Modular Sensor-testing Framework for Autonomous Rail Vehicles

Proposal

David Christian Horn

Bachelor Theses
March 2026

Real-Time/Embedded Systems
Department of Computer Science
Kiel University

Advised by
Prof. Dr. Reinhard von Hanxleden, Dr.-Ing. Alexander Schulz-Rosengarten

Contents

List of Acronyms

ROS2 Robot Operating System. 3, 4

Chapter 1

Proposal for Modular Sensor-testing Framework for Autonomous Rail Vehicles

1.1 The Problem

Autonomous rail vehicles rely heavily on accurate perception systems to detect obstacles and interpret their surroundings. Developing and validating such systems in real-world environments is expensive, time-consuming, and often difficult to reproduce. Therefore, reliable simulation environments are essential to test sensor configurations and perception logic safely and consistently before deploying them on physical platforms.

This thesis focuses on designing and implementing a modular simulation framework that enables virtual testing of different sensor setups for an autonomous rail vehicle.How can a modular simulation framework be structured to enable reproducible testing and comparison of virtual sensor configurations for autonomous rail vehicles? The goal is to create a basic yet extensible environment in which environmental conditions and obstacles can be systematically varied to evaluate their impact on perception performance. The framework should emphasize reproducibility, maintainability, and future extensibility, providing a clear structure that other researchers can build upon.

The work is carried out in the context of the **DataTrain** project, which develops an autonomous rail vehicle platform for research on perception, control, and decision-making. Within this larger initiative, this thesis contributes the foundation for a Software-in-the-Loop (SIL) testing environment.

1.2 Planned Solutions and Goals

The primary objective of this work is to design and implement a modular simulation framework that enables testing and evaluation of virtual sensor configurations for autonomous rail vehicles under controlled and reproducible conditions. The focus lies on establishing a functional and extensible foundation rather than on the exhaustive implementation of all possible components.

The proposed framework will allow researchers to deploy a simplified train model within a virtual environment and analyze its perception performance under varying environmental and obstacle configurations. Emphasis is placed on modularity, reusability, and reproducibility, ensuring that future extensions and sensor models can be integrated with minimal effort.

1.2.1 Hard Goals

These goals define the core deliverables of the thesis. They are achievable within the project timeframe and directly support the main research question.

1. **Develop a basic simulation environment** that represents a train moving along a track and interacting with simple obstacles and environmental conditions (e.g., lighting or weather variations).
   Rationale: This forms the minimal virtual testbed necessary to evaluate perception logic without hardware dependencies.
   Measurability: The framework supports at least two distinct environments and two obstacle types, detectable by at least two different sensor components.

2. **Design a modular software architecture** that allows the integration and configuration of multiple virtual sensors in a consistent and scalable way.
   Rationale: A modular structure ensures the framework can evolve as new sensing modalities are introduced in future research.
   Measurability: A concise user manual demonstrates how to adjust environment parameters, weather, and sensor placement or type without modifying the core code.

3. **Establish standardized data interfaces** for sensor outputs, environment descriptions, and evaluation results.
   Rationale: Standardized data structures enable comparability between simulation runs and facilitate automated evaluation.
   Measurability: The output data allows clear identification of whether, when, and where an object was detected, and can be easily exported or streamed in a consistent format for further analysis.

1.2.2 Soft Goals

These goals are desirable extensions that contribute to the overall usability and flexibility of the framework but are not essential for the successful completion of the thesis. They may be addressed if time permits.

1. Extend the environment to include additional sensor types or more complex scenarios.
   Rationale: This broadens the applicability of the framework for future research.

2. Provide a minimal YAML-based configuration interface for simulation control.
   Rationale: Enables quick and reproducible setup of different scenario configurations.

3. Implement an extended evaluation and visualization workflow for sensor data.
   Rationale: Provides quantitative feedback on sensor performance and supports result interpretation.

4. Introduce debug configurations for sensors and simulation parameters.
   Rationale: Facilitates systematic troubleshooting and validation during development.

5. Enable reproducible deployment through containerization or similar techniques.
   Rationale: Ensures portability and consistent behavior across different systems.

6. Implement a simple actuation simulation (e.g., PID-based control).
   Rationale: Adds realism to the simulation by modeling train acceleration and braking behavior.

This project thus prioritizes conceptual clarity, modular structure, and reproducible results over exhaustive technical coverage. The resulting framework aims to provide a robust and extensible foundation for future research and development within the DataTrain project.

1.3 Technical Approach

1.3.1 Concept and Tooling Overview

1.4 Technical Approach

1.4.1 Concept and Tooling Overview

The implementation follows a modular and extensible design philosophy. The simulation framework is based on established open-source robotics tools that provide reliable communication between components and allow the integration of custom sensor and evaluation modules. While technologies such as Robot Operating System (ROS2) and modern 3D simulation environments (e.g., Gazebo) serve as the underlying platforms, the focus of this thesis lies on the conceptual architecture, the interaction between components, and the reproducible setup of simulation experiments.

The framework is structured around clearly defined interfaces connecting simulation, perception, and evaluation layers. This modular design enables new sensors or analysis modules to be added without significant structural changes. All components exchange data through standardized formats, facilitating monitoring, debugging, and systematic evaluation of the simulation.

To achieve these objectives, the framework adopts a ROS2-based communication architecture. ROS2 (Robot Operating System 2) provides modularity, interoperability, and scalability through standardized publish–subscribe communication patterns such as topics, services, and actions. Leveraging ROS2 ensures that the developed system aligns with state-of-the-art practices in robotics research and remains compatible with future extensions or integration into larger projects.

The implementation is primarily carried out in Python using the `rclpy` client library. This allows for rapid prototyping and straightforward integration of data analysis and evaluation tools while maintaining consistency with the overall system design.

1.4.2 Simulation Environment

For the simulation environment, the project employs **Gazebo** a 3D physics simulator that integrates seamlessly with ROS2.

Gazebo provides realistic simulation of train dynamics, track geometries, and sensor models, while supporting the inclusion of physical effects such as noise, occlusion, friction, and weather.

This combination enables the controlled evaluation of perception and control layers within a fully virtual, reproducible environment.

A simplified virtual rail environment will be created, containing a basic train model, a track, and configurable environmental conditions such as obstacles or lighting variations. The goal is not to achieve physical realism, but to provide a controllable and repeatable testing setup that supports experimentation with different sensor configurations.

The simulation setup allows:

- ▻ Controlled variation of environmental and obstacle conditions for testing perception robustness.
- ▻ Reproducible execution of predefined scenarios for comparison and evaluation.
- ▻ Integration of multiple virtual sensors through a modular interface layer.

1.4.3 Architecture and Communication

The overall system architecture follows a modular design, where each component (e.g., simulation, sensor, perception, evaluation) operates independently but communicates through well-defined interfaces. This approach supports flexible data exchange and simplifies debugging and system expansion.

Dedicated interfaces will be designed for:

- ▻ **Sensor data exchange:** Providing structured outputs for detection and localization.
- ▻ **Environment and configuration management:** Allowing parameterized setup of weather, lighting, and obstacle types.
- ▻ **Evaluation and logging:** Capturing relevant information to verify detection performance and enable later analysis or visualization.

1.4.4 Debugging and Evaluation Interfaces

In addition to the simulation core, the framework will include tools for observing and analyzing system behavior. This is mainly part of the soft featrues, because the extend of this will strongly depend on the project's progress throughout the semester. Ideally i am able to configure some automated output evaluation pipelines for the sim using preexisting framework options. These interfaces are intended to support both development debugging

and later quantitative evaluation. The goal is to make system outputs transparent and easily interpretable.

The debugging and evaluation setup will include:

- ▻ Configurable outputs for program and sensor-level debugging.
- ▻ Data logging and export mechanisms to enable later plotting, comparison, and performance evaluation.
- ▻ A basic structure for visual inspection of simulated sensor data and perception results.

This approach emphasizes modularity, traceability, and simplicity, providing a foundation for future extensions.

Chapter 2

Schedule

2.1 Implementation Schedule

2.2 Writing Schedule

Writing and Documentation Deadlines

To ensure steady progress and consistent feedback, the following internal writing deadlines are defined:

1. **16.01.2026:** Submission of first draft chapters — *Introduction, Related Work, and Deployment Setup.*
2. **09.02.2026:** Submission of draft for *System Design* and start of implementation documentation.
3. **02.03.2026:** Submission of draft chapters — *Evaluation and Results.*
4. **15.03.2026 (latest):** Submission of the *final complete thesis draft* for review and final feedback.

Table 2.1. Planned schedule for implementation and documentation.

Phase	Timeframe	Objectives and Deliverables
1. Project Setup & Planning	03.11.–10.11.2025	Finalize project scope and technical requirements. Set up version control, simulation environment, and basic documentation structure. Verify compatibility of development tools and middleware.
2. Simulation Environment Setup	11.11.–30.11.2025	Create the base simulation environment with a simplified rail track and train model. Implement minimal motion behavior (start, stop, acceleration) and define environment parameters (lighting, weather). Verify reproducibility of basic scenarios.
3. Sensor Framework Integration	01.12.–20.12.2025	Design modular sensor interfaces and integrate at least two basic sensor types. Ensure consistent data structure for sensor outputs. Validate sensor functionality and scenario variability (e.g., different obstacles or weather).
4. Data Interface and Logging Design	05.01.–20.01.2026	Implement standardized data interfaces for sensor output, environment setup, and evaluation data. Enable structured logging and basic export functionality for later analysis.
5. Evaluation and Analysis Preparation	21.01.–04.02.2026	Define basic evaluation criteria (e.g., object detection success rate). Create mechanisms for recording and analyzing simulation results. Prepare initial test cases and validation runs.
6. Debugging & Visualization Tools (Optional)	05.02.–18.02.2026	Introduce optional visualization and debugging features. Configure simple dashboards or outputs for observing sensor data and detection performance. Optional: integrate batch testing or simple configuration interface.
7. Final Integration & Documentation	19.02.–28.02.2026	Integrate all system components, verify reproducibility, and perform final test runs. Finalize user instructions and internal documentation. Prepare for evaluation by supervisors.
8. Thesis Writing & Submission	01.03.–30.03.2026	Finalize the written report, figures, and evaluation summary. Prepare presentation slides and submit the completed thesis.