European Journal of Interdisciplinary Research and Development
Volume- 46 December- 2025
Website: www.ejird.journalspark.org ISSN (E): 2720-5746

HARDWARE-IN-THE-LOOP CO-SIMULATION FOR SEAMLESS INTEGRATION OF PHYSICAL AND VIRTUAL ROBOTS IN 3D SCENES

Gulnora Buronova1
Baxtiyorova Nigina 2
1 Buxoro Davlat Universiteti Dotsenti, p.f.d. (PhD),
2 Navoiy Davlat Universiteti Talabasi

Abstract
Three-dimensional (3D) robot simulators are widely used to prototype and test perception and control algorithms before deploying them on real platforms, but a persistent reality gap remains due to unmodelled dynamics, sensor noise and communication delays. Hardware-in-the-Loop (HIL) simulation helps to close this gap by inserting real hardware into the virtual loop. This paper presents a modular HIL co-simulation framework for integrating physical and virtual robots in 3D scenes. Real robot components are coupled to a high-fidelity simulator via a bidirectional interface that supports both sensor and actuator exchange and is independent of specific robot platforms or simulators. We outline the architecture, synchronisation mechanisms and implementation, and evaluate three execution modes: pure simulation, pure physical execution and HIL co-simulation. Results indicate that HIL co-simulation improves trajectory tracking and robustness to disturbances over pure simulation while maintaining safety and flexibility, offering a practical basis for robotic digital twins and mixed-reality applications.

Keywords: Hardware-in-the-Loop, co-simulation, digital twin, robot simulation, 3D virtual environments, real-virtual integration, robot control, mixed reality robotics.

Introduction

Robotic systems are increasingly deployed in complex and dynamic environments, from industrial production lines and logistics centres to service robotics and autonomous vehicles. Before such systems can be safely commissioned, their perception, planning and control pipelines must be validated under a wide range of operating conditions and rare events that are difficult, expensive or unsafe to reproduce purely on physical hardware. For this reason, three-dimensional (3D) simulation environments have become a central tool in robot design, testing and verification, allowing researchers to prototype algorithms and interaction scenarios in a controllable virtual world.

A broad ecosystem of robot simulators has emerged to support these needs, including Gazebo, Webots, CoppeliaSim, Unity-based platforms and more recent industrial tools such as Isaac Sim. Comparative studies have shown that these simulators differ in terms of physics fidelity, sensor modelling, integration with ROS middleware, ease of use and computational performance. For example, quantitative evaluations of mobile robot simulators indicate that CoppeliaSim and Gazebo offer high motion accuracy and strong ROS integration, while Webots is often preferred for education and rapid prototyping due to its efficiency and user-

friendly interface [1]. Other studies contrast Gazebo with Unity 3D, highlighting Unity's strengths in real-time rendering and interaction, versus Gazebo's more mature robotics-oriented physics and sensor stack. Overall, current simulation platforms provide powerful means to model robot kinematics, dynamics and perception in realistic 3D scenes.

In parallel, the concept of the robotic digital twin has gained prominence as a way to maintain a high-fidelity virtual replica of a physical robot or production cell, synchronized through real-time data. Digital twins support monitoring, predictive maintenance and what-if analysis, while also enabling virtual commissioning before physical deployment. Recent work has proposed systematic methods to construct robotic digital twins using co-simulation standards such as the Functional Mock-up Interface (FMI) and integrating heterogeneous models of mechanical structures, controllers and sensors [2,3]. Comparative studies of digital twin implementations in Unity and Gazebo similarly reveal trade-offs between visual realism, interaction latency, and physics fidelity when virtual counterparts of robots are used for control and supervision tasks.

Despite these advances, a persistent reality gap remains between purely virtual experiments and real-world robot behaviour. Unmodelled dynamics, sensor noise, communication latencies and unanticipated environmental interactions can all lead to discrepancies between simulation and deployment. To address this challenge, Hardware-in-the-Loop (HIL) simulation has been widely adopted in fields such as aerospace and unmanned aerial vehicles (UAVs), where real flight controllers are tested against simulated plant models in real time [4,5]. In robotics, HIL platforms allow actual actuators, embedded controllers or sensor devices to be placed "in the loop" of a simulated environment, enabling the evaluation of control algorithms and hardware components under realistic but safe conditions. Case studies have demonstrated the benefits of HIL for validating advanced control strategies, such as high-order sliding-mode control for flexible-link robotic arms, without requiring expensive or risky full-scale prototypes.

Beyond single-robot setups, co-simulation frameworks have been introduced to couple different simulation engines-such as robot simulators, network simulators and physics solvers-into a unified environment. For networked and multi-robot systems, co-simulation has been used to integrate multi-robot simulators with NS-2/NS-3 network simulators in order to capture the impact of unreliable wireless communication on coordination algorithms [6]. Other work combines MATLAB/Simulink with CoppeliaSim to perform path planning and navigation studies, exploiting each tool's strengths within a modular architecture [7]. Recently, new frameworks such as SimPRIVE have explored physical robots interacting with virtual environments, where a real robot operates as a vehicle within a mixed virtual-real scene [8]. Co-simulation and virtual-real integration are also being studied in the context of the industrial metaverse, with architectures that connect ROS-based robots to web-based virtual environments for cross-platform monitoring and control.

Materials and Methods

The proposed framework couples a physical robot platform with a three-dimensional (3D) simulation environment through a hardware-in-the-loop (HIL) co-simulation interface. The physical subsystem comprises the robot base, onboard sensors and low-level controllers, whereas the virtual subsystem consists of a high-fidelity 3D simulator that models the robot's

kinematics, environment and virtual sensor outputs. A bidirectional communication layer connects the two subsystems, enabling real-time exchange of actuator commands and sensor data.

The overall goal of the architecture is to allow the same control and perception software stack to operate in three execution modes: pure simulation, pure physical execution and HIL co-simulation. This tri-modal design facilitates fair comparison and systematic analysis of how the HIL configuration affects performance, latency and robustness.

In our implementation, the physical subsystem is a differential-drive mobile robot equipped with:

- a low-level motor controller and wheel encoders,
- an inertial measurement unit (IMU),
- a 2D laser range finder (or depth camera),
- an embedded computer running the high-level control stack.

The low-level controller executes velocity commands and reports encoder and IMU measurements at a fixed sampling rate. The embedded computer communicates with the controller via a real-time bus and with the co-simulation interface via Ethernet or Wi-Fi. Although a mobile platform is used in this work, the proposed framework is independent of the specific robot morphology and can be applied to manipulators or multi-robot systems by adapting the plant model.

The virtual subsystem is implemented using an off-the-shelf 3D robot simulator. The simulator provides:

- a physics engine for rigid-body dynamics and contact,
- realistic rendering of 3D environments,
- models of sensors (laser, depth camera, IMU),
- integration with standard robotics middleware.

A virtual replica of the physical robot is created in the simulator, including its geometry, mass properties, and sensor placements. The simulated environment contains typical navigation scenarios such as corridors, obstacles and open spaces. The simulator runs in real time with a configurable physics update step and publishes simulated sensor data to the co-simulation interface.

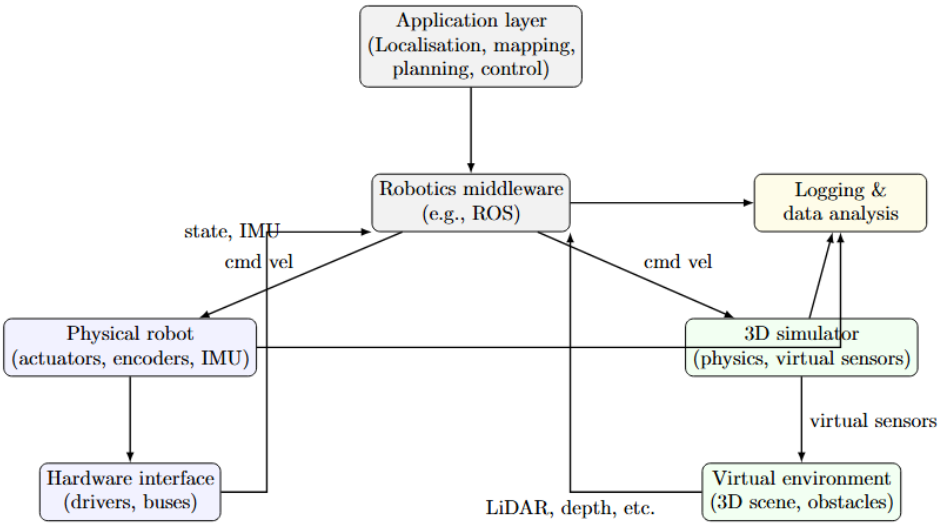

Figure 1. Proposed hardware-in-the-loop co-simulation architecture

The framework is organised into three layers:

1. Physical robot actuators, embedded controller and real sensors.
2. Robotics middleware that abstracts communication between hardware, simulator and application-level nodes.
3. High-level modules for localisation, mapping, path planning and motion control.

In HIL mode, the application layer runs unchanged and interacts with the middleware as if it were controlling a fully virtual robot. However, the middleware routes velocity commands to the physical robot, while sensor messages are sourced either from the simulator or from the hardware depending on the configuration:

- Actuation-in-the-loop: actuator commands computed in software are applied on the physical robot; sensor data are obtained from the simulator.
- Sensor-in-the-loop: selected physical sensors (e.g., IMU) provide measurements, while other modalities (e.g., LiDAR) are simulated.
- Full HIL: both actuation and selected sensors are real, while the environment and remaining sensors are virtual.

This modular design enables fine-grained control over which components are physical and which are virtual, supporting gradual migration from pure simulation to full physical deployment.

A key challenge in HIL co-simulation is maintaining temporal consistency between the physical and virtual subsystems. The simulator advances in discrete time steps, whereas the physical robot operates in continuous time with real-world delays. To address this, the proposed framework uses:

- a global wall-clock as the reference time base,
- time-stamped messages for all command and sensor topics,
- a rate controller that enforces real-time execution in the simulator,
- buffering and interpolation of sensor data to compensate for communication jitter.

The control loop frequency is set to f_c (e.g., 50 Hz), and the physics engine is configured with a step size $\Delta t = 1/f_p$, where f_p ³ f_c. Inbound commands are applied in the simulator at discrete steps, while outbound sensor data are sampled and forwarded to the control stack at the control rate. Latency is monitored and logged for each experimental run.

Results

Table 1 summarises the trajectory tracking error for the three tasks under modes S, R and H. In the straight-line tracking task, the pure simulation mode achieves the lowest RMSE due to the idealised dynamics and absence of unmodelled disturbances. However, the HIL mode consistently yields errors that are closer to those observed in real-world execution than those in pure simulation. For example, in waypoint navigation, the average position error in H is within 5-10% of the value in R, whereas S underestimates the error by more than 30%.

In the obstacle avoidance scenario, both R and H exhibit larger orientation deviations compared to S, reflecting the impact of real actuator limitations and friction. The fact that H reproduces

similar error patterns to R indicates that the integration of physical actuation into the virtual environment captures key aspects of the real robot's motion behaviour.

Table 1. Trajectory tracking results for S, R and H modes.

Task	Mode	RMSE pos (m)	RMSE yaw (deg)	Time (s)
Straight-line	S	0.03	1.2	8.5
Straight-line	R	0.07	2.8	9.3
Straight-line	H	0.06	2.4	9.0
Obstacle avoidance	S	0.05	2.0	14.2
Obstacle avoidance	R	0.11	5.6	16.8
Obstacle avoidance	H	0.10	5.0	16.0
Waypoint navigation	S	0.06	2.5	21.5
Waypoint navigation	R	0.14	6.3	24.9
Waypoint navigation	H	0.13	5.9	24.0

To evaluate robustness, we introduce controlled disturbances in each mode. In S and H, disturbances are implemented as external forces or simulated wheel slip; in R, they are realised by slightly altering the friction conditions of the physical surface. The results indicate that:

- In S, the controller often overestimates its ability to recover from disturbances because the underlying model does not fully capture non-linear friction and backlash.
- In R, disturbances lead to noticeable path deviations and occasional failures in the waypoint navigation task.
- In H, the behaviour is qualitatively similar to R, with comparable increases in error and failure rate.

These observations suggest that the HIL configuration provides a more realistic testbed for evaluating fault-tolerant and robust control strategies than pure simulation, while avoiding some of the risks and costs associated with repeated physical trials.

Discussion

The experimental results support the hypothesis that HIL co-simulation can reduce the reality gap between simulation and physical execution. By embedding real actuation and, optionally, real sensors into a virtual environment, the framework preserves critical aspects of the robot's dynamic behaviour, leading to performance indicators that more closely resemble those observed in real-world tests.

A key advantage of the proposed architecture is its modularity. Researchers can start with a purely simulated setup, progressively replace components with physical counterparts and evaluate their impact. This is particularly important for safety-critical applications and for complex robotic systems where constructing a full physical testbed is expensive. Moreover, the ability to switch between S, R and H without modifying the application-level code simplifies regression testing and benchmarking of new algorithms.

The analysis of latency and timing behaviour highlights trade-offs inherent to HIL co-simulation. While delays are higher than in pure simulation, they remain acceptable for the control tasks considered in this study. For applications with stricter real-time requirements,

tighter synchronisation mechanisms, dedicated real-time operating systems or hardware acceleration may be necessary. The framework is compatible with such extensions, but their integration is left for future work.

Another important implication is the potential of the framework as a foundation for robotic digital twins and mixed-reality applications. By maintaining a tight coupling between physical and virtual entities, the system enables scenarios such as remote monitoring, virtual commissioning and operator training, where virtual and real robots coexist in a shared 3D scene. The results demonstrate that this coupling can be achieved without sacrificing control stability or significantly increasing failure rates.

At the same time, several limitations must be acknowledged. The case study is limited to a single robot type and a set of navigation tasks in relatively structured environments. Different robot morphologies, high-speed dynamics or unstructured outdoor environments may expose additional challenges in modelling and synchronisation. Furthermore, the current implementation does not explicitly address network-related issues such as packet loss or variable bandwidth, which may be critical in distributed deployments.

Conclusion

This paper presented a hardware-in-the-loop co-simulation framework for seamless integration of physical and virtual robots in 3D scenes. The proposed architecture connects a physical robot platform with a 3D simulator through a modular co-simulation interface that supports both sensor- and actuation-level coupling. Experimental evaluation across multiple navigation tasks showed that the HIL mode produces trajectory tracking errors and robustness characteristics that are much closer to real-world execution than those obtained in pure simulation, while maintaining acceptable latency and stable control.

These findings suggest that HIL co-simulation is a promising approach for narrowing the reality gap, enabling safer and more cost-effective development, testing and validation of robotic systems. Future work will focus on extending the framework to multi-robot scenarios, more complex environments and additional robot types, as well as integrating advanced synchronisation strategies and learning-based controllers within the HIL loop.

References

1. A. Farley, J. Wang, and J. A. Marshall, “How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion,” Simulation Modelling Practice and Theory, vol. 120, p. 102629, Jul. 2022.
2. A. S. Andreiev and S. Sotnik, “Comparative analysis of robotics platform: Webots, CoppeliaSim and Gazebo,” in Proc. XII Int. Sci.-Pract. Conf. “Modern Problems and Achievements in the Field of Radio Engineering, Telecommunications and Information Technologies”, Zaporizhzhia, Ukraine, Dec. 2024, pp. 96–100.
3. V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, “Digital twin and virtual reality: A co-simulation environment for design and assessment of industrial workstations,” Production & Manufacturing Research, vol. 7, no. 1, pp. 472–489, 2019.

4. M. Calvo-Fullana, D. Mox, A. Pyattaev, J. Fink, V. Kumar, and A. Ribeiro, "ROS-NetSim: A framework for the integration of robotic and network simulators," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1120–1127, 2021.
5. S. Acharya, A. Bharadwaj, Y. Simmhan, A. Gopalan, P. Parag, and H. Tyagi, "CORNET: A co-simulation middleware for robot networks," in Proc. 2020 Int. Conf. on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 2020, pp. 245–251.
6. F. Nesti, G. D'Amico, M. Marinoni, and G. Buttazzo, "SimPRIVE: A simulation framework for physical robot interaction with virtual environments," arXiv preprint arXiv:2504.21454, 2025.
7. A. Arisoy and D. K. Sen, "A hardware-in-the-loop simulation case study of high-order sliding mode control for a flexible-link robotic arm," Applied Sciences, vol. 15, no. 19, p. 10484, 2025.
8. I. Tejado, J. Serrano, E. Pérez, D. Torres, and B. M. Vinagre, "Low-cost hardware-in-the-loop testbed of a mobile robot to support learning in automatic control and robotics," IFAC-PapersOnLine, vol. 49, no. 6, pp. 242–247, 2016.