Universität Osnabrück
Institut für Informatik
AG Technische Informatik
Prof. Dr.-Ing. Mario Porrmann

Master's Thesis

Mixed Reality Environment for Robot-in-the-Loop-Testing using Digital Twins

Jonas Kittelmann

Betreuer: Prof. Dr.-Ing. Mario Porrmann
M.Sc. Philipp Gehricke

Osnabrück, 23 December 2025 M 00

Abstract

Hier sollte in einem Abstract kurz der Inhalt der Arbeit erläutert werden.
Zuerst auf deutsch.

Then an abstract of the thesis in english should follow.

Contents

1 Introduction

Robotic and Autonomous Systems (RAS) combine multiple disciplines, including control engineering and robotics, mechanical engineering, electronics, and software engineering. Testing these systems is not as straightforward as using traditional methods, and they require more because they span multiple disciplines. For researchers and engineers specializing in software testing, adapting existing techniques for RAS presents a significant challenge. This is why there is extensive literature dedicated to proposing and evaluating different testing techniques and processes, showing how essential it is to establish structured methods for ensuring the reliability of these complex systems. [AMV23]

This validation challenge highlights a fundamental tension between simulation-only testing and full-scale physical experimentation, which are the two established methods in robotics evaluation. Simulation is central to the robotic development, offering a way to test control logic and experiment with system configurations before committing them to hardware [Hu05; Mic04]. Simulations are often easier to set up, less expensive, and can run faster than real-world tests, allowing for rapid design exploration and to get a better grasp of complex algorithms [Mic04; DRC+17; BM18]. However, a gap between the virtual and physical worlds emerges when simulation models are abandoned during the implementation phase. Virtual models, by their nature, cannot perfectly represent every characteristic of physical hardware, such as the exact effects of friction, sensor noise, or small differences in motors [Hu05; BM18]. On the other hand, testing with real hardware provides the highest fidelity and allows for controls to be refined based on the presence of all the unpredictable physics and variability that simulations can't replicate [Hu05; CCC+21]. But this approach has its own severe drawbacks. Conducting physical experiments can be resource-intensive and time-consuming and require significant funds, manpower, and infrastructure [Hu05; DRC+17; Mic04]. Furthermore, some critical scenarios, such as emergencies, are too dangerous to be recreated in the physical world [Hu05]. For complex, large cooperative systems, testing the entire system with only real components may not even be feasible due to the issues of complexity and scale [Hu05]. Since neither approach on its own is

sufficient, there is a need for an intermediate solution that can connect the simulation with real-world experimentation [Hu05].

Hybrid methods such as Hardware-in-the-Loop Simulation (HILS) have emerged to bridge the validation gap by replacing segments of pure simulations with actual hardware components [Hu05]. This research focuses on a specific application of this concept known as "robot-in-the-loop" (RitL) simulation, a more recent approach that allows physical robots and digital robot models to work together for comprehensive system testing and evaluation [Hu05]. RitL allows researchers to work with actual robots inside a simulated environment, even when a full physical setup is unavailable [Hu05]. Central to this approach is the Digital Twin [AA23]. The digital twin is a live, virtual replica of a physical robot that tracks its real-world counterpart in real time [AA23]. With a continuous connection and rapid data exchange, the digital twin duplicates the robot's actions, ensuring both remain in sync [AA23]. To manage this hybrid test environment, Augmented Reality (AR) is increasingly utilized as a way for interaction and information exchange with autonomous systems, enhancing the efficiency of Human-Robot Interaction (HRI) [MV20]. The combination of RitL testing [Hu05], synchronized Digital Twins [AA23], and AR interfaces [MV20] paves the way for the development and validation of future robotic systems.

The "Virtual Environment for mobile Robotic Applications" (VERA) framework integrates digital twins, AR, and vehicle-in-the-loop testing together into one system [Geh24]. VERA provides a modular platform for creating, managing, and visualizing synchronized virtual environments, which are presented both in simulations and as real-world projections [Geh24]. This master's thesis builds directly upon the foundation laid by this original framework.

The VERA framework provides a good foundation, but its initial version has several limitations that this thesis seeks to address [Geh24]. Firstly, the custom environment manager does not feature a full physics simulation, with the integration of enhanced physics noted as a direction for future work [Geh24]. The 2D visualizer that is created with Pygame faced performance problems while handling a higher number of dynamic objects [Geh24]. If there are too many simultaneous updates, its capacity was exceeded, which led to delayed and incomplete visualizations [Geh24]. Additionally, while the framework supports AR projections, a more immersive Virtual Reality (VR) interaction was not implemented and was identified as potential future work [Geh24]. Therefore, the main problem is that although VERA's concepts are promising, its custom components have technical constraints that limit it, particularly regarding physical realism, scalability, and user interaction [Geh24].

This thesis proposes replacing VERA's custom simulation and visualization components [Geh24] with the Unity Engine [Uni23]. This implementation will use the Unity Engine [Uni23], which was selected for its graphics, integrated physics, and VR support, because these features directly address the technical limitations previously restricting VERA.

The core goal is to create a real-time digital twin [AA23] of the EMAROS robot [Geh24], integrating its complete model, live sensor data, and physical properties such as mass, inertia, and collision models. This digital twin will serve as the foundation for "robot-in-the-loop" [Hu05] testing scenarios. Additionally, this project will implement interaction by extending the original AR projections [Geh24] to include Virtual Reality (VR) [EM21] and desktop interfaces for scenario modification, robot control, and data visualization. This also includes reimplementing and enhancing VERA's AR floor projection feature by using Unity's rendering tools to improve visual quality and system capabilities [Uni23].

[**NOT FINAL**]The remainder of this thesis is structured as follows: Chapter 2 reviews the foundational concepts of Robot-in-the-Loop testing, Digital Twins, and Mixed Reality, and includes a detailed review of the original VERA framework. Chapter 3 defines the new system requirements based on VERA's limitations. Chapter 4 details the implementation of the new architecture, including the Unity/ROS 2 integration, the EMAROS digital twin and the implemented scenarios. Chapter 5 presents the evaluation and discusses the results. Finally, Chapter 6 summarizes the thesis contributions and provides an outlook on future research.[**NOT FINAL**]

2 Background and Related Work

Einführung in das kapitel [**NOT FINAL**]

2.1 Robot-in-the-Loop-Testing

The validation and verification of modern Robotic and Autonomous Systems (RAS) is a significant challenge due to their complexity [AMV23]. This is because they integrate software, mechanical, and electrical engineering all at once. [AMV23]. A central problem is ensuring that the software and hardware work together seamlessly, especially when testing both simultaneously [AMV23]. The X-in-the-Loop (XitL) simulation paradigm addresses this challenge [Bra+19]. It offers a way to combine the flexibility of software simulation with the realism of physical experiments [Bra+19].

A foundational XitL method is Hardware-in-the-Loop (HIL) simulation, which is a technique for testing mechatronic systems [Mih+22; Bra+19]. The main principle of HIL is creating a closed loop between the real hardware that is being tested and a simulation that represents the rest of the system or its operational environment [Mih+22]. This setup effectively tricks the hardware into behaving as if it were operating in a real system, which allows for testing across a wide range of virtual scenarios [Bra+19]. The main reason for using HIL is to shorten development cycles and prevent costly or dangerous failures by making exhaustive testing possible before the system is actually completely implemented [Mih+22].This is why HIL is essential in industries like automotive, aerospace, and robotics, where real-world testing can be too expensive, dangerous, or even impossible [Bra+19].

Robot-in-the-Loop (RitL) [Mih+22] simulation extends the Hardware-in-the-Loop (HIL) concept. Instead of just a component, the hardware under test is a complete robotic system, such as an uncrewed vehicle [Mih+22]. RitL replaces components of an pure simulated setup with the actual robot, increasing the realism of testing [Hu05]. As illustrated in Figure **??**, a typical RitL configuration has the robot's real actuators operating in the physical world, while while its sensors interact with a simulated

environment instead (Hu05, Mihalic2022). This creates a hybrid setup where, for example, the robot might use virtual sensors to see objects in the simulation but use its real motors to move physically [Hu05]. To keep the physical and virtual worlds synchronized, the real robot often has a virtual counterpart in the simulation which state is updated as the physical robot acts and moves [Hu05].

Figure 2.1: Conceptual diagram of a Robot-in-the-Loop (RitL) simulation, showing the real robot interacting with a virtual environment. The real robot's actions affect its virtual counterpart, and the virtual environment provides sensor data back to the real robot.

The key advantage of RitL is that it allows repeatable testing of high level software, like navigation algorithms, while still using the dynamics of real hardware [Mih+22]. This approach provides a safe and practical alternative to full physical testbeds and avoiding their cost, complexity, and risk [Mih+22]. As a result, RitL becomes an vital tool for safely evaluating system performance in the lab [Hu05; Mih+22]. This is especially important when safety is at stake or when working on projects that are difficult to reproduce, like Mars rover missions or large scale robot swarms [Hu05; Mih+22].

The RitL paradigm has been widely applied to validate complex autonomous systems, particularly in the automotive field using Vehicle-in-the-Loop (VitL) testing. For example, the Dynamic Vehicle-in-the-Loop (DynViL) architecture integrates a real test vehicle with the high-fidelity CARLA simulator, which operates on the Unreal Engine [DSR+22]. As shown in Figure **??**, this approach allows a vehicle on an empty track to be stimulated by sensor data from a virtual world [DSR+22]. This enables the safe and repeatable testing of automated driving functions in scenarios that would be dangerous to replicate physically [DSR+22]. A similar Vehicle-in-Virtual-Environment (VVE) method also uses CARLA and the Unreal Engine to create a closed loop where the real vehicle's motion is tracked and reflected in the virtual world, allowing its control systems to react to simulated events [Cao+23].

Figure 2.2: An example of a Vehicle-in-the-Loop (VitL) setup, illustrating a real car on a test track connected to a high-fidelity simulator like CARLA that generates virtual traffic and sensor data. Adapted from Drechsler et al. (2022) and Cao et al. (2023).

These examples show a trend towards using game engine based simulators to evaluate autonomous vehicles. This approach sits between pure software simulation, which often lacks vehicle dynamics fidelity, and physical testing, where safety and consistency can

pose challenges [Cao+23]. Some systems even stimulate the vehicle's actual sensors. For instance, the Radar Target Simulator (RTS) can feed artificial radar echoes from a virtual scene to the vehicle's real radar sensor [Die+21]. This allows for end-to-end validation of the entire perception and control pipeline [Die+21].

The X-in-the-Loop approach is not limited to a single domain, it is used in a variety of fields. In marine robotics, a VIL framework was developed to test the long term autonomy of a robot swarm. In this system, a Autonomous Surface Vehicle (ASV) interacts with many simulated underwater sensor nodes to validate complex cooperative behaviors without the high logistical cost of deploying a full swarm [Bab+20]. In the aerospace domain, the RFlySim platform uses an FPGA-based HIL system to create a high fidelity simulation for testing UAV autopilot systems in a lab, which reduces the need for expensive and risky outdoor flights [Dai+21].

These approaches continue progressing toward deeper integration. This includes the introduction of Scenario-in-the-Loop (SciL) frameworks that aim to completely blur the lines between real and virtual testing [Sza+21]. These systems rely on creating comprehensive digital twins of the entire test environment and combining real and virtual components to run complex, mixed reality test scenarios [Sza+21].

2.2 Digital Twins

A key concept that underpins many modern "X-in-the-Loop" frameworks is the Digital Twin (DT). The DT serves as the virtual counterpart to a physical system, acting as a virtual copy that enables real-time monitoring and simulation [AA23]. The core idea is to create a digital information model of a physical system that stays linked to it throughout its entire lifecycle [Gri17].

This model, first called the "Information Mirroring Model," has three primary components: the physical product, its corresponding virtual model, and the data connection that links them [Gri17; AA23]. A conceptual diagram of this structure is shown in Figure **??**. The virtual model is more than a simple geometric shape; it is often a complex simulation that can model a system's mechanical, electrical, and software properties [Len+21]. The data connection is bi-directional, allowing sensor data to flow from the real world to update the virtual model. In turn, the virtual model can send commands back to control or optimize the physical system [Gri17; Len+21]. This continuous, synchronized data exchange is the defining feature of a Digital Twin [AA23].

Figure 2.3: The foundational concept of a Digital Twin, illustrating the three core components: the physical entity, the virtual model, and the bi-directional data connection that links them. Adapted from Grieves et al. (2017).

In robotics, Digital Twins are often built using specialized simulation software. The Robot Operating System (ROS) typically acts as the middleware connecting the virtual simulation to the physical hardware. For instance, Stączek et al. used the Gazebo simulator with ROS to create a DT of a factory floor, which they used to test and optimize the navigation algorithms of a mobile robot in narrow corridors [Stą+21]. A similar approach was taken by R. Singh et al., who developed a DT of a greenhouse in Gazebo to validate a deep learning-based harvesting robot [Sin+24b]. Other simulators like CoppeliaSim and Webots are also common. Magrin et al. used CoppeliaSim and ROS to create a DT as a learning tool for mobile robot design [Mag+21], while Marques et al. used Webots for an Automated Guided Vehicle (AGV), synchronizing it via an OPC-UA server [Mar+24]. These examples show a common method of using traditional simulators to model robot kinematics and sensor feedback for closed-loop testing.

More recently, modern game engines have become a popular choice for creating Digital Twins, as they offer higher-fidelity graphics and more realistic physics. The Unity game engine, in particular, is used to build powerful, performant, and visually rich virtual environments. This can be seen in Figure **??**. For example, Pérez et al. used Unity3D to develop a DT of a multi-robot cell that included an immersive Virtual Reality (VR) interface for virtual commissioning and operator training [Pér+20].

Figure 2.4: A comparison of robotic Digital Twin environments, showing a typical view from a traditional simulator (left) versus a high-fidelity visualization from a modern game engine like Unity (right).

The technical capabilities of these engines are also a key factor. Yang et al. used Unity and its integrated NVIDIA PhysX engine to simulate the complex physics of a UAV. They also demonstrated how to create virtual sensors, such as a LiDAR, directly within the game engine using its raycasting API [Yan+20]. The performance and reliability of these game engine-based frameworks have also been a focus of research. Kwon et al. developed a safety-critical DT in Unity and ROS 2, achieving a low data-transmission latency of just 13 ms for predicting collisions [Kwo+25]. Similarly, M. Singh et al. created a Unity and ROS-based DT and conducted extensive performance validation, reporting a communication latency of 77.67 ms and a positional accuracy of 99.99% [Sin+24a].

2.3 Mixed Reality

Beyond the testing paradigm and the digital twin, the way a user interacts with the system is another critical part of a modern robotics framework. Virtual, Augmented, and Mixed Reality (VAM) have become promising technologies for improving the information exchange between humans and robots [Wal+23]. In robotics, Augmented Reality (AR) is an especially powerful tool for enhancing Human-Robot Interaction (HRI) by integrating 3D virtual objects into a real-world environment in real-time [MV20].

The relationship between these technologies is formally described by the foundational "Reality-Virtuality (RV) Continuum" concept, first introduced by Milgram and Kishino [MK94; Ska+21; MV20]. As illustrated in Figure **??**, this continuum is a scale that is anchored by a purely real environment at one end and a completely virtual one at the other [MK94; Ska+21; MV20].

Figure 2.5: The Reality-Virtuality Continuum, illustrating the spectrum from a real environment to a completely virtual one. Mixed Reality (MR) encompasses both Augmented Reality (AR) and Augmented Virtuality (AV). Adapted from Milgram & Kishino (1994) [MK94] and Walker et al. (2023) [Wal+23].

A conventional Virtual Reality (VR) environment represents the continuum's virtual endpoint. In VR, the user is totally immersed in and can interact with a completely synthetic world [MK94]. This approach is very useful for HRI research, as it allows for testing interactions with virtual robots in scenarios that might be unsafe or too expensive for physical hardware [Wal+23]. The general term Mixed Reality (MR) describes the entire spectrum between the two extremes, where real and virtual worlds are merged within a single display [MK94]. MR is made up of two main categories: Augmented Reality (AR) and Augmented Virtuality (AV) [MK94; MV20]. AR involves augmenting a real environment with virtual objects [MK94]. This allows HRI researchers to place a robot's 3D data and intentions directly into the user's physical space [Wal+23]. The opposite is Augmented Virtuality (AV), where a primarily virtual world is enhanced with elements from the real world, like live video feeds [MK94].

2.4 The VERA Framework and EMARO

2.5 Comparison of Simulation Engines?

3 Requirements

4 Concept and Implementation

4.1 System Architecture

4.1.1 Integration of Unity, ROS 2, and EMAROS

components, their responsibilities, and deployment setup.

4.1.2 Communication Concept

Topics, message types, timing, synchronization ...

4.2 Development Milestones

4.2.1 Base Integration

Initial data flow, messaging bridge, and minimal scenes.

4.2.2 Realization of the Digital Twin

Robot modeling, environment setup, and synchronization.

4.2.3 Interaction Scenarios

Scenarios and scripting.

4.2.4 VR Integration

Devices and implementation, mabybe before Interaction Scenarios?

4.2.5 Pure Virtual Model

Implementation and use cases without the physical counterpart.

5 Evaluation

5.1 Methodology

5.1.1 Test Scenarios

Define representative scenarios and justification.

5.1.2 Metrics

Latency, frame rate, synchronization accuracy, and resource usage.

5.2 Execution and Results

5.2.1 Base Integration Performance

baseline measurements and analysis

5.2.2 Scenario-dependent Performance

performance under different interaction and load conditions.

5.3 Discussion

results, limitations, and implications.

6 Conclusion

summary of the work and discussion of future research directions.

List of Figures

Listings

Bibliography

[AA23] K. K. Alnowaiser and M. A. Ahmed. “Digital Twin: Current Research Trends and Future Directions”. In: *Arabian Journal for Science and Engineering* 48.2 (2023), pp. 1075–1095.

[AMV23] Hugo Araujo, Mohammad Reza Mousavi, and Mahsa Varshosaz. “Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review”. In: *ACM Transactions on Software Engineering and Methodology* 32.2 (2023). ISSN: 1049-331X. DOI: 10.1145/3542945. URL: https://doi.org/10.1145/3542945.

[Bab+20] Author Babic et al. “Vehicle-in-the-Loop in Marine Robotics”. In: *Proceedings*. Update with full citation. 2020.

[BM18] Mordechai Ben-Ari and Francesco Mondada. *Elements of Robotics*. Springer Open, 2018. ISBN: 978-3-319-62533-1. DOI: 10.1007/978-3-319-62533-1.

[Bra+19] Author Brayanov et al. “X-in-the-Loop Simulation Overview”. In: *Proceedings*. Update with full citation. 2019.

[Cao+23] Author Cao et al. “Vehicle-in-Virtual-Environment method”. In: *Journal/-Conference* (2023). Update with full citation.

[CCC+21] Giuseppina Lucia Casalaro, Giulio Cattivera, Federico Ciccozzi, et al. “Model-driven engineering for mobile robotic systems: a systematic mapping study”. In: *Software and Systems Modeling* 21 (2021), pp. 19–49. DOI: 10.1007/s10270-021-00908-8.

[Dai+21] Author Dai et al. “RFlySim: FPGA-based HIL for UAVs”. In: *Proceedings*. Update with full citation. 2021.

[Die+21] Author Diewald et al. “Radar Target Simulator for End-to-End Validation”. In: *Proceedings*. Update with full citation. 2021.

[DRC+17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, et al. “CARLA: An Open Urban Driving Simulator”. In: *1st Conference on Robot Learning (CoRL 2017)*. 2017. arXiv: 1711.03938.

[DSR+22] Maikol Drechsler, Varun Sharma, Fabio Reway, et al. “Dynamic Vehicle-in-the-Loop: A Novel Method for Testing Automated Driving Functions”. In: *SAE International Journal of Connected and Automated Vehicles* 5 (June 2022), pp. 1–14. DOI: 10.4271/12-05-04-0029.

[EM21] Maria Engberg and Blair Macintyre. *Reality Media: Augmented and Virtual Reality*. Nov. 2021. ISBN: 9780262366250. DOI: 10.7551/mitpress/11708.001.0001.

[Geh24] P. Gehricke. “Virtual Environment for mobile Robotic Applications”. MA thesis. Universität Osnabrück, 2024.

[Gri17] Michael Grieves. *Origins of the Digital Twin Concept*. White paper; update with full citation. 2017.

[Hu05] Xiaolin Hu. “Applying robot-in-the-loop-simulation to mobile robot systems”. In: *2005 IEEE International Conference on Robotics and Automation*. 2005, pp. 506–513. ISBN: 0-7803-9178-0. DOI: https://doi.org/10.1109/ICAR.2005.1507456.

[Kwo+25] Author Kwon et al. “Safety-critical DT in Unity + ROS 2”. In: *Journal/Conference* (2025). Update with full citation.

[Len+21] Author Leng et al. “Digital Twin Technologies Review”. In: *Journal/Conference* (2021). Update with full citation.

[Mag+21] Author Magrin et al. “DT for Mobile Robot Design with CoppeliaSim”. In: *Proceedings*. Update with full citation. 2021.

[Mar+24] Author Marques et al. “Webots-based AGV with OPC-UA Synchronization”. In: *Journal/Conference* (2024). Update with full citation.

[Mic04] Olivier Michel. “Webots TM: Professional Mobile Robot Simulation”. In: *International Journal of Advanced Robotic Systems* 1.1 (2004), pp. 39–42. ISSN: 1729-8806.

[Mih+22] Author Mihalic et al. “Hardware-in-the-Loop for Mechatronics”. In: *Journal/Conference* (2022). Update with full citation.

[MK94] P. Milgram and F. Kishino. “A Taxonomy of Mixed Reality Visual Displays”. In: *IEICE Transactions on Information Systems* E77-D.12 (1994), pp. 1321–1329.

[MV20] Z. Makhataeva and H. A. Varol. “Augmented Reality for Robotics: A Review”. In: *Robotics* 9.2 (2020), p. 21.

[Pér+20] Author Pérez et al. “Unity3D DT for Multi-robot Cell with VR”. In: *Proceedings*. Update with full citation. 2020.

[Sin+24a] M. Singh et al. “Performance Validation of Unity + ROS DT”. In: *Journal/-Conference* (2024). Update with full citation.

[Sin+24b] R. Singh et al. “Greenhouse DT for Harvesting Robot in Gazebo”. In: *Proceedings*. Update with full citation. 2024.

[Ska+21] Richard Skarbez et al. “Revisiting the RV Continuum”. In: *Journal/Conference* (2021). Update with full citation.

[Stą+21] Author Stączek et al. “Factory Floor DT with Gazebo and ROS”. In: *Proceedings*. Update with full citation. 2021.

[Sza+21] Author Szalay et al. “Scenario-in-the-Loop (SciL) Frameworks”. In: *Journal/Conference* (2021). Update with full citation.

[Uni23] Unity Technologies. *Unity*. Version 2023.2.3. Game development platform. 2023. URL: `https://unity.com/`.

[Wal+23] Author Walker et al. “VAM for HRI”. In: *Journal/Conference* (2023). Update with full citation.

[Yan+20] Author Yang et al. “Unity + PhysX for UAV and Virtual Sensors”. In: *Proceedings*. Update with full citation. 2020.

Declaration of Authorship

I hereby declare that I have written this thesis independently and without unauthorized assistance. I have not used any sources or aids other than those indicated. All passages taken verbatim or in spirit from the works of other authors have been properly acknowledged and cited.

Osnabrück, 23 December 2025

Jonas Kittelmann