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 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 [AAR+19]. It offers a way to combine the flexibility of software simulation with the realism of physical experiments [AAR+19].

A foundational XitL method is Hardware-in-the-Loop (HIL) simulation, which is a technique for testing mechatronic systems [MTH22; AAR+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 [MTH22]. 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 [AAR+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 [MTH22].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 [AAR+19].

Robot-in-the-Loop (RitL) [MTH22] 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 [MTH22]. RitL replaces components of an pure simulated setup with the actual robot, increasing the realism of testing [Hu05]. [**NOT FINAL**]As illustrated in Figure **??**[**NOT FINAL**], 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; MTH22]. 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: [**NOT FINAL**]The real robot's actions affect its virtual counterpart, and the virtual environment provides sensor data back to the real robot.[**NOT FINAL**]

The main benefit of RitL is that it uses the dynamics of actual hardware for repeatable testing of high-level software, such as navigation algorithms [MTH22]. This method avoids the expense, complexity, and risk of full physical testbeds while being a secure and useful substitute [MTH22]. RitL becomes therefore an tool for safely evaluating system performance in the lab [Hu05; MTH22]. This is especially important when working on projects that are hard to replicate, such as large-scale robot swarms or Mars rover missions, or when safety is at risk [Hu05; MTH22].

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 2.2, this approach allows a vehicle on an empty track to be stimulated by sensor data from a virtual world [DSR+22]. This lets automated driving functions to be safely and reliably tested in situations that would be hazardous to physically replicate [DSR+22]. CARLA and the Unreal Engine are also used in a similar Vehicle-in-Virtual-Environment (VVE) method which establishes a closed loop in which the motion of the real vehicle is tracked and reflected in the virtual world, making it possible its control systems to respond to simulated events [CCG+23].

These examples show a trend towards using game engine based simulators to evaluate autonomous vehicles. This method falls somewhere between physical testing, where safety and consistency can be problematic, and pure software simulation, which frequently lacks vehicle dynamics fidelity [CCG+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 actual radar sensor [DKK+21]. This allows for end-to-end validation of the entire perception and control pipeline [DKK+21].

The X-in-the-Loop approach is found in many different areas, not just a single field. In marine robotics, a VIL framework was developed to test the long term autonomy of

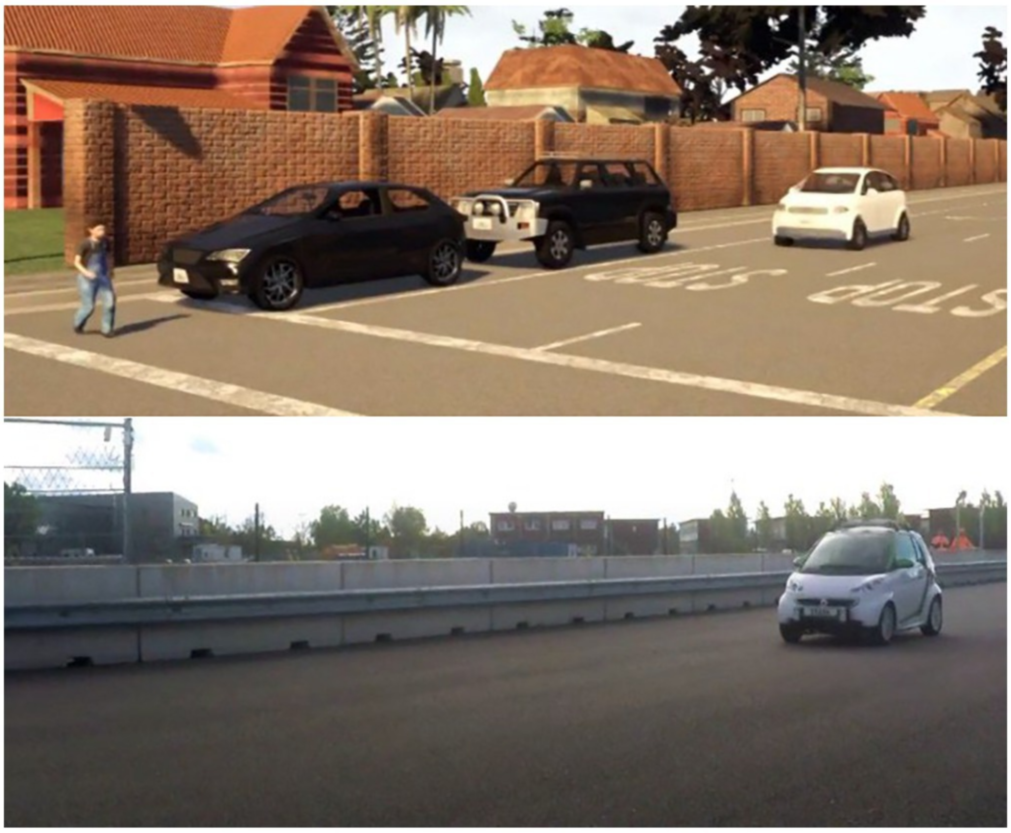

Figure 2.2: An example of a Vehicle-in-the-Loop (VitL) setup. A real car on a test track connected to a high-fidelity simulator like CARLA that generates virtual traffic and sensor data. [DSR+22]

a robot swarm. In this system, an Autonomous Surface Vehicle (ASV) interacts with multiple simulated underwater sensor nodes to test cooperative behaviors without the logistical cost of deploying a full swarm [BVM20]. 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 [DKQ+21].

These approaches continue to move 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 [VTS21]. These systems rely on creating detailed digital twins of the entire test environment and combining real and virtual components to run complex, mixed reality test scenarios [VTS21].

2.2 Digital Twins

The Digital Twin (DT) is an important concept in many current X-in-the-Loop frameworks. The DT serves as the virtual counterpart to a physical system, acting as a virtual copy that allows real-time monitoring and simulation [AA23]. The idea is to create a digital information model of a physical system that stays linked to it for the duration of its lifecycle [GV17].

This concept was initially known as the Information Mirroring Model and consists of three core elements: the physical product, its corresponding virtual model, and the data connection that connects them [GV17; AA23]. A diagram of this structure is shown in Figure 2.3. The virtual model is more than a basic geometric shape, it is often a detailed simulation that can model mechanical, electrical, and software properties of a system [LWS+21]. Information moves in both directions, so sensor data can flow from the real world to update the virtual model [GV17; LWS+21]. In turn, the virtual model can also send commands back to control or optimize the physical system [GV17; LWS+21]. This continuous, bidirectional data exchange characterizes a Digital Twin [AA23].

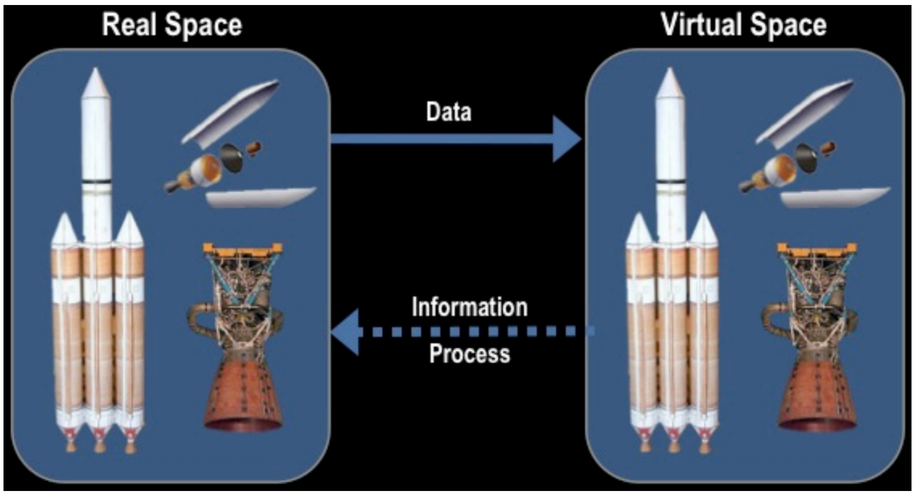

Figure 2.3: The foundational concept of a Digital Twin, illustrating the three core components: the physical entity, the virtual model, and the bi-directional data connection that links them [GV17; LWS+21]. [Gri15]

Digital Twins in robotics are frequently created using simulation software and the middleware that connects the virtual simulation to the actual hardware is often the Robot Operating System (ROS) [MFG+22]. 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 [SPD+21]. To validate a deep learning-based harvesting robot, R. Singh et al. adopted a similar strategy and created a DT of a greenhouse in Gazebo [BSH24]. 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 [MCT21], while Marques et al. used Webots for an Automated Guided Vehicle (AGV), synchronizing it via an OPC-UA server [MRS24]. These examples show a common approach of using traditional simulators to simulate robot kinematics and sensor feedback for closed-loop testing.

More recently, game engines have become a popular choice for creating Digital Twins, as they offer more realistic graphics and physics. Unity [Uni23], for instance, is used developers to build virtual environments that are highly realistic. [**NOT FINAL**]This can be seen in Figure **??**[**NOT FINAL**]. Pérez et al. used Unity to develop a DT of a multi-robot cell with a Virtual Reality (VR) interface for virtual commissioning and operator training [PRR+20].

Figure 2.4: [**NOT FINAL**]A comparison of robotic Digital Twin environments, showing a view from a traditional simulator (left) versus a high-fidelity visualization from a modern game engine like Unity (right).[**NOT FINAL**]

Another important consideration are the technical capabilities these engines. Yang et al. [YMZ20] simulated the physics of a UAV using Unity and its built-in NVIDIA PhysX engine. They also demonstrated the creation of virtual sensors, such as a LiDAR, directly within the game engine using its raycasting API [YMZ20]. Research has also been done on the performance and reliability of these game engine-based frameworks. Kwon et al. developed a safety-critical DT in Unity and ROS 2, reaching a data-transmission latency of 13 ms for predicting collisions [KYS+25]. Similarly, M. Singh et al. created a DT using ROS and Unityand and conducted performance validation with a communication latency of 77.67 ms and a positional accuracy of 99.99% [SKG+24].

2.3 Mixed Reality

Beyond the testing paradigm and the digital twin, the way a user interacts with the system is another 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 [WPC+22]. In robotics, Augmented Reality (AR) is an especially useful 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]. As illustrated in Figure 2.5, 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; SSW21; MV20].

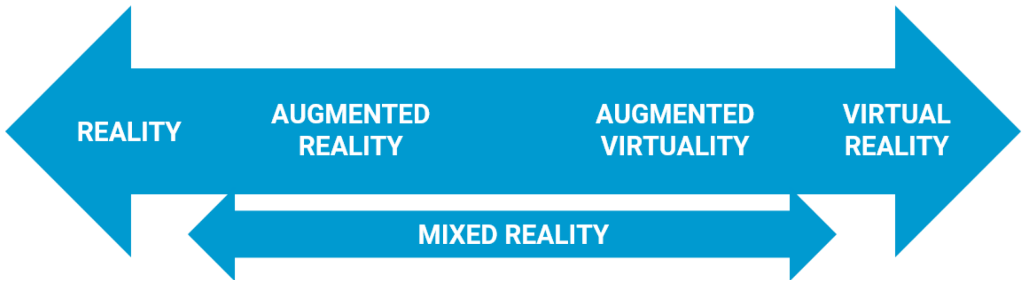

Figure 2.5: The Reality-Virtuality Continuum, illustrating the spectrum from a real environment to a completely virtual one. [WPC+22]

A Virtual Reality (VR) environment is an endpoint of the continuum, where the user is totally immersed in and can interact with a fully synthetic world [MK94]. This approach is useful for HRI research, as it allows for testing interactions with virtual robots in scenarios where it might be unsafe or too expensive for physical hardware [WPC+22]. The general term Mixed Reality (MR) describes the entire spectrum between the two extremes, where real and virtual worlds are combined into a single display [MK94]. MR is made up of two main categories: Augmented Reality (AR) and Augmented Virtuality (AV) [MK94; MV20]. AR is the process of adding virtual objects to a real environment [MK94]. This lets for example, HRI researchers to place 3D data and intentions of a robot directly into the physical space of an user [WPC+22]. 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 (Seperates Kapitel?)

This thesis builds directly upon the "Virtual Environment for mobile Robotic Applications" (VERA) framework, a preceding master's thesis by Gehricke [Geh24]. The VERA framework was designed as a modular platform to bridge the gap between pure simulation and real-world testing [Geh24]. To achieve this, it integrated concepts from Digital Twins [AA23], Augmented Reality (AR) [MV20], and Vehicle-in-the-Loop (ViL) testing [Geh24].

The core concept of VERA is to project a dynamic, interactive virtual environment onto a physical floor, where a real robot operates [Geh24]. This system enables synchronized virtual and projected environments, allowing the robot to interact with both representations simultaneously [Geh24]. This combination of simulation with real-world dynamics and AR-based visualization provides a flexible platform for developing and evaluating complex robotic tasks [Geh24]. The physical setup of the VERA platform is shown in Figure **??**.

Figure 2.6: The VERA platform, including the overhead projector and 3D camera mounted on a cross-beam, and the physical test area on the floor. Adapted from [Geh24].

2.4.1 The EMARO Test Platform (Mehr detaillierte Beschreibung?)

The capabilities of the VERA framework were demonstrated using a specific hardware testbed, the EMARO (Educational Modular Autonomous Robot Osnabrück) [Geh24]. EMARO is a mobile, autonomous, and modular mini-robot developed at Osnabrück University, designed specifically for teaching and research in robotics, artificial intelligence, and computer engineering [Geh24].

The robot's hardware architecture is modular, based on three main Printed Circuit Boards (PCBs): a **Host-PCB** that integrates the compute module, a **Power-PCB** for energy management and motor control, and a **Base-PCB** for integrating ground and distance sensors [Geh24]. The variant used in the original VERA thesis was equipped with a Raspberry Pi Compute Module 4, an Inertial Measurement Unit (IMU) for orientation, and two wide-angle cameras configured for stereo vision applications [Geh24]. The EMARO robot is shown in Figure **??**.

Figure 2.7: The EMARO (Educational Modular Autonomous Robot Osnabrück) platform used as the physical testbed for the VERA framework. Adapted from [Geh24].

Critically for its integration, the entire system, including the EMARO, operates on the Robot Operating System 2 (ROS 2) [Geh24]. Within the original VERA framework, EMARO navigated the projected environment autonomously by employing a camera-based, OpenCV line-following algorithm [Geh24]. This `pathfinder` node processed the camera feeds to detect the projected line, calculate its centroid, and publish steering commands to the `/cmd_vel` topic, enabling the robot to follow the path [Geh24]. Data from the robot's IMU and system status (like CPU and RAM usage) were also published to ROS 2 topics, where VERA's visualizer could subscribe to them for AR projections [Geh24].

2.4.2 Original VERA System Architecture

The software architecture of the original VERA framework was built on ROS 2 and consisted of three main components: a positioning system (VEPS), an environment manager, and a dual-instance visualization system [Geh24]. The interaction between these components, as described in the original thesis, is shown in Figure **??**.

Figure 2.8: The software architecture of the original VERA framework, showing the data flow between the EMARO, the 3D Camera, the core VERA nodes (VEPS, Manager, Visualizer), and RViz. Adapted from [Geh24].

2.4.2.1 Virtual Environment Positioning System (VEPS)

The first component, the Virtual Environment Positioning System (VEPS), was responsible for the real-time localization of the EMARO [Geh24]. This system did not use traditional vision, but instead relied on depth data [Geh24]. It utilized an Allied Vision Ruby 3D stereo camera, mounted above the test area, to capture a 3D point cloud of the environment [Geh24].

This point cloud data was processed in several steps. First, it was filtered to remove the ground plane and sensor noise, isolating the points corresponding to the robot [Geh24]. Second, the **Median Absolute Deviation (MAD)** algorithm was applied to the remaining points to statistically find their center [Geh24]. This calculated centroid was used as the robot's 2D (x, y) position. This position was then smoothed using a Kalman filter and published as a ROS 2 transform (`/tf`) for other nodes to consume [Geh24]. The evaluation of VEPS showed it could achieve processing times under 5ms at a 30 FPS update rate [Geh24].

2.4.2.2 Virtual Environment Manager

The second component was the Environment Manager, which acted as the central "brain" of the simulation [Geh24]. This C++ ROS 2 node was responsible for parsing, managing, and updating the state of the virtual world [Geh24]. It would load environments from standard Gazebo-compatible `world.sdf` files, which defined the models, poses, and custom properties (e.g., "movable" or "removable") of all virtual objects [Geh24].

The manager ran a main loop on a 50ms timer [Geh24]. In each loop, it would look up the robot's current position (published by VEPS) and perform collision checks between the robot and all spawnable objects in the environment [Geh24]. If a collision was detected, the manager would update the object's state (e.g., marking it for removal) and publish these changes to the visualizers using ROS 2 `MarkerArray` messages over the `/objects` topic [Geh24]. This node was also responsible for path tracking, recording the robot's position history and publishing it as a `/path` message for the AR visualization [Geh24].

2.4.2.3 Visualization System (RViz and Pygame)

Finally, the visualization system consisted of two separate, synchronized instances that both subscribed to the manager's `/objects` and `/path` topics [Geh24].

1. **RViz:** The standard ROS visualization tool, RViz, was used as a 3D simulation interface [Geh24]. It provided a detailed 3D view for debugging and monitoring, displaying the robot's digital twin (from its URDF model), the virtual objects, the robot's driven path, and other debug data like the VEPS point cloud [Geh24]. An example of the RViz interface from the original thesis is shown in Figure **??**.

Figure 2.9: The RViz simulation interface in the original VERA framework, showing a 3D view of the virtual objects (green), the EMARO's digital model (grey), and its driven path. Adapted from [Geh24].

2. **2D Projector Node:** This was the custom-built component responsible for projecting the virtual world onto the physical floor [Geh24]. This ROS 2 node, written in Python, used the **Pygame library** to render a 2D orthogonal, top-down view of the environment [Geh24]. It subscribed to the manager's topics to draw all the virtual objects and the robot's path [Geh24]. This node also rendered the AR-like features, such as a status display (showing EMARO's CPU and RAM usage) that was projected onto the floor behind the robot, following its position and orientation in real-time [Geh24]. The final rendered output of this Pygame node is shown in Figure **??**.

Figure 2.10: The 2D orthogonal view rendered by the custom Pygame visualizer, as it would be sent to the projector. It displays the objects (green), the robot's position (red dot), its path (cyan), and the AR status display. Adapted from [Geh24].

2.4.3 Platform Changes and Identified Limitations

While the `[Geh24]` thesis established the VERA framework's concept, this thesis builds upon an evolved version of the platform and directly addresses the specific software limitations documented in the original work.

2.4.3.1 Changes of the Positioning System

The VERA platform is modular, and its components have been iterated upon. The point-cloud-based VEPS [Geh24] is no longer in use. Instead, the platform for this thesis employs a more robust, marker-based tracking system. This system, developed in a separate bachelor's thesis [**Your_BA_Thesis_Citation**], uses ArUco markers and multiple cameras to provide a high-precision 6D pose (position and orientation) of the robot, which it publishes over a ROS 2 topic. This thesis consumes this pose data as a service. The internal implementation of the ArUco tracking system itself is outside the scope of this project.

2.4.3.2 Identified Limitations of the Original Software

While the positioning system was updated, the core software limitations of the original VERA manager and visualizer remained. These limitations, explicitly documented in the `[Geh24]` thesis, form the primary justification for this work:

- **Lack of Realistic Physics:** The custom-built environment manager was a significant limitation as it did not include an enhanced physics engine for realistic interactions. The `[Geh24]` thesis concludes by listing the "integration of enhanced physics into the manager's implementation" as a primary direction for future work [Geh24].
- **Performance and Scalability Bottlenecks:** The evaluation of the **Pygame-based 2D visualizer** revealed significant performance limitations [Geh24]. In scenarios with a high number of object updates (e.g., >800–1250 objects), the system's responsiveness failed, and marker latencies were observed to "spike into

seconds" [Geh24]. The original thesis identified the cause of this bottleneck: the visualizer's message queue could not process the high frequency of individual marker messages, leading to "delayed and incomplete updates" [Geh24].

- **Limited Interaction Modalities:** Finally, while the AR projections were successful, more immersive and interactive modalities were not implemented. The `[Geh24]` thesis concludes by suggesting a "virtual reality integration to interact with the simulated environment via VR headsets" as another "possible extension" of the system [Geh24].

These three documented limitations—a lack of physics, a non-scalable custom visualizer, and the absence of VR interaction—are the specific problems this thesis solves by replacing VERA's custom manager and Pygame visualizer with the Unity Engine.

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.

[AAR+19] Bulgaria Aleksandrov, Chavdar Acad, Bulgaria Rumenin, et al. “Review of hardware-in-the-loop -a hundred years progress in the pseudo-real testing”. In: 54 (Dec. 2019), pp. 70–84.

[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.

[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.

[BSH24] Rajmeet Bhourji, Lakmal Seneviratne, and Irfan Hussain. “A Deep Learning-Based Approach to Strawberry Grasping Using a Telescopic-Link Differential Drive Mobile Robot in ROS-Gazebo for Greenhouse Digital Twin Environments”. In: *IEEE Access* PP (Jan. 2024), pp. 1–1. DOI: 10.1109/ACCESS.2024.3520233. URL: https://doi.org/10.1109/ACCESS.2024.3520233.

[BVM20] Anja Babić, Goran Vasiljevic, and Nikola Miskovic. “Vehicle-in-the-Loop Framework for Testing Long-Term Autonomy in a Heterogeneous Marine Robot Swarm”. In: *IEEE Robotics and Automation Letters* PP (June 2020), pp. 1–1. DOI: 10.1109/LRA.2020.3000426.

[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.

[CCG+23] X. Cao, H. Chen, S. Y. Gelbal, et al. “Vehicle-in-Virtual-Environment (VVE) Method for Autonomous Driving System Development, Evaluation and Demonstration”. In: *Sensors* 23.11 (2023), p. 5088. DOI: 10.3390/s23115088. URL: https://doi.org/10.3390/s23115088.

[DKK+21] Axel Diewald, Clemens Kurz, Prasanna Venkatesan Kannan, et al. “Radar Target Simulation for Vehicle-in-the-Loop Testing”. In: *Vehicles* 3 (May 2021), pp. 257–271. DOI: 10.3390/vehicles3020016.

[DKQ+21] Xunhua Dai, Chenxu Ke, Quan Quan, and Kai-Yuan Cai. “RFlySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations”. In: *Aerospace Science and Technology* 114 (Apr. 2021), p. 106727. DOI: 10.1016/j.ast.2021.106727.

[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.

[Gri15] Michael Grieves. “Digital Twin: Manufacturing Excellence through Virtual Factory Replication”. In: (Mar. 2015).

[GV17] Michael Grieves and John Vickers. “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems”. In: Aug. 2017, pp. 85–113. ISBN: 978-3-319-38754-3. DOI: 10.1007/978-3-319-38756-7_4.

[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.

[KYS+25] Woojin Kwon, Jieun Yang, Seunghwa Song, et al. "Real-Time Digital-Twin-Based Cobot-Worker Collision Risk Prediction Using Unity, ROS, and UWB". In: *IEEE Access* 13 (2025), pp. 85967–85978. DOI: 10.1109/ACCESS.2025.3569332. URL: https://doi.org/10.1109/ACCESS.2025.3569332.

[LWS+21] Jiewu Leng, Dewen Wang, Weiming Shen, et al. "Digital twins-based smart manufacturing system design in Industry 4.0: A review". In: *Journal of Manufacturing Systems* 60 (July 2021), pp. 119–137. DOI: 10.1016/j.jmsy.2021.05.011.

[MCT21] Carlos Magrin, Gustavo Conte, and Eduardo Todt. "Creating a Digital Twin as an Open Source Learning Tool for Mobile Robotics". In: *2021 Latin American Robotics Symposium (LARS), Brazilian Symposium on Robotics (SBR) and Workshop on Robotics in Education (WRE)*. Oct. 2021, pp. 13–18. DOI: 10.1109/LARS/SBR/WRE54079.2021.9605457.

[MFG+22] Steven Macenski, Tully Foote, Brian Gerkey, et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[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.

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

[MRS24] Nuno Marques, Marco Rodrigues, and Diogo Sousa. "Driving Forward Mobile Robotics: A Digital Twin Architecture Case Study for AGVs Data-Driven Autonomy". In: (Jan. 2024). DOI: 10.21203/rs.3.rs-3837578/v1. URL: https://doi.org/10.21203/rs.3.rs-3837578/v1.

[MTH22] F. Mihalič, M. Truntič, and A. Hren. "Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges". In: *Electronics* 11.15 (2022), p. 2462. DOI: 10.3390/electronics11152462. URL: https://doi.org/10.3390/electronics11152462.

[MV20] Zhanat Makhataeva and Atakan Varol. "Augmented Reality for Robotics: A Review". In: *Robotics* 9.2 (Apr. 2020), p. 21. DOI: 10.3390/robotics9020021. URL: https://doi.org/10.3390/robotics9020021.

[PRR+20] L. Pérez, S. Rodríguez-Jiménez, N. Rodríguez, et al. “Digital Twin and Virtual Reality Based Methodology for Multi-Robot Manufacturing Cell Commissioning”. In: *Applied Sciences* 10.10 (May 2020), p. 3633. DOI: 10.3390/app10103633. URL: https://doi.org/10.3390/app10103633.

[SKG+24] M. Singh, J. Kapukotuwa, E. L. S. Gouveia, et al. “Unity and ROS as a Digital and Communication Layer for Digital Twin Application: Case Study of Robotic Arm in a Smart Manufacturing Cell”. In: *Sensors* 24.17 (2024), p. 5680. DOI: 10.3390/s24175680. URL: https://doi.org/10.3390/s24175680.

[SPD+21] P. Stączek, J. Pizoń, W. Danilczuk, and A. Gola. “A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR’s) Operating Environment—A Case Study”. In: *Sensors* 21.23 (2021), p. 7830. DOI: 10.3390/s21237830. URL: https://doi.org/10.3390/s21237830.

[SSW21] Richard Skarbez, Missie Smith, and Mary Whitton. “Revisiting Milgram and Kishino’s Reality-Virtuality Continuum”. In: *Frontiers in Virtual Reality* 2 (Mar. 2021). DOI: 10.3389/frvir.2021.647997. URL: https://doi.org/10.3389/frvir.2021.647997.

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

[VTS21] Balázs Varga, Tamas Tettamanti, and Zsolt Szalay. “System Architecture for Scenario-In-The-Loop Automotive Testing”. In: *Transport and Telecommunication Journal* 22 (Apr. 2021), pp. 141–151. DOI: 10.2478/ttj-2021-0011. URL: https://doi.org/10.2478/ttj-2021-0011.

[WPC+22] Michael Walker, Thao Phung, Tathagata Chakraborti, et al. *Virtual, Augmented, and Mixed Reality for Human-Robot Interaction: A Survey and Virtual Design Element Taxonomy*. 2022. arXiv: 2202.11249 [cs.RO]. URL: https://arxiv.org/abs/2202.11249.

[YMZ20] Yuanlin Yang, Wei Meng, and Shiquan Zhu. “A Digital Twin Simulation Platform for Multi-rotor UAV”. In: *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*. 2020, pp. 591–596. DOI: 10.1109/ICCSS52145.2020.9336872. URL: https://doi.org/10.1109/ICCSS52145.2020.9336872.

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