Proceedings of the Seventh International Conference on
Railway Technology: Research, Development and Maintenance
Edited by: J. Pombo
Civil-Comp Conferences, Volume 15, Paper 14.1
Civil-Comp Press, Edinburgh, United Kingdom, 2026
ISSN: 2753-3239, doi: 10.4203/ccc.15.14.1

CPU-Based Real-Time LiDAR Simulation for Automatic Train Operation Scenarios

T. Hofmeier[1,2] and M. Cichon[2]

[1]Automotive Software Systems Engineering, Nuremberg Institute of Technology, Germany
[2]Institute of Vehicle System Technology, Karlsruhe Institute of Technology, Germany

Abstract

Recent European railway initiatives such as Shift2Rail emphasise sustainability, capacity increase, and cost efficiency through automation and digitalisation. In this context, Automatic Train Operation increasingly relies on robust environment perception for safe on-sight operation, where LiDAR sensors provide accurate geometric information for obstacle detection and infrastructure monitoring.
Driven by advances in automotive and robotics research, many LiDAR simulation approaches focus either on high physical fidelity at significant computational cost or on offline dataset generation, limiting their suitability for real-time closed-loop validation. For railway-oriented Automatic Train Operation testing, however, a lightweight, engine-native, and reproducible real-time simulation is required.
This paper presents a CPU-based LiDAR simulation implemented natively in Unreal Engine 5. The approach combines engine-native ray casting, JSON-configurable parameters, and UDP-based point cloud streaming with a pragmatic plausibility model. Instead of reproducing full physical sensor behaviour, selected effects such as distance attenuation, material-based reflectivity proxies, incidence-angle weighting, and measurement noise are incorporated to generate perception-relevant point cloud characteristics under real-time constraints. The simulation enables reproducible and configurable LiDAR perception for on-sight Automatic Train Operation validation and is released as part of the open-source Vir2Rail environment, which provides railway-specific assets and scenarios.

Keywords: automatic train operation, on-sight train operation, LiDAR simulation, Unreal Engine, real-time simulation, virtual validation

1 Introduction

Recent European railway research and innovation programmes such as Shift2Rail and Europe's Rail emphasise digitalisation and automation as key enablers for increasing capacity, improving efficiency, and supporting a more sustainable railway system. [1], [2]

In this context, Automatic Train Operation (ATO) has become a central development direction and can be introduced incrementally across different Grades of Automation (GoA). Higher levels of automation, such as GoA4, are already deployed in controlled environments, but extending ATO to open railway systems introduces significantly higher operational complexity.[3], [4]

Open railway systems introduce higher scenario diversity, stronger infrastructure interaction, and safety-critical degraded modes under uncertainty. Consequently, safety assurance and verification for on-sight operation scenarios increasingly depend on reliable environment perception. [2]

This paper focuses on on-sight operation as a key operational principle for ATO development. On-sight operation is commonly defined as a mode in which movement authority and speed are constrained to enable safe driving based on direct observation of the track ahead ("line-of-sight"), typically under low-speed conditions and requiring reactions to obstacles [3], [5]. In conventional on-sight operations (GoA0), these decisions are made by human operators. In on-sight ATO, the same operational principle is retained, but the decision-making and reaction capability, including the detection of relevant objects and infrastructure conditions, the assessment of clearance and stopping distance, and the execution of appropriate speed and braking commands, is performed by an automated system [6]. This shifts the functional burden from human perception and judgement to machine perception and safety-relevant decision logic, which fundamentally increases the relevance of sensor-centric verification and validation and the need for traceable evidence across the lifecycle [7].

Given these characteristics, virtual simulation is an essential tool for the verification and validation of perception-relevant ATO functions. It enables systematic scenario variation, repeatability, and early closed-loop testing before costly field campaigns are required, reducing development risk while increasing coverage of safety-critical scenarios [8], [9]. Simulation-based testing thus complements established railway lifecycle processes and supports the generation of traceable evidence for safety-oriented development and approval workflows [7].

Among available perception modalities, LiDAR sensors are well suited for on-sight railway applications as they provide accurate geometric information for obstacle detection, clearance monitoring, and infrastructure perception [10]. Consequently, realistic and configurable LiDAR simulation is a key enabler for the virtual testing of perception-driven ATO functions in on-sight operation.

In parallel to these developments, applied research projects address the transition of automated railway vehicles towards series maturity. One example is the VAL (Vollautomatische Abdrücklokomotive) project, which targets the automation of low-speed on-sight shunting operations in marshalling yards, where decisions are traditionally based on direct environment perception from the vehicle [11]. Within VAL, simulation-based testing is essential to evaluate perception-driven components prior

to large-scale field trials, requiring representative virtual sensor data to assess system behaviour across relevant scenarios under limited safety margins. At the start of development in early 2022, commercially available automotive simulation solutions were assessed but found insufficient for railway-specific requirements, particularly regarding real-time closed-loop integration, configurability, and railway assets. Consequently, a custom simulation environment was developed for flexible and reproducible testing, with the LiDAR module presented here forming a core, reusable, engine-native component.

2 State of the Art and Requirements of LiDAR Simulation

LiDAR simulation approaches span a spectrum from physically detailed radiometric models to dataset-oriented offline frameworks and real-time engine-based implementations. While these approaches address different objectives, their suitability depends on the application context.

For on-sight ATO, LiDAR simulation must operate within a closed-loop validation environment, where perception, decision logic, and vehicle motion interact in real time. Consequently, the key evaluation criteria are not maximum physical fidelity alone, but the balance between geometric plausibility, computational tractability, integration effort, and reproducibility.

Recent comparative work structures LiDAR simulation approaches according to realism–performance trade-offs and the sensing phenomena they model, such as ray dropout, multi-echo, motion distortion, and reflectivity effects [12].

This categorisation provides a useful lens for assessing relevant modelling effects under closed-loop railway ATO constraints.

2.1 Physically Motivated and Radiometric Models

Physically motivated LiDAR models approximate the sensing process at a high level of fidelity by modelling emission characteristics, surface interaction, material-dependent reflectance, atmospheric attenuation, and potentially multi-return phenomena. In automotive-oriented literature, such approaches incorporate material properties and sensor capability limits to produce more realistic return behaviour and intensity-related effects[13]. These models are particularly valuable for sensor characterisation, calibrated intensity modelling, and detailed studies of how material and environmental conditions affect returns.

From the perspective of on-sight ATO validation, however, these approaches are challenging to deploy in real-time closed-loop systems. Computational complexity increases with beam count and scene detail when radiometric interactions are evaluated per ray, limiting scalability under deterministic runtime constraints. Moreover, for geometry-driven railway perception tasks such as obstacle detection, clearance monitoring, and coarse infrastructure localisation, the benefit of full radiometric fidelity is often limited compared to its computational cost.

2.2 Dataset-driven and Offline Generation Frameworks

In contrast to physically motivated approaches, a second class of LiDAR simulation methods focuses on synthetic dataset generation for training and benchmarking

perception algorithms. In these frameworks, sensor data are generated offline and exported as labelled point clouds for downstream processing. In the railway domain, TrainSim provides a simulation framework for sensor dataset generation in trackside environments, enabling systematic scenario variation and evaluation. [14]

The main strength of dataset-driven approaches lies in their scalability and controllability during data generation. Large scenario variations can be produced systematically, and precise ground truth annotations can be attached to each frame, making such frameworks suitable for supervised learning workflows and benchmarking studies.

However, offline generation inherently decouples sensor simulation from vehicle control. The interaction between perception output and subsequent control decisions is not represented during data production. For on-sight ATO validation, where braking behaviour, obstacle response, and motion dynamics depend on real-time perception feedback, this separation limits applicability. In addition, systematic variation of sensor parameters typically requires regeneration of datasets, which reduces iteration speed in engineering-oriented development cycles.

2.3 Engine-based Real-time Simulation Approaches

To address these limitations in closed-loop validation, real-time engine-based approaches aim to generate LiDAR point clouds interactively within modern simulation engines, balancing geometric realism and computational efficiency. Game engines provide mature tooling for scenario authoring, environment variation, and the integration of vehicle dynamics and control systems, making them attractive platforms for closed-loop autonomy testing. In the automotive domain, CARLA demonstrates configurable sensor suites and scenario-based evaluation within an Unreal Engine (UE) environment [15]. Similarly, AirSim provides a UE–based platform for autonomous systems research, including configurable LiDAR sensors and integration with external middleware such as ROS, illustrating the feasibility of real-time sensor simulation within interactive validation environments [16]. In railway research, UE-based environments have been employed to support digital-twin concepts and scenario-driven ATO validation [17], [18].

Within this class of solutions, LiDAR implementations differ substantially along several design axes. A primary distinction concerns the sensing abstraction: some implementations are geometry-first and rely mainly on ray casting or depth sampling, optionally augmented with lightweight noise models. Others attempt to integrate additional sensor effects such as intensity proxies, probabilistic dropout, motion distortion, or multi-echo behaviour. Comparative studies indicate that not all modelling effects contribute equally to downstream autonomy performance, and that selective inclusion of high-impact phenomena may offer a favourable realism–performance trade-off under real-time constraints [12].

For on-sight ATO engineering workflows, further practical considerations become decisive. Closed-loop validation requires deterministic update rates and predictable latency. Implementations relying on GPU-heavy pipelines, custom engine forks, or complex plugin ecosystems may introduce integration overhead, hardware dependencies, or reduced reproducibility across development environments. In safety-oriented validation contexts, controllable abstraction, deterministic behaviour, and ease of integration often take precedence over maximum visual or radiometric fidelity. While

these approaches demonstrate the feasibility of real-time LiDAR simulation, their direct integration into existing ATO simulation environments is often limited. LiDAR simulation inherently requires tight coupling with scene representation, collision handling, and material properties to capture the interaction between sensor and environment. This coupling can conflict with existing system architectures and sensor implementations, making reuse and transfer between projects non-trivial, including in railway applications where existing solutions are not designed for tightly coupled integration within ATO validation frameworks.

2.4 Derived Requirements for an ATO-Oriented LiDAR Simulator

In on-sight ATO, perception outputs directly influence low-speed braking decisions and clearance assessment under constrained safety margins. Consequently, predictable runtime behaviour and reproducibility are essential for systematic verification and validation, shifting the design focus from maximising physical realism towards controlled abstraction levels suitable for geometry-driven perception tasks in closed-loop validation.

Based on the preceding discussion, LiDAR simulation approaches can be characterised by their level of sensing abstraction and computational complexity. These abstraction levels describe how geometric information and additional sensing effects are represented and processed within a simulation environment.

For on-sight ATO validation, the key question is not which platform is used, but which abstraction level provides sufficient geometric plausibility under real-time closed-loop constraints. Existing approaches indicate that geometry-first implementations augmented with selected modelling effects provide a favourable balance between realism and computational tractability for railway-specific validation.

From this synthesis, the following technical requirements for an ATO-oriented LiDAR simulator are derived:

R1 Real-time execution at representative update rates: Sustaining typical LiDAR update rates and point throughput under representative railway scene complexity for stable closed-loop testing.

R2 Engine-native integration without modification: Deployment as a native Unreal Engine module or actor without engine forks or external plugins, ensuring portability and low integration overhead.

R3 Configurable scan geometry and sensor abstraction: External configuration of scan pattern, angular resolution, maximum range, coordinate alignment, and update rate for systematic parameter studies.

R4 Targeted plausibility effects with favourable realism–cost trade-off: Implementation of selected modelling effects (e.g., distance attenuation, incidence-angle weighting, material reflectivity) to balance realism and computational cost.

R5 Reproducibility and controllable stochasticity: External control of configuration parameters and stochastic components (e.g., noise, dropout)

R6 Lightweight and toolchain-agnostic data interfaces: Low-latency point cloud output (e.g., UDP streaming) for integration with external perception and evaluation systems.

These requirements define the architectural and modelling principles guiding the implementation described in Section 3.2

3 Unreal Engine-Based LiDAR Simulation Implementation

Within the VAL project [11], perception-driven ATO components are evaluated under controlled simulation conditions prior to large-scale field testing. The LiDAR simulator presented in this paper forms a core sensing component within this validation workflow. To facilitate reproducible experimentation and to lower the entry barrier for railway-oriented ATO simulation research, a deliberately reduced and self-contained subset of the development environment is provided as an Unreal Engine 5 (UE5)–based open-source project named Vir2Rail [19]. The released project represents a technically consistent extraction of the sensor simulation and its immediate runtime context, while project-specific integration components and advanced validation tooling developed within VAL are intentionally excluded.

Vir2Rail is distributed as a complete in-editor UE5 setup. It provides parametrised vehicle motion with configurable velocity profiles, railway infrastructure, and real-time streaming of LiDAR and camera data. The environment itself does not implement a control loop; instead, it serves as a deterministic sensor data generator that can be connected to external perception and control systems if required.

3.1 Vir2Rail: Open-Source Unreal Engine Railway Simulation Environment

Vir2Rail is released as a complete in-editor UE5 project and does not rely on third-party plugins or proprietary middleware components. This deliberate design decision ensures portability, full source-level transparency, and unrestricted extensibility. By distributing the project in editor form, users obtain direct access to all C++ classes, configuration files, and assets, enabling individual adaptations, performance optimisations, and integration into custom research toolchains without requiring engine modifications. The environment is structured around two central actor classes, DemoTrack and DemoVehicle, which together define the spatial and kinematic context of the simulation. Figure 1 provides an overview of the environment and assets.

Figure 1: Vir2Rail Overview and Railway Assets

The DemoTrack class encapsulates spline-based track generation and associated catenary infrastructure. Track curvature, alignment, and length can be configured

directly within the editor, while masts and catenary elements are positioned parametrically along the spline. This enables controlled variation of scene complexity and infrastructure density without manual geometry editing. The DemoVehicle actor represents the moving platform on which sensor modules are instantiated. Vehicle motion is defined kinematically by an editor-configurable velocity and start position along the spline. During runtime, the vehicle traverses the track deterministically while attached sensor actors generate data streams. No internal control logic is implemented; the vehicle serves as a reproducible motion carrier for perception experiments.

Figure 2: Vir2Rail Railway Assets

To support perception-oriented test scenarios, rolling stock assets are included, as illustrated in Figure 2. These allow structured generation of approach, clearance, and coupling scenarios relevant to shunting and low-speed operations. Object placement is handled by the helper class DemoObjects, positioning wagons or obstacles at defined spline coordinates, ensuring reproducible configurations and systematic scenario variation without manual repositioning.

Figure 3: Vir2Rail Ego-Vehicle View

Figure 3 shows the in-vehicle perspective together with the corresponding camera simulation output, which is obtained from UE5's native rendering pipeline and transmitted without additional modelling.

An overarching weather system allows environmental parameters such as rain, fog, snow, and solar position to be adjusted. In addition, the helper class RandomWeather generates randomized sunny, foggy and rainy or snowy weather configurations for scenario variation, if needed. All major subsystems are configured via external JSON files (Weather.json, Interfaces.json, Camera.json, Lidar.json), supporting reproducibility and controlled parameter studies. The structure and functionality of the LiDAR configuration are described in the following section.

3.2 Modular LiDAR Simulation Architecture

The LiDAR simulator follows a modular processing architecture that separates an initialisation phase from the repeated runtime sensing loop, as illustrated in Figure 4. To analyse architectural trade-offs between computational performance, data handling overhead, and modelling fidelity under identical scan geometry and scene conditions, the simulator was implemented as a sequence of four modules with increasing architectural complexity.

The simulator employs CPU-based ray casting using UE5's native line tracing facilities rather than GPU-based rendering approaches. While GPU-based LiDAR simulation can provide high throughput through shader-based rendering pipelines, integrating such approaches with scene collision information, material properties, and asset-level metadata often increases implementation complexity and limits portability across projects. In contrast, UE5's native collision queries provide direct access to hit events, surface normals, and physical materials associated with scene geometry. This enables straightforward integration of material-dependent reflection proxies and environment-aware sensing effects without requiring custom rendering pipelines, while maintaining portability across different simulation setups.

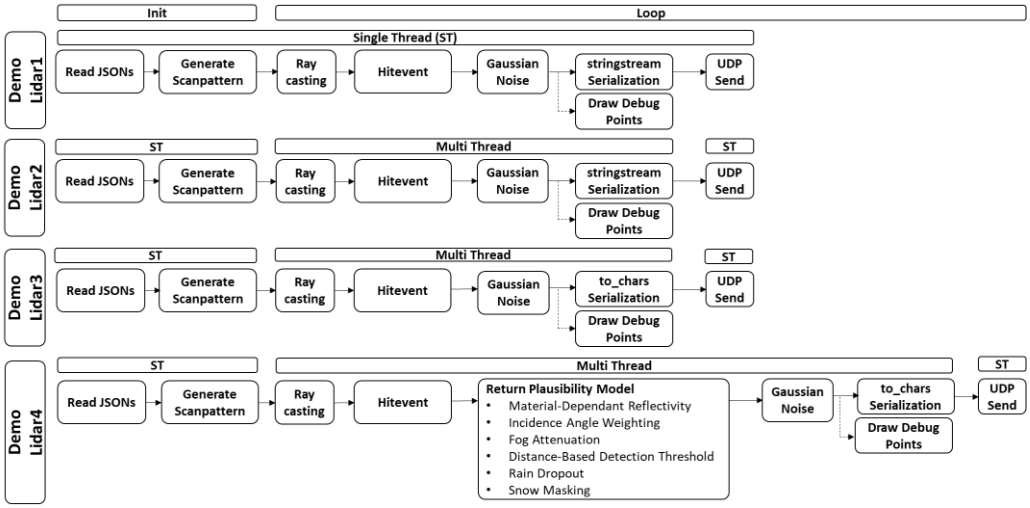

Figure 4: LiDAR Module Architecture

Across all modules, the simulator follows the same fundamental sensing principle. A configurable scan pattern is generated from externally defined sensor parameters, including field of view, angular resolution, maximum range, and update frequency, which are loaded from an external JSON configuration during initialisation. Beam directions are precomputed once and reused during runtime to minimise computational overhead. During operation, periodic scan cycles are executed via the UE5 tick mechanism, depending on the pre-set target frequency, where each beam is evaluated using the engine's collision system to obtain the first blocking intersection with the scene geometry. The resulting hit event provides the impact position, surface normal, and associated physical material, forming the basis for subsequent sensor modelling and also contains additional information that could be used for further model enhancement. The four modules implement successive extensions of this sensing pipeline in

order to isolate architectural and modelling effects. Starting from a single-threaded geometric baseline (Module 1), the simulator introduces multithreaded beam processing, using the ParallelFor function (Module 2) and optimised serialisation (Module 3), allowing the impact of computation and data handling to be evaluated independently. This incremental design enables a controlled analysis of performance improvements, which is presented in Section 4.

After the architectural optimisation stages introduced by Modules 1-3, Module 4 extends the sensing pipeline with a lightweight plausibility model to increase the realism of the simulated point cloud while preserving real-time performance. Rather than modelling the full physical sensing process, the approach introduces a small set of effects that directly influence the spatial distribution and reliability of LiDAR returns.

The plausibility model evaluates each geometric hit in a sequential manner, combining surface properties, incidence angle, range, and atmospheric attenuation. First, a return proxy is computed based on a material-specific reflection coefficient, obtained from the associated physical material where available, and weighted by the cosine of the incidence angle. This suppresses grazing-angle detections and introduces a simplified material dependency. The resulting proxy is further scaled by an exponential attenuation term based on the configured fog density parameter. This parameter corresponds to the UE5 exponential height fog setting and is used here as an engineering proxy to approximate visibility-related signal degradation with increasing distance. A return is accepted only if the attenuated return proxy exceeds a minimum threshold. This threshold is defined from a configurable reference value at a reference distance and, in the current implementation, increases with the squared distance ratio relative to the reference distance.

In addition to this deterministic plausibility evaluation, stochastic weather effects are applied to the remaining valid returns. All weather-related parameters are externally configured via a dedicated JSON interface, enabling reproducible scenario definition and systematic variation. Rain is modelled as a distance-dependent dropout process that removes points with increasing probability over path length and rain intensity. The rain intensity parameter is mapped to approximate precipitation rates (mm/h) based on empirical observations, allowing a coarse alignment with real-world conditions. In contrast, fog density and snowfall are treated as engineering proxies without direct physical calibration. Concretely, snow is represented as near-range clutter by injecting or replacing returns near the sensor origin. These synthetic snow returns may overwrite existing geometric points and thereby introduce masking effects within the point cloud. Finally, additive Gaussian noise is applied to the resulting point positions with a variance that increases with distance.

A central design objective of this modelling approach is the use of parameters that can be obtained or approximated with limited effort, for example from sensor specifications, simple measurements, or coarse material categorisation. The model therefore does not aim to reproduce the exact physical behaviour of a specific LiDAR device. Instead, it focuses on generating point cloud characteristics that are sufficiently realistic for downstream perception tasks. This includes angle-dependent return loss, range-limited detectability, and weather-induced degradation effects. The resulting abstraction provides a practical balance between realism, configurability, and computational efficiency for real-time ATO validation.

4 Evaluation and Results

The proposed LiDAR simulation was evaluated with respect to computational performance, scalability, and the influence of environmental scene complexity on runtime behaviour. The evaluation focuses on two aspects: the architectural progression of the four simulation modules and the influence of environment modelling choices on the computational load of the sensing pipeline.

All experiments were conducted within the Vir2Rail simulation environment introduced in Section 3.1 on a workstation equipped with an Intel Xeon W-1370P CPU (3.60 GHz), 128 GB RAM, and an NVIDIA RTX A5000 GPU. As the proposed LiDAR simulation is CPU-based, the GPU was not utilised for sensor computation. To ensure reproducibility and comparability, the simulated vehicle remained stationary during all measurements, and a fixed LiDAR configuration was used. The sensor was configured with a horizontal field of view of 60°, a vertical field of view of 30°, an angular resolution of 0.25° in both directions, a maximum range of 300 m, and a target update rate of 10 Hz. This configuration provides a representative but computationally demanding scan setup for comparing module performance under identical conditions.

The performance evaluation is based on the game time reported for the LiDAR module within the UE5 profiling environment. For each configuration, the module was executed repeatedly under identical conditions twice, and the reported game time was recorded over ten consecutive scans each time and averaged to reduce variability caused by background CPU activity. The resulting values therefore represent the engine-side execution cost attributed to the LiDAR module under the respective benchmark configuration. In UE5, the achievable frame rate (FPS) is given by the inverse of the frame time and is limited by the maximum of the CPU (game thread) and GPU pipeline times, including both the sensor simulation and the underlying scene load.

LiDAR-Module	Mean Game Time [ms]	Standard Deviation [ms]	Min Game Time [ms]	Max Game Time [ms]	Speed-up to Module 1 [-]
Module 1	168.38	2.73	165.1	178.1	1.00x
Module 2	135.33	2.71	132.1	141.7	1.24x
Module 3	42.68	4.07	39.4	45.7	3.95x
Module 4	42.56	2.14	40.3	46.7	3.96x

Table 1: Game Time (CPU cost) comparison of the different LiDAR modules

The results in Table 1 highlight the impact of the architectural changes across the four modules. While multithreading in Module 2 reduces execution time by approximately 24% compared to the single-threaded baseline, the observed speedup remains significantly below the theoretical expectation for parallel beam evaluation. This is primarily due to synchronisation overhead introduced by thread-safe data handling, including the use of locks to prevent race conditions during shared data access.

Profiling results further indicate that, in Module 2, the parallelised beam evaluation accounts for about 30% of the total execution time, with a substantial portion of runtime spent in data handling and serialisation. This observation reveals that the computational bottleneck is not dominated by ray casting itself, but by downstream processing. This is confirmed by Module 3, where optimised serialisation reduces

runtime by nearly a factor of four compared to the baseline. The results demonstrate that data handling overhead has a significantly larger impact on performance than geometric ray evaluation under the given configuration. Module 4 maintains a comparable runtime to Module 3 despite additional plausibility modelling and weather effects, indicating that the proposed modelling approach introduces negligible computational overhead while preserving real-time capability.

These results confirm that the proposed architecture satisfies the requirements derived in Section 2.4. In particular, real-time execution (R1) is maintained even with extended modelling, while the engine-native implementation (R2) and externally configurable parameters via JSON files (R3, R5) enable flexible experimentation. Furthermore, the results demonstrate that the selected plausibility effects (R4) can be integrated without compromising computational efficiency, and that the data interface (R6), implemented via UDP streaming, does not introduce a performance bottleneck.

While the previous analysis isolates the computational cost of the LiDAR modules, the real-time capability of the overall simulation is not determined by the sensor alone. In UE5, the effective tick rate is governed by both the game thread time (CPU) and the GPU rendering time, with the slower path limiting the simulation rate. To account for this, additional measurements were conducted under varying environment configurations. Baseline times were first recorded without active sensors, followed by enabling the camera module to capture camera processing overhead. Subsequently, scene complexity was modified stepwise by removing foliage, enabling collision for vegetation, and activating shadow casting for grass elements. This allows the impact of rendering and collision complexity on real-time performance to be assessed and highlights the importance of environment design alongside sensor implementation.

Level Configuration	Mean GPU Time [ms]	Mean Game Time [ms]	GPU Time Std. Dev. [ms]	Game Time Std. Dev. [ms]
BaseLevel	16.60	7.26	0.23	0.41
BaseLevel + Camera	16.54	19.80	0.19	0.98
No Foliage	10.26	5.71	0.11	0.09
No Shadow on Foliage	16.74	7.48	0.22	0.36
Collision on Grass	17.02	6.96	0.28	0.80

Table 2: Influences on GPU & Game Time by different additional Settings

The results in Table 2 highlight that real-time performance in UE5 is not solely determined by sensor computation, but strongly influenced by scene configuration and rendering workload. While the LiDAR module itself operates within the game thread, the effective simulation rate is jointly limited by CPU and GPU (render time). Removing foliage results in a substantial reduction of GPU time and also lowers game time, indicating that vegetation is a dominant contributor to overall scene complexity. In contrast, disabling shadows on foliage shows only minor performance improvements, suggesting that geometric complexity rather than shading dominates the rendering cost in this configuration. Enabling the camera module leads to an increase in game time, while GPU time remains largely unchanged. This indicates that the overhead is not caused by additional rendering, but by the extraction, processing, and transmission of the already rendered image data within the game thread.

Notably, enabling collision on grass has only a minor impact on both GPU and CPU runtime, indicating that collision queries themselves do not represent a critical bottleneck for real-time execution. However, this configuration has a substantial impact on level loading and unloading times due to the increased number of collision-enabled assets. While this overhead remains manageable in small-scale environments such as the Vir2Rail demo setup, it becomes a limiting factor for large-scale maps and closed-loop system testing, where high scenario throughput is required.

These findings underline that achieving real-time capability in UE5–based simulation requires careful balancing of rendering complexity and collision modelling. In particular, large numbers of collision-enabled foliage assets should be avoided in scalable validation environments, even if their runtime impact appears negligible. Overall, the evaluation demonstrates that simulation performance can be tailored to specific application requirements by balancing LiDAR configuration parameters (e.g., resolution, range, and modelling fidelity) with environment complexity.

5 Conclusions and Contributions

This paper presented a CPU-based real-time LiDAR simulation approach implemented natively in UE5 for on-sight ATO validation. The evaluation demonstrates that real-time performance can be achieved using a geometry-first sensing approach combined with lightweight plausibility modelling. The results further show that computational performance is not primarily limited by ray-based sensing itself, but by data handling and engine-level factors such as rendering load and scene complexity. In particular, the modular analysis reveals that multithreading alone provides limited speedup due to synchronisation overhead, whereas optimised data serialisation yields a substantially larger performance gain. At the system level, the achievable simulation rate is jointly determined by CPU-based sensor processing and GPU-based rendering, highlighting that real-time capability depends not only on sensor implementation but also on appropriate environment design.

The proposed modelling approach focuses on a small set of configurable parameters that can be obtained or approximated from sensor specifications, datasheets, or simple comparative measurements. This enables a pragmatic alignment of simulated sensor behaviour with expected real-world characteristics without requiring full physical calibration. At the same time, the approach introduces limitations. For landscape-based materials, missing collision representations for foliage can lead to unrealistically steep incidence angles, resulting in excessive attenuation of returns. In the current implementation, this effect can be compensated by increasing the reflectivity of the underlying landscape material, representing an engineering workaround rather than a physically grounded solution. Similarly, highly reflective smooth surfaces, such as the running surface of rails, are overrepresented in the model, as specular reflections are not explicitly handled and returns are always assumed to reach the sensor. This effect can be mitigated by manually reducing reflectivity values for such materials, but again requires case-specific adjustments. These observations highlight that the current approach relies partly on manual parameter tuning to achieve plausible results.

The main contribution of this work is therefore not a physically complete sensor model, but a reproducible and engineering-oriented simulation framework that

balances realism, configurability, and computational efficiency. If a closer approximation of a specific real-world LiDAR sensor is required, the model parameters must be adjusted accordingly based on measurements or available reference data. All necessary interfaces are exposed to support such calibration. In this context, the Vir2Rail environment is released as an open-source UE5 project, providing a transparent and extensible basis for LiDAR-based ATO research and validation. Future work will focus on reducing the need for such manual adjustments by extending the model with additional surface descriptors, such as roughness and material-dependent scattering behaviour, as well as incorporating more advanced return modelling to better capture real sensor characteristics.

Acknowledgements

This work was partly accomplished within the project VAL, FKZ 5320000013, EBA Ref. 8fd/003-1255#008-VAL, funded by the German Federal Ministry of Transport.

References

[1] Europe's Rail Joint Undertaking, 'Shift2Rail Strategic Master Plan', European Union, 2015. Accessed: Jan. 08, 2026. [Online]. Available: https://rail-research.europa.eu/wp-content/uploads/2021/06/Shift2Rail-Master-Plan_approved-by-S2R-GB.pdf

[2] Europe's Rail Joint Undertaking, 'Europe's Rail Multi-Annual Work Programme', European Union, 2022. Accessed: Jan. 08, 2026. [Online]. Available: https://rail-research.europa.eu/wp-content/uploads/2022/03/EURAIL_MAWP_final.pdf

[3] IEC, IEC 62290-1:2014 Railway applications – urban guided transport management and command/control systems – part 1: system principles and fundamental concepts, IEC 62290–1, 2014.

[4] UITP, 'World Report on Metro Automation 2018: Statistics Brief', UITP – International Association of Public Transport, Brussels, Belgium, Statistics Brief, 2018. Accessed: Dec. 01, 2025. [Online]. Available: https://www.uitp.org/wp-content/uploads/sites/7/2025/04/Statistics-Brief-Metro-automation_final_web03.pdf

[5] European Union Agency for Railways (ERA), 'ERTMS/ATO Operational Principles', European Union Agency for Railways (ERA), Dec. 2023. Accessed: Jan. 08, 2026. [Online]. Available: https://www.era.europa.eu/sites/default/files/2024-12/index64_12e108_2_ato_operational_principles.pdf

[6] Office of Rail Regulation, 'Guidance on Tramways: Railway Safety Publication 2', Office of Rail Regulation, 2006. [Online]. Available: https://www.metroalliance.co.uk/wp-content/uploads/2017/05/BEE-C1-Guidance-on-Tramways-Railways-Safety-Publication-Part-2-Nov-2006.pdf

[7] 'Railway applications – The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS) – Part 1: Generic RAMS process', CENELEC – European Committee for Electrotechnical Standardization, EN 50126-1:2017, 2017. Accessed: Jan. 08, 2026. [Online]. Available: https://edstar.eda.europa.eu/Standards/Details/b649ab17-c207-4e3e-bf6a-16f5da4e1e0f

[8] M. Wild, J. S. Becker, A. Buinoschi-Tirpescu, and E. Mohlmann, ‘Towards Virtual Testing of Perception in the Railway Domain’, 2025 IEEE Int. Autom. Veh. Valid. Conf. IAVVC, doi: 10.1109/IAVVC61942.2025.11219512.
[9] L. Greiner-Fuchs, S. Schäfer, T. Hofmeier, and M. Cichon, ‘Database-supported methodical approach for the development of a toolchain for the evaluation of ATO functions using a scenario-based test methodology’, in Proceedings of the Fifth International Conference on Railway Technology: Research, Development and Maintenance, in Paper 13.4. Montpellier: Civil-Comp Press, 2022. doi: 10.4203/ccc.1.13.4.
[10] R. Tagiew et al., ‘Sensor system for development of perception systems for ATO’, Discov. Artif. Intell., vol. 3, no. 1, p. 22, Jun. 2023, doi: 10.1007/s44163-023-00066-4.
[11] ‘Vollautomatische Abdrücklokomotive - VAL - DB Cargo Lab’. Accessed: Dec. 11, 2023. [Online]. Available: https://www.dbcargo.com/rail-de-de/gruen-und-innovativ/dbcargo-lab/val
[12] S. Manivasagam, I. A. Bârsan, J. Wang, Z. Yang, and R. Urtasun, ‘Towards Zero Domain Gap: A Comprehensive Study of Realistic LiDAR Simulation for Autonomy Testing’, in 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France: IEEE, Oct. 2023, pp. 8238–8248. doi: 10.1109/ICCV51070.2023.00760.
[13] S. Muckenhuber, H. Holzer, and Z. Bockaj, ‘Automotive Lidar Modelling Approach Based on Material Properties and Lidar Capabilities’, Sensors, vol. 20, no. 03309, pp. 1–25, 2020, doi: 10.3390/s2003309.
[14] G. D’Amico et al., ‘TrainSim: A Railway Simulation Framework for LiDAR and Camera Dataset Generation’, IEEE Trans. Intell. Transp. Syst., vol. 24, no. 12, pp. 15006–15017, 2023, doi: 10.1109/TITS.2023.3297728.
[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, ‘CARLA: An Open Urban Driving Simulator’, in Proceedings of the 1st Annual Conference on Robot Learning, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., in Proceedings of Machine Learning Research, vol. 78. PMLR, Nov. 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html
[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, ‘AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles’, Jul. 18, 2017, arXiv: arXiv:1705.05065. doi: 10.48550/arXiv.1705.05065.
[17] T. Hofmeier and M. Cichon, ‘Scaled Model Development and Testing for Automatic Train Operation: Creating a Digital Twin’, SN Comput. Sci., 2025, doi: 10.1007/s42979-025-04529-6.
[18] L. Greiner-Fuchs, J. Feldhoff, S. Schäfer, T. Hofmeier, B. Schlereth-Groh, and M. Cichon, ‘Virtual Railway Scenario: Testing Automated on-Sight Train Operation’, in 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), 2025, pp. 1–6. doi: 10.1109/IAVVC61942.2025.11219626.
[19] T. Hofmeier and M. Cichon, Vir2Rail – Modular LiDAR Simulation Framework for ATO Research in Railway Environments. (Mar. 04, 2026). Zenodo. doi: 10.5281/zenodo.18769047.