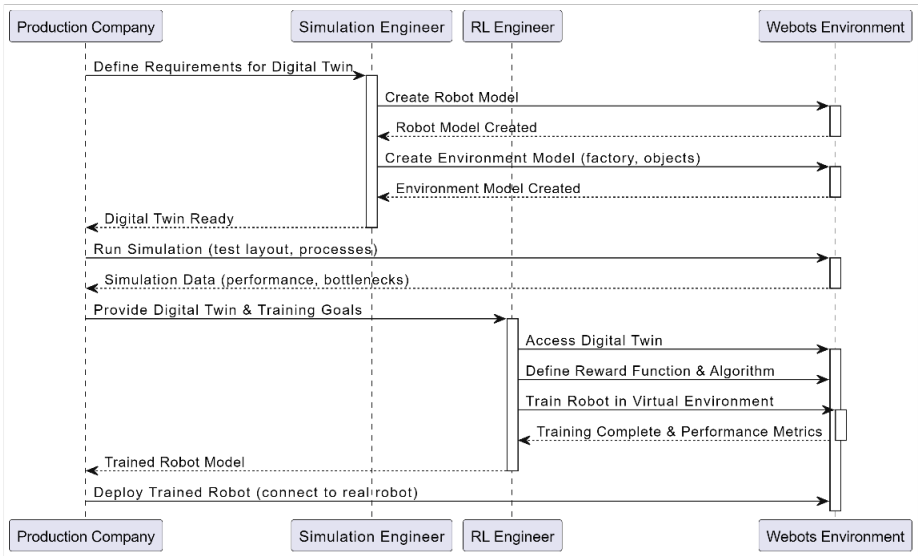

Figure 3. The sequence diagram of the generalized DT creation process

behaviors are established in the virtual realm, the trained models can be effectively deployed onto the physical robot for additional training in the physical environment, ensuring compatibility, efficiency, and real-world readiness. This streamlined workflow not only accelerates the robot development cycle but also bridges the gap between virtual experimentation and practical deployment, enabling the creation of smarter, more adaptable robotic solutions.

The creation of a DT begins with designing an RL-compatible environment, following standard APIs like OpenAI Gym to ensure compatibility with RL libraries such as Stable-Baselines3 and TensorFlow Agents. The RL agent perceives the environment through simulated sensor data, selects actions from a predefined action space, and receives rewards based on task performance. The reward structure is designed to align with training objectives, guiding the agent toward optimal behavior.

To enhance real-world applicability, domain randomization introduces variations in textures, lighting, and physical parameters, improving generalization. After training, learned policies are validated in simulation and refined for real-world deployment. This approach accelerates robotic development while reducing costs and risks.

In this study, Webots is integrated with Stable-Baselines3 to provide a structured RL testing environment. Stable-Baselines3 offers scalable RL algorithms, facilitating efficient communication between the agent and simulation. This integration supports iterative optimization, task design, and real-world readiness. Figure 3 presents the workflow for developing and deploying a DT in a production environment, involving key roles such as the Production Company, Simulation Engineer, and RL Engineer. The following section details the steps of this process.

1. *Defining Requirements for the DT*. The Production Company first defines the digital twin's requirements, which are communicated to the Simulation Engineer. This includes specifying the robot's tasks (e.g., pick-and-place, welding), operational constraints

(speed, payload, workspace limits), and necessary precision. Key data requirements for RL include Computer-Aided Design (CAD) models, sensor data, and historical operational data. Data fusion integrates information from sources like robot controllers and industrial protocols. Effective data management ensures accuracy and reliability, enabling the DT to replicate real-world conditions for training and validation. The following Table 1 summarizes these data requirements.

Table 1. Summary of the key data requirements for a DT of a robot designed for RL

Data Category	Specific Data Points	Importance for RL
Robot Model Data	CAD models (geometry, mass, inertia), kinematic and dynamic parameters (joint limits, motor characteristics).	Accurate representation of the robot's physical properties and movement capabilities is crucial for realistic simulation and effective RL training.
Environment Data	CAD models of the production line layout, machinery, fixtures, and objects the robot interacts with.	Enables the creation of a realistic virtual environment where the robot can learn to interact with its surroundings.
Operational Data	Robot controller data (joint positions, velocities, accelerations, torques), sensor data (proximity, force, vision), PLC data, system logs.	Provides real-time data on the robot's behaviour and the state of the production line, which can be used to validate the DT and inform the reward function for RL.
Task Definition Data	Specific goals and constraints of the task the robot needs to learn (e.g., target positions, assembly sequences, cycle times).	Defines the objective for the RL agent and helps in designing an appropriate reward function.
Performance Data	Metrics for evaluating the robot's performance (e.g., success rate, cycle time, energy consumption).	Used to assess the effectiveness of the RL training in the simulation and to evaluate the performance of the deployed robot in the real world.
Communication Data	Details of communication protocols and data links between the DT, the physical robot, sensors, and other systems (e.g., UR-RTDE, ROS/ROS2).	Enables the transfer of trained models to the physical robot and the potential for real-time synchronization and monitoring.

2. *Creating the Detailed Robot and Environmental Model in Webots*. The Simulation Engineer develops a detailed robot and environment model within Webots. The robot model includes kinematics, dynamics, actuators, sensors, and an end-effector, ensuring accurate simulation of motion and interactions. The environment model replicates the production layout, including objects, obstacles, and sensor placements. Balancing model fidelity and computational efficiency is essential. While high-detail models improve realism, they increase computational cost. The appropriate level of detail depends on task complexity, ensuring effective RL training without unnecessary overhead resource.
3. *Simulation and Performance Bottleneck Identification*. Once the DT is created in Webots, the Production Company runs simulations to test different layouts, optimize workflows, and identify bottlenecks. By analyzing robot performance under varying conditions, such as speed, load, and potential failure scenarios, engineers can detect inefficiencies,

reachability issues, and collision risks before physical deployment. Simulation data, including cycle times, trajectory analysis, and energy consumption, provides insights for refining system design and improving operational efficiency. This virtual testing approach reduces costs and minimizes disruptions to the production process.

4. *Formulating Training Goals for the RL Engineer*. The Production Company defines training objectives based on simulation analysis and provides the DT to the RL Engineer. These objectives specify tasks, such as pick-and-place or path planning, along with performance metrics like success rate and cycle time. Constraints, including safety limits and operational boundaries, are established to ensure feasibility in real-world deployment.
5. *RL Training with DT*. The RL Engineer utilizes the DT in Webots to define the reward function, select training algorithms, and train the robot. The reward function guides learning by assigning positive rewards for task completion and penalties for undesired actions. The choice of algorithm, such as Soft Actor-Critic, depends on task complexity and learning stability. Through iterative trial and error, the RL agent refines its policy to maximize cumulative rewards. DT enables safe, cost-effective training, allowing multiple simulations to accelerate learning without risks to physical equipment or production.
6. *Delivery of the Trained Robot Model and Performance Metrics*. The RL Engineer delivers the trained robot model and performance metrics to the production company. The model, typically a control policy or neural network weights, represents the learned behavior. Performance metrics, such as cumulative reward progression, policy stability, task completion rate, and cycle time, assess training effectiveness. These metrics help determine if the learned policy meets operational requirements before real-world deployment.
7. *Deployment of the Trained Robot and Real-World Connection*. The production company deploys the trained model on the physical robot, ensuring compatibility between simulation and hardware. The robot then executes tasks autonomously based on the learned policy. Optionally, real-time integration with the DT allows continuous monitoring, performance evaluation, and further refinement of the control policy using real-world data.

This sequence showcases the importance of collaboration and the utility of tools like Webots and Stable-Baselines3 in developing efficient, well-trained robotic systems. The diagram also highlights how virtual environments, such as Webots, are indispensable for bridging simulation and real-world applications in a controlled and iterative manner.

4. Results

To evaluate the RL patterns from Section 3 and the DT method from Section 3.2, we selected two Webots environments, as shown in Figure 4. These experiments were designed not only to demonstrate the learning efficacy within the proposed WebotsRL framework but also to highlight its comparative strengths and practical advantages over existing approaches, particularly in standardizing Webots for modern RL libraries and improving the pathway to real-world deployment. Both environments involve the inverted pendulum problem (Barto et al.,

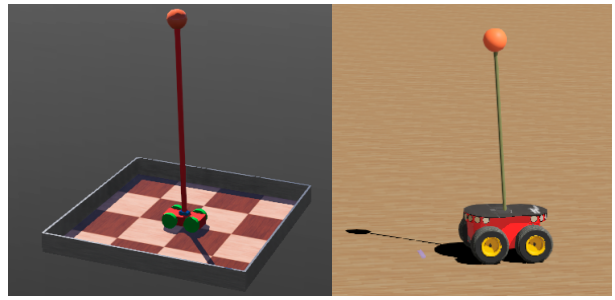

Figure 4. Two Webots environments used to test methods presented in this paper

1970). The first (Figure 4, left) replicates the Deepbots RL framework for Webots (Kirtas et al., 2020). It consists of a cart with a one-meter pole attached via a free hinge, equipped with a sensor to measure the pole's angle. The task requires maintaining the pole in a vertical position by moving the cart forward or backward, using a discrete PPO algorithm (Schulman et al., 2017). The observation space includes the cart's position and velocity, the pole's angle, and its angular velocity. The agent selects between two actions – moving forward or backward – and receives a reward of +1 per step, including the termination step. Episodes terminate after 1950 steps or when the pole falls or the cart moves beyond ±0.4 meters. A task is considered solved if the agent achieves an average score above 1950 over 100 consecutive episodes.

We used this environment to validate the consistency of our framework against results from Kirtas et al. (2020) and to assess the impact of Webots' "speed-up simulation" mode on learning efficiency. The PPO agent, implemented with a two-layer neural network (10 ReLU neurons per layer), successfully solved the problem within approximately 3.5 hours of simulated time, consistent with prior work. However, running the simulation in "speed-up" mode reduced execution time to less than 10 minutes without affecting performance. The learning curve (Figure 5, left) confirms alignment with previous results. This result not only validates the core functionality of our framework but also demonstrates an immediate advantage: compatibility with the updated OpenAI Gymnasium standard, allowing seamless integration with current RL libraries like Stable-Baselines3. This contrasts with the original Deepbots framework, which was developed for the older, now deprecated, OpenAI Gym. Furthermore,

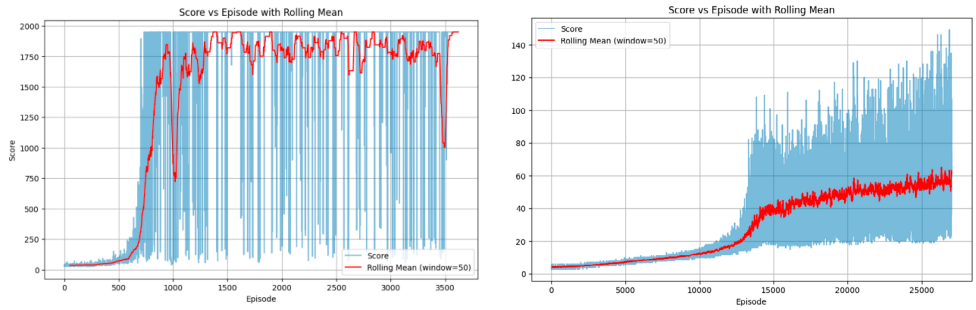

Figure 5. The learning curves for both inverted pendulum environments

the successful use of Webots' "speed-up" mode underscores the framework's ability to capitalize on simulator features for efficient experimentation.

The second environment utilizes a Webots model of the Pioneer 3-AT, a four-wheel, skid-steer robot. This platform is selected for its suitability in RL experiments due to its well-defined action space (skid-steer drive with motor control), sensor availability (wheel encoders), and robust physical characteristics (12 kg weight, 0.7 m/s max speed, 35% max traversable grade). The robot's microcontroller and I/O capabilities, including digital and analog inputs, allow for diverse sensor integration and control. These features facilitate the development and testing of RL algorithms for tasks involving navigation and manipulation in varied terrains.

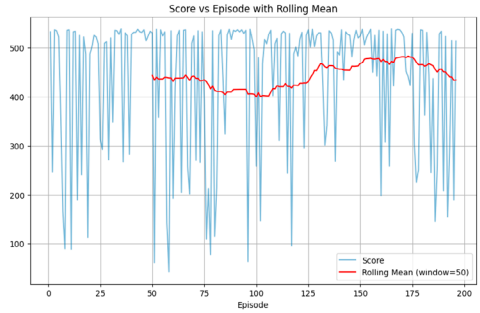

Figure 6. The learning curves of Pioneer 3-AT after 32 million steps

The Pioneer 3-AT Webots model closely replicates its physical counterpart. This study evaluates whether an RL algorithm can achieve stable inverted pendulum control, similar to the toy cart example. Figure 5 (right) shows the learning curve over the first 1 million steps, where the model required approximately 30,000 training episodes but achieved only around 60 cumulative reward per episode, indicating slower learning compared to the simpler system. To test long-term learning potential, we extended training to 32 million steps, randomizing the initial pendulum angle for each episode to enhance realism. We observed faster convergence when initial conditions matched the toy cart experiment but by randomizing the initial pendulum angle in each episode we noticed significant drop in learning curve convergence. Figure 6 presents results using the trained model, where the average cumulative reward reached 500. However, occasional low rewards persisted, highlighting the challenge of RL in complex environments and the potential for errors in learned policies.

The observed challenges with the Pioneer 3-AT namely the slower convergence compared to the simpler cart-pole system, the significant drop in learning curve convergence when randomizing initial pendulum angles, and the persistence of occasional low rewards despite extensive training warrant a more detailed consideration. These issues are likely attributable to several factors inherent in more complex robotic systems. The Pioneer 3-AT platform, with its skid-steer dynamics and increased degrees of freedom compared to the basic cart, presents a substantially harder control problem. This complexity amplifies the difficulty of effective exploration in a larger state-action space and can exacerbate the credit assignment problem for the RL agent. Furthermore, while randomizing initial conditions enhances realism and the potential for generalization, it also greatly expands the state space the agent must learn to master, thereby slowing down the learning process.

To address these convergence and performance issues, several advanced DRL strategies could be explored. Curriculum learning, for instance, would involve initially training the agent in simpler scenarios perhaps by starting with less randomization in the pendulum's initial angle (like the initial conditions of the toy cart experiment where faster convergence was noted), a restricted range of cart movement, or even a temporarily stabilized pendulum and gradually increasing the complexity. This staged approach could allow the agent to develop foundational control policies before tackling the full difficulty of the task.

5. Conclusions

This study introduced an open-source system and novel reinforcement learning (RL) design patterns for training and evaluating robotic models in Webots. By integrating RL frameworks with a structured digital twin methodology, we demonstrated an efficient approach for simulating and optimizing robot behavior before deployment in real-world environments. We analyzed three RL design patterns, highlighting the limitations of traditional Supervisor-based implementations and proposing an alternative that removes reliance on Webots' Supervisor class. This new approach, centered around an external robot server for task processing, improves transferability by ensuring a more realistic framework that aligns with physical constraints.

The broader implications of this work are significant for the field of robotics. The proposed methodology and framework are intended to facilitate a more seamless adaptation from virtual training to real-world execution, potentially streamlining the development workflow for roboticists and engineers. The emphasis on the third design pattern directly contributes to addressing the persistent sim-to-real challenge by minimizing reliance on simulator-specific information often unavailable in physical robots. Furthermore, we demonstrated the digital twin process, detailing the steps required for creating an RL-compatible simulation environment. Through techniques like domain randomization and iterative training, the digital twin approach enhances the robustness of RL models, allowing them to generalize better across varying conditions. The effective integration of Webots with Stable-Baselines3 for structured RL experimentation supports scalable and efficient learning, paving the way for more complex DRL investigations within this environment. This structured approach can also yield economic benefits by reducing development time, minimizing risks to physical hardware, and accelerating the deployment of robotic solutions.

Our experiments with the inverted pendulum and Pioneer 3-AT robot models provided insights into the impact of environment complexity on RL training efficiency. While the toy cart example achieved rapid convergence, the more realistic Pioneer 3-AT scenario required significantly longer training durations. The introduction of randomized initial conditions further slowed convergence, emphasizing the challenges inherent in applying RL to complex systems. Despite prolonged training, the persistence of occasional low rewards in the Pioneer 3-AT experiments highlighted the potential for errors in learned policies and underscored that current RL methodologies still require refinement for robust real-world applications.

Reflecting on these findings, several limitations of the current approach also warrant discussion. Firstly, while the third design pattern with an external robot server is proposed for

better real-world transferability, it introduces architectural complexity compared to simpler integrated patterns. The setup, management, and potential communication overhead of this server were not deeply explored and could present challenges. Secondly, the experimental validation, while covering a classic problem and a production-grade robot, was primarily focused on single-agent inverted pendulum tasks. The framework's scalability and the digital twin methodology's robustness in more complex scenarios, such as multi-agent RL or intricate manipulation tasks, require further investigation. Thirdly, the effectiveness of our approach heavily relies on the fidelity of the digital twin; creating and maintaining this accuracy, while balancing computational costs, remains a non-trivial challenge that can directly impact sim-to-real transfer success. The observed difficulties in the Pioneer 3-AT training also point to the broader limitations of current DRL algorithms in terms of generalization and sample efficiency when applied to complex, high-dimensional systems. Finally, while the framework is tailored for Webots, its direct adaptability and the transfer of all its benefits to other simulation platforms may be limited without significant modification.

References

Ayala, A., Cruz, F., Campos, D., Rubio, R., Fernandes, B., & Dazeley, R. (2020). A comparison of humanoid robot simulators: A quantitative approach. In *Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics* (*ICDL-EpiRob*) (pp. 1–6). IEEE. https://doi.org/10.1109/ICDL-EpiRob48136.2020.9278116

Barto, A., Sutton, R., & Anderson, C. (1970). Neuron like elements that can solve difficult learning control problems. *IEEE Transactions on Systems*, *Man and Cybernetics*, *13*, 834–846. https://doi.org/10.1109/TSMC.1983.6313077

Behrens, T., Hindriks, K. V., Bordini, R. H., Lars Braubach, Mehdi Dastani, Dix, J., Hübner, J. F., & Pokahr, A. (2012). An interface for agent-environment interaction. *Lecture Notes in Computer Science*, *6599*, 139–158. https://doi.org/10.1007/978-3-642-28939-2_8

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI gym*. arXiv. https://arxiv.org/abs/1606.01540

Ferigo, D., Traversaro, S., Metta, G., & Pucci, D. (2020, January). Gym-ignition: Reproducible robotic simulations for reinforcement learning. In *2020 IEEE/SICE International Symposium on System Integration* (*SII*) (pp. 885–890). IEEE. https://doi.org/10.1109/SII46433.2020.9025951

Garbev, A., & Atanassov, A. (2020, October). Comparative analysis of RoboDK and robot operating system for solving diagnostics tasks in off-line programming. In *2020 International Conference Automatics and Informatics* (*ICAI*) (pp. 1–5). IEEE. https://doi.org/10.1109/ICAI50593.2020.9311332

Jonban, M. S., Romeral, L., Marzband, M., & Abusorrah, A. (2024). A reinforcement learning approach using Markov decision processes for battery energy storage control within a smart contract framework. *Journal of Energy Storage*, *86*, Article 111342. https://doi.org/10.1016/j.est.2024.111342

Kilinc, O., & Montana, G. (2022). Reinforcement learning for robotic manipulation using simulated locomotion demonstrations. *Machine Learning*, *111*(2), 465–486. https://doi.org/10.1007/s10994-021-06116-1

Kirtas, M., Tsampazis, K., Passalis, N., & Tefas, A. (2020). Deepbots: A webots-based deep reinforcement learning framework for robotics. In I. Maglogiannis, L. Iliadis, & E. Pimenidis (Eds.), *IFIP Advances in Information and Communication Technology: Vol. 584. Artificial Intelligence Applications and Innovations. AIAI 2020* (pp. 64–75). Springer International Publishing. https://doi.org/10.1007/978-3-030-49186-4_6

Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). Exploration in deep reinforcement learning: A survey. *Information Fusion*, *85*, 1–22. https://doi.org/10.1016/j.inffus.2022.03.003

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M., I., & Stoica, I. (2017). *RLLIB: Abstractions for Distributed Reinforcement Learning*. arXiv. https://arxiv.org/abs/1712.09381

Lopez, N. G., Nuin, Y. L. E., Moral, E. B., Juan, L. U. S., Rueda, A. S., Vilches, V. M., & Kojcev, R. (2019). *gym-gazebo2, a toolkit for reinforcement learning using ROS 2 and Gazebo*. arXiv. https://arxiv.org/abs/1903.06278

Michel, O. (2004). Cyberbotics ltd. webots™: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, *1*(1), Article5. https://doi.org/10.5772/5618

NVIDIA. (2025). NVIDIA Isaac ROS. https://developer.nvidia.com/isaac/ros

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1–8. http://www.jmlr.org/papers/v22/20-1364.html

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. arXiv. https://arxiv.org/abs/1707.06347

Talaat, F. M. (2022). Effective deep Q-networks (EDQN) strategy for resource allocation based on optimized reinforcement learning algorithm. *Multimedia Tools and Applications*, *81*(28), 39945–39961. https://doi.org/10.1007/s11042-022-13000-0

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., & Stone, P. (2024). Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, *8*, 153–188. https://doi.org/10.1146/annurev-control-030323-022510

Todorov, E., Erez, T., & Tassa, Y. (2012, October). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5026–5033). IEEE. https://doi.org/10.1109/IROS.2012.6386109

Tseeva, F. M., Shogenova, M. M., Senov, K. M., Liana, K. V., & Bozieva, A. M. (2024). Comparative analysis of two simulation environments for robots, Gazebo, and CoppeliaSim in the context of their use for teaching students a course in robotic systems modeling. In 2024 International Conference "Quality Management, Transport and Information Security, Information Technologies"(QM&TIS&IT) (pp. 186–189). IEEE. https://doi.org/10.1109/QMTISIT63393.2024.10762908

Uslu, E., Cakmak, F., Altuntaş, N., Marangoz, S., Amasyalı, M. F., & Yavuz, S. (2017). An architecture for multi-robot localization and mapping in the Gazebo/Robot Operating System simulation environment. *Simulation*, *93*(9), 771–780. https://doi.org/10.1177/0037549717710098

Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). *Extending the openai gym for robotics: A toolkit for reinforcement learning using ROS and Gazebo*. arXiv. https://arxiv.org/abs/1608.05742

Zhang, L., Shen, L., Yang, L., Chen, S., Yuan, B., Wang, X., & Tao, D. (2022). *Penalized proximal policy optimization for safe reinforcement learning*. arXiv. https://arxiv.org/abs/2205.11814

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., Cola, D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., & Younis, O. G. (2024). *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. ArXiv. https://arxiv.org/abs/2407.17032