State-of-the-Art Sensor Simulation for Autonomous Vehicles: Analyzing Techniques and Error Modeling in Virtual Environments

Yusuf Dalli[6048501], Lara de Geus[4965868], Quirine Japikse[5040930], and Hsuan-An Chu[5914647]

Technische Universiteit Delft
H.Chu-3, Lhdegeus, Q.C.Japikse, Y.D.Dalli@tudelft.nl

Abstract. With the rise of AVs, the automotive sector has undergone major transformations. AVs hold the potential to improve traffic safety and environmental sustainability, but they face challenges in testing and deployment. This is particularly due to the reliance on complex perception systems like cameras, LiDAR, and GPS. Simulation has emerged as a crucial tool for testing AVs, offering a safer, more cost-effective alternative to real-world trials, though challenges remain in replicating real-world conditions accurately. This paper reviews the state-of-the-art techniques for simulating AV sensor systems in virtual environments. The advantages and limitations of different sensor simulation methods are analyzed, focusing on their fidelity, computational requirements, and ability to replicate real-world conditions. In particular, this paper focuses on the simulation of errors in sensor models, to increase realism. While there is considerable research on simulating sensor errors in isolation, there is a notable gap in the high-fidelity simulation of multi-modal sensor fusion models that incorporate sensor noise to achieve more realistic outcomes.

Keywords: Autonomous vehicles · Simulation · Sensor models · Sensor noise

1 Introduction

In recent years, the automotive sector has undergone major transformations (Mallozzi et al., 2019). One of the most significant developments is the increasing prevalence of autonomous vehicles (AVs) entering the market. The transition to AVs is supported by strong justifications, particularly the potential for long-term improvements in traffic safety and environmental sustainability. AVs are expected to reduce traffic accidents and congestion, as well as lower emissions. However, there remains a pressing need for regulatory frameworks to guide the testing and deployment of these vehicles, especially as they rely on sophisticated perception systems, including cameras, LiDAR, RADAR, and ultrasonic sensors (Espineira et al., 2021).

Machine learning components, such as deep neural networks, have demonstrated considerable success in tasks like object detection, classification, and image segmentation. These models are also effective in predicting the behavior of other road users like human driven vehicles or pedestrians. Nevertheless, these models can be vulnerable to opposed examples, and there have been well-documented real-world failures of AVs where the breakdown of ML-based perception systems played a significant role. Consequently, it is critical to develop more robust techniques for testing and verifying AV systems (Fremont et al., 2020).

There are several approaches to testing autonomous vehicles, including public road testing, closed track testing, and simulation. Among these, simulation has become an essential tool for the design and validation of AVs with machine learning components due to its cost-effectiveness and enhanced safety. Advanced photorealistic simulators enable AV designers to simulate billions of miles of driving, addressing hard-to-replicate corner cases and diagnosing issues that occur during real-world testing, such as disengagements. However, key challenges still exist with this testing method. One major difficulty lies in integrating AVs with human-driven vehicles, particularly in situations involving conflicts over right-of-way. Additionally, concerns remain about the fidelity of simulations—how accurately do they reflect real-world conditions? And what is the comparative value of simulation versus physical testing in environments that include other vehicles, pedestrians, and road users? (Fremont et al., 2020).

Effective AV simulations require consideration of numerous different factors. An AV must recognize human-driven vehicles, interpret their intentions, predict their future movements, and take appropriate actions to avoid collisions. Currently, no existing algorithm or analytical model perfectly accomplishes this (Zhang et al., 2022). For example, human driving behavior models for AV testing need to be highly detailed at the microscopic or even sub-microscopic level, as AVs operate in environments with tight spatial and temporal constraints. Moreover, the environment in which an AV operates must be carefully simulated, since sensors data is entirely based on the virtual environment. Sensors detect objects, roads conditions, and pedestrians, and AV decisions are driven by this input. Therefore, aligning simulated input data with real-world conditions is essential for reliable testing.

Several studies have investigated methods for simulating both the environment and AV sensors. For instance, Elmquist and Negrut (2020) discuss techniques for simulating GPS, IMU, and LiDAR sensors. This research aims to compare and evaluate various approaches to simulating autonomous vehicle sensor systems, with a particular focus on modeling sensor noise to enhance realism and reliability. The objective is to identify state-of-the-art methods for simulating AV sensors and their associated noise in realistic virtual environments, ultimately discovering gaps in the current literature. The next section describes the research methods used in this review. This is followed by a review of the state-of-the-art techniques in simulating sensors and sensor noise. This section will also compare different packages and methods. The final part of the third section presents a literature review on time synchronization in different packages. Section 4 discusses the advantages and disadvantages of different packages explored in section 3. The final section concludes the review by summarizing the key findings from the literature and showing the main gaps identified, accompanied with recommendations for future research.

2 Research methods

A comprehensive search was conducted to obtain studies that align with the defined research objective. Scopus was the initial database used, with the search terms specified in Table 1, yielding respectively 9 and 8 papers. However, after reviewing the titles and abstracts, only three articles were found to meet the research objective. Subsequently, the backward and forward snowballing technique was applied to the obtained papers, leading to the papers listed in Table 1. Although both the search term and snowballing provided limited relevant results, making the search terms broader lead quickly to a tremendously amount of papers, which was also not desirable. Therefore, Google scholar was used to find more papers, utilizing the same search terms. To ensure the quality of the papers retrieved from Google Scholar, only papers with a high citation count were considered. The titles and abstract were carefully reviewed to determine whether each paper aligned with the research objective.

Furthermore, official documentation and white papers of simulation packages Chrono and CARLA are also investigated to dive deeper into their inner workings and possibilities.

3 State of Art

3.1 Simulating sensors of AVs

This subsection will dive deeper into the simulation of AV sensors and their associated sensor noise, including Cameras, GPS and LiDAR.

Cameras The paper by Elmquist and Negrut (2021) discusses two approaches to simulating camera models. The first is closed loop simulation, where the camera model is integrated into the autonomous system, which includes components like perception, control, and planning. The camera captures images of the virtual environment, which are processed to identify objects, enabling the control system to make decisions and send commands to the simulated vehicle. The system's decisions (such as adjusting speed or turning) feed back into the simulation, affecting the environment and the subsequent camera images. Closed-loop simulations, especially for testing things like obstacle avoidance and lane-keeping, require the camera model to operate in real-time or faster. This often necessitates a trade-off between accuracy and simulation speed, as high computational

Search engine	Search term	Papers yielded
Scopus	*Keywords*: "autonomous vehicles" AND simulation AND (LiDAR OR camera OR GPS OR sensor) AND noise	(Elmquist & Negrut, 2020)
	Article title, Abstract, Keywords: autonomous AND vehicles AND simulation AND (LiDAR OR camera OR GPS OR sensor) AND noise AND "sensor model"	(Durst & Goodin, 2012), (Espineira et al., 2021)
Google scholar	autonomous vehicles AND simulation AND (LiDAR OR camera OR GPS OR sensor) AND noise	(Manivasagam et al., 2020)
Snowballing		(Elmquist et al., 2021) (Elmquist & Negrut, 2021) (Elmquist et al., 2023) (Balaguer & Carpin, 2008) (Sobh et al., 2018) Sakic et al. (2020)
Official documentation & White Papers		(CARLA team, 2024) (Project Chrono, 2024) (Negrut et al., 2018)

Table 1. Literature search engines and their yields

demands can hinder performance. Gaming engines like Unreal Engine or Unity are commonly used for rendering in closed-loop simulations due to their great performance. However, they are not designed for simulating precise camera artifacts like sensor noise or optical distortions. To meet the real-time requirements of closed-loop simulation, simplified models introduce sensor noise, distortion, and delay, sacrificing some realism. Ensuring temporal consistency remains a challenge.

The second approach mentioned by Elmquist and Negrut (2021), synthetic data generation, enables the production of large amounts of automatically labeled training data in virtual environments for training machine learning models like object detectors. Errors are handled using complex models to ensure synthetic data can serve as an effective substitute for real-world data. Unlike closed-loop simulation, this approach allows for higher image fidelity and more sophisticated sensor models, though it may be more time-consuming. A significant challenge is ensuring synthetic images are realistic enough to be used for training, as simple physics-based models may not capture all the nuances of real-world environments.

Another paper in 2021 by Elmquist et al. (2021) use the Chrono::Sensor module to simulate sensors. The camera is modeled with real-time ray tracing, which accurately shows how light interacts with objects in the simulation. The model accounts for factors like reflectance and refractance. Chrono::Sensor can run sensor simulations in real time, producing accurate camera data at speeds comparable to real world sensor operations, ensuring high-fidelity simulations. However, this approach is computationally expensive and can significantly increase simulation time, particularly when running multiple sensors simultaneously.

Chrono::Sensor allows for customization of key parameters for the camera like image resolution, horizontal field of view, exposure time and lag Elmquist et al. (2021), enabling accurate representation of real-world cameras. For instance, lag is used to model temporal errors, simulating real-world processing delays. While many sensor simulations don't consider this, it is crucial for understanding how systems controls respond when sensor data is delayed. The simulation also includes motion blur and lens distortion, adding realism by mimicking how cameras capture moving objects and peripheral distortions. A noise model is used to account for intensity-dependent noise, where noise varies with light intensity. The camera model provides a framework for understanding and quantifying different types of sensor noise. However, it does not simulate the image signal processor, so the final processed image noise is not fully represented.

Elmquist et al. (2023) enhance the camera model to improve the realism of simulated images, addressing limitations in traditional rendering methods that simplify real-world camera operations. The model incorporates key physical aspects such as lens distortion, sensor noise, and lighting conditions. Different lens distortion models simulate light passing through lenses, and sensor noise is handled using additive white Gaussian noise, though this method doesn't fully capture real-world complexity. Basic models simulate demosaicing and color balancing in the image signal processing pipeline to better represent real-world lighting and color conditions, though they don't fully replicate real camera internals. Modifications to exposure levels and brightness help match real data, and more complex lighting setups with multiple light sources are introduced to create more realistic lighting conditions, providing a better representation of diverse environments. While modifications such as lens distortion and noise improve realism, they increase computational complexity. Also, the impact of these modifications is context-dependent, highlighting the need for a tailored approach depending on the specific use case.

GPS Simulating GPS in autonomous vehicles involves replicating complex interactions between the sensors, satellites, and the surrounding environment. This has usually been done by adding noise to the system since the simulation knows the exact position of the vehicle at all times. One paper by Elmquist and Negrut (2020) uses the Two-Line Element data, adding appropriate noise based on the number of visible satellites. The system tracks the current GPS satellite locations and determines which satellites are visible to the GPS sensor by casting rays from the sensor to the satellites. This accounts for environmental factors, such as buildings that may obstruct the view of some satellites. The noise distribution is then adjusted based on the number of visible satellites, reflecting the reduced accuracy with fewer satellites. While this basic noise model accounts for satellite visibility, it omits more complex real-world effects such as signal multi-pathing (where signals bounce off surfaces before reaching the receiver) and atmospheric distortions.

Both Elmquist and Negrut (2020) and Elmquist et al. (2021) use the Chrono::Sensor module to simulate GPS sensors. The sensor can be placed at any location on the simulated autonomous vehicle, providing flexibility in modeling different autonomous vehicle configurations. The GPS sensor is parameterized by settings such as update frequency, data collection time, and lag, which are empirical and require calibration or noise estimation to achieve realistic results. Additionally, the collection window remains a key parameter in the GPS model, even though the sensor doesn't have an exposure time like a camera. This parameter is used to replicate real-world behavior, where GPS sensors do not provide instantaneous position data but instead collect signals from multiple satellites over a small time frame. The lag setting accounts for the delay in receiving GPS data, reflecting the time it takes for signals to be processed and made available. To simulate sensor noise, the GPS model supports the addition of a basic parameterized Gaussian drift model. This model accounts for drift, a common phenomenon in real-world GPS systems where measured values gradually deviate over time due to inaccuracies. However, real-world GPS noise is often more complex than a simple Gaussian distribution. Incorporating factors like the number of satellites visible to the receiver would enhance the noise model.

Similarly, a paper by Balaguer and Carpin (2008) discusses a simulated GPS sensor for outdoor robotics, emphasizing a tradeoff between speed and accuracy. Their model also relies on real satellite data and includes noise models based on the number of satellites visible. They further refined the simulation by eliminating satellites below a certain elevation and those obstructed by buildings. However, the model lacks more complex signal phenomena such as multipath errors and dilution of precision.

However, Durst and Goodin (2012) demonstrated that simulating multipath errors and dilution of precision effects is feasible, though it requires significant computational resources. The paper introduces a high-fidelity GPS sensor model that calculates the receiver's position based on distances to GPS satellites, incorporating material reflectance. The model accounts for line-of-sight obstructions or multipath reflections, which occur when GPS signals are reflected off surfaces like buildings or foliage before reaching the receiver, causing inaccuracies in position calculation. This process is shown in Figure 1.

The model applies error functions to simulate atmospheric delays and includes satellite ephemeris errors, which introduce additional range errors due to inaccuracies in the satellite's reported position. By incorporating these factors, the model deterministically recreates dilution of precision and multipath effects, providing a more realistic simulation of GPS errors that significantly impact autonomous navigation. While computationally intensive, this method significantly

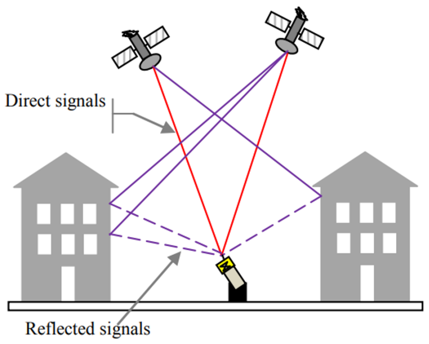

Fig. 1. Principle of multipath signal error, direct signal (red) and reflected interference (purple) (Kumar et al., 2013)

improves the realism of GPS signal degradation, particular beneficial in complex environments like urban canyons or dense foliage. Such detailed models bridge the gap between simulated and real-world performance, supporting the development of more robust navigation algorithms for autonomous vehicles. Durst and Goodin (2012)

LiDAR The Chrono::Sensor module also includes a LiDAR sensor (Elmquist et al., 2021). Multiple LiDAR rays are emitted simultaneously producing highly realistic point clouds, closely replicating real-world LiDAR output. The LiDAR model supports customization of key sensor parameters like field of view, and maximum range, allowing the simulation to accurately replicate specific LiDAR systems. Similar to the camera model, the LiDAR model accounts for temporal errors by incorporating lag and collection time settings. Additionally, sensor noise is simulated through models that introduce measurement errors. Rather than applying noise directly to the point cloud data. Noise is introduced into the raw measurements, which are then converted into noisy point cloud data. This method ensures that the noise is more realistic and mirrors real-world sensor performance. The noise levels can be adjusted based on calibration data from real-world sensors or from specifications found in LiDAR sensor datasheets.

Furthermore, the LiDAR implementation supports both beam discretization and beam divergence (Elmquist et al., 2021). Beam discretization addresses the challenge of a laser beam spreading out as it travels, which reduces its ability to capture fine details. By discretizing the beam, the system improves data accuracy. Beam divergence allows the laser to hit multiple objects at varying distances, resulting in multiple returns. This increases the number of points in the point cloud, providing richer and more detailed information about the environment.

Manivasagam et al. (2020) introduce a novel simulation system called LiDARsim that combines both physics-based and machine learning-based approaches to generate realistic LiDAR point clouds. The system works by first building a large catalog of 3D static maps (city environments) and dynamic objects (e.g., vehicles, pedestrians) from real-world data collected by a self-driving fleet equipped with LiDAR sensors. The simulator then places a virtual autonomous vehicle and dynamic objects in the 3D environment and uses ray-casting to simulate the LiDAR scanning process. LiDARsim accounts for real-world effects like the rolling shutter, adjusting for the vehicle's movement during scanning to prevent distortions and ensure accuracy. To improve the realism of the physics-based LiDAR point-cloud, sensor errors are simulated through the introduction of 'raydrop' and sensor noise. Raydrop occurs when certain LiDAR rays do not return because the signal is too weak to be detected by the receiver. LiDARsim simulates this phenomenon using a deep neural network that predicts the probability of raydrops based on factors such as material reflectivity, distance from the sensor and incidence angle. Additionally, sensor noise is simulated by sampling from a probability mask, which assigns the likelihood of noise affecting certain points in the LiDAR data. This approach creates a more realistic LiDAR point cloud.

In addition to generating point clouds, LiDARsim can be used in closed-loop systems, where the sensor data directly influence the vehicle's decision-making and planning algorithms (Manivasagam et al., 2020). This is crucial for evaluating the full autonomy stack, including the vehicle's ability to respond to dynamic objects or unexpected events.

Espineira et al. (2021) use the WMG 3xD simulator, a high fidelity driving simulator that integrates Driver-in-the-Loop, real-time sensor simulation and highly accurate vehicle dynamics. The simulator leverages the Unreal Gaming Engine to create the virtual environment and simulate how sensors interact with this environment. Compared to other software, Unreal Engine stands out for its flexibility and fidelity, making it particularly suitable for developing custom features, such as a LiDAR model. When simulating LiDAR rays, the intensity of the returned rays is calculated based on factors like distance surface reflectivity, and only points with sufficiently strong signals are added to the 3D point cloud, resulting in a highly realistic point cloud that closely matches real-world data. This process is similar to the 'raydrop' simulation described by Manivasagam et al. (2020). To further enhance realism, the sensor model is integrated with a noise model, which introduces real-world imperfections into the simulated data. In particular, a probabilistic rain model is used to simulate the impact of raindrops on the LiDAR signals. The model calculates the likelihood of raindrops reflecting the laser beams, potentially causing false positives (detecting raindrops as objects) or false negatives (failing to detect real objects due to rain attenuation). This article sets the foundation for incorporating additional noise sources into sensor models.

Multi-sensor fusion Sobh et al. (2018) develops and evaluates an end-to-end multi-modal fusion architecture for autonomous driving, integrating LiDAR and camera data within the CARLA simulator. Cameras provide detailed visual information but struggle with depth perception and poor lighting conditions, while LiDAR excels at spatial mapping and distance measurements but lacks visual detail. By fusing these two sensors, the system overcomes the limitations of each, improving object detection, lane recognition, and obstacle avoidance. To address the issue of raw camera images being sensitive to lighting variations, Sobh et al. (2018) use data augmentation to make the model more robust to lighting changes, and semantic segmentation to convert raw camera images into a more abstract form, reducing sensitivity to lighting. While data augmentation improves the camera's ability to handle varying lighting conditions, and abstraction simplifies the data, the system's capacity to manage real-world sensor noise and unpredictable conditions may still be limited.

Sobh et al. (2018) also create an abstract view of the environment for modeling LiDAR. The 3D point cloud data is transformed into a 2D image, significantly reducing computational costs while retaining crucial information about object locations and distances. There is no specific focus on noise modeling, but the abstractions minimize the impact of noise and improve generalization.

The system developed by Sobh et al. (2018) uses a late-fusion approach, where data from the camera and LiDAR are processed independently using separate algorithms to extract features such as object location and shape. After each sensor has processed its data, the feature maps are combined later in the pipeline, creating a more comprehensive understanding of the environment. This fusion enhances the system's ability to detect objects, navigate through the environment, and improve driving behavior. Additionally, if one sensor performs poorly due to environmental factors (e.g., the camera struggling in low-light conditions), the other sensor can compensate, improving the overall reliability of the system.

For late fusion to be effective, the camera and LiDAR systems must be properly calibrated so that their data aligns accurately in space and time Sobh et al. (2018). The accuracy of the sensor fusion model is highly dependent on this calibration, and misalignment can lead to inaccuracies in object detection and pose estimation. While abstraction techniques are used to reduce complexity, processing large amounts of data in real time still requires significant computational power.

Sakic et al. (2020) also employ late fusion of LiDAR and camera sensors simulated in CARLA, using a calibration process to align the 3D LiDAR point cloud data with the 2D camera data. Sensor errors for the camera and LiDAR are not explicitly modeled in detail. Moreover, the system relies heavily on the camera's object detection accuracy. If the camera fails to detect an object, the LiDAR data is not independently used for detection, which implies that errors or limitations in the camera's performance could affect the overall accuracy.

3.2 Combining AV models and environment simulation

An autonomous vehicle is a complex system that integrates multiple subsystems, including perception, trajectory planning, and vehicle dynamics. After modeling the individual components like LiDAR, GPS, and cameras, it is essential to test their performance on a simulated AV platform. There are two primary approaches to simulating an integrated AV, each offering a trade-off between physical and communication resolution versus environmental fidelity. Simulations can either focus on accurately reproducing driving dynamics and agent communication, or provide a lower-resolution but more complex environment, incorporating factors like traffic and dense foliage. This section discusses both simulator packages, with the previously mentioned Synchrono (Negrut et al., 2018) and CARLA (Dosovitskiy et al., 2017) as examples.

Synchrono (Negrut et al., 2018) is an open-source framework built on the physics-based engine Chrono (Tasora et al., 2016), designed for simulating collaborating robots. Chrono provides various simulation modules, including rigid body dynamics for vehicles and a sensor module that generates synthetic data from RGB cameras, LiDAR, radar, and GPS. This high-fidelity simulation capability enables detailed modeling of driving dynamics and sensor integration in a variety of scenarios, including off-road conditions or underwater environments. Synchrono excels in scenarios requiring precise physics and communication modeling. However, the complexity of its physics-based simulation can limit the creation of visually rich environments. Additionally, the computational intensity of its models makes it less suitable for hardware-in-the-loop or human-in-the-loop simulations. Another challenge lies in synchronization issues, especially when multiple agents have different physics simulation resolutions.

CARLA (Dosovitskiy et al., 2017) is an open-source simulator built on the Unreal Engine, known for its ability to create visually detailed environments with various assets. This game-engine foundation makes CARLA ideal for end-to-end testing of AV functionalities such as perception, mapping, localization, and vehicle control (Kaur et al., 2021). Its distributed client-server architecture enhances flexibility: the client modules manage actor logic and world conditions, while the server handles the core simulation tasks, including sensor rendering, physics computations, and world-state updates. This design allows for easy integration of custom scripts and modules. However, the distributed nature of CARLA's architecture can introduce synchronization challenges, particularly when coordinating time and events across different client modules.

Time and Synchronization within Synchronos Due to Chrono's physics-based nature, the simulation operates at an extremely high update frequency to ensure the accuracy of the physics equations. In contrast, sensors typically have lower update rates. To accommodate this difference, Chrono manages sensor data processing on a separate thread, as illustrated in Figure **??**.

However, this approach can lead to slowdowns when the sensor thread becomes computationally expensive, causing the main thread to wait for the sensor data. This issue becomes more pronounced in simulations involving multiple agents, as sensor-heavy agents may advance in larger time steps, leading to a time and information gap between them. The Synchrono framework addresses this problem by introducing "simulation heartbeats," which act as synchroniza-

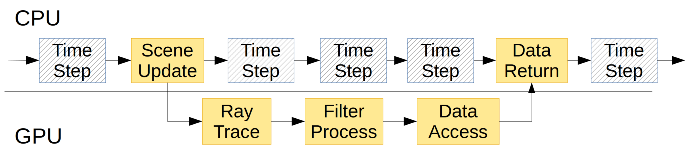

Fig. 2. Chrono's main and sensor threads (Project Chrono, 2024)

tion points and state caches. The state cache periodically checks the simulation times of external agents. If an external agent's simulation time lags behind by more than two heartbeat intervals, the local simulation pauses until it receives an updated message, as shown in Figure 3. This mechanism prevents one agent from advancing too far ahead in time, ensuring consistent information exchange between agents through cached data at each heartbeat.

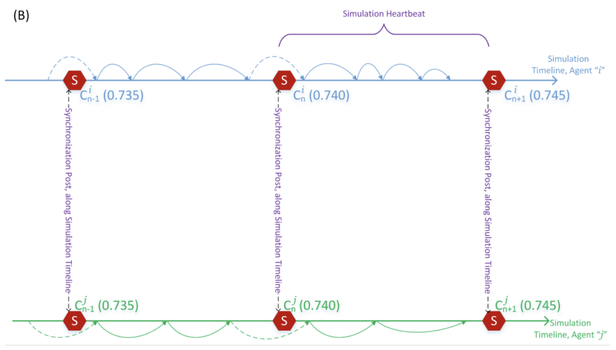

Fig. 3. Chrono's simulation heartbeat concept, the simulation of Agent "i" will get blocked if there are no stored state of Agent "j" within two heartbeats (Negrut et al., 2018)

Time and Synchronization within CARLA Due to CARLA's game engine based nature, the simulation runs in real-time by default using a variable time step. In this mode, the server operates as fast as possible, transmitting and receiving information to and from client scripts in real-time. While this approach provides fast and efficient visualization, it poses challenges for the reproducibility of the simulation. When rerunning the same simulation with identical seeds, floating-point arithmetic errors may occur due to race conditions, or the variable time step may differ based on the server's hardware configuration. This mode is ideal for scenarios that prioritize fast visualization or where reproducibility is not a concern, such as in human-in-the-loop or hardware-in-the-loop testing. For simulations requiring precision and synchronization, a more controlled method is necessary. One way is to run the simulation with a fixed time step, decoupling the simulation from real-time execution. This approach introduces a delay in communication at each tick, allowing for more consistent event tracking and easier reproducibility throughout the simulation. However, even in fixed time step mode, client scripts remain unsynchronized, meaning race conditions between client scripts can still occur. The asynchronous mode structure of CARLA is shown in Figure 4

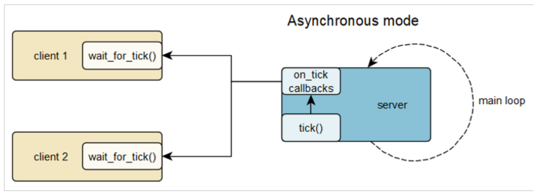

Fig. 4. CARLA asynchronous mode (López & Pina, 2020)

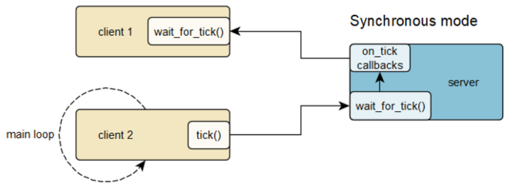

Fig. 5. CARLA synchronous mode (López & Pina, 2020)

The later developed Synchronous mode in CARLA gives clients scripts complete control over the advancement of time ticks (Figure 5). This mode is especially suited for scenarios where client scripts run slowly or when precise synchronization between different elements is required. For instance, modeled sensors may need to wait for specific world snapshots, or driving algorithms might depend on the completion of an image processing pipeline. In such cases, synchronous mode, combined with a fixed time step, is ideal for ensuring accurate and coordinated execution in complex simulations. While it is technically possible to run synchronous mode with a variable time step, it is generally not recommended. Larger time steps caused by slower computations can lead to the breakdown of the physics engine, compromising the integrity of the simulation (CARLA team, 2024). Therefore, using synchronous mode with a fixed time step is the most reliable approach, particularly for tasks like sensor modeling and development where precision is crucial.

4 Discussion

In this section the literature from the previous chapter is discussed. There are several sources found explaining different methods and packages for simulating AV sensors and their associated sensor noise. Table 2 summarizes the key findings, highlighting the advantages and disadvantages of each method. As can be seen, most sensor models are tested in Chrono::Sensor (Synchrono) and CARLA.

Camera models are simulated using Unreal engine and Chrono::Sensor, which both employ different approaches. Unreal engine uses closed loop simulation is used, while Chrono::Sensor uses synthetic data generation. Both methods make the trade-off between fidelity versus performance. Closed loop simulation allows real time testing, sacrificing the fidelity, where Chrono::Sensor creates realistic imaging which are computationally expensive. These trade-offs also affect error handling: Unreal Engine uses simpler models, while Chrono::Sensor focuses more on sensor noise modeling.

Most GPS sensor models prioritize computational efficiency by using simple noise models, ignoring the noise from satellite reflections and dilution of precision. However, Durst and Goodin (2012) addressed these specific errors, though at a high computational cost.

LiDAR sensor simulation is covered across three different modules. Chrono::Sensor shares similar trade-offs as with the camera model. The LiDARsim module, developed by Manivasagam et al. (2020), is very cost-efficient and uses real-world data, but is less accurate compared to Chrono::Sensor. Espineira et al. (2021) use the Unreal engine, where only rain is considered as noise, ignoring other circumstances which may cause errors in the sensor output.

Since each sensor has its strengths and limitations, sensor fusion is an effective strategy to overcome individual weaknesses. Both Sobh et al. (2018) and (Sakic et al., 2020) implement late fusion of camera and LiDAR data. However, neither explicitly models sensor noise, reducing the realism of the simulation.

Two packages were reviewed for time synchronization: Synchrono and CARLA. Synchrono offers better vehicle dynamics and sensor simulation, while CARLA provides more expandability and faster speeds. No simulator currently achieves the best of both worlds. Although Chrono's physics component can be integrated into CARLA, this option only supports vehicle physics without collision or sensor integration. Therefore, further development is needed to create a high-fidelity simulation that balances both environmental complexity and physics accuracy.

Table 2: Overview of sensor simulation

Sensor type	**Reference**	**Simulation module**	**Pros**	**Cons**	**Error handling**

Continued on the next page

Table 2: Overview of sensor simulation

Sensor type	**Reference**	**Simulation module**	**Pros**	**Cons**	**Error handling**
Camera	(Elmquist & Negrut, 2021)	Unreal engine	Allows real-time testing	Sacrificing fidelity in order to maintain high performance	Errors are handled with simplified models, may not be as realistic as in slower models
	(Elmquist et al., 2021)	Chrono::Sensor	High fidelity and realism, customizable settings (flexible)	Computationally expensive, no multi-sensor fusion, dependency on high-quality environments, no simulation of the image signal processor noise	Intensity-dependent noise model, motion blur and lens distortion, temporal errors through lag
	(Elmquist et al., 2023)	Chrono::Sensor	Improved realism, customizable settings (felxible)	Computationally expensive, no multi-sensor fusion, dependency on high-quality environments, limited sensor noise modeling	lens distortion, simple sensor noise, image signal processor noise (demosaicing, exposure, color balance)
GPS	(Elmquist & Negrut, 2020)	Chrono::Sensor	Flexible sensor placement, adjustable parameters, simple GPS model, satellite obstruction, efficient	No multipath and Dilution of Precision simulation	Noise based on visible satellites
	(Elmquist et al., 2021)	Chrono::Sensor	Flexible sensor placement, adjustable parameters	No multipath and Dilution of Precision simulation, no noise based on visible satellites	Temporal errors through lag and collection window, sensor noise through drift model
	(Balaguer & Carpin, 2008)	USARSim	Satellite obstruction, computationally efficient	No multipath and Dilution of Precision simulation	

Continued on the next page

Table 2: Overview of sensor simulation

Sensor type	**Reference**	**Simulation module**	**Pros**	**Cons**	**Error handling**
	(Durst & Goodin, 2012)	Vehicle Autonomous Navigation Environment Computational Test Bed	Accurate multipath and dilution of precision simulation	Computationally intensive	Multipath, dilution of precision, atmospheric delays
LiDAR	(Elmquist et al., 2021)	Chrono::Sensor	High fidelity and realism, parallel computing, customizable settings (flexible)	Computationally expensive, no multi-sensor fusion, dependency on high-quality environments	Noise applied to range, angular and intensity measurements, temporal errors trough lag & collection time
	(Manivasagam et al., 2020)	LiDARsim	Cost-efficient, usage of real-world data, high fidelity	Computationally expensive, no multi-sensor fusion	Raydrop, sensor noise by probability mask
	(Espineira et al., 2021)	Unreal Engine within the WMG 3xD simulator	High fidelity and realism, noise integration	Only rain is included as noise, no multi-sensor fusion	Attenuation equation and noise model
Multi-sensor fusion	(Sobh et al., 2018)	CARLA	Multi-sensor fusion of LiDAR and Camera	Computationally expensive, highly dependent on calibration, no (limited) sensor noise modeling	Abstraction of data (LiDAR), data augmentation and abstraction (Camera)
	(Sakic et al., 2020)	CARLA	Multi-sensor fusion of LiDAR and Camera	Computationally expensive, high dependence on camera, no sensor noise modeling	no specific noise modeling

5 Conclusion

This literature review explored current methods for simulating AV sensors, including cameras, LiDAR, and GPS, and how errors are modeled within these simulations. While significant progress has been made in simulating individual sensors, a key gap remains: the lack of comprehensive integration of error simulation across all sensors in an AV.

Most studies focus on either simulating the sensors themselves or modeling the errors for a specific sensor type. For instance, high-fidelity simulations might include detailed noise models for cameras or LiDAR under certain conditions like rain. However, these efforts are often isolated, and there is a lack of integration in simulating the sensors simultaneously.

This gap means current simulation platforms do not fully capture the integration between sensors. For example, a camera cannot detect range accurately even when multiple cameras are used, but it can identify objects. On the other hand, LiDAR is perfect for range detection but can't differentiate a statue from a pedestrian. Multi-sensor integration is essential for AVs to accurately understand their environment, and without integrated error simulation, it's hard to assess the true robustness of these simulations under realistic conditions.

Future research should aim to develop simulation frameworks that combine realistic error models across all sensors and evaluate their collective effect on the AVs' performance. By addressing this gap, the validation and testing of AVs can be enhanced, ultimately contributing to the development of safer, more reliable AV technologies.

References

Balaguer, B., & Carpin, S. (2008). Where am I? A simulated GPS sensor for outdoor robotic applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5325 LNAI*, 222 – 233. doi: https://doi.org/10.1007/978-3-540-89076-8_23

CARLA team. (2024). *Synchrony and time-step — CARLA simulator.* https://carla.readthedocs.io/en/latest/adv_synchrony_timestep/. (Accessed 06-10-2024)

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st annual conference on robot learning* (pp. 1–16). doi: https://doi.org/10.48550/arXiv.1711.03938

Durst, P., & Goodin, C. (2012, 01). High fidelity modelling and simulation of inertial sensors commonly used by autonomous mobile robots. *World Journal of Modelling and Simulation*, *8*, 172-184.

Elmquist, A., & Negrut, D. (2020, December). Methods and models for simulating autonomous vehicle sensors. *IEEE Transactions on Intelligent Vehicles*, *5*(4), 684–692. doi: https://doi.org/10.1109/tiv.2020.3003524

Elmquist, A., & Negrut, D. (2021, November). Modeling cameras for autonomous vehicle and robot simulation: An overview. *IEEE Sensors Journal*, *21*(22), 25547–25560. doi: https://doi.org/10.1109/jsen.2021.3118952

Elmquist, A., Serban, R., & Negrut, D. (2021, February). A sensor simulation framework for training and testing robots and autonomous vehicles. *Journal of Autonomous Vehicles and Systems*, *1*(2). doi: https://doi.org/10.1115/1.4050080

Elmquist, A., Serban, R., & Negrut, D. (2023, August). Synthetic image generation for robot simulation: Quantifying the impact of model modifications on perception. *IEEE Sensors Journal*, *23*(16), 18304–18315. doi: https://doi.org/10.1109/jsen.2023.3288488

Espineira, J. P., Robinson, J., Groenewald, J., Chan, P. H., & Donzella, V. (2021, April). Realistic lidar with noise model for real-time testing of automated vehicles in a virtual environment. *IEEE Sensors Journal*, *21*(8), 9919–9926. doi: https://doi.org/10.1109/jsen.2021.3059310

Fremont, D. J., Kim, E., Pant, Y. V., Seshia, S. A., Acharya, A., Bruso, X., . . . Mehta, S. (2020, September). Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC).* IEEE. doi: https://doi.org/10.1109/itsc45102.2020.9294368

Kaur, P., Taghavi, S., Tian, Z., & Shi, W. (2021). A survey on simulators for testing self-driving cars. In *2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD)* (pp. 62–70). doi: https://doi.org/10.1109/MetroCAD51599.2021.00018

Kumar, G., Rao, G., & Kumar, M. (2013, 01). Gps signal short-term propagation characteristics modeling in urban areas for precise navigation applications. *Positioning*, *04*, 192-199. doi: https://doi.org/10.4236/pos.2013.42019

López, A., & Pina, B. (2020, Jun). *Core implementations: Synchrony, snapshots and landmarks.* CARLA Team.

Mallozzi, P., Pelliccione, P., Knauss, A., Berger, C., & Mohammadiha, N. (2019). Autonomous vehicles: State of the art, future trends, and challenges. In *Automotive systems and software engineering* (p. 347–367). Springer International Publishing. doi: https://doi.org/10.1007/978-3-030-12157-0_16

Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., ... Urtasun, R. (2020). LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 11164-11173). doi: https://doi.org/10.1109/CVPR42600.2020.01118

Negrut, D., Serban, R., Elmquist, A., & Hatch, D. (2018). Synchrono: An open-source framework for physics-based simulation of collaborating robots. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (pp. 101–107). doi: https://doi.org/10.1109/SIMPAR.2018.8376278

Project Chrono. (2024). *Chrono::Sensor Reference Manual.* https://api.projectchrono.org/9.0.0/manual_sensor.html. (Accessed 15-10-2024)

Sakic, N., Krunic, M., Stevic, S., & Dragojevic, M. (2020). Camera-LIDAR Object Detection and Distance Estimation with Application in Collision Avoidance System. In *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin* (Vol. 2020-November). doi: https://doi.org/10.1109/ICCE-Berlin50680.2020.9352201

Sobh, I., Amin, L., Abdelkarim, S., Elmadawy, K., Saeed, M., Abdeltawab, O., ... El Sallab, A. (2018). End-to-end multi-modal sensors fusion system for urban automated driving. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018).*

Tasora, A., Serban, R., Mazhar, H., Pazouki, A., Melanz, D., Fleischmann, J., ... Negrut, D. (2016). Chrono: An open source multi-physics dynamics engine. In T. Kozubek (Ed.), *Chrono: An open source multi-physics dynamics engine* (pp. 19–49). Springer. doi: https://doi.org/10.1007/978-3-319-40361-8_2

Zhang, K., Chang, C., Zhong, W., Li, S., Li, Z., & Li, L. (2022, November). A systematic solution of human driving behavior modeling and simulation for automated vehicle studies. *IEEE Transactions on Intelligent Transportation Systems*, *23*(11), 21944–21958. doi: https://doi.org/10.1109/tits.2022.3170329