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, there is a gap between the virtual and physical worlds when simulation models are abandoned during the implementation phase. By nature, a virtual model cannot capture every characteristic of the physical hardware perfectly, such as the exact effects of friction, sensor noise, or small differences in motors [Hu05; BM18]. Testing with real hardware allows for the highest fidelity and 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]. Testing the entire system using only real components is also not feasible for complex, large cooperative systems because of issues relating to complexity and scale [Hu05]. As neither of these approaches seems satisfactory on its own, an intermediate solution is required that could connect simulation with real-world experimentation [Hu05].

To bridge this gap, hybrid methods like "robot-in-the-loop" (RitL) simulation have emerged, which allow physical robots to operate within virtual environments [Hu05]. This is often centered on a "Digital Twin," which is a real-time virtual replica of the physical robot [AA23], and managed through Mixed Reality interfaces like Augmented Reality (AR) to enhance interaction [MV20]. The combination of these technologies provides a robust paradigm for the comprehensive testing and validation of robotic systems.

The “Virtual Environment for mobile Robotic Applications” (VERA) framework integrates digital twins, AR, and vehicle-in-the-loop testing together into a single 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.

While the VERA framework provides a strong conceptual foundation, its original technical implementation contained various critical limitation regarding physical realism, scalability, and user interaction [Geh24]. Its custom environment manager lacked a realistic physics engine, while its 2D visualizer, built with Pygame, showed performance degradation when handling a large number of dynamic objects, leading to delayed and incomplete visualizations [Geh24]. Furthermore, while AR was implemented, the original thesis identified immersive Virtual Reality (VR) interaction as a key area for future work [Geh24].

This thesis overcomes those limitations by offering a novel, high-performance architecture replacing the custom manager and visualizer used within VERA. To address these needs, a game engine is selected to replace the custom solutions, specifically chosen to fulfill the requirements for high-fidelity graphics, integrated physics, and native support for VR. All communications and data synchronization between the physical robot and the simulation environment are carried out with the **Robot Operating System 2 (ROS 2)** [MFG+22], a middleware framework used within state-of-the-art robotics.

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 the selected engine’s rendering tools to improve visual quality and system capabilities.

[**NOT FINAL**]The remainder of this thesis is structured as follows: after reviewing the basic concepts and technologies, such as RitL, Digital Twins, Mixed Reality, ROS 2, and several simulation engines in Chapter 2, an in-depth review of the original framework of VERA is presented, along with its limitations. In Chapter 3, the functional and non-functional requirements for the new system are identified. The architecture and implementation of the new framework are explained in detail in Chapter 4, with particular reference to the integration of **the game engine** with ROS 2, the development of the digital twin, and the mixed reality interaction system. Then, a quantitative evaluation regarding the performance of the system is presented along with a discussion of the results in Chapter 5. Finally, Chapter 6 summarizes the thesis contributions and gives an outlook on possible 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].

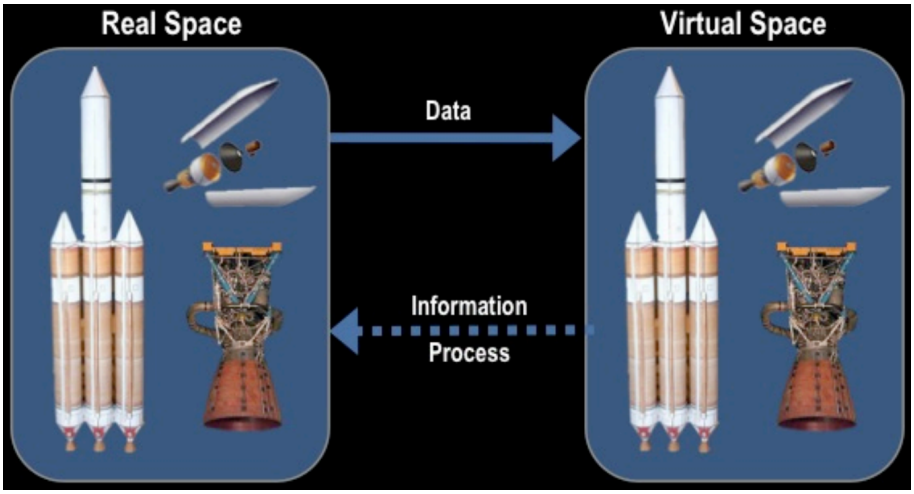

Figure 2.3: The foundational concept of a Digital Twin, illustrating the three core components: the physical entity, the virtual model, and the bi-directional data connection that links them [GV17; LWS+21]. [Gri15]

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 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 robotics 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+24a].

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 Robot Operating System 2 (ROS 2)

ROS 2 is not a operating system but a middleware framework to simplify the creation of complex robotic systems [MFG+22]. It was redesigned from scratch to meet the challenges posed by modern robotics in domains ranging from logistics and agriculture to space missions and became the standard for both research and industry [MML+23; MFG+22]. At its heart, ROS 2 is designed based on principles of distribution, abstraction, asynchrony, and modularity that together allow the development of

scalable and robust applications [MFG+22]. The architecture of ROS 2 is based on a distributed network of independent programs called **Nodes** [MFG+22]. A node is a single, self-contained executable that performs a specific task, such as controlling a motor, processing sensor data, or, in the case of this thesis, bridging communication to the Unity simulation [MFG+22]. The nodes communicate through a set of defined patterns. The most common pattern is the publish-subscribe mechanism called **Topics**, illustrated in Figure 2.6 [MFG+22]. In this model, nodes can publish data as messages to a topic, while other nodes can subscribe to that topic to receive data asynchronously [MFG+22]. For tasks that require a direct request and a guaranteed response, ROS 2 offers **Services**, following a synchronous request-response pattern [MFG+22]. For long-running tasks that require continuous feedback and the ability to be preempted, ROS 2 provides a unique communication pattern called **Actions** [MFG+22]. An action consists of a goal, a feedback stream, and a final result, making it optimal for managing tasks like navigation where the progress of a robot toward a goal has to be monitored over time [MMW+20].

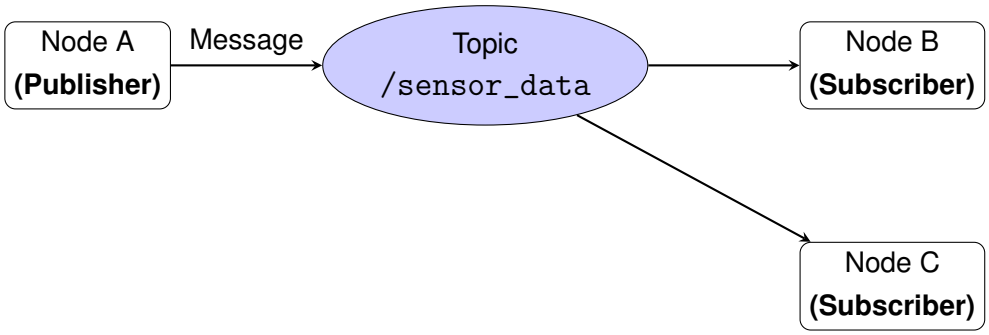

Figure 2.6: The ROS 2 publish-subscribe model. Node A publishes messages onto a central topic, and any number of subscriber nodes (B and C) may receive that data without direct knowledge of the publisher.

An essential part of any mobile robot is coordinate frame management, which in ROS 2 is accomplished through its transform library, **tf2** [Foo13]. The tf2 library standardizes the tracking of spatial relationships between the various parts of the robot and environment, placing them into a tree-like data structure, as illustrated in Figure 2.7 [Foo13]. This permits any node in the system to request at any time the position and orientation of any frame relative to another, a feature critical for transforming sensor data into a useful frame of reference [Foo13].

In the architecture of this thesis, ROS 2 provides the central **data backbone**, a term borrowed from digital engineering that defines an integrated communication layer for all relevant system knowledge [PKP+20]. This is the "glue" for connecting the

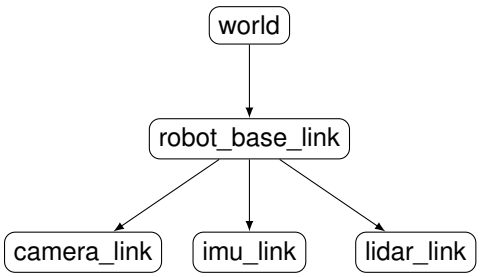

Figure 2.7: A simplified example of a ROS 2 transform tree (tf tree). The library maintains the hierarchical relationships so that a program can easily determine the transform from the `camera_link` to the `world` frame, for instance.

physical EMAROs robot and its tracking system with the virtual Unity environment. The digital twin of the EMAROs robot subscribes to ROS 2 topics to receive real-time data from the physical world and can synchronize its state through this communication layer [SKG+24b]. In this project, the `ros2-for-unity` asset of Robotec.AI is utilized [Rob21]. This approach has the particular advantage of not bridging the communication but instead implementing the ROS 2 middleware stack (RCL and below) directly in Unity. This makes entities in the simulation native ROS 2 nodes, letting them respect QoS (Quality of Service) settings and obtain latencies significantly lower than possible with bridged solutions [Rob21]. For navigation for both the physical and virtual robots, this thesis utilizes **Navigation2 (Nav2)**, the official, next-generation autonomous navigation framework for ROS 2 [MMW+20]. Nav2 was designed from the ground up to orchestrate planning, control, and recovery tasks using configurable Behavior Trees that are highly modular and can be changed at runtime to create custom navigation behaviors [MMW+20]. Core to its operation, Nav2 separates the navigation task into two closely related, yet distinct components, namely a **global planner**, which seeks out an acceptable, long-range path through the environment, and a **local trajectory planner** or controller, which generates velocity commands in order to follow that route while reacting to immediate obstacles [MML+23]. Using the Nav2 stack, a high-level goal, e.g. a coordinate in the environment, can be sent via a ROS 2 action to either the physical EMAROs robot, or the purely virtual model, and the system will take care of all complex path planning and collision avoidance autonomously.

2.5 The VERA Framework

This thesis directly builds on the "Virtual Environment for mobile Robotic Applications" framework, which was developed earlier by Gehricke [Geh24]. VERA was designed

initially as a modular and scalable platform to bridge the validation gap between pure software simulation and real-world testing. It combines the concepts Digital Twins, Augmented Reality, and Vehicle-in-the-Loop testing to enable robots to interact with a virtual environment projected into the real world [Geh24].

The core idea of VERA is projecting a dynamic, interactive virtual world onto the physical floor where a real robot operates. The system synchronizes the virtual state with the actions of the physical robot in such a way that scenarios are possible where the robot can detect and react to virtual obstacles as if they were real. This system provides a flexible testbed for the development and evaluation of robotic applications without having to physically build complex environments [Geh24].

Figure 2.8: The physical setup of the VERA platform: A projector and tracking system are mounted on a frame above the test area. [Geh24]

2.5.1 The EMAROs Test Platform

The capabilities of VERA are demonstrated by a mobile mini-robot called EMAROs (Educational Modular Autonomous Robot Osnabrück), which was developed at Osnabrück University specifically for research and education in robotics and artificial intelligence [Geh24].

The modular hardware architecture of EMAROs is based on three main PCBs: a
textbfHost-PCB containing high-level computing, a
textbfPower-PCB managing energy supply and motor control and a
textbfBase-PCB integrating ground sensors [Geh24]. The specific variant used in this work is equipped with a Raspberry Pi Compute Module 4 (CM4) running Ubuntu and the Robot Operating System 2 (ROS 2) [Geh24].

Figure 2.9: The EMAROs robot is the robotic platform used in the testbed, equipped with a modular sensor suite and running ROS 2 [Geh24].

For environmental perception, the robot is equipped with Time-of-Flight (ToF) distance sensors and an Inertial Measurement Unit (IMU) for orientation tracking [Geh24]. Moreover, there are two wide-angle cameras that can be configured for stereo vision applications or line-following tasks [Geh24]. Within VERA, EMAROs takes on the role of a "Physical Twin," streaming real-time telemetry including odometry, battery status, and sensor data to the simulation framework via ROS 2 topics [Geh24].

2.5.2 Original VERA System Architecture

The software architecure of the original VERA framework was mainly a ROS 2 implementation, and its structure included only three main components: the **Virtual Environment Positioning System (VEPS)**, the **Environment Manager**, and the **Visualization System** [Geh24].

Accurate localization is important to synchronize the physical robot with the projected virtual world and was provided by the **VEPS** in the original VERA framework [Geh24]. The original VEPS implementation relied on a depth-based approach where a ceiling-mounted Allied Vision Ruby 3D stereo camera captured point clouds of the test area [Geh24]. The point clouds were filtered by a Median Absolute Deviation (MAD) algorithm to calculate the centroid of the robot [Geh24]. However, for maximum robustness and precision for this thesis, the positioning system was updated to use an ArUco marker-based tracking system developed in parallel research by [**JuliaBA**]. Although the point-cloud method provided a markerless solution, the ArUco-based approach has higher reliability and lower noise under changing light conditions [**JuliaBA**; Geh24]. By attaching a fiducial marker to the top of the EMAROs robot, the system is able to calculate the 6D pose (position and orientation) of the robot [**JuliaBA**]. This pose is published into the ROS 2 network, where it can instantly update the position of the digital twin [**JuliaBA**].

Figure 2.10: Visualization of the tracking system. The original VERA used point clouds (left), while the current iteration uses ArUco markers (right) for robust pose estimation [**JuliaBA**].

The **Virtual Environment Manager** is the central brain of the simulation [Geh24]. Implemented as a C++ ROS 2 node, it parses environment definitions from standard Gazebo `world.sdf` files [Geh24]. It keeps track of the state of all virtual objects, including their positions and properties, such as whether an object is movable or static [Geh24]. One of the key roles of the manager is to manage the interactions of the robot with the virtual world [Geh24]. It operates a continuous loop, typically at 20 Hz, which checks the distance between the position of the robot given by the VEPS and the virtual objects [Geh24]. The manager triggers predefined actions if a collision is detected, such as removing a collectible object or stopping the robot [Geh24]. It relies on basic distance heuristics and not a physics engine, hence limiting interactions to simple ones [Geh24].

The **Visualization System** is responsible for rendering the virtual environment so it can be projected onto the physical floor [Geh24]. In the original VERA framework,

this was achieved using a custom ROS 2 node built with Pygame, a Python library designed for 2D game development [Geh24]. The visualizer subscribes to the object lists published by the Manager and the robot's path history [Geh24]. It provides a 2D, top-down, orthogonal view of the scene, drawing obstacles as simple geometric shapes [Geh24]. It gives Augmented Reality features, projecting dynamic information directly into the workspace [Geh24]. The driven of the robot is displayed as a trail, while a system status panel with real-time CPU and battery usage is shown behind the robot [Geh24].

Figure 2.11: Original VERA visualization via Pygame, showing a projection of the robot's path and virtual obstacles onto the floor. [Geh24]

2.5.3 Identified Limitations

While VERA successfully demonstrated the concept of Robot-in-the-Loop testing with AR projections, its original software implementation contains several critical limitations, that this thesis sets out to overcome:

- **Lack of Realistic Physics:** The custom Environment Manager does not feature a physics engine [Geh24]. Collision detection is performed through simple radius checks, meaning the robot cannot push objects or experience friction nor can it interact with complex geometries [Geh24]. This reduces the realism of the Digital Twin [Geh24].
- **Performance Scalability:** Pygame-based visualizer utilizes CPU for rendering and has performance-related issues for complex scenarios [Geh24]. Gehricke [Geh24] noted that in an environments with a high number of dynamic objects (e.g., >800), the message queue became saturated, leading to considerable latency and incomplete visual updates.
- **Limited Interaction (No VR):** The system was designed strictly for 2D floor projections [Geh24]. It does not support any kind of immersive Virtual Reality (VR) interfaces that would enable a human operator to view the scene in 3D or to interact with the robot remotely [Geh24]. Therefore, the integration of Virtual Reality was identified as a necessary future step for Human-Robot Interaction (HRI) [Geh24].

2.6 Simulation Engines for Robotics

The choice of a capable simulation engine is an important part in solving the limitations of the original VERA framework. The choice of platform affects the efficiency, accuracy, and applicability of the Digital Twin [SKG+25]. Based on this, the review of simulation engines in this thesis is guided by four criteria which are required for a functional Mixed Reality Digital Twin:

- **Visual Fidelity:** It is not only about rendering quality but to support realistic sensor simulation, such as cameras, and to provide an immersive experience to the user in VR that lacks in traditional robotics simulators [DBF+25].
- **Physics Accuracy:** To accurately predict the kinematic and dynamic behaviour of the EMAROs robot, the engine needs to have a strong physics backend [SKG+25].
- **Native VR/AR Support:** In order to extend the VERA with the proposed Virtual Reality interfaces, the engine must have a mature framework for VR devices and provide capabilities for creating interactive environments without extensive middleware [CIR23].
- **Robust ROS 2 Integration:** ROS 2 forms the data backbone of the system and should enable low-latency, high-throughput communication to synchronize the digital twin with the physical robot in real-time [SKG+25].

2.6.1 Robotics Simulators

Gazebo is the most used simulator in Industry 4.0 applications because of its deep integration with ROS and and the fact that it is open-source [SKG+25]. It delivers detailed physics simulations along with sensor data emulations, making it ideal for precise engineering applications [SKG+25]. However, while this simulator enjoys benefits from a large community and extensive documentation, there are noticeable drawbacks concerning visualization [DBF+25]. For example, the rendering quality of Gazebo is only moderate and not photorealistic to an extent that would enable immersive VR experiences [DBF+25]. Moreover, some works note that Gazebo has a steep learning curve for new users due to its complex configuration and less intuitive user interface compared to modern commercial tools [SKG+25; FCY25].

Webots well established opensource platform, appreciated for cross-platform support and a large library of robot models [KYH+24]. Although it is quite effective in educational applications, the rendering capabilities of the software are somewhat dated

compared to the current game engines, making it not fitting for highfidelity Mixed Reality graphics [KYH+24].

2.6.2 Physics-Based Simulators

A few other simulators were found in literature but considered less appropriate for the particular VR needs of this thesis. **CoppeliaSim (previously known as V-REP)** is widely applied in automation, yet it has less physics realism compared to the professional ones, and its scaling is limited [FCY25]. **MuJoCo** does well with contact-rich tasks and Reinforcement Learning but works almost exclusively on the stability of the physics and not the visualization, so it lacks the inbuilt VR frameworks necessary for Mixed Reality interaction [FCY25].

2.6.3 High-Fidelity and Game-Based Engines

Game engines have been widely adopted within the robotics community to overcome the visualization limitations of traditional robotics simulators.

NVIDIA Isaac Sim is the state-of-the-art in industrial simulation [DBF+25]. The platform is capable of photorealism through ray-tracing and has advanced GPU-accelerated physics through PhysX [DBF+25]. It excels in generating synthetic data for AI perception [KYH+24]. But Isaac Sim has a very high hardware barrier where high-end NVIDIA RTX GPUs are specifically mandatory [DBF+25]. Being proprietary, it more restricting for use in educational contexts compared to more open-source or widely accessible game engines [DBF+25].

Unreal Engine is renowned for photorealistic visuals is capable of handling big, open-world environments [FCY25]. But it has a steep learning curve because of its dependency on C++ and therefore is not very well-suited for rapid prototyping compared to other engines like Unity [CIR23].

Unity Engine is establishing itself as the primary platform for Industry 5.0 applications, in particular those focusing on HRI and MR [FCY25]. It boasts a balance of high-fidelity graphics with a robust physics engine, PhysX [SKG+25]. Unity is also known for having one of the most mature MR ecosystems, along with a huge asset store, which lowers the barrier to entry for developers[SKG+25; CIR23]. Literature points out that the user-friendly interface of Unity and supportive community help with a softer learning curve compared to Gazebo or Unreal, particularly for users without deep C++ expertise [SKG+25; CIR23]. ROS 2 is not natively supported by Unity. Support must be provided

through third-party plugins. For the work described in this thesis, the textbfROS2 For Unity plugin from Robotec.AI has been used citeRob24. This asset offers high-performance communication by loading the ROS 2 middleware layer (RCL layer) directly inside the Unity engine [Rob21]. Unlike bridged solutions, simulation entities now can act as native ROS 2 nodes in the context of Unity, thus respecting QoS and reaching lower latencies [Rob21].

2.6.4 Comparison and Selection

Based on this analysis, Unity is selected as the simulation engine. It has an integrated MR framework for Mixed Reality [CIR23] and better visualization compared to Gazebo [FCY25]. It maintains a balance between accessibility and performance, while C# scripting, combined with large community support, makes learning it simpler than Unreal Engine [CIR23]. In addition the `ros2-for-unity` plugin brings high-performance and low-latency communication [Rob21].

Table 2.1: Comparison of Simulation Platforms for Mixed Reality Digital Twins [FCY25; KYH+24; SKG+25; CIR23].

Feature	Gazebo	Isaac Sim	Unreal Engine	Unity
Primary Use Case	Control & Navigation	AI & Photorealism	Photorealism	HRI & MR (VR/AR)
Visual Fidelity	Moderate	Very High	Very High	High
Physics Engine	ODE/Bullet/DART	PhysX 5 (GPU)	Chaos/PhysX	PhysX
Learning Curve	Steep	Advanced	Steep (C++)	Moderate (C#)
Community Support	High (ROS specific)	Moderate	High (Gaming)	Very High (MR/Gaming)
ROS 2 Integration	Native	Bridge	Bridge	Plugin (Native Stack)
Hardware Req.	Low	Very High (RTX)	High	Moderate

2.7 Synthesis? Discussion?

The review of related work confirms that RitL testing [Hu05] combined with Digital Twin technology [AA23] provides a robust paradigm for validating robotic systems. Current research highlights a necessary transition from simple kinematic simulations toward high-visual-fidelity, physics-based environments that can replicate complex interactions and sensor data with greater realism [**SKG+25**; **DSR+22**].

However, the original VERA framework possesses severe technical limitations due to its reliance on custom software components [Geh24]. As detailed in Section 2.5.3,

the lack of a dedicated physics engine prevents realistic object manipulation, while the Pygame-based visualizer suffers from performance bottlenecks in complex scenes [Geh24]. Furthermore, the absence of native Virtual Reality support restricts the potential for investigating Human-Robot Collaboration [Geh24].

To overcome these limitations, this thesis proposes integrating a professional simulation engine into the architecture of VERA using the **Robot Operating System 2 (ROS 2)** [**MFG+22**]. The primary goal is to establish a robust, physics-enabled Mixed Reality environment that bridges the gap between simulation and reality. This new development will be verified by creating demonstration scenarios where the EMAROS robot actively interacts with virtual elements, serving as a proof-of-concept for the system's improved physical realism, scalability, and immersive interactive capabilities.

3 Requirements

This chapter defines the comprehensive requirements for the Mixed Reality Environment designed for the **EMAROS** (Educational Modular Autonomous Robot Osnabrück) platform. The primary objective is to create a versatile validation tool for Robot-in-the-Loop (RitL) testing. The requirements are categorized into **Platform Capabilities**, **System Architecture**, **Functional Capabilities**, and **Automated Verification**, ensuring that the system supports both interactive demonstration and rigorous, automated testing.

3.1 Simulation Platform Capabilities

These requirements define the necessary technical capabilities of the underlying simulation engine. These criteria form the basis for the technology selection process in the implementation phase.

- **FR-01: High-Fidelity Rendering:** The simulation engine must support photo-realistic rendering pipelines to generate synthetic image data that accurately represents real-world lighting and material properties. This is essential for stimulating the EMAROS robot's computer vision algorithms with realistic input.
- **FR-02: Advanced Physics Simulation:** The engine must provide a rigid-body physics core capable of resolving mass, friction, linear drag, and angular drag. This is required to simulate physical interactions, such as the robot pushing obstacles or tools interacting with the ground surface.
- **FR-03: Extended Reality (XR) Support:** The platform must provide a mature framework for Virtual Reality (VR) and Augmented Reality (AR), enabling the projection of 3D content onto physical surfaces and the integration of immersive input devices without requiring extensive custom driver development.
- **FR-04: Native Middleware Integration:** The engine must support high-performance integration with the Robot Operating System 2 (ROS 2). To minimize communi-

cation latency, the simulation should function as a first-party participant in the network rather than relying on external bridge applications.

3.2 System Architecture & Middleware

The system architecture is defined by the constraints of the EMAROS hardware and the need for standard robotic communication.

- **FR-05: Exclusive ROS 2 Communication:** All data exchange—including sensor streams, telemetry, control commands, and system management—must occur exclusively via standard ROS 2 communication patterns. This ensures the simulation remains compatible with the existing EMAROS software stack.
- **FR-06: Deterministic Time Authority:** The system must act as the simulation time source by publishing a clock signal synchronized with the internal physics simulation. All simulated sensors must derive their timestamps from this source to prevent data drift in the robot's control algorithms.
- **FR-07: Dynamic Scenario Management:** The system must provide a mechanism to unload the current environment and load a different validation scenario at runtime via a network command. This capability is required to execute automated test suites that span multiple environments without restarting the application.
- **FR-08: Interface Standardization:** The system must utilize standard robotic message definitions for all external communication. This ensures that the simulation is interchangeable with the physical robot, allowing the same control software to operate in both real and virtual contexts.

3.3 EMAROS Digital Twin Interfaces

The system must provide a digital representation specifically tailored to the capabilities and kinematics of the EMAROS platform.

- **FR-09: Physical Twin Synchronization:** The system must accept real-time pose updates from an external tracking system to synchronize the Digital Twin with the physical robot. In this mode, the virtual model must align its collision boundaries with the physical chassis to interact accurately with virtual objects.

- **FR-10: Standalone Simulation Mode:** In the absence of physical hardware, the system must simulate the kinematic behavior of the EMAROS robot. It must convert velocity commands into physics-based movement and publish the resulting odometry, allowing navigation algorithms to be validated in a pure virtual environment.

- **FR-11: Flexible Coordinate Management:** The system must publish dynamic transforms representing the robot's movement. Additionally, it must support the definition of static sensor offsets either internally or via external configuration, allowing the simulated sensor setup to be adapted to hardware changes without recompiling the software.

3.4 Perception & Sensor Simulation

To support "Scenario-in-the-Loop" testing, the system must simulate the specific sensor suite equipped on the EMAROS robot.

- **FR-12: Synthetic Camera Feed:** The system must render the scene from the perspective of the robot's onboard camera and publish it as a video stream. The resolution and update rate must be configurable to match the specific camera hardware used on the robot.

- **FR-13: Visual Consistency for Perception:** The simulation must support consistent material properties for task-relevant objects. This ensures that computer vision algorithms can reliably detect objects based on color signatures regardless of the environmental lighting conditions.

- **FR-14: Volumetric Sensor Simulation:** The system must simulate a 2D planar laser scanner by detecting geometry within a defined field of view. This sensor must detect both static environment walls and dynamic, user-placed obstacles to update the robot's navigation map.

- **FR-15: Sensor Verification Visualization:** The system must provide a visual debugging mode that displays the active field of view and ray paths of sensors, allowing users to verify sensor coverage and mounting orientation during setup.

3.5 General Interaction Infrastructure

These requirements define the generic capabilities for interacting with the simulation via the network, independent of specific scenarios.

- **FR-16: Universal Command Interface:** The system must provide a standardized network interface to receive text-based commands and return execution status feedback. This generic interface allows the robot to trigger arbitrary simulation events (e.g., attaching tools, resetting states).
- **FR-17: Object Manipulation Capability:** The system must provide a mechanism for the robot to logically attach ("grasp") and detach ("release") objects via a network command, facilitating logistics and transport simulations.
- **FR-18: Surface Modification Capability:** The system must allow the robot to dynamically modify the texture of the floor surface at its current location via a command. This capability is required to generate persistent visual trails for navigation tasks.
- **FR-19: Real-Time Telemetry Visualization:** The system must support the visualization of real-time robot status data (e.g., battery level, system load) directly within the 3D environment via network subscription.

3.6 Scenario-Specific Logic

These requirements define the logic for specific environments used to validate the framework's versatility.

3.6.1 Smart Farming Scenario

- **FR-20: Discrete Agricultural States:** The environment must simulate field areas with discrete states (e.g., Untouched, Cultivated, Seeded). These states must be visually distinct to be detectable by the robot's camera.
- **FR-21: Tool-Based Interaction:** The scenario must simulate specific virtual implements (e.g., Seeder, Cultivator). Environmental state transitions must only occur when the robot interacts with a field area while the correct tool is active.

3.6.2 Logistics & Exploration Scenario

- **FR-22: Goal Validation Logic:** The environment must contain logical zones capable of detecting the presence of specific target objects. Upon valid entry, the system must trigger a visual state change in the object to provide ground-truth confirmation that a transport task is complete.
- **FR-23: Dynamic Obstacle Handling:** The scenario must allow a user to place obstacles in real-time. The simulation must immediately update the physics and visibility calculations so the robot's sensors can detect the new blockage.

3.6.3 Adaptive Navigation Scenario

- **FR-24: Proximity-Based Environment Mutation:** To test adaptive re-planning algorithms, the system must be capable of dynamically altering the environment layout (e.g., changing road segments) in areas distant from the robot, simulating a changing world.

3.6.4 Competitive Game Scenario

- **FR-25: Competitive Game Loop:** The system must manage a game state that tracks scores based on object positions and automatically resets the environment when a point is scored.
- **FR-26: Automated Opponent Entity:** The system must provide a physics-constrained, autonomous entity that reacts to dynamic objects in the scene, providing a moving target or competitor for the robot.

3.7 User Interface & Mixed Reality

These requirements define how human users view and interact with the system.

- **FR-27: Calibrated AR Projection:** The system must render a top-down, orthographic view of the scene. The projection parameters must be configurable to align the virtual view 1:1 with the physical dimensions of the testbed floor.
- **FR-28: VR Visualization:** The system must support a Virtual Reality mode allowing users to view the Digital Twin in 3D space. Status displays attached

to the robot must automatically orient themselves to remain readable from the user's perspective.

- **FR-29: Input Abstraction:** Navigation goals and interaction commands must be triggerable interchangeably by different input methods (e.g., Mouse, VR Controller, or Network Command) without altering the core logic.
- **FR-30: Interactive Goal Setting:** Users must be able to define navigation targets by pointing at the virtual ground using the active input device.

3.8 Automated Verification Agents

To validate the system's utility for testing, requirements are defined for the external automation agents (scripts).

- **FR-31: Sensor-Only Autonomy:** The automated agents must rely solely on simulated sensor data (Camera, LiDAR, Odometry) to make decisions, strictly avoiding access to the simulation's internal ground truth data.
- **FR-32: Visual Perception Integration:** The agents must utilize computer vision algorithms to detect environmental states (e.g., crop maturity, object color, road markings) from the simulated camera feed.
- **FR-33: Closed-Loop Control:** The agents must implement control logic that sends commands and waits for feedback or sensor verification before proceeding, ensuring robust task execution.

3.9 Non-Functional Requirements

- **NFR-01: Latency Minimization:** The end-to-end latency between a physical event and its virtual visualization must be minimized to ensure the Digital Twin remains synchronized with the real world. Ideally, this latency should be lower than the simulation frame time.
- **NFR-02: Frame Rate Stability:** The simulation must maintain a stable target frame rate (e.g., 60 FPS) to prevent simulator sickness in VR and to ensure that AR projections do not exhibit visual artifacts that could confuse the robot's perception system.

- **NFR-03: Interface Modularity:** The automation scripts must be decoupled from the simulation implementation. A script written to control the robot in the simulation must work on the physical EMAROS robot without modification.
- **NFR-04: Scalability:** The system architecture must support scenes with a high density of dynamic objects without violating the frame rate stability requirement.
- **NFR-05: Configuration Flexibility:** Key system parameters, such as network topic names, robot dimensions, and projection resolution, must be accessible via a configuration interface to adapt to different laboratory setups without requiring source code modification.

4 Concept and Implementation

This chapter details the realization of the Mixed Reality Environment based on the requirements defined in Chapter 3. It begins with the selection of the simulation engine, followed by the system architecture, and concludes with the implementation of specific interaction logic and scenarios.

4.1 Technology Selection

To satisfy the platform capabilities defined in Section 3.1 (FR-01 to FR-04), a suitable simulation engine must be selected. The following analysis compares four industry-standard platforms: Gazebo, NVIDIA Isaac Sim, Unreal Engine, and Unity.

4.1.1 Comparative Analysis

The comparison is based on visual fidelity, physics capabilities, learning curve, and native support for XR and ROS 2.

Table 4.1: Comparison of Simulation Platforms for Mixed Reality Digital Twins [FCY25; KYH+24; SKG+25; CIR23].

Feature	Gazebo	Isaac Sim	Unreal Engine	Unity
Primary Use Case	Control & Navigation	AI & Photorealism	Photorealism	HRI & MR (VR/AR)
Visual Fidelity	Moderate	Very High	Very High	High
Physics Engine	ODE/Bullet/DART	PhysX 5 (GPU)	Chaos/PhysX	PhysX
Learning Curve	Steep	Advanced	Steep (C++)	Moderate (C#)
Community Support	High (ROS specific)	Moderate	High (Gaming)	Very High (MR/Gaming)
ROS 2 Integration	Native	Bridge	Bridge	Plugin (Native Stack)
Hardware Req.	Low	Very High (RTX)	High	Moderate

4.1.2 Selection Rationale

Based on this analysis, the **Unity Engine** is selected as the implementation platform for this thesis. This decision is driven by three key factors that align with the requirements:

- **XR Framework (FR-03):** Unity offers a mature, integrated framework for Mixed Reality (AR/VR) [CIR23], whereas Gazebo lacks native VR support.
- **Visual & Physics Balance (FR-01, FR-02):** Unity provides high-fidelity visualization and robust PhysX integration, offering a better balance between performance and quality compared to the high hardware demands of Isaac Sim [FCY25].
- **Development Efficiency:** The use of C# scripting, combined with extensive community support, makes Unity more accessible for rapid prototyping compared to the C++ complexity of Unreal Engine [CIR23].

Furthermore, the availability of the `ros2-for-unity` asset allows the simulation to function as a native ROS 2 node, satisfying the low-latency communication requirement (FR-04) without relying on external bridges [Rob21].

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.

[CIR23] Enrique Coronado, Shunki Itadera, and Ixchel G. Ramirez-Alpizar. “Integrating Virtual, Mixed, and Augmented Reality to Human–Robot Interaction Applications Using Game Engines: A Brief Review of Accessible Software Tools and Frameworks”. In: *Applied Sciences* 13.3 (2023). ISSN: 2076-3417. DOI: 10.3390/app13031292. URL: https://www.mdpi.com/2076-3417/13/3/1292.

[DBF+25] Juliana Dos Santos, Murilo Bicho, Tony Froes, et al. “Benchmarking Digital Twins for Tower Cranes: Isaac Sim vs. Gazebo”. In: *IECON 2025 – 51st Annual Conference of the IEEE Industrial Electronics Society*. 2025, pp. 1–8. DOI: 10.1109/IECON58223.2025.11221823.

[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.

[FCY25] Jose M. Flores Gonzalez, Enrique Coronado, and Natsuki Yamanobe. “ROS-Compatible Robotics Simulators for Industry 4.0 and Industry 5.0: A Systematic Review of Trends and Technologies”. In: *Applied Sciences* 15.15 (2025). ISSN: 2076-3417. DOI: 10.3390/app15158637. URL: https://www.mdpi.com/2076-3417/15/15/8637.

[Foo13] Tully Foote. “tf: The transform library”. In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. Apr. 2013, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

[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.

[KYH+24] Seyed Mohamad Kargar, Borislav Yordanov, Carlo Harvey, and Ali Asadipour. “Emerging Trends in Realistic Robotic Simulations: A Comprehensive Systematic Literature Review”. In: *IEEE Access* 12 (2024), pp. 191264–191287. DOI: 10.1109/ACCESS.2024.3404881.

[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.

[MML+23] Steve Macenski, Tom Moore, David V. Lu, et al. “From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the Robot Operating System 2”. In: *Robotics and Autonomous Systems* 168 (2023), p. 104493. ISSN: 0921-8890. DOI: 10.1016/j.robot.2023.104493. URL: https://www.sciencedirect.com/science/article/pii/S092188902300132X.

[MMW+20] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. “The Marathon 2: A Navigation System”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2718–2725. DOI: 10.1109/IROS45743.2020.9341207.

[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.

[PKP+20] Alexander Perzylo, Ingmar Kessler, Stefan Profanter, and Markus Rickert. "Toward a Knowledge-Based Data Backbone for Seamless Digital Engineering in Smart Factories". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. 2020, pp. 164–171. DOI: 10.1109/ETFA46521.2020.9211943.

[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.

[Rob21] Robotec.AI. *ROS2 For Unity*. GitHub repository. 2021. URL: https://github.com/RobotecAI/ros2-for-unity.

[SKG+24a] 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.

[SKG+24b] Maulshree Singh, Jayasekara Kapukotuwa, Eber Lawrence Souza 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). ISSN: 1424-8220. DOI: 10.3390/s24175680. URL: https://www.mdpi.com/1424-8220/24/17/5680.

[SKG+25] Maulshree Singh, Jayasekara Kapukotuwa, Eber Lawrence Souza Gouveia, et al. "Comparative Study of Digital Twin Developed in Unity and Gazebo". In: *Electronics* 14.2 (2025). ISSN: 2079-9292. DOI: 10.3390/electronics14020276. URL: https://www.mdpi.com/2079-9292/14/2/276.

[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