PRISSMA Project
Plateforme de Recherche et d'Investissement pour la Sûreté
et la Sécurité de la Mobilité Autonome
04/2021 - 04/2024

[DELIVERABLE 2.5] DEFINITION OF INTERFACES AND SIMULATION ENVIRONMENT: STRENGTH, WEAKNESS, ADVANTAGE, RISK, LIMIT, CONSTRAINTS

[LIVRABLE 2.5] DÉFINITION DES INTERFACES ET DE L'ENVIRONNEMENT DE SIMULATION : FORCE, FAIBLESSE, AVANTAGE, RISQUE, LIMITE, CONTRAINTES

Main authors : D. Gruyer (UGE), Cedric Gava (SPHEREA), R. Régnier (LNE), F. Bernardin (Cerema), Sio-Song Ieng (UGE), G. Mouchard (CEA), G. Durand (AVSimulation), C. Chaves (AVSimulation), S. Prabakaran (AVSimulation), A. Ridzuan (AVSimulation), K. Quintero (IRT SystemX), Keilatt Andriantavison (Valeo), Pierre-Antoine Laharotte (UGE), Wei Xu (UGE), Laurent Benet (ATC France) ...

Keywords: Simulation tools, sensor models, traffic simulation, graphical and physic engines, simulation environments, interfaces

Abstract. This document propose a generic definition with requirements about the needed models, tools, and interfaces for the building of a simulation environment dedicated to the evaluation and validation of the AI-based systems included in Automated Driving Systems.

Résumé. Ce document propose une définition générique avec des exigences concernant les modèles, les outils et les interfaces nécessaires à la construction d'un environnement de simulation dédié à l'évaluation et à la validation des systèmes basés sur l'IA inclus dans les systèmes de conduite automatisée.

Contents

List of Figures

List of Tables

1 Introduction (ALL)

This deliverable addresses the presentation of the work carried out in tasks T2.2 and T2.3.

The first section concerns the definition of the evaluation environment through simulation based on the objectives sought. This task identifies the critical subsystems to be implemented to meet the evaluation and validation needs. This includes identifying different platforms and simulation systems (sensors, mobile dynamics, communication means, traffic generation, simulation of degraded conditions, etc.) for evaluation and approval purposes (design-related aspects are addressed in pillar 1 of the grand challenge). It also involves defining the relevance domains of the simulation, the levels of realism and representativeness (ranging from simple to physico-realistic), the events to be taken into account, and more.

This deliverable also addresses the architecture to be implemented and its properties to adapt to the use of formal modeling in the evaluation. This level of formal modelling takes into account the behaviour of these subsystems and the formalisation of their functional properties (e.g., absence of blocking, correct collision detection and prediction). Efforts are also focused on the integration and interoperability aspects of different platforms and simulation tools, allowing for the approval of applications and services that integrate AI-based systems, system-of-systems, communication, and cyber-security systems (addressed in pillar 1). The objective of this deliverable (L2.5) is to propose requirement definitions and evaluation environments to build interoperable simulation platforms for critical systems. These platforms must also ensure a high level of representativeness to provide an ”acceptable” proof value to simulation as evaluation tools (this aspect is addressed in WP2 and T2.5, corresponding to deliverable L2.7).

A second section addresses the critical issue of interconnection and communication between platforms, models, applications, and resources used for virtual testing in the field of automated mobility. Identifying the needs and their specifications allows for an evaluation of the relevance of proposed solutions. An overview of ongoing initiatives and standardisation’s is provided (ASAM: standardisation work on simulation and scenarios, FMI: vehicle dynamics, OSI: sensor standardisation, etc.). Defining the interfaces allows for their implementation with an evaluation and quantification of the quality of the software mechanisms used. The objective of this section is to propose definitions and software libraries that guarantee interoperability, generality, and scalability of distributed solutions proposed in POCs. Additionally, the limitations of such interfaces are identified, as well as the underlying implementation constraints.

2 Definition of the framework and the interfaces for the building of simulation environment

2.1 Overall issues and objectives

This section is dedicated to the definition of the verification and evaluation protocol for the models and tools used be the simulation platform.

2.1.1 Definition of the evaluation and certification framework

2.1.1.1 Actors

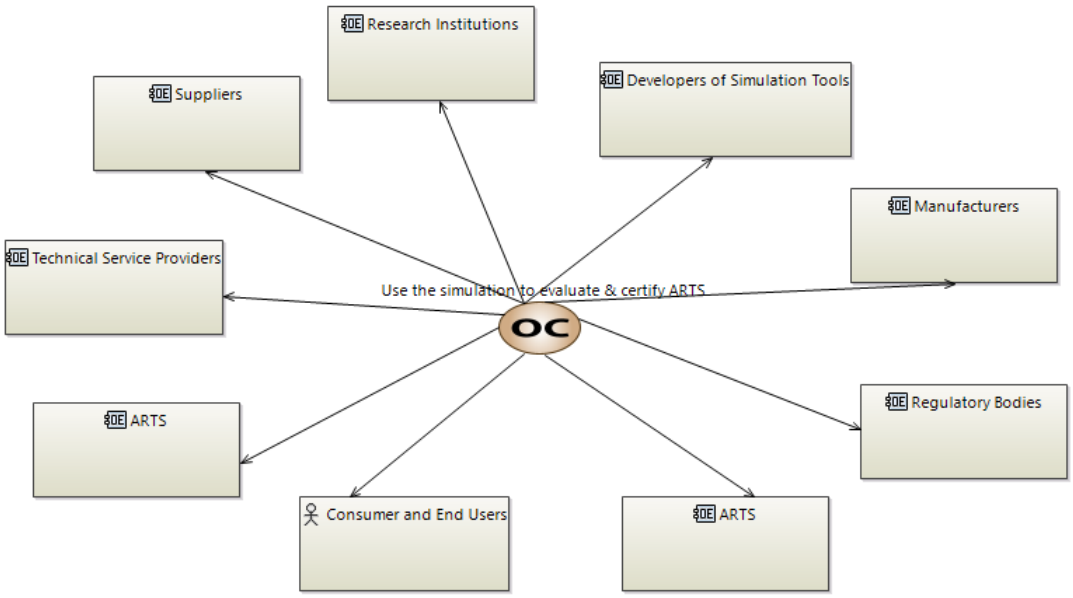


Figure 1: Operational context of simulation usage for ARTS evaluation and certification

In the context of utilising simulation in the evaluation and certification process for autonomous vehicles, various stakeholders play integral roles. Each actor contributes distinct expertise and resources to ensure the efficacy, safety, and regulatory compliance of autonomous systems. These key actors include:

- Regulatory Bodies: Regulatory agencies oversee the development and deployment of autonomous vehicles, setting forth standards and regulations to ensure safety and adherence to legal requirements. They play a crucial role in establishing guidelines for testing procedures and certification criteria.
- Manufacturers: Original equipment manufacturers (OEMs) are responsible for designing, developing, and testing autonomous vehicle systems. They employ simulation tools and techniques throughout the development life-cycle to assess system performance, refine algorithms, and validate functionalities.

- Suppliers: Component suppliers provide essential hardware and software components for autonomous vehicles, including sensors, processors, and control systems. They collaborate with manufacturers to integrate components seamlessly and ensure compatibility with simulation environments.
- Technical Service Providers: Technical service providers offer expertise in simulation software, modelling methodologies, and testing procedures. They assist manufacturers in conducting comprehensive simulations, analysing results, and validating system behaviour to meet regulatory requirements.
- Research Institutions: Academic institutions and research organisations contribute to advancing simulation techniques and validating autonomous vehicle systems. They conduct studies, develop simulation models, and provide insights into emerging technologies and best practices.
- Developers of Simulation Tools: Companies and organisations that develop simulation software and tools provide essential resources for evaluating and certifying autonomous vehicles. Their expertise in creating realistic and scalable simulation environments enables manufacturers and service providers to conduct extensive testing and validation.
- Consumers and End Users: Consumers and end users interact with autonomous vehicles and rely on them for transportation needs. Their acceptance and trust in autonomous technology are influenced by rigorous testing and certification processes that ensure safety and reliability.
- Public Authorities: Local and national government agencies oversee the integration of autonomous vehicles into existing transportation infrastructure. They collaborate with regulatory bodies and industry stakeholders to address regulatory gaps, manage public concerns, and ensure compliance with safety standards.

Understanding the roles and interactions of these stakeholders is essential for leveraging simulation effectively in the evaluation and certification of autonomous vehicles. By fostering collaboration and knowledge-sharing among diverse actors, the industry can establish robust frameworks that prioritise safety, innovation, and regulatory compliance throughout the certification process.

2.1.1.2 Tools

Simulation tools form the backbone of the evaluation and certification process for autonomous vehicles, providing a virtual environment where system behaviours can be tested and validated under diverse conditions. These tools encompass various software and hardware solutions tailored to specific simulation needs. Key types of simulation tools include:

- Software Simulation Platforms: Software-based simulation platforms offer comprehensive environments for modelling and simulating the behaviour of autonomous vehicles and their interactions with the surrounding environment. These platforms provide tools for creating virtual worlds, defining vehicle dynamics, implementing control algorithms, and analysing simulation results.

- Hardware-in-the-Loop (HIL) Systems: HIL systems combine physical components, such as vehicle hardware and sensors, with simulation software to emulate real-world operating conditions. By interfacing with actual hardware, HIL systems enable thorough testing of control algorithms, sensor fusion, and actuator response in a controlled environment.
- Sensor Simulation Software: Sensor simulation software replicates the behaviour of various sensors used in autonomous vehicles, including cameras, LIDARs, radars, and ultrasonic sensors. These tools generate realistic sensor data, allowing developers to evaluate perception algorithms, object detection, and environmental awareness in simulated scenarios.
- Scenario Generation Tools: Scenario generation tools enable the creation of diverse and complex scenarios to test the capabilities of autonomous vehicles. These tools facilitate the generation of realistic traffic patterns, road conditions, weather effects, and unexpected events, allowing for comprehensive validation of system robustness and adaptability.
- Modelling and Visualisation Software: Modelling and visualisation software aids in the creation of detailed virtual environments, vehicle models, and sensor configurations. These tools enhance the realism of simulations, enabling developers to accurately represent real-world scenarios and visualise system behaviour for analysis and debugging.
- Automatic test sequencer: Such kind of tools operate in coordination with any of the previously mentioned tools to assert properties and issues sanctions against the validation of those properties.

Leveraging these simulation tools, stakeholders can conduct extensive testing and validation of autonomous vehicle systems, identifying potential issues, optimising performance, and ensuring compliance with regulatory standards. By integrating simulation into the evaluation and certification process, the industry can accelerate innovation, enhance safety, and expedite the deployment of autonomous vehicles.

2.1.1.3 Temporality/Sequencing

In the evaluation and certification process for autonomous vehicles using simulation, temporal organisation is pivotal. This process unfolds in distinct phases:

- Development and Iterative Testing: Initially, simulation aids in the iterative refinement of autonomous vehicle systems during development. Manufacturers use simulation tools to validate algorithms, assess performance across varied conditions, and pinpoint areas for enhancement.
- Validation and Verification: As development advances, simulation serves validation and verification purposes. This entails rigorous testing of the system against predetermined criteria, safety standards, and regulatory benchmarks. Simulation ensures that autonomous vehicles function predictably and reliably in diverse scenarios.
- Certification and Regulatory Compliance: Simulation assumes a central role in the certification process, enabling manufacturers to demonstrate compliance with regulatory and safety requirements. Comprehensive simulations validate system performance, evaluate risk factors, and ensure adherence to legal mandates prior to commercial deployment.

- Life-cycle Management and Continuous Improvement: Even after certification, simulation remains integral for life-cycle management and ongoing enhancements. Manufacturers utilise simulation to monitor system performance, analyse real-world data, and implement updates or modifications to optimize safety, efficiency, and user experience. Analysing near-accident situations by replaying the situation in simulation and varying the different parameters to identify hazardous situation.

By adhering to a well-structured temporal framework, stakeholders can effectively utilise simulation to streamline the evaluation and certification process for autonomous vehicles. This approach ensures robustness, reliability, and regulatory compliance throughout the development life-cycle.

2.1.1.4 System of Systems - AI Module Roles

In the realm of autonomous vehicles and simulation, understanding the role of AI modules within the system of systems is paramount. Each AI module performs specific functions and tasks, contributing to the overall functionality and performance of the autonomous vehicle. Here's a breakdown of the key AI modules and their respective roles:

- Perception AI: The Perception AI module is responsible for interpreting sensor data and understanding the vehicle's surroundings. It identifies objects, pedestrians, road markings, and other relevant elements in the environment to inform the vehicle's decision-making process.

- Decision AI: The Decision AI module processes information from the Perception AI and other sources to make decisions regarding vehicle control and navigation. It determines actions such as accelerating, braking, steering, and lane changes based on the perceived environment and predefined rules or objectives.

- Control AI: The Control AI module translates decisions made by the Decision AI into specific control commands to execute vehicle manoeuvres. It regulates vehicle dynamics, stability, and trajectory adherence to ensure safe and efficient operation under varying conditions.

By delineating the roles of these AI modules, manufacturers and developers can effectively design, implement, and validate autonomous vehicle systems using simulation. Simulation facilitates the testing and optimisation of each AI module's performance within the larger system, ensuring seamless integration and robust functionality.

2.1.1.5 Demonstration Process

In the realm of autonomous vehicle evaluation and certification, the demonstration process holds paramount importance. This process involves presenting and validating the performance of the autonomous system in simulated and realistic scenarios. Here are the key aspects of the demonstration process:

- Identifying Key Scenarios: Firstly, it's crucial to identify relevant demonstration scenarios that highlight the key capabilities and functionalities of the autonomous system. These scenarios may include urban driving, highway driving, emergency situations, and safety-critical events.

- Creating Realistic Simulations: The identified scenarios are then replicated in simulation environments, taking into account factors such as geography, traffic, weather conditions, and the behaviours of other road users. Creating realistic simulations allows testing the system's capabilities in diverse and complex conditions.
- Executing Scenarios: Simulations are executed with the autonomous system to evaluate its performance and responsiveness in each scenario. Relevant metrics are collected to assess the quality of the system's decisions, its ability to detect and respond to obstacles, and its reliability in various driving situations.
- Analyzing Results: The results of the demonstration are analysed to assess the system's compliance with performance objectives and safety criteria. Any potential gaps are identified, and corrective measures are considered to improve the system's performance.
- Validation and Certification: Finally, the system's performance during demonstrations is used as evidence for the validation and certification of the autonomous vehicle. Regulatory bodies and relevant authorities may rely on demonstration results to grant the necessary approvals for the vehicle to enter the market.

In summary, the demonstration process plays a crucial role in evaluating the performance and certifying autonomous vehicles. By using simulation to recreate realistic scenarios, manufacturers can demonstrate the reliability and safety of their systems, thus contributing to the advancement of autonomous vehicle technology towards wider adoption and secure integration on our roads.

2.2 Description of interfaces

2.2.1 Some requirements about interfaces

In the development of simulation environments for evaluation and validation processes, the integration of various simulation tools is often necessary to capture the complexity and detail of real-world scenarios. These tools, whether software-based or incorporating hardware components, must interact seamlessly to exchange data efficiently and ensure the accuracy of simulation results.

- Interface Standardisation: Ensuring that all simulation blocks adhere to the same interface definition is crucial for seamless data sharing and interoperability.
- Real-Time Execution: Hardware components within the simulation system must operate in real-time, aligning simulated time with real-time to maintain accuracy.
- Time Scaling for Software Components: Software-only simulations may run faster or slower than real-time, depending on the simulation's purpose and computational requirements.
- Performance-Dependent Simulated Time: The simulated time of a software-based simulation depends on the model's performance, with complex computations potentially slowing down the simulation.
- Synchronisation of Tools: When multiple tools are used together, synchronisation is essential to ensure they operate on the same simulated time, preventing discrepancies in results.

- Asynchronous Processing: In scenarios where there's no feedback loop and faster processing of certain data segments doesn't impact result realism, asynchronous operation may be feasible.
- Data Sharing Interface: Standard protocols like Functional Mock-up Interface (FMI) are often used for data exchange, although specific connectors may require custom development to integrate certain tools.
- Communication Protocols: Alongside standard interfaces, effective communication protocols are necessary to facilitate seamless interaction between simulation components.
- Scalability: Interfaces should support scalability to accommodate simulations of varying complexity and scale, ensuring flexibility in system design.
- Robustness and Error Handling: Interfaces should be resilient to errors and capable of handling unexpected scenarios gracefully, minimising disruptions during simulation runs.

Expanding on the text, consider discussing the importance of modularity in interface design to facilitate the integration of new simulation tools or components, as well as the need for comprehensive documentation and support to assist developers in utilising the interfaces effectively. Additionally, exploring methods for verifying interface compatibility and conducting thorough testing procedures to validate the integrity of data exchange mechanisms could enhance the discussion.

2.2.2 The need for open architectures

The demand for robust data exchange architectures emerged alongside the rapid advancement of electronic components for critical applications. Throughout the 1980s and 1990s, military systems faced a challenge: the processors they relied on became outdated even before deployment in operational settings ([36]).

To address this issue, open architectures were conceived, aiming to decouple software from hardware dependencies. This approach enabled hardware to evolve independently to match the pace of market advancements, without necessitating extensive software rework.

Open architectures rely on hardware abstraction layers (HALs) of varying complexity to ensure software independence from underlying hardware, even when utilising standards like PCI, PCI Express, RS232, ATA, and SATA. HAL encompasses all software directly reliant on the underlying hardware, including boot code, context switch routines, configurations, and hardware resource access such as Memory Management Units (MMU), on-chip buses, bus bridges, and timers.

While HAL serves as an abstraction of hardware architecture, its implementation often varies between operating system vendors. Consequently, most HALs are inherently tied to specific operating systems, as each vendor defines its unique HAL specifications ([37]).

2.2.3 The need for distributed architectures

While significant progress has been made through the standardisation of execution nodes with hardware components and the broader adoption of Hardware Abstraction Layers (HAL), these efforts have encountered limitations:

Firstly, the reliance on standards within hardware and HALs has not adequately facilitated the seamless reuse of software across diverse platforms. Despite attempts at standardisation, interoperability challenges persist, hindering true cross-platform compatibility.

Furthermore, with system capabilities increasingly distributed across networked components developed by various suppliers, the mere standardisation and openness of individual equipment technologies prove insufficient. This inadequacy becomes apparent as technological advancements continue to shape the components comprising these systems, making it challenging to maintain system capabilities over time.

Therefore, the crux of the matter lies not solely in standardising individual computation nodes, but also in standardising the distribution of tasks across multiple nodes. Achieving this level of standardisation is essential for ensuring interoperability, scalability, and long-term viability of complex systems composed of diverse hardware and software elements.

The logical architecture for distributed test system has been largely introduced in the deliverable 2.3 of the PRISSMA project. This architecture specifies the main functions to be interconnected through a communication bus (or middleware) to realise the simulation tools for the operational needs of the evaluation certification on the ARTS:

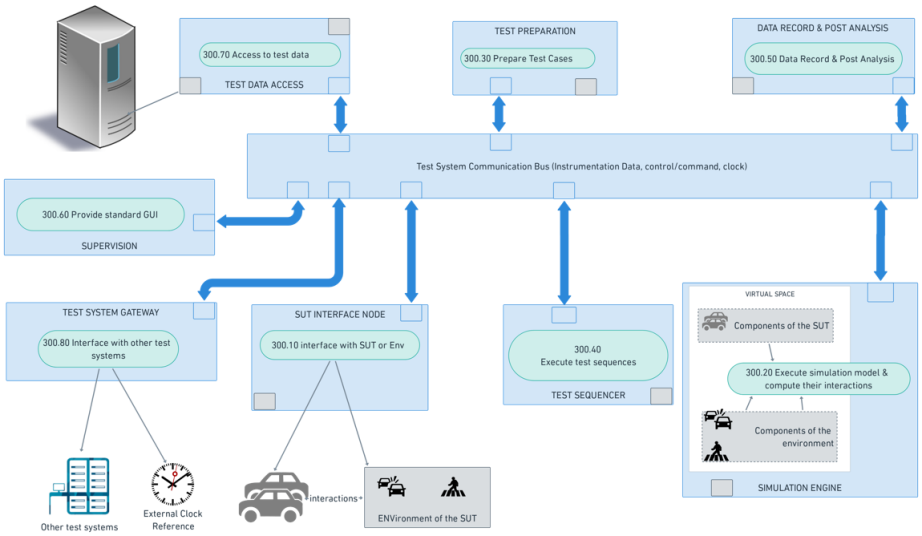


Figure 2: Logical architecture for distributed test and simulations platforms

2.2.4 The network OSI layer as a foundation

A network of computation nodes typically falls into two main categories:

- Broadcast Mode (e.g., bus or ring topology): This mode operates using a single transmission medium, where messages are broadcasted across the network. Any network unit can intercept and analyse the message to determine if it's intended for them based on the recipient's address.
- Point-to-Point Mode (e.g., star or mesh topology): In this mode, each pair of network units is directly connected via a physical medium. Communication between two units requires traversing through an intermediary node.

Each topology offers distinct advantages and disadvantages depending on the circumstances:

- Fully Connected Topology: While this topology facilitates communication between all nodes, implementing it at the physical level requires a significant number of electrical wires, resulting in higher costs and weight. This drawback is particularly challenging for applications such as aircraft, where weight considerations are crucial.

The OSI ISO/IEC 7498 model has played a pivotal role in delineating connectivity concerns across different equipment within a system, organising them into distinct layers. This model enables a systematic approach to network design, management, and troubleshooting by separating functionalities into discrete layers, such as the physical, data link, network, transport, session, presentation, and application layers.

2.2.5 The rise of the middlewares

The standardisation of various layers of inter-connectivity within software-intensive electronic computing resources has paved the way for the emergence of "middleware," facilitating the decoupling of component behaviour from their individual implementations.

Middleware, in essence, is software that facilitates communication and data management in distributed applications. Initially defined as services positioned above the transport layer (such as TCP/IP) but below the application environment, middleware serves as the conduit for interactions between components. Conceptually, middleware can be likened to the hyphen ("-") in client-server relationships or the "to" in peer-to-peer setups.

Middlewares come in various forms ([38]):

- Transactional Middleware: Primarily involved in processing multiple synchronous/asynchronous transactions, handling clusters of associated requests from distributed systems like bank transactions or credit card payments.
- Procedural Middleware: Facilitates remote procedure calls, enabling the connection, passing, and retrieval of software responses in asynchronous system communications.
- Message-oriented Middleware: Implements message queue and message passing architectures to support synchronous/asynchronous communication between distributed components.
- Object-oriented Middleware: Similar to procedural middleware but incorporates principles of object-oriented programming. This type of middleware encompasses object references, exceptions, and inheritance of properties through distributed object requests.

The architecture of any technical system revolves around determining the assembly of components that fulfil the system's identified functions. Ideally, these functions are executed at the application layer. A robust physical platform simplifies the implementation of architectural tasks and reduces realisation costs, emphasising the importance of choosing the right components to achieve system objectives effectively.

2.2.6 Tests systems architectures

In the VHTNG reference architecture, the exchanges among the different components are grouped into three different categories ([39]):

- The physical system under test communication channels, which regroups the communication of real components using real communication channels.

- The virtual system under test communication channels, based on the ED247 standard simulate the communication between virtual components.
- The VHTNG Instrumentation networks which regroups the communication between the components of the test system itself.

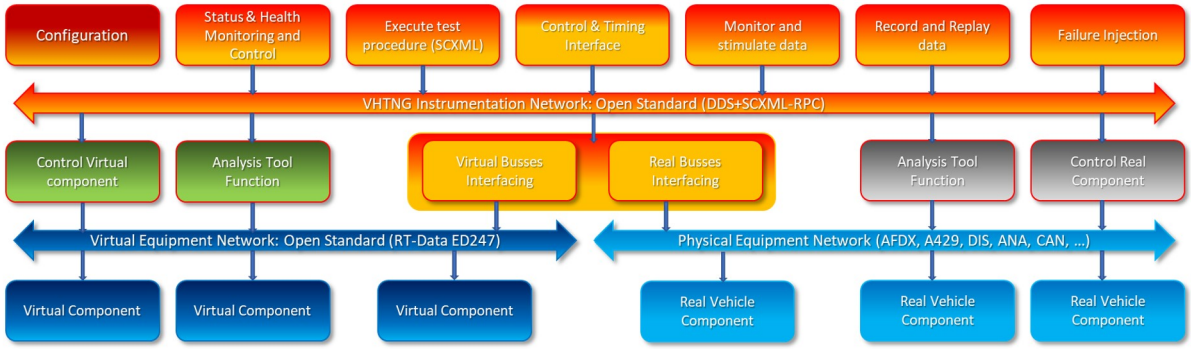


Figure 3: Allocation of functions to implementation

By enforcing standard interfaces for the Virtual Equipment network and the Instrumentation network, the VHTNG project aims at enabling the setup of distributed test systems provided by multiple suppliers. In the [40] article, the authors proposed a referenced implementation of their simulation platform using DDS middleware for real time distribution of simulation data. In addition to the DDS middleware, the FMU/FMI standard has been used for integrating simulation to this platform. In both VHTNG and this article we see the advantages of using standards for designing open architectures: the integration of new components adding features can rely on their compatibility on these standard for easier integration.

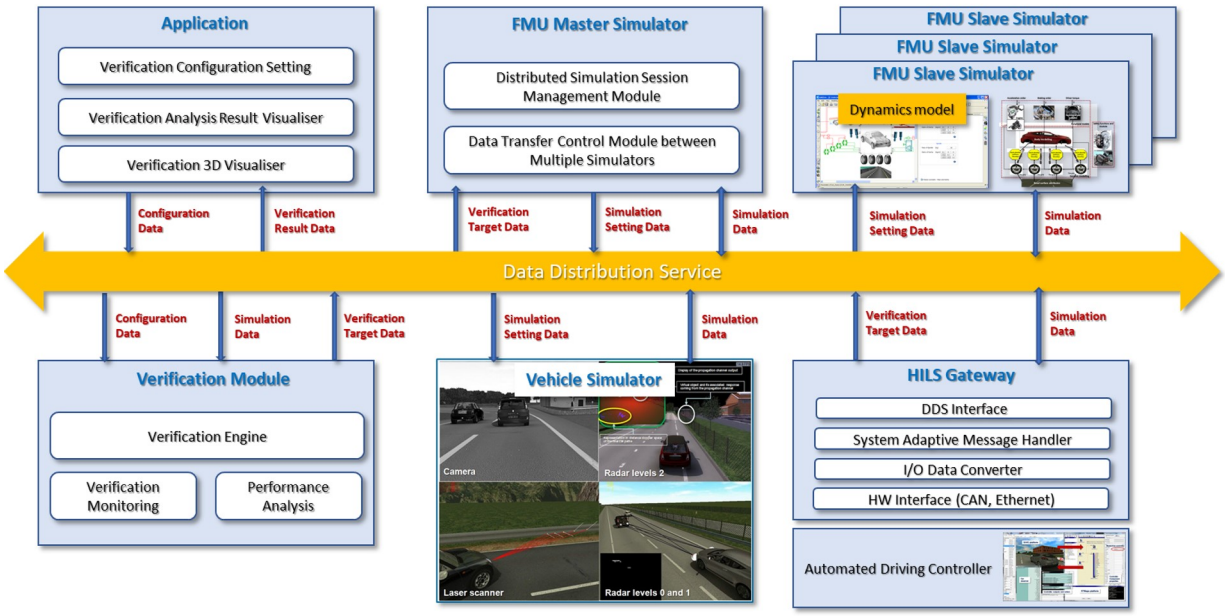


Figure 4: Simulation platform based on DDS and FMU

2.2.7 Data exchange library

EPICS, or Experimental Physics and Industrial Control System , stands as a collaborative endeavour, offering a suite of open-source software tools, libraries, and applications. Its global utilisation extends to the creation of distributed soft real-time control systems for scientific instruments, ranging from particle accelerators to telescopes and large-scale scientific experiments. EPICS empowers researchers and engineers to orchestrate complex systems with precision and efficiency, facilitating breakthroughs in various domains of scientific exploration.

In the realm of Internet-of-Things (IoT), **MQTT** reigns as a dominant protocol. Functioning on a "publish and subscribe" model, MQTT excels in transmitting sensor data with exceptional efficiency owing to its lightweight nature. However, its design may not be optimised for the transmission of larger data payloads, such as video clips.

Conversely, **XMPP**, or Extensible Messaging and Presence Protocol, offers swift and real-time communication capabilities. Built upon the foundation of Extensible Markup Language (XML), XMPP provides the flexibility to define message formats, enabling structured data exchange across network nodes. Moreover, XMPP boasts robust security features, including identity management, authentication, authorisation, and encryption, making it an ideal choice for applications requiring stringent security measures.

AMQP, or Advanced Message Queuing Protocol, delineates an efficient, binary, peer-to-peer communication protocol tailored for transporting messages between network processes. By decoupling message structure from delivery mechanisms, AMQP affords flexibility and scalability. With support for multiple messaging protocols, AMQP can be seamlessly integrated into distributed configurations to meet high-scale and high-availability demands.

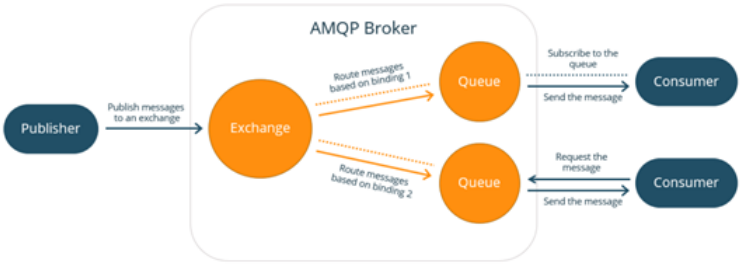


Figure 5: AMQP Broker

DDS, or Data Distribution Service, implements a publish-subscribe pattern for seamless data exchange among distributed system nodes. By creating topics and publishing samples, DDS facilitates the dissemination of information to subscribers expressing interest in specific topics. Notably, DDS empowers users to configure quality of service parameters upfront, streamlining the development of distributed applications while promoting modularity and structural integrity.

Finally, the **ED247** virtualisation protocols offer standardised mechanisms for exchanging simulation data across virtual components. This standardisation simplifies the interconnection of benches across different locations in a non-intrusive manner, enhancing the versatility and interoperability of virtual data exchange within simulation environments.

2.2.7.1 Implementation of DDS in Pro-SiVIC

In Pro-SiVIC, the proposed solution is based on the principle of a distributed storage space through which SiVIC instances exchange object state vectors:

- A SiVIC instance responsible for a simulation aspect (e.g., vehicle model) publishes the corresponding state vector on DDS.
- A SiVIC instance wishing to replicate this simulation aspect obtains the corresponding state vector from DDS.

The mechanism of the distributed storage space itself is developed in a separate C++ library called ”libdds”. This library manages the publication and subscription to raw data frames (byte arrays). Data on DDS is identified by the DNS alias or the IP address of the computer publishing it and a string name. ”libdds” can also be used to exchange information other than Pro-SiVIC simulation actor state vectors (sensor data, etc.). For networking aspects, ”libdds” relies on the C ENet library ((http://enet.bespin.org/))and operates on a peer-to-peer logic. This means that each computer acts as both a data-consuming client and a data-providing server. Using ENet ensures lower latency than TCP, minimising Pro-SiVIC instance desynchronisation, and provides flexibility in quality of service management (optional data frame robustness, etc.). Additionally, ”libdds” offers high-performance data sharing on the same computer by implementing faster paths via IPC. The raw data frames exchanged via ”libdds” are produced and consumed in Pro-SiVIC by a new plug-in called ”sivicDDS”. This plug-in manages, for the Pro-SiVIC instance in which it is loaded, the list of simulated states that are to be published on DDS or replicated from it. The ”sivicDDS” plugin contains the logic to convert an object’s state into a data frame / byte array and vice versa. The general architecture is illustrated in the diagram below. Red arrows indicate data publication on DDS, while blue arrows represent consumption of shared data.

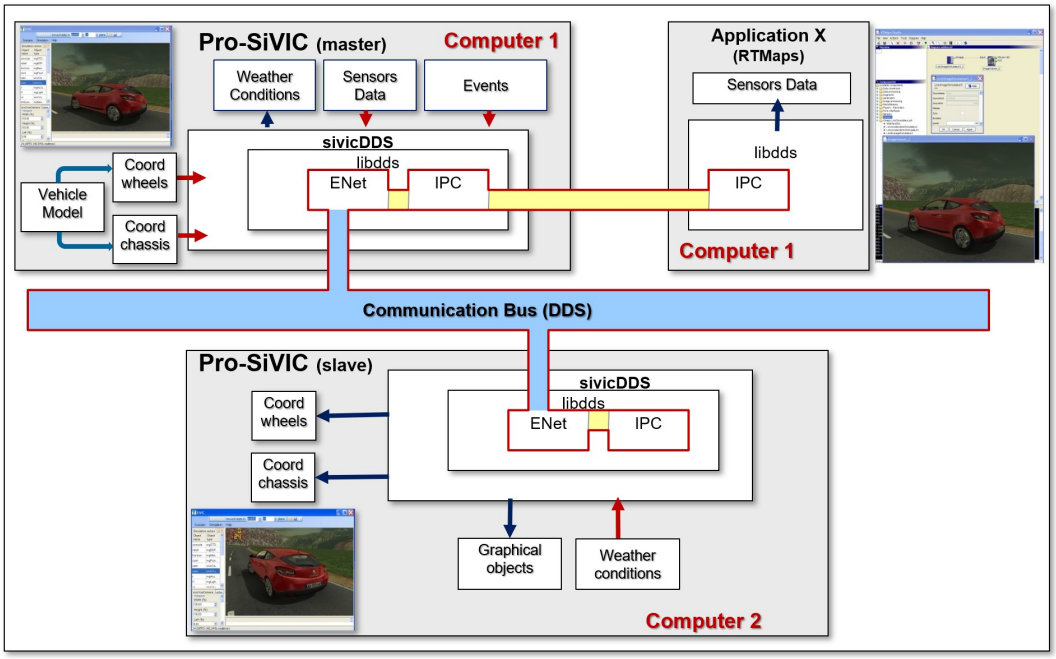


Figure 6: Implementation of DDS in Pro-SiVIC in order to exchange Data and Objects-Components-Environment attributes, and to synchronise the multiple platforms, tools, softwares. (Source: UGE)

2.2.8 Data exchange format

The Functional Mock-up Interface (FMI) stands as a pivotal standard in the realm of dynamic model exchange, offering a structured framework for sharing and simulating models across

diverse simulation environments. FMI delineates a comprehensive approach, comprising a ZIP archive format and an application programming interface (API), aimed at facilitating seamless model interchange. Within this framework, models are encapsulated within Functional Mock-up Units (FMUs), encompassing a combination of XML files, binaries, and C code.

The FMI API serves as the linchpin for simulation environments, acting as the conduit for interacting with FMUs. Simulation environments utilise this API, known as the importer, to instantiate one or more instances of an FMU and simulate them, often alongside other models. This interoperability enables the integration of diverse models into cohesive simulations, fostering collaboration and innovation across various domains.

FMI defines three distinct interface types, each tailored to cater to different simulation scenarios:

- Co-Simulation (CS): In this mode, the FMU typically embeds its solver or scheduler, enabling autonomous execution within the simulation environment. This approach affords flexibility and autonomy to individual FMUs, enhancing their capability to interact with other components within the simulation.
- Model Exchange (ME): Contrary to co-simulation, model exchange necessitates the importer to undertake numerical integration tasks. The importer assumes responsibility for synchronising and coordinating the execution of FMUs, facilitating seamless integration into the simulation environment.
- Scheduled Execution (SE): In SE mode, the importer orchestrates the execution of model partitions within the FMU. By triggering the execution of specific segments of the model at predefined intervals, SE mode offers granular control over the simulation process, optimising performance and resource utilisation.

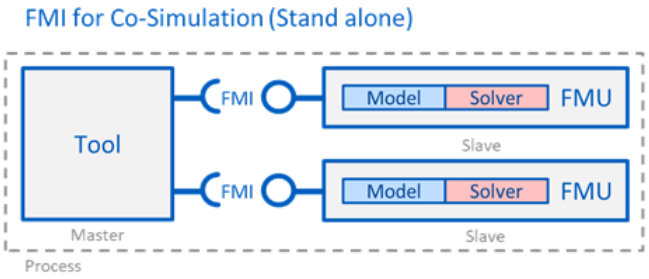


Figure 7: FMU and FMI using

By offering these diverse interface types, FMI caters to a wide spectrum of simulation requirements, ranging from autonomous FMU execution to tightly integrated model exchange scenarios. This versatility empowers users to tailor their simulation setups to suit specific needs, fostering innovation and collaboration within the dynamic modelling community.

2.2.9 Synchronisation and time management

Here are the requirements related to time management and synchronisation for a simulation framework for automated vehicles prototyping, test, evaluation, and validation:

Time Period for the World:

- Flexible Time Scale: The simulation framework should allow users to define the time period for the simulated world, enabling them to adjust the simulation speed according to specific scenarios and testing requirements.
- Temporal Resolution: The framework should support high temporal resolution to accurately simulate dynamic events within the world, such as vehicle movements, traffic interactions, and environmental changes.
- Dynamic Day-Night Cycle: Incorporating a dynamic day-night cycle feature with adjustable time periods allows for realistic simulation scenarios, enabling testing under varying lighting conditions and environmental contexts.
- Seasonal Variations: The framework should support the simulation of seasonal variations, including changes in weather patterns, road conditions, and daylight hours over extended time periods.
- Simulation Time Management: Users should be able to control and manipulate the simulation time, including pausing, resuming, and rewinding the simulation, to analyze specific events or replay scenarios for evaluation purposes.
- Time-Stamped Data Logging: The framework should provide support for time-stamped data logging, allowing users to capture and analyze simulation data with accurate temporal references for performance evaluation and debugging.

Time Period for Each Sensor:

- Sensor Time Synchronisation: Ensure that each sensor within the simulation framework operates according to its specified time period, with synchronised timestamps to maintain temporal coherence between sensor outputs.
- Adjustable Sensor Sampling Rate: Allow users to configure the sampling rate and time period for each sensor independently, providing flexibility to match real-world sensor behaviour and capture relevant data for analysis.
- Temporal Alignment of Sensor Data: Implement mechanisms to align sensor data streams temporally, accounting for processing delays and communication latencies, to ensure accurate sensor fusion and perception algorithms.
- Dynamic Sensor Activation and Deactivation: Enable dynamic activation and deactivation of sensors based on scenario requirements, allowing users to simulate sensor failures, occlusions, or power-saving modes while maintaining temporal consistency.
- Realistic Sensor Behaviour Modelling: Incorporate realistic sensor behaviour models that simulate sensor response times, noise characteristics, and temporal dependencies to accurately replicate sensor performance in diverse environmental conditions.
- Temporal Calibration: Provide tools for temporal calibration of sensors to synchronise their internal clocks and align their outputs with ground truth data or reference sources for validation and accuracy assessment.

Time Period for Vehicle Dynamics:

- Vehicle Dynamics Simulation Time Step: Define a suitable time step for simulating vehicle dynamics, ensuring sufficient temporal resolution to capture vehicle motion dynamics, control inputs, and interactions with the environment.
- Real-Time Vehicle Dynamics Update: Implement real-time vehicle dynamics update mechanisms to compute vehicle states, such as position, velocity, and orientation, at each simulation time step based on dynamic models and control algorithms.
- Time-Synchronised Vehicle Control: Synchronise vehicle control commands with the simulation time step to ensure accurate vehicle response and behavior in dynamic scenarios, enabling realistic interaction with the simulated environment and other vehicles.

Time Management for Scenario:

- Scenario Time Management: Allow users to define and manage scenarios with adjustable time periods, enabling the creation of complex, time-sensitive scenarios for testing various automated driving functionalities and safety-critical scenarios.
- Temporal Event Triggering: Implement mechanisms to trigger events, such as traffic incidents, pedestrian crossings, or vehicle maneuvers, at specific time intervals or relative to the simulation time, facilitating scenario orchestration and control.
- Temporal Constraints and Deadlines: Support the specification of temporal constraints and deadlines for scenario execution, ensuring that critical events occur within predefined time windows and enabling the evaluation of system responsiveness and performance under time-critical conditions.

Time Synchronisation Mechanism for Distributed Architecture:

- Network Time Protocol (NTP) Integration: Integrate NTP or similar time synchronisation protocols to achieve consistent time synchronisation across distributed simulation nodes, minimizing time drift and ensuring temporal coherence in multi-agent simulations.
- Distributed Clock Synchronisation: Implement distributed clock synchronisation algorithms, such as the Network Time Protocol (NTP) or Precision Time Protocol (PTP), to maintain accurate and synchronised simulation time across distributed computing resources.
- Latency Compensation: Compensate for communication latencies and processing delays between distributed simulation nodes to achieve tight synchronisation of simulation events and maintain temporal consistency in collaborative simulation scenarios.
- Error Handling and Resilience: Implement error handling mechanisms to detect and mitigate synchronisation errors or discrepancies between distributed simulation nodes, ensuring robustness and reliability of the time synchronisation mechanism.
- Dynamic Load Balancing: Employ dynamic load balancing techniques to distribute computational workload evenly across distributed simulation nodes while preserving temporal coherence, minimizing simulation time deviations, and maximizing overall system performance.

By incorporating these requirements, a simulation framework for automated vehicles can effectively manage time periods, synchronize simulation components, and ensure temporal coherence, enabling accurate and reliable evaluation of automated driving systems in diverse scenarios.

3 Definition and requirement of the simulation environment

3.1 Overview of the Generic simulation architecture

This section propose to provide a global definition of what is needed in order to develop and to build a simulation environment dedicated to the evaluation and the validation of Automated Mobility Systems involving AI-based algorithms, functions, modules, and components. This section will address the main requirements, properties, capabilities, expectations, limits, weaknesses of the different models and tools that are part of a simulation architecture.

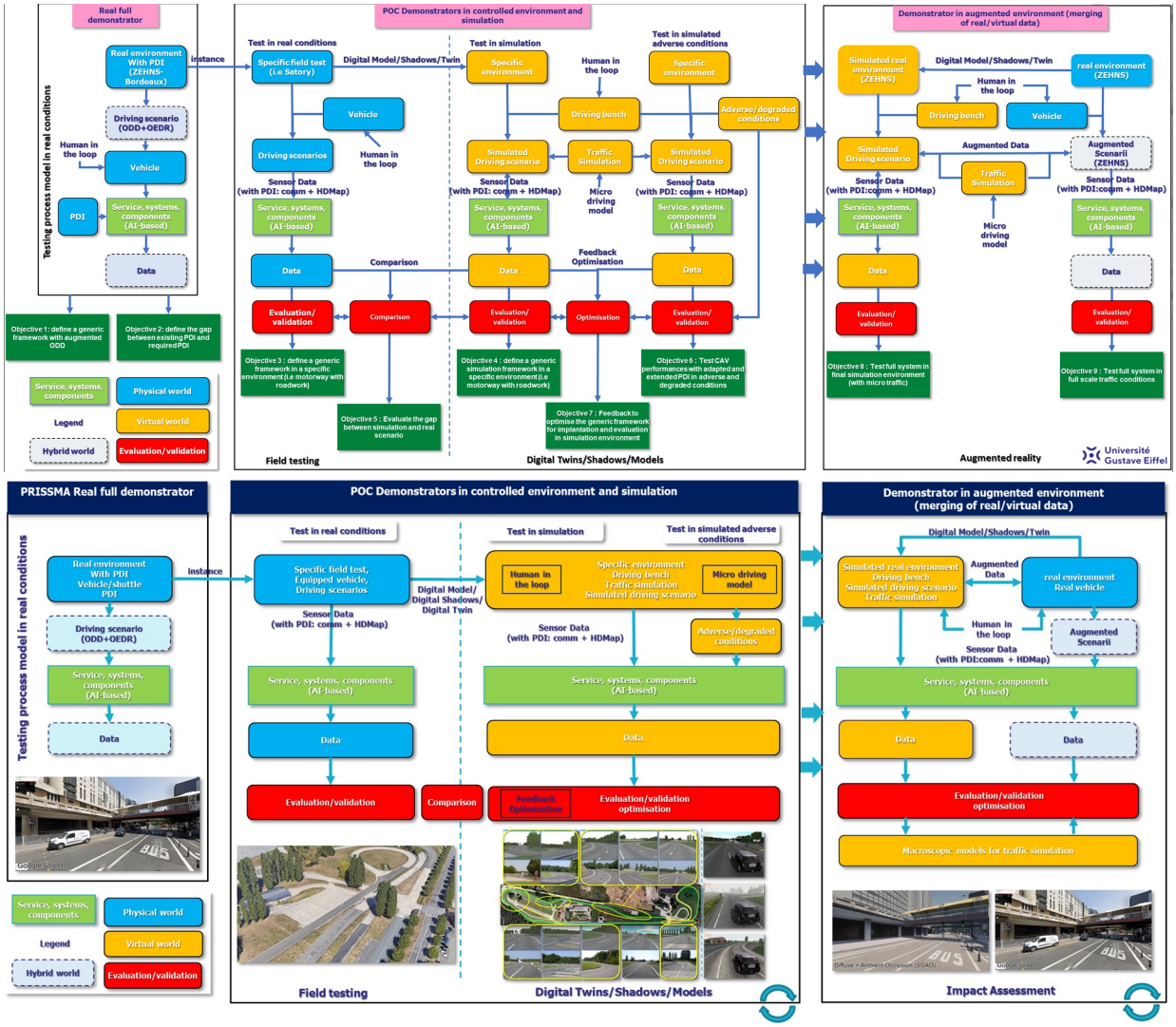


Figure 8: Overview of the general framework for the virtual prototyping, test, evaluation, and validation of ADS and AI-based systems (source UGE)

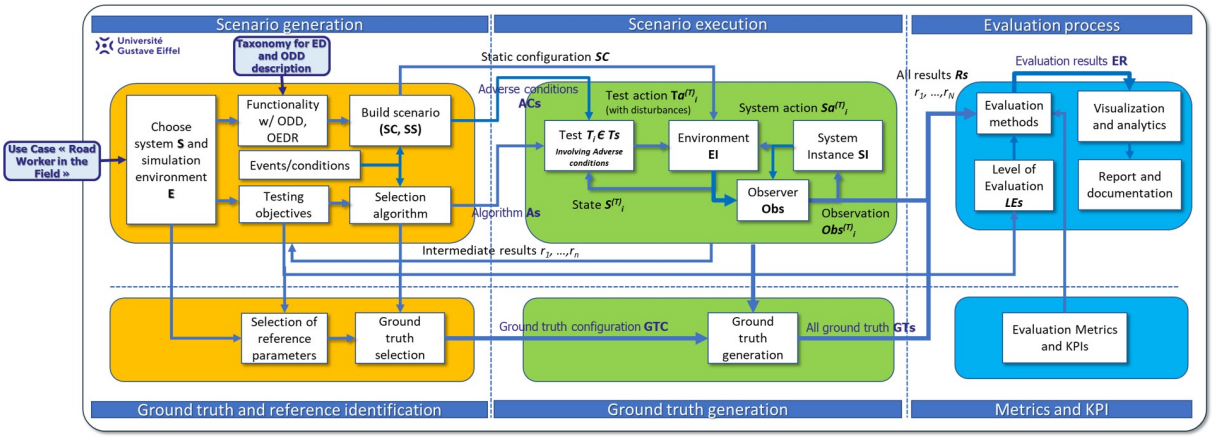


Figure 9: Overview of the generic PRISSMA framework for the evaluation and validation process in simulation (source UGE)

3.2 Requirements for the models, functions, components and tools involved in the simulation environment

The main requirement in order to build a virtual simulation environment for the evaluation and validation of AI-based modules and systems involved in CAV development and deployment is summarised in the figure 8, figure 61, and figure 9. In the figure 8 we can see the 3 main stages of a global evaluation process. The first stage consists to:

- Define generic framework of the service using (with OD, ODD, and OEDR spaces based on taxonomy developed in WP8)
- Define the scene and scenarios allowing to cover the ODD and the possible situations and events encountering
- Define the service, systems, and components involved in the application under test (involving AI-based modules and functions)
- Define the real vehicle embedded architecture (hardware and software)
- Define the topology of sensors
- Define the data recording process
- Define the evaluation and validation procedure

Unfortunately, in open road and real situations, it is impossible to generate and to control all the possible scenarios. Indeed, we don't have the capability to control accurately the movements and behaviours of road users, and the degraded and adverse conditions like bad weather configurations. It is for this reason that it is necessary to address the second stage. This second stage involved the implementation of the application/component/service under test in a real controlled environment and in the same simulated environment. This need the building of an accurate and realistic (high level of fidelity) Digital Twin (involving the Digital Shadow). The hardware and software components involved in the evaluation bench for the real controlled environment are detailed in the WP3. In this deliverable, we will focus the definition and requirement only on

the simulation environment. About the simulated controlled environment (test site), the main components and functions needed in order to execute the scenario are provided in the figure

3.2.1 General requirements on the simulation and multiple spectral graphical engine

In order to be usable for the evaluation and validation stages, Simulation engines and graphical engines (games engines) have to respect a set of requirements and constraints about software (capabilities, rendering level realism, physics engine realism, time management ...) and hardware aspects (sharing of treads, memory control and access, CPU and GPU control, ...).

3.2.1.1 General requirements for graphical engines

Here are the main mandatory requirements for a graphical engine in a simulation environment:

- **Rendering aspects**:
  - Rendering Quality: Provide high-quality rendering capabilities to ensure realistic visualization of simulation elements, including vehicles, sensors, road users, environments, weathers, and the other road and infrastructure objects and furniture. In a simulation environment, rendering quality refers to the fidelity and realism of the visual output produced by the graphical engine. This requirement is paramount as it directly influences the user's perception or the sensors' perception of the simulated environment's accuracy and immersion.
  - Fidelity of Visual Modelling: The graphical engine must accurately depict simulation elements, including vehicles, sensors, road users, environments, weathers, and the other road and infrastructure objects and furniture, with high fidelity. This entails rendering details such as textures, geometry, lighting, shadows, reflections, and particle effects realistically to resemble their real-world counterparts.
  - Realism and Fidelity: The rendered scenes should evoke a sense of realism and fidelity, allowing users and sensors to engage with the simulated environment as if it were real. This involves employing advanced rendering techniques such as physically-based rendering (PBR), global illumination, and atmospheric effects to create convincing visuals. For sensors, it means to take into account multi-spectral rendering capabilities. Multi-spectral rendering needs to use specific physical models, materials, and textures to reproduce the interaction between sensors and the environment.
- **Consistency Across Platforms and API**: Regardless of the hardware specifications or operating system used, the rendering quality should remain consistent across different platforms. Users and sensors should experience similar fidelity and realism whether running the simulation on a high-end gaming PC or a mobile device.
  - Hardware Compatibility will ensure compatibility with a wide range of hardware configurations, including graphics cards, processors, and memory capacities, to maximize accessibility and performance.
  - Cross-Platform Support will offer cross-platform compatibility to run the graphical engine seamlessly on various operating systems, including Windows, macOS, and Linux.

  - API Support will provide support for standard graphics APIs such as OpenGL, Vulkan, or DirectX to leverage hardware acceleration and optimize rendering performance.

- **Dynamic and Optimised Rendering**: The graphical engine should dynamically adapt rendering quality based on the available hardware resources and performance constraints. It should prioritize rendering critical elements of the scene at high quality while adjusting less critical components to maintain smooth frame rates and responsiveness. In order to guarantee a real time operating and the scalability, some resources management mechanism need to be implemented. For instance, the BSP is interesting in order to take into account vertices, faces, materials, textures ... only in the field of view of the human vision or the sensors. The scale rendering performance is essential to accommodate large and complex simulation scenes with numerous objects and detailed environments. Support real-time rendering to maintain interactive responsiveness and fluidity, crucial for dynamic simulations with changing scenarios.

  - Post-Processing Effects: Integrate post-processing effects like bloom, depth of field, motion blur, and color grading to enhance visual aesthetics and realism.
  - Level of Detail (LOD): Implement level of detail techniques to dynamically adjust object detail and polygon count based on distance from the viewer, optimizing rendering performance without sacrificing visual quality.

- **Materials and Textures**: Objects and textures within the simulation should feature sufficient detail and resolution to appear crisp and detailed, even at close inspection.

  - Texture levels: High-resolution textures, bump mapping, and normal mapping techniques can enhance the perceived quality of rendered surfaces. Moreover, it is essential to implement generative and procedural textures.
  - Reflections: In order to take into account reflection effects, the dynamic generation of planar, cubic, and spherical texture is mandatory. These reflection textures allow to apply reflection of the environment on car bodies, wet road surfaces, shop windows ...
  - HDR texture: High Dynamic Range texture is mandatory in the last graphical engines in order to take into account the light intensity for each pixel of a texture. It is useful for skybox textures allowing to generate the blurring effect of the sun.
  - Shaders: In order to optimise the rendering time, the use of shader and GPU capabilities (CUDA) is mandatory.
  - Meta material and texture: Essential for multi-spectral rendering allowing to reproduce the interaction between sensors and the environment.

- **Anti-Aliasing and Filtering**: Implement anti-aliasing techniques to reduce jagged edges and aliasing artifacts, resulting in smoother edges and improved visual clarity. Additionally, apply texture filtering methods such as anisotropic filtering and mipmapping to enhance texture quality and reduce visual distortion.

- **Dynamic Lighting and Shadows**: Render dynamic lighting effects such as realistic shadows, reflections, and light scattering to simulate the interplay of light and shadow within

the environment accurately. This includes dynamic shadow mapping, soft shadows, and volumetric lighting to create immersive lighting scenarios. It is also needed to implement mechanism of ambient occlusion map and self-shadowing in order to improve the visual rendering. About dynamic lighting, it is now essential to have the capability to manage and generate in same time a great number of light sources with specific characteristics. This means the need to provide adaptive and physical capabilities to merge several shadows mask (objects and light sources)

- **Particle Effects and Special Effects**: Incorporate particle effects and special visual effects such as smoke, fire, explosions, weather effects, and environmental phenomena to add dynamism and realism to the simulation environment. These effects should blend seamlessly with the rendered scene and respond dynamically to user interactions and environmental conditions.

- **Raytracing capability**: Integrate a ray tracing engine to enhance rendering quality by simulating the behaviour of light rays as they interact with objects in the scene. Ray tracing enables advanced effects such as accurate reflections, refractions, and global illumination, resulting in photorealistic visuals with lifelike lighting and shadows. Moreover, raytracing functions allow to manage the interaction between wheels and the ground, the collision events, the simulation of LIDAR beams, RADAR, GPS (multiple reflection effect) ...

- **Terrain Generation**: Generate realistic terrain meshes and landscapes, including hills, valleys, roads, and vegetation, to simulate diverse environmental conditions.

- **Physics Integration and animation support**: Integrate physics simulation and physics engine or library with the graphical engine to accurately depict object interactions, collisions, and dynamic behaviour within the simulation environment. The animation support enables animation capabilities for objects, vehicles, and characters within the simulation environment, including skeletal animation, keyframe animation, and procedural animation.

- **User Interface Integration**: Seamlessly integrate graphical user interface (GUI) elements, menus, and overlays within the simulation environment for user interaction, configuration, and data visualisation. This requirement will be developed and detailed in a future section.

- **Multi-View Support**: Support multiple viewport configurations and camera perspectives to enable simultaneous viewing of different simulation aspects and viewpoints. In the framework of the development of specific simulation engine for mobility purposes, it is necessary to provide work windows with landmarks and specific references. Moreover, a multiple layers mechanism is necessary to handle different level of information needed for the sensor simulation (RADAR, GPS, IR ...). A set of data viewers and oscilloscopes functionalities are mandatory for the analysing and display of data coming from the components of the simulation (vehicles, pedestrians, sensors, weather conditions, ...).

- **Engine architecture and monitoring**:

  - Plugin Architecture: Provide a flexible plugin architecture to extend and enhance the graphical engine's functionality through custom modules, shaders, and rendering

techniques. This aspect is essential for a dynamic, adaptive, and real time simulation platform. The dynamic class loading system will allow, during the simulation, to load, adapt, modify, delete components.

- Performance Monitoring: Include performance monitoring tools and diagnostics to track rendering frame rates, memory usage, GPU/CPU load, and rendering bottlenecks for optimisation.

- **Documentation and Support**: Offer comprehensive documentation, tutorials, and technical support resources to assist users in understanding and effectively utilizing the graphical engine's features and capabilities.

These requirements aim to ensure that the graphical engine delivers high-quality, real-time rendering performance, scalability, cross-platform compatibility, customizability, and integration with other simulation components. At this moment, the 2 main graphical engine respecting these constraints and requirement are Unreal Engine 5 from NVIDIA and Unity.

Figure 10: Simulation platform from DSPace using Unreal Engine 5 (https://www.unrealengine.com/en-US/spotlights/dspace-drives-advancements-in-autonomous-vehicle-testing)

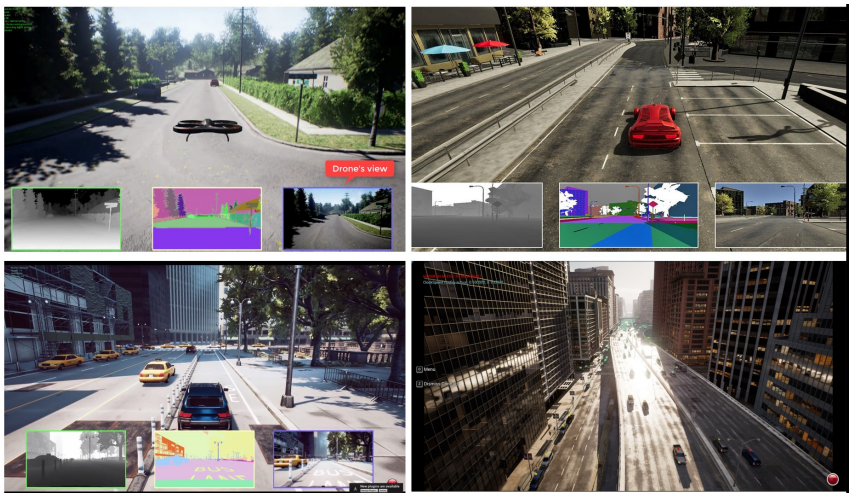


Figure 11: Simulation platform from Microsoft (AirSim) using Unreal Engine 5 (https://microsoft.github.io/AirSim/)

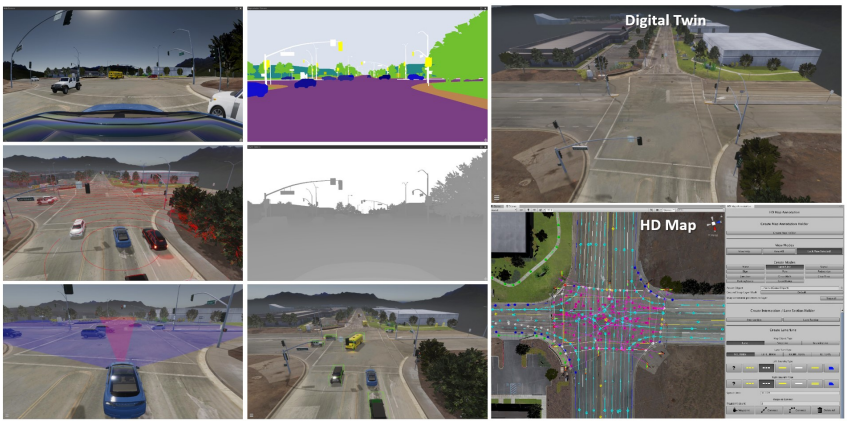


Figure 12: Simulation platform from LG (LGVSL) using Unity (https://microsoft.github.io/AirSim/)

Figure 13: Simulation platform from NVIDIA (NVIDIA Omniverse Replicator For DRIVE Sim – Synthetic Data Generation) using Unreal Engine 5 (https://microsoft.github.io/AirSim/)

3.2.1.2 Additional requirement for the simulation and multiple spectral graphical engine in PRISSMA

From [41] and with an adaptation from PRISSMA objectives, a set of requirements are provided in order to validate the use of a simulation platform, and more specifically the graphic and simulation engine:

- **Multi-resource constraint**: The platform must support multiple sources of input data to facilitate world creation and scenario generation.
- **Scenario management**: The platform must support test automation across multiple created worlds and scenarios. This also involves the use of a specific scenario format such as OpenScenario and an event management mechanism to manage transitions between the scenes constituting the scenario. This also involves the use of a task and action scheduler.
- **Scripting Language**: Scripting of the test automation process should be possible using standard scripting languages. This language must be usable at any time and allow the

management, modification, addition, and deletion of any object, any parameter, and any action in real time.

- **Transparent code**: The platform should use open source code as much as possible for control and decision logic.
- **Modularity and adaptability**: The platform must provide common core functionality with a configurable modular design. This means that the platform must be made up of easily loadable or unloadable plug-ins. This architecture must also offer an architecture allowing this processing to be distributed across several processors and several remote computers.
- **Simulation Fidelity and Quality**: The platform will support physics-based worlds.
- **Sensor Modelling**: The platform must support editable sensor models.
- **Sensor Types**: The platform must support the most common sensors in the automotive domain. The platform must support at least RADAR, LIDAR, GPS, IMU, camera and ultrasound sensors.
- **References and ground truths**: The platform must provide ground truth data during simulation execution.
- **Ego-Vehicle Control**: The platform must provide the ability to enable a control channel to control the simulated ego-vehicle.
- **Control of actors and extras**: The platform must offer the possibility of controlling several actors, that is to say vehicles other than the vehicle concerned or pedestrians. The platform must be able to populate the scene to increase loyalty.
- **Ensure process parallelisation**: The platform must support the ability to run and evaluate multiple control channels simultaneously.
- **Signal scheduler**: The platform must offer the possibility of prioritising control signals between the different channels.
- **Control Signals**: Each control channel must provide control signals conforming to the same specification.
- **Data flow management and scheduler**: The platform must be able to distinguish control signals coming from different channels.
- **Sensors and sources separability**: The platform must be able to distinguish between several sensors of the same type.
- **Unexpected events and rare scenarios**: The platform will support the creation of hand-crafted worlds and scenarios, as well as their subsequent adaptation to take into account rare and unexpected scenes and scenarios.
- **Usability of Datasets**: The scenario database must be reusable in the sense that it must be independent of the perception and control logic used.

- **Data channel adaptability**: The connection of the sensors to the control channel must be configurable to adapt to the needs of each channel.
- **Cyber security and operating safety**: The platform must have the ability to inject faults during execution. Moreover, the platform must provide inputs and mechanism allowing to simulate cyber attacks (perception, communication, component).
- **Generic scripting language**: The platform must offer the ability to script fault injection in the same interface as test automation.
- **Generic architecture respecting standard**: The platform must support co-simulation standards such as FMI for large and specialised simulations.
- **Reproductibility**: The portability of code generated from the platform should not be limited to a particular hardware configuration.
- **Repeatability**: The platform must provide the same result after n times the same scenario with the same platform configuration.
- **Real-time constraint**: The platform must guarantee a real-time processing of the involved model, plug-ins, module. For instance, in order to feed a camera model, the rendering stage needs to guarantee an operating with an accurate timestamp and a frequency at least 2 times higher than the simulated sensors. In a offline process, this mean that the time engine must operate 2 times higher that the plug-in included the camera model. Some issues could occur for High frequency sensors like neuromorphic camera operating at 100 khz.

Moreover in [42], the author presents the main requirements and functionalities needed to have a graphic engine usable for evaluation and validation stages. In this report, a simple comparison is done between the main industrial platforms with respect to their vehicle dynamics models, sensor support, 3D rendering, real-time physics and collisions, actor manoeuvres and behaviours, model libraries and hardware dependencies.

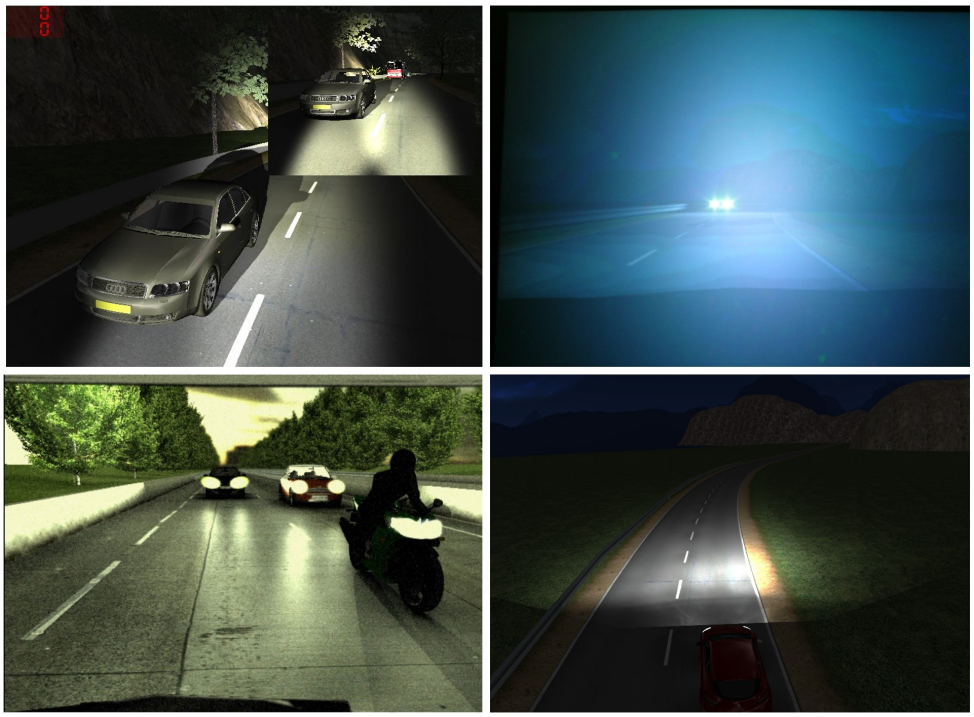


Figure 14: Light management by night with Pro-SiVIC with light map and mask for head light, pixelic rendering, and HDR texture(Source: UGE and ESI group)

Figure 15: The different level of shadows managed in Pro-SiVIC: cast, catch, self shadowing, occlusion (Source: UGE and ESI group)

From the different studies and the research work develop in University Gustave Eiffel [43], it appears that a graphic engine is not enough for evaluation and validation of system of systems, and component AI-based. Graphic engine needs to be extend in order to obtain simulation engine usable with a high level of fidelity, quality, representativeness. The following functions and capabilities need to be validated:

- **Multi-spectral rendering and modelling of the propagation channel**: The simulation engine needs to have the capability to mimic signals provided on several bandwidth. This capability is essential for sensors modelling. For instance, the simulation of intrinsic parameters and operating of the RADAR needs to generate and to process high frequency signals (24 Ghz or 79 Ghz). Currently, this type of high frequency management is obtain by using specific libraries and GPU capacities (use of CUDA language).

- **Lights generation and management** (see figure 14): Provide the capability to manage a large set of light sources with an accurate and efficient pixel level rendering (for real time processing). Generation of accurate and dynamic light masks for instance for nigh conditions.

- **Shadows generation and management (see figure 15)**: The simulation engine needs to manage properly the different light sources and their interaction with the objects and the environment. This means to propose mechanism providing several shadows renderings (ambient occlusion map, occlude shadow, cast and catch shadows, self-shading, ...)

- **Material and meta material**: The simulation engine needs to provide large range and efficient resource management (graphics (material, texture, ...), Cross Radar Section, Bidirectional Reflectance Distribution Function (BRDF), IR material emission, ...)

- **Textures management and generation**: Provide shaders and functions allowing to manage and generate HDR texture (coding light intensity, see figure **??**), procedural and animated textures, Multiple reflexion mechanism (environment reflection on car body, windows, wet road, ... with a resolution fitting with requirement of sensors

- **Ray tracing mechanism**: In order to manage with a high level of accuracy and fidelity specific dynamic models, interaction between environment objects, or propagation channel properties (GPS, RADAR, ...), an efficient and real time ray tracing mechanism is needed.

- **Filter mechanism**: Library of shaders usable by sensors and implemented specific physical models allows to apply specific modification and transformation to a raw data generated by sensors. This mechanism is useful and essential for camera and for weather conditions generation. These filters implement for instance planar, cubic, cylindrical reflections as shown in figure 34, noise, blur, fog, Depth of Field, optical deformation, colour, self Exposure, auto focus, ...

- **Spatial management**: The simulation engine need to implement quaternion library in order to avoid errors generation in the positioning and the orientation of the objects.

- **Physical engine**: Library allowing to apply dynamic model for dynamic object with the management of physical interactions between objects. For instance, a truck modelling needs to implement dynamic modelling of the cabin and the trailer with the physical link between both. At this moment, for robot simulation with physical interaction, the main physical engine are given in [44] and are ODE, Bullet, Vortex, and MuJoKo. For the game engine ([45]), the main physic engine are PhysX, Bullet, Havok, MuJoCo, and ODE. In [46], a comparison of these physical engine is made. Unity also proposes eXpanSIM in order to model vehicle with physic engine capabilities.

- **Particle filter**: Provide a efficient mechanism for adverse conditions simulation. Particle engine allows to generate rain, snow, fog, cloud, smoke, fire effects with a high level of fidelity.

- **Multiple layers management**: Provide the capability to manage in same time several parallel processing for specific resources and models (i.e. simulation of camera, GPS, RADAR, and IR in same time with their own physical resources and requirements)

- **Time management**: Have the capabilities to provide an accurate mechanism of time management for orchestration/synchronisation of the various simulators and models. This function needs to generate real-time operating with a high level of repeatability (several same scenarios and simulations provide the same result with the same time stamping of the data). The time generator and manager needs to provide a large set of time modelling (see figure Time). It is essential to control the operating period and frequency of each sensors.

- **Event generation and management**: The simulation engine needs to implement event mechanisms and functions with specific conditions, relations, constraint, situations (spatial, temporal, semantic, climate, ...). Moreover, event variable are essential to provide an automated validation process with the coverage of a large set of values for significant parameters and variables under test or generated relevant situations under test.

- **Interfaces**: the need to interconnect different tools and models with one another has become a crucial need. It is with this intention that a general standard with the acronym FMI (Functional Mockup Interface) was created, for easing up the exchange of models and standardising the way of connecting and sequencing them. This interface needs to support the transfer of large amounts of information (video streaming, for example). At this moment, the more efficient and used library is DDS.

- **Plug-in mechanism**: The using of an architecture based on plug-ins is essential in order to obtain a highly dynamic and modular simulation platform. This aspect is crucial and essential in order to have the capability to load plug-in instances during the operating in real time.

- **Modular and distributed architecture**: This aspect is based on the 3 previous functionalities (plug-in, interface, and time management) and is essential to share the specific processing and component simulation for time consumption or specific resource using. This means that the different modules and plug-ins can run either on several process (multiple CPU/GPU and parallel processing), and/or several remote computer using a network of computers and an accurate synchronisation mechanism.

- **A efficient and dynamic script language**: The script language is essential in order to manage all the parameters, objects, components, filters, environments, conditions, events, times aspects ... of the simulation in real time. The script language is useful and essential for the real time and automated scenario management and execution. Some simulation platforms use their own propriety language but XML based, Python based, LUA based script languages seems to become a standard. Moreover Open-Scenario seems to become the standard for the scenario definition, generation, and management.

- **Reference and Ground truth generation**: provide the different plug-ins, functions, methods to generate references and ground truth essential and useful in the evaluation and validation process. The quality of the ground truth needs to be evaluated and labelled in order to guaranty its fidelity and its accuracy.

- **Large scale outdoor environment**: This aspect becomes more and more important. The question concerns the capability of the simulation environment to generate and to use in real time large outdoor environments in order to simulated long distance travels. this requirement is faced with the generation of a large quantity of resources having the constraint of being faithful to reality and operating in real time to power sensors on board a vehicle. Here it is no longer a question of producing a digital twin of a restricted area of a few square kilometres but of environments of several tens of square kilometres with a faithful rendering of the terrain and the road environment. Some work attempts to propose solutions. This is the case of [47] which proposes offers a photo-realistic terrain Simulation pipeline for unstructured outdoor environments.

- **Digital Twin of test benches**: In order to validate the fidelity of the simulation platform, it is mandatory to generate not only Digital Twin of real environment but also the virtual test benches using on these road environment (open road or controlled environment). For instance, it was the case in UGE for the rain bench and the crash facilities (foam dummy). In a next section, this Digital Shadows (DS) and Digital Twin (DT) are addressed and a generic framework is proposed for the generation of such a DT and DS. In PRISSMA, 2 new DS and DT have been developed (see figures 70 and 72)

In order to provide synthetic data usable in an evaluation and validation process, it is mandatory to address the synthetic data fidelity, quality, and diversity. These 3 concepts are related concepts but represent distinct aspects in the context of data generation:

- **Synthetic Data Fidelity**:
  - Definition: Fidelity refers to the accuracy with which synthetic data replicates the statistical properties and patterns of the original data. This aspect is link to the physical accuracy of the synthetic structure or data generated by a model. In this context, the generated synthetic data must be experimentally validated and must fit with characteristics of the real system, sensor, data. Generated samples resemble real samples from $\mathbb{P}r$ (real distribution of data). A high-fidelity synthetic data set should contain "realistic" samples, e.g. visually-realistic images.
  - Focus: It emphasises how closely the synthetic data mimics the key characteristics, distributions, and relationships present in the real dataset.

- **Synthetic Data Quality**:
  - Definition: Quality encompasses a broader range of characteristics and features that contribute to the overall usefulness and appropriateness of synthetic data for a specific purpose.
  - Focus: It considers factors beyond statistical accuracy, including the relevance, utility, and effectiveness of synthetic data in achieving specific objectives.

- **Synthetic Data Diversity**:

  - Definition: Synthetic data diversity refers to the variety and richness of information captured within a dataset generated through artificial means. It emphasises the representation of different patterns, characteristics, and scenarios to ensure a comprehensive and realistic reflection of the underlying real-world data. Generated samples are diverse enough to cover the variability of real data, i.e., a model should be able to generate a wide variety of good samples. Diversity could be link to the generalisation: Generated samples should not be mere copies of the (real) samples in training data, i.e., models that overfit to real datasets are not truly "generative".
  - Focus: The primary focus of synthetic data diversity is to create a dataset that mimics the complexities and variations present in the authentic data it aims to replace. This involves capturing a wide range of features, relationships, and distributions to enhance the model's adaptability and generalisation.

3.2.2 Vehicle dynamics model

In order to develop realistic dynamic vehicle model, it is often necessary to use Physical engine with solvers (ODE), ray-tracing library, and mechanic properties in order to manage physical interaction between several dynamic objects. Moreover, as presented in the figure 16, a dynamic vehicle modelling must involve at least this several modules:

- Multi-body dynamics model: represents the mechanical systems of the vehicle, including:
  - shock absorbers (with either linear, or non linear modelling (potentially with use of LUT))
  - chassis and car body (with aerodynamic coefficient)
  - Steering wheel column with auto alignment, driver, and controller torques
  - Wheels and Tires involving tire grid modelling (longitudinal and lateral)

  It simulates the movement, forces, and interactions between these components based on Newtonian physics principles. Car body and chassis use differential system, aerodynamic coefficient
- Braking system
- Powertrain model and drivetrain: simulates the engine, transmission, and other powertrain components, accounting for torque, gear ratios, fuel consumption, and performance characteristics. These models need to address thermal engine as well as hybrid or electric engine (involving battery modelling) with powertrain and gearbox (automatic or manual)
- main ADAS systems like EPS, ABS, ACC and Stop&Go

Moreover, as mentioned in the JAMA report [1], a set of disturbances must to be considered in order to obtain a physical realistic behaviour of a vehicle in realistic and constrain configurations. The figure 17 shows some of these disturbances applied in the ODD of a real vehicle.

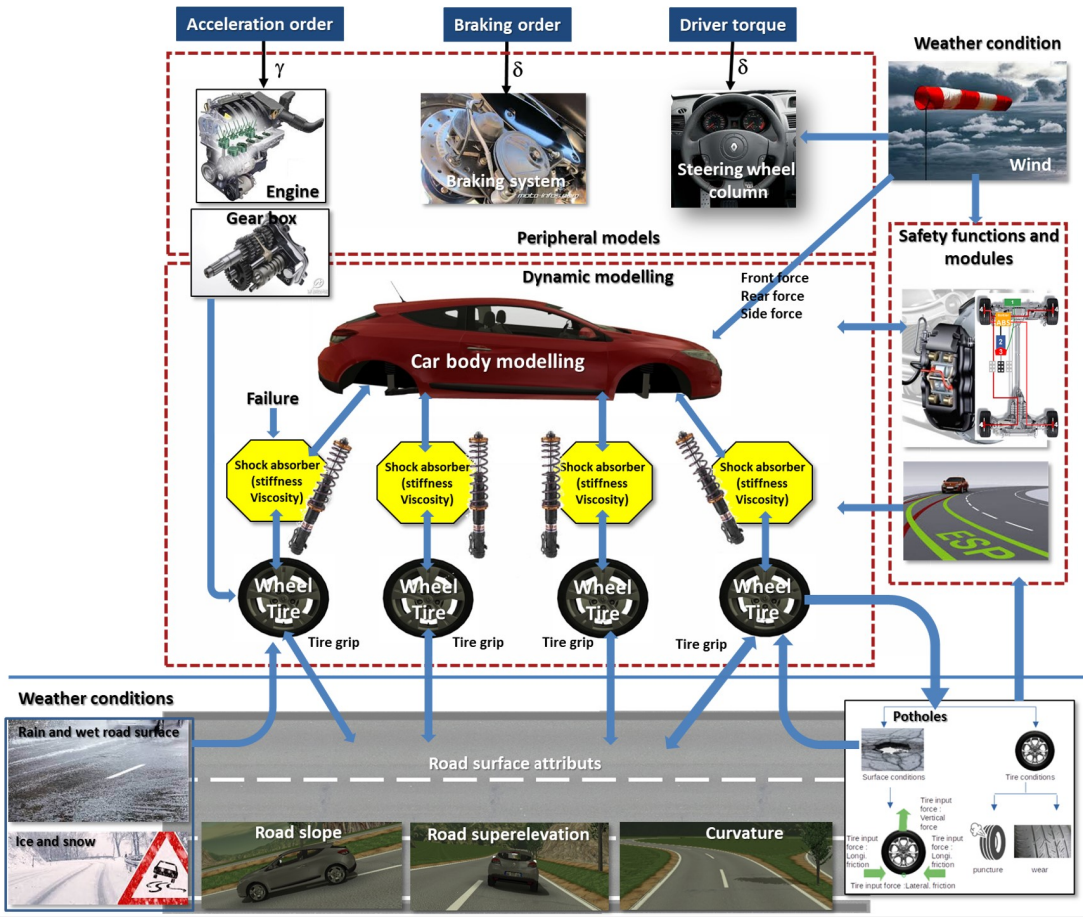


Figure 16: Different parts to consider in a realistic and dynamic vehicle modelling (source: UGE)

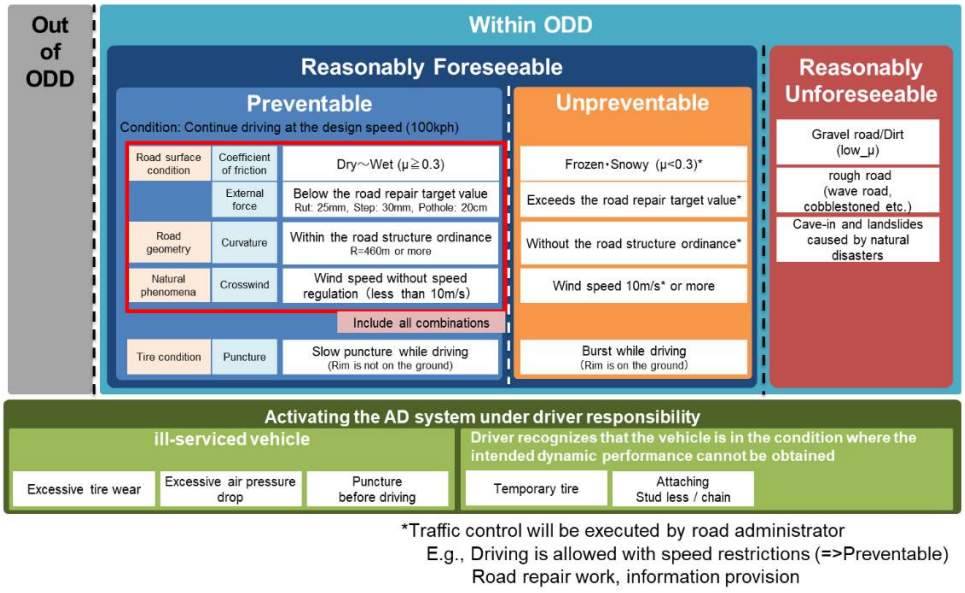


Figure 17: Preventability/Unpreventability boundary conditions in vehicle movement disturbance (source: JAMA's report [1])

3.2.2.1 Some academic dynamic vehicle part modelling

About Academic dynamic vehicle modelling, the PhD thesis of Laetitia Li (2021) [48] gives a good overview of car body, wheels, and tires modelling. In this work, cinematic models are firstly presented then after, a set of 4 different dynamic model are enumerated and presented:

- Dynamic bicycle modelling with 3 DoF
- Dynamic 4 wheels modelling with 7 DoF
- Dynamic 4 wheels modelling with 10 DoF
- Dynamic 4 wheels modelling with 14 DoF

About the tire grip modelling, the main well-known models are:

- Model of Brosse
- Model of Gim
- Model of Dugoff
- Model of Kiencke/Burckhardt
- Model of Pacejka (full and simplify (order 3))

In [49], an overview is given about the modelling of drive-train components:

- Combustion engine
- automatic gearbox
- Limited slip differential
- braking system
- Steering column

In the same work, the main stability controllers are presented:

- Anti-lock Braking System
- Traction Control System
- Electronic Stability Control

In [50], Sébastien Glaser propose a full enough modelling of the dynamic of a vehicle for the simulation in limit conditions. In this model implemented in Pro-SiVIC platform, the following parts are implemented:

- The car body using the fundamental principles of dynamics
- The shock absorbers
- the anti-roll bars
- The wheels and tire with Dugoff and Pacejka models

3.2.3 Environment Model and propagation channel

In order to generate a representative and relevant simulated environment, it is mandatory to take into account the main factors of the environment having an impact on the operating of the vehicles, the sensors, and the ADAS. The following list, though not exhaustive, identifies some of the most common disturbances to generate in the virtual environment with a high level of fidelity:

- **Extreme adverse weather conditions (heavy rain, snow, or fog)** : Simulates weather conditions (such as rain, snow, fog), lighting, visibility, and other environmental factors that influence sensors and vehicles performances. Weather conditions can strongly reduce the maximum range of a sensor and its signal quality (acuity, contrast, excessive visual clutter) for human vision, AV visual systems (cameras, LIDAR), and DSRC transmissions (though to a lesser extent).
  - *Mist and fog* are the result of the condensation of water vapour on atmospheric nuclei remaining in suspension close to the ground. Mist and fog are classified based on the visibility, or Meteorologic Optical Range (MOR). This latter corresponds to the length of path in the atmosphere over which the light, from a known source, is reduced to 5% of its original intensity. We talk about fog when the visibility is below 1 km. When the visibility is greater than 1 km we talk about mist. Fog droplets size range from a few tenths of a micron to a few tens of microns [51]. The visibility decreases with the increase of droplets number in the medium and also depends on the droplet size distribution [52]. A same visibility value can correspond to different droplets size distribution.
  - *Precipitations, including rain, snow, hail and sleet*, consist in liquid or frozen water drops falling to the ground. It is the result of water vapour condensation on particles in the colder atmosphere. The intensity of the precipitations is defined by the size and the distribution of the droplets in the medium. The visibility impairment due to precipitations is also related to the speed of fall of droplets which causes the appearance of streaks linked to the camera shutter. The diameter of raindrops can range between 0.1 to $\sim$ 6 mm. In scattering and absorption calculations, raindrop shape is often assumed to be spherical even though large droplets typical of heavy rains look like oblate spheroidal [53]. Unlike rain drops, snow grains cannot be considered as spherical. Their shape and size are complex and hence difficult to simulate numerically. Räisänen et al. constructed a reference phase function for the scattering of snow grains, taking into account the diversity of snow grains shape and size [54].
- **Excessive dirt or physical obstructions (such as snow or ice) on the vehicle**: Interferes with or reduces maximum range and signal quality (acuity, contrast, physical occlusion of field of view) for human vision and all basic AV sensors (cameras, LIDAR, radar).
  - *Dust, natural haze or smog* occur by the accumulation of particles of dust, smoke or any air pollutant, in relatively dry air. The extinction coefficient of haze is wavelength dependant. Zhang et al. (2021) [17] pointed out an adverse weather condition typical in East Asia during spring months, named Asian dust, which is made of mineral dust from crustal sources transportation from desert areas eastward.

- **Darkness, low illumination, and specific light conditions**: Reduces maximum range and signal quality (acuity, contrast, possible glare from external light sources) for human vision and AV camera systems. The sunset can generate IR signal which could disturb the IR sensor signal and perception performance.
- **Terrain Elevation and Obstructions**: Integrate accurate elevation data and obstructions such as buildings, trees, and other obstacles to simulate line-of-sight (LOS) and non-line-of-sight (NLOS) signal paths. Large physical obstructions (buildings, terrain, heavy vegetation, etc.) interferes with line of sight for human vision and all basic AV sensors (cameras, radar, LIDAR); some obstructions can also reduce the maximum signal range for DSRC.
- **Dense traffic**: Interferes with or reduces line of sight for human vision and all basic AV sensors (cameras, radar, LIDAR); can also interfere with effective DSRC transmission caused by excessive volumes of signals/messages. (However, human drivers do have some limited ability to see through the windows of adjacent vehicles.). Moreover, it is more difficult to detect and discriminate the close objects with the same dynamic (same speed and same heading).
- **Small objects**: Difficult discrimination/detection/identification of the small objects like negative obstacle (pothole) or positive objects like little animal (fox, cat, dog) or static object (piece of wheel tire, or bumper). In the second case, the IR camera could not make the difference between the road surface and the object.

Moreover, the 3D environment needs to reproduce the following information:

- **Realistic Geographical Data**: Utilise high-quality geographical data, including terrain, buildings, vegetation, and other environmental features, to accurately model the physical 3D environment. this requirement involves the road and ground spatial configuration representing the road surface and terrain features, including elevation changes, curvature, friction characteristics, and surface irregularities affecting vehicle dynamics.
- **Maps, materials, and textures**: The environment needs to use materials, textures, shadows, energy intensities map, temperature and radiosity map in order to fed the different sensors technologies. Even with the best intrinsic modelling of a sensors, without the same quality for the propagation channel and the road environment, the signal coming from the sensors will be weak and not faithful enough to reality and to the signals that can be generated by real sensors.
- **Multi-Path Modelling**: Incorporate models for simulating multi-path effects caused by signal reflections, diffractions, and scattering from various objects in the environment.
- **Atmospheric Effects**: Model atmospheric conditions, including temperature, pressure, humidity, and air density, to simulate their impact on signal propagation through the atmosphere. This information is useful in order to fed the GPS model. For the camera, this information allows to generate in far distance the effect of the atmospheric veil. In these atmospheric effects, we can add the wind which could interact with the vehicle and generate a force impacting the speed and consumption of the vehicle.

- **Propagation Loss Models**: Implement propagation loss models to calculate signal attenuation over distance, considering factors like free space loss, path loss, and environmental obstructions.
- **Signal Fading and Shadowing**: Simulate signal fading and shadowing effects caused by obstacles blocking or interfering with the direct path between transmitter and receiver.
- **Frequency and Bandwidth Considerations**: Account for signal frequency and bandwidth characteristics to accurately model signal behaviour in different frequency bands and under varying bandwidth conditions.
- **Time-Variant Channel Modelling**: Incorporate time-variant channel models to simulate dynamic changes in the propagation environment over time, such as signal fading, interference, and channel capacity fluctuations.

By meeting these requirements, a simulation environment can provide an accurate representation of the environment and propagation channel, enabling thorough testing and evaluation of wireless communication systems and devices. The figure 19 summarises the signal propagation channel with a large set of potential effects on the sensors. The simulation of a road environment needs to reproduce with the more efficient and realistic way these phenomena.

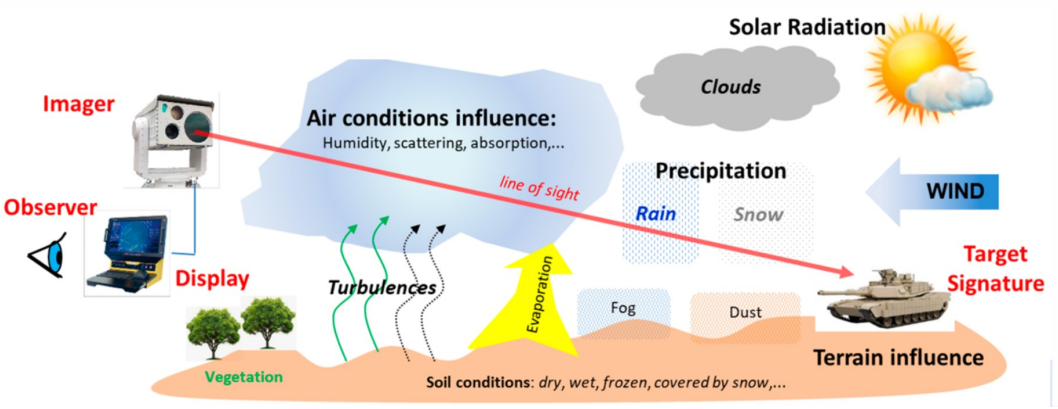


Figure 18: Atmospheric disturbances impacting the visibility and the sensors operating

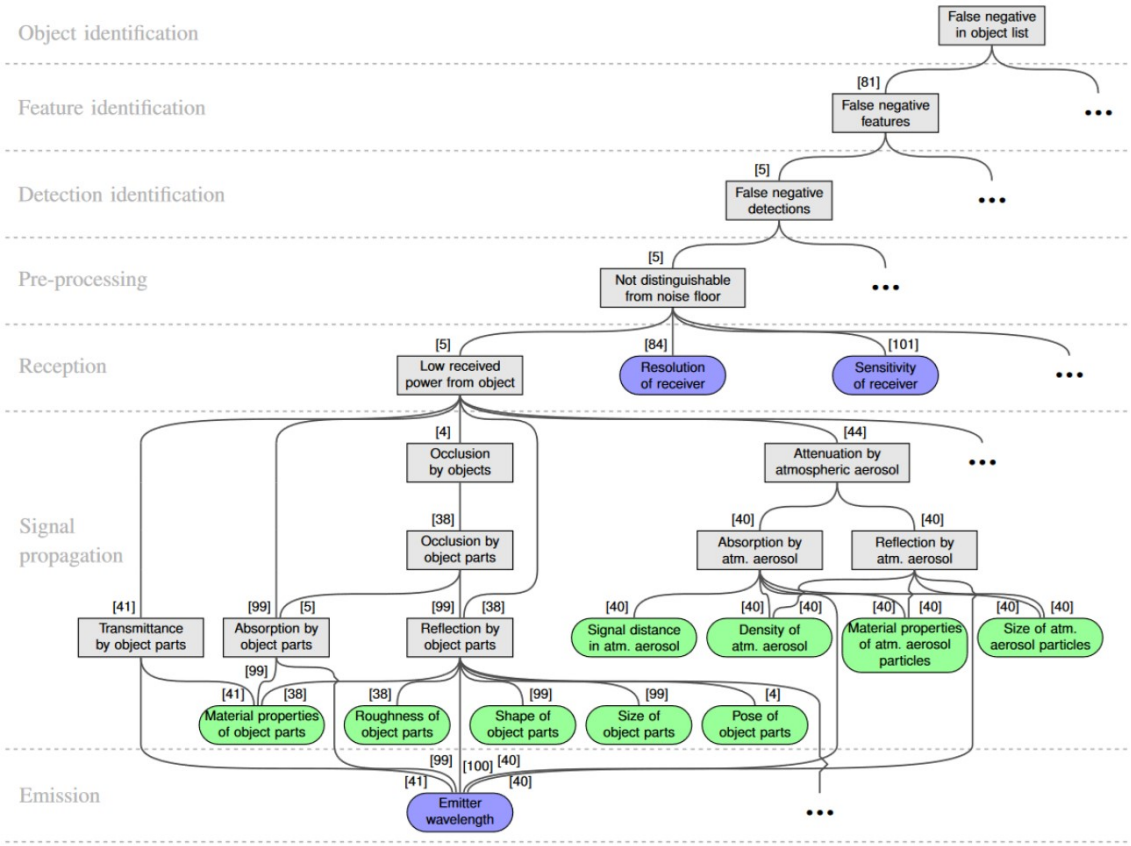


Figure 19: Different layers and phenomena impacting the propagation of a signal and by extension the efficiency of the object detection and identification by a sensor and a AI-based perception system. Grey rectangles provide the effects, green rounded rectangles the system independent causes, and the blue rounded rectangles give the design parameter causes (source: [2])

3.2.4 Exteroceptive Sensor Models

3.2.4.1 LIDAR technologies and model requirements

LIDAR model: simulates Light Detection and Ranging sensors that use laser pulses to measure distances and create 3D point clouds, commonly used for perception in autonomous driving simulations.

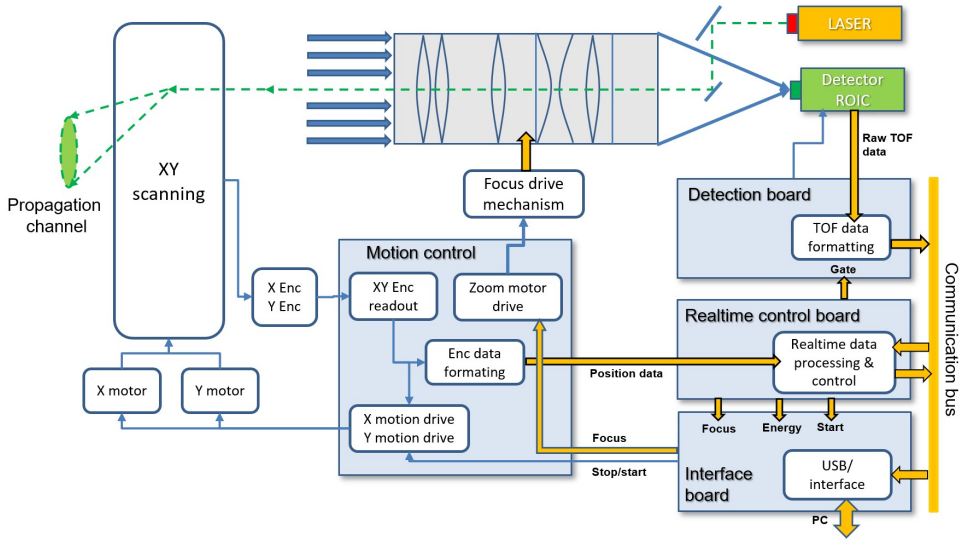


Figure 20: Different functions and modules in the scanning LIDAR technology (source: UGE)

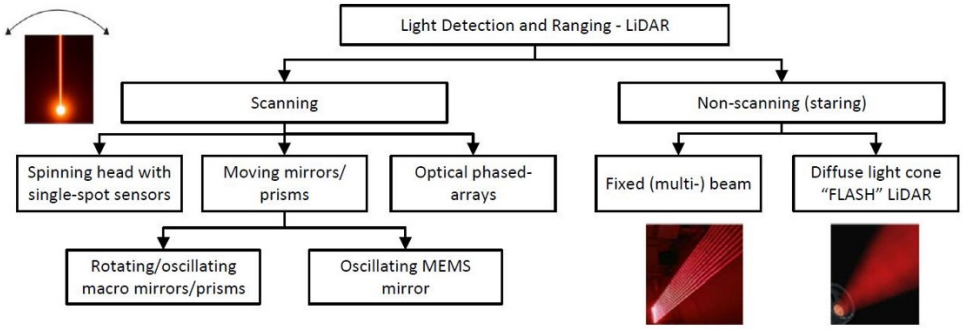


Figure 21: Fundamental classification of various LIDAR concepts (source: [3])

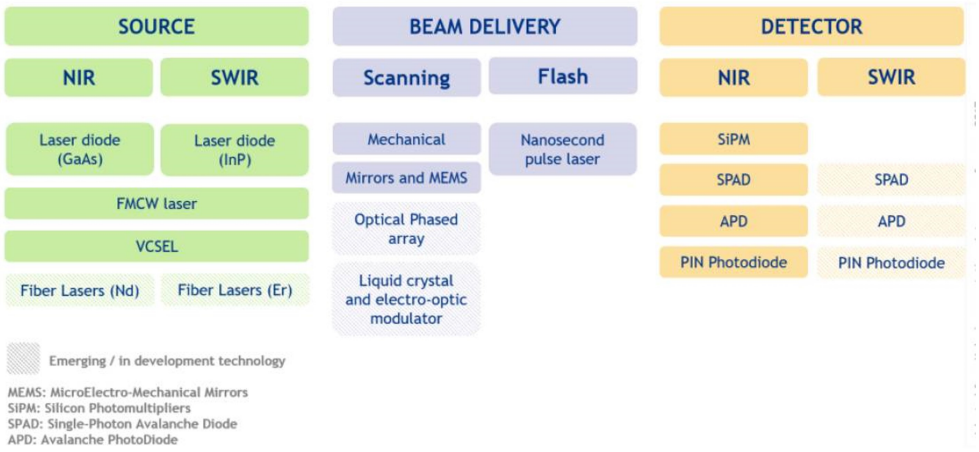


Figure 22: LIDAR technologies for automotive domain and AV (source: https://tematys.fr/Publications/fr/29-sensors)

Parameter	MQ-8-Plus	MQ-8-PoE-Plus	MQ-8-PoE-Ultra
Laser Class	Class I Laser Product (Eye Safe, IEC 60825-1)		
Wavelength	905nm		
Measurement Technique	Time of Flight (TOF)		
Minimum Sensor Range	2m		
Maximum Sensor Range	150m (80% reflectivity) 50m (10% reflectivity)		200m (80% reflectivity) 70m (10% reflectivity)
Range Accuracy (1σ at 50m)	<3cm		
Frame Rate	5-20Hz		
Angular Resolution	0.03°-0.13° dependent on frame rate		
Detection Layers	8		
Field of View (FOV)	Horizontal: 360°, Vertical: 12.4° (-1.6°/-14°)		
Output Connection	M12 Connector: 100/1000Mbps Ethernet, NMEA/PPS, Power	RJ45 802.3at (PoE+)	
Data Outputs	Angle, Distance, Intensity, Time Stamps (synchronized to GPS when available)		
Returns	Up to 3		
Output Rate	430,000 points per second (1 return), 1.3M points per second (3 returns)		
Nominal Power	16W	18W	
Operating Voltage	9-30VDC	42.5-57VDC	
Operating Temperature	-20°C to +60°C (-4°F to +140°F)		
Storage Temperature	-40°C to +105°C (-40°F to +220°F)		
Nominal Weight	900g	1375g	
Dimensions	103mm (D) x 87mm (H)	115mm (D) x 134mm (H)	
Shock & Vibration	ETSI EN 300 019-2-5, IEC Class 5M3		
Environmental Protection	IP69K	IP67	
Certifications and Compliance	FDA, FCC, CE, RoHS, WEEE		

Figure 23: LIDAR parameters in use - Paris2Connect

Simulating a LIDAR system for embedded vehicles involves capturing the key characteristics of LIDAR technology while ensuring realism and accuracy. Here are the main requirements for creating a realistic LIDAR model:

- **Laser Emission and Reflection Modelling** : Accurate modelling of laser emission and reflection processes to simulate how LIDAR sensors emit laser beams and detect their reflections from surrounding objects. This aspect involve the wave length of the laser beams and the number of shot by beam. Last generation of LIDAR also apply signal modulation similar to FMCW in order to limit the impact of distances in the propagation channel.
- **Sensor Geometry**: Modelling of the physical geometry of the LIDAR sensor, including the number and arrangement of laser emitters and receivers. This ensures that the simulated LIDAR sensor closely matches the real-world sensor configuration. For instance, the laser impacts arrangement of a IBEO, Velodyne, of OUSTER sensors could be different.
- **Scanning Patterns**: Simulation of scanning patterns used by LIDAR sensors to sweep laser beams across the environment. This includes modelling rotational or oscillating movements to capture a 360-degree field of view. This aspect is in relation with the previous requirement.
- **Range Measurement**: Implementation of algorithms to calculate range measurements based on the time-of-flight or phase-shift of laser pulses. Accurate range measurement is essential for determining the distance to objects in the LIDAR's field of view. This requirement depends of the beam power and the material reflection level (flat or granular, clear of dark material).
- **Resolution and Accuracy**: Modelling of the spatial resolution and accuracy of the LIDAR sensor, which determines its ability to detect and distinguish objects with high precision. This involves considering factors such as beam divergence, noise, and sensor calibration.

- **Point Cloud Generation**: Generation of point clouds representing the 3D spatial distribution of objects in the LIDAR's field of view. This involves processing the raw LIDAR data to generate a digital representation of the environment.
- **Environmental Factors**: Consideration of environmental factors that affect LIDAR performance, such as ambient light, weather conditions, and surface reflectivity. Accurate modelling of these factors is crucial for realistic simulation results.
- **Real-time Operation**: Optimisation of the LIDAR simulation model for real-time operation, ensuring that it can run efficiently within the computational constraints of embedded vehicle systems.

By addressing these requirements, a simulated LIDAR model can provide valuable insights into the performance and capabilities of LIDAR-based perception systems for embedded vehicle applications, including autonomous driving, collision avoidance, and environmental mapping.

Regarding the new generation of LIDAR systems, several noteworthy developments are underway:

- Firstly, MEMS-based LIDAR products have been under development for approximately a decade and were expected to be launched before 2020. These systems are anticipated to experience significant growth provided the right balance between cost, performance, and manufacturability is achieved. MEMS LIDAR systems employ tiny mirrors whose tilt angle varies in response to a stimulus, such as voltage. These mirrors substitute mechanical scanning hardware with an electromechanical equivalent. However, aligning multiple mirrors for multidimensional laser beam movement poses challenges, particularly susceptibility to shocks and vibrations encountered in moving vehicles. Additionally, automotive specifications necessitate operation down to -40°C, which can be demanding for MEMS devices.
- Concurrently, flash LIDARs are in development, operating akin to standard digital cameras utilizing an optical flash. In flash LIDAR, a single large-area laser pulse illuminates the environment, while a focal plane array of photodetectors captures back-scattered light. This method captures the entire scene in a single image, enabling faster data capture rates and immunity to vibration effects. However, the presence of retroreflectors in the environment can blind the sensor, and high peak laser power is required for illumination over longer distances. Despite these drawbacks, flash LIDARs offer advantages such as absence of moving parts, compactness, and cost-effectiveness.
- In the short term, Optical Phased Arrays (OPA) technology is anticipated to challenge MEMS and flash LIDARs. Based on Photonics Integrated Circuits (PIC), OPA steers the beam without moving parts, offering cost-effectiveness and high performance. However, OPA technology requires further maturation, with ongoing developments focused on improving range and resolution specifications.
- Additionally, techniques such as Frequency-Modulated Continuous Wave (FMCW) and liquid crystal beam steering are being monitored for long-term prospects. FMCW, utilizing a coherent method with frequency-modulated laser light, offers simpler optics and computational load compared to traditional Time-of-Flight (ToF) methods.

- Looking ahead, the Short-Wave Infrared (SWIR) range (1550 nm) is expected to witness substantial growth due to its wider range and superior performance in adverse weather conditions. The main challenge currently lies in the cost of SWIR detectors, but promising technologies such as quantum dots detectors are already in development.

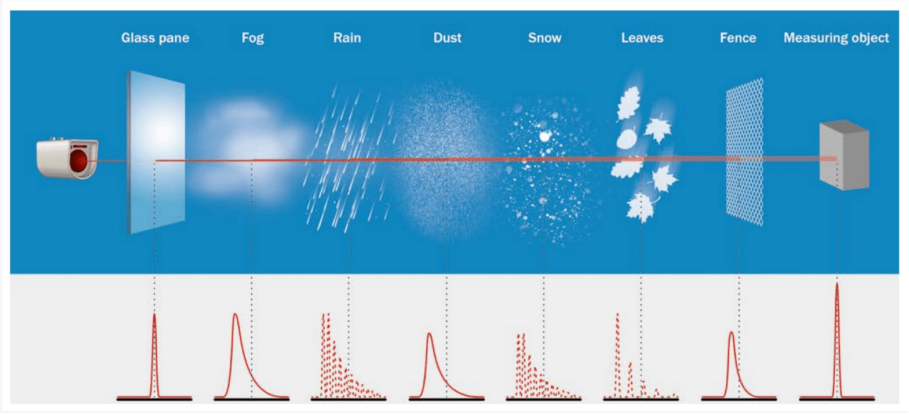


Figure 24: LIDAR interaction with disturbers in the environment ([4])

	Company	Model	Channels or Layers	FPS (Hz)	Acc. (m)	RNG (m)	VFOV (°)	HFOV (°)	HR (°)	VR (°)	λ (nm)	Ø (mm)
Mechanical/Spinning LiDARs	Velodyne	VLP-16	16	5–20	±0.03	1…100	30	360	0.1–0.4	2	903	103.3
		VLP-32C	32	5–20	±0.03	1…200	40	360	0.1–0.4	0.33 [1]	903	103
		HDL-32E	32	5–20	±0.02	2…100	41.33	360	0.08–0.33	1.33	903	85.3
		HDL-64E	64	5–20	±0.02	3…120	26.8	360	0.09	0.33	903	223.5
		VLS-128 Alpha Prime	128	5–20	±0.03	max 245	40	360	0.1–0.4	0.11 [1]	903	165.5
	Hesai	Pandar64	64	10,20	±0.02	0.3…200	40	360	0.2,0.4	0.167 [1]	905	116
		Pandar40P	40	10,20	±0.02	0.3…200	40	360	0.2,0.4	0.167 [1]	905	116
	Ouster	OS1–64 Gen 1	64	10,20	±0.03	0.8…120	33.2	360	0.7,0.35,	0.53	850	85
		OS1-16 Gen 1	16	10,20	±0.03	0.8…120	33.2	360	0.17	0.53	850	85
	RoboSense	RS-Lidar32	32	5,10,20	±0.03	0.4…200	40	360	0.1–0.4	0.33 [1]	905	114
	LeiShen	C32-151A	32	5,10,20	±0.02	0.5…70	32	360	0.09,	1	905	120
		C16-700B	16	5,10,20	±0.02	0.5…150	30	360	0.18,0.36	2	905	102
	Hokuyo	YVT-35LX-F0	-	20 [3]	±0.05 [3]	0.3…35 [3]	40	210	-	-	905	◊
Solid State LiDARs	IBEO	LUX 4L Standard	4	25	0.1	50 [2]	3.2	110	0.25	0.8	905	◊
		LUX HD	4	25	0.1	50 [2]	3.2	110	0.25	0.8	905	◊
		LUX 8L	8	25	0.1	30 [2]	6.4	110	0.25	0.8	905	◊
	SICK	LD-MRS400102S01 HD	4	50	-	30 [2]	3.2	110	0.125…0.5		-	◊
		LD-MRS800001S01	8	50	-	50 [2]	6.4	110	0.125…0.5		-	◊
	Cepton	Vista P60	-	10	-	200	22	60	0.25	0.25	905	◊
		Vista P90	-	10	-	200	27	90	0.25	0.25	905	◊
		Vista X90	-	40	-	200	25	90	0.13	0.13	905	◊

Figure 25: LIDAR Parameters for the main type of automotive LIDAR [5]

3.2.4.2 RADAR technologies and model requirements

Radar model: emulates Radar sensors that use radio waves for object detection and speed measurement.

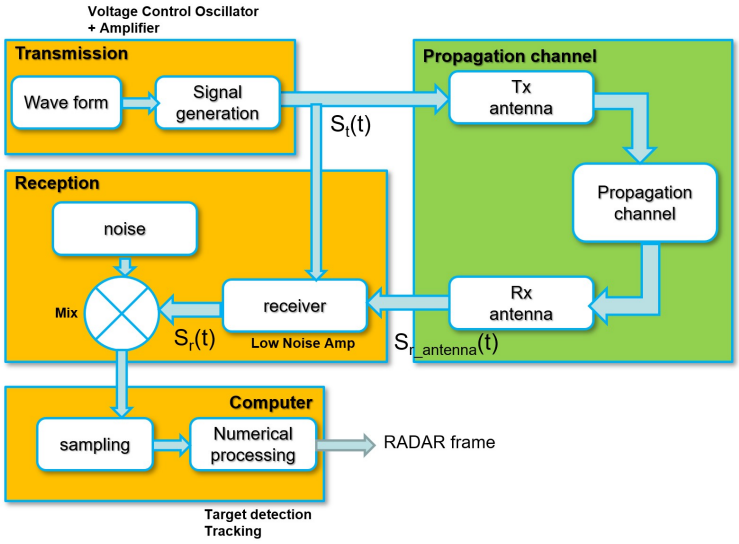


Figure 26: RADAR functions in a Digital Twin of the sensor (Source: UGE)

Creating a realistic simulation of an automotive RADAR system involves addressing various aspects to accurately replicate its behaviours and outputs. Here are the main requirements for simulating an automotive RADAR and obtaining a realistic model:

- **Antenna and Transceiver Modelling** : Accurate modeling of the RADAR antenna and transceiver components, including their physical characteristics, radiation patterns, and transmission/reception properties. This ensures that the simulated RADAR system behaves realistically in capturing and processing radio frequency (RF) signals.
- **Frequency, Bandwidth, and modulation**: Specification of the RADAR's operating frequency and bandwidth, which determine its sensitivity, resolution, and range capabilities. The simulated RADAR model should accurately replicate the frequency modulation and bandwidth constraints of real automotive RADAR systems. In the automotive domain, the main frequencies are either around 24 GHz or around 76 and 79 GHz. Moreover, a large set of modulations and waveforms exists like Frequency Continuous Modulated wave (FMCW). Moreover, different types of technologies exist like SRR (Short Range Radar), MRR (Middle Range Radar) and LRR (Long Range Radar). Each technology impact the field of view of the RADAR. For instance, a LRR currently have a 20° FOV with a long detection range (up to 200 and 300 meters) while a SRR has a 60° FOV with a shorter range.
- **Signal Propagation Modelling**: Modelling of signal propagation effects such as attenuation, reflection, diffraction, and scattering, which influence the behaviour of RF signals in the environment. Accurate signal propagation modelling is essential for realistic RADAR simulations in various driving scenarios.

- **Target Detection and Tracking**: Implementation of algorithms for target detection, tracking, and localisation based on RADAR measurements. This involves processing RADAR return signals to identify and track objects in the vicinity of the vehicle, including vehicles, pedestrians, and other obstacles.
- **Range and Doppler Measurements**: Generation of range and Doppler measurements from RADAR return signals, which provide information about the distance, relative velocity, and motion characteristics of detected objects. Accurate range and Doppler measurements are critical for effective object detection and tracking.
- **Clutter and Noise Simulation**: Simulation of background clutter and noise in RADAR return signals, including environmental noise, interference from other RF sources, and electronic noise from the RADAR system itself. Realistic clutter and noise simulation ensure robust performance of RADAR signal processing algorithms.
- **Environmental Factors**: Consideration of environmental factors that affect RADAR performance, such as weather conditions, road surface properties, and surrounding terrain. Accurate modelling of these factors allows for realistic RADAR simulations under various driving conditions.
- **Integration with Vehicle Dynamics** : Integration of the RADAR simulation model with vehicle dynamics and control systems to enable realistic interaction between RADAR measurements and vehicle motion. This allows for the simulation of RADAR-based collision avoidance, adaptive cruise control, and other driver assistance functions.

By fulfilling these requirements, a simulated automotive RADAR model can provide valuable insights into the performance, capabilities, and limitations of RADAR-based sensing systems for automotive applications, including advanced driver assistance systems (ADAS) and autonomous driving technologies. It is important to mention the development of the new generation of RADAR like the RADAR imaging. The figure 27 presents the comparison of 3 of the main automotive RADAR technologies.

	Aptiv Delphi ESR 2.5	Aptiv Delphi SRR2	Continental ARS 408-21	SmartMicro UMRR-96 T-153 [1]
Freq (GHz)	76.5	76.5	76 … 77	79 (77 … 81)
HFOV (°) Short-Range Mid-Range Long-Range	 ±45 ±10	 ±75	 ±9 ±60	 ≥130 ≥130 ≥100 (squint beam)
VFOV (°) Short-Range Long-Range	 4.4	 10	20 14	 15
Range (m) Short-Range Mid-Range Long-Range	1–60 1–175 [2]	0.5–80 [2]	0.2–70/100 0.2–250	0.15–19.3 [3] 0.4–55 [3] 0.8–120 [3]
Range Acc (m) Short-Range Mid-Range Long-Range	-	±0.5 noise and ±0.5% bias	-	<0.15 or 1% (bigger of) <0.30 or 1% (bigger of) <0.50 or 1% (bigger of)
Vel Range (km/h) Short-Range Mid-Range Long-Range	-	-	-400 … +200 [4]	−400 … +140 [4] −340 … +140 [4] −340 … +140 [4]
IO Interfaces	CAN/Ethernet [5]	PCAN	CAN	CAN/Automotive Ethernet
ROS Drivers	[101,102]		[103]	[104]
Reference	[51,105–109]		[110–112]	[113]

[1] It is recommended to use PCAN-USB adapter from PEAK System for connections of Controller Area Network (CAN) to a computer via Universal Serial Bus (USB) [114]. [2] Range indicated for ESR 2.5 (long-range mode) and SRR2 is measured at 10dB and 5 dB, respectively. [3] Range may vary depending on the number of targets in the observed environment and will not achieve a 100% true-positive detection rate. [4] A negative velocity range indicates the object is moving away from the radar (opening range) and a positive value indicates the object is moving toward the radar (closing range) [115]. [5] Internet Protocol (IP) address specified on request with a sale unit and is not modifiable by user [116].

Figure 27: General specifications of main automotive RADAR sensors (from SmartMicro, Continental and Aptiv Delphi). The parameters presented are frequency (Freq), horizontal FoV (HFOV), vertical FoV (VFOV), range accuracy (Range Acc), velocity range (Vel Range), input/output interfaces (IO Interfaces) and ROS (Robotic Operating System) drivers. (source: [5])

Waveform Type	Transmit Waveform s(t)	Detection Principle	Resolution	Comments
CW	$e^{j2\pi f_c t}$	Conjugate mixing	$\triangle f_d = 1/T$	No range information
Pulsed CW	$\Pi(T_p)\, e^{j2\pi f_c t}$	Correlation	$\triangle R = cT_p/2$ $\triangle f_d = 1/T_p$	Range-Doppler performance tradeoff
FMCW	$e^{j2\pi(f_c+0.5Kt)t}$, $K = \frac{B}{T_0}$	Conjugate mixing	$\triangle R = c/2B$ $\triangle f_d = 1/PT_0$	Both range and Doppler information
SFCW	$e^{j2\pi f_n t}$, $f_n = f_c + (n-1)\triangle f$	Inverse Fourier transform	$\triangle R = c/2B$ $\triangle f_d = 1/PT_0$	$\triangle f$ decides maximum range
OFDM	$\sum_{n=0}^{N-1} I(n)\, e^{2\pi(f_c+n\triangle f)t}$	Frequency domain channel estimation	$\triangle R = c/N\triangle f$ $\triangle f_d = 1/PT_N$	Suitable for vehicular communication

B denotes bandwidth of the radar. T is the amount of time for which data is captured.
N stands for a number of samples in CW and number of carriers in OFDM.
$\Pi(T_p)$ is rectangular pulse of duration T_p. P is number of FM/SF-CW or OFDM blocks of duration T_0 and T_N, respectively.
$I(n)$ is arbitrary sequence and $\triangle f$ is carrier/frequency separation in OFDM/SFCW.

Figure 28: Main automotive RADAR waveforms (source: [6])

The RADAR waveforms, outlined in figure 28, are categorised based on whether they are continuous wave (CW), pulsed, and frequency or phase modulated. Modulated radar waveforms include FM CW, stepped frequency (SF) CW, orthogonal frequency-division multiplexing (OFDM), and frequency shift keying (FSK). Each waveform type offers specific advantages in processing, implementation, and performance:

- CW radar utilises conjugate mixing of a high-frequency transmitted and received signal to

produce the output signal at the Doppler frequency of the target. Although it easily detects target speed, the continuous waveform lacks round-trip delay measurement necessary for range estimation.

- Pulsed CW radar, on the other hand, estimates range information by lengthening each pulse and measuring the frequency difference between transmitted and received pulses. Pulse duration and pulse repetition frequency (PRF) are critical parameters for designing pulsed CW radar with desired range and velocity resolution.
- Frequency Modulated Continuous Wave (FMCW), also known as linear frequency modulation (LFM) or chirp, enables simultaneous range and velocity estimation due to pulse compression. With FMCW radar, range resolution is inversely proportional to signal bandwidth, and Doppler resolution is determined by pulse-width and the number of pulses used for estimation.
- Frequency Shift Keying (FSK) and Stepped Frequency Continuous Wave (SFCW) waveforms vary in a discrete manner, offering a unique approach where range profiles of targets are derived from data collected at discrete frequencies. Hybrid waveform types, such as combining FSK with multislope FMCW, enhance radar processing performance.
- Orthogonal Frequency Division Multiplexing (OFDM) waveform provides joint implementation capabilities for automotive radar and vehicle-to-vehicle communications. OFDM radar processing estimates range profiles through frequency domain channel estimation.

Optimisation of radar waveforms can be achieved based on target statistics. But it is necessary to keep in mind that the accurate modelling and simulation of intrinsic signal (waveform) generation and processing needs to use the capacities of GPU and some specific signal libraries in order to handle the real signal high frequencies.

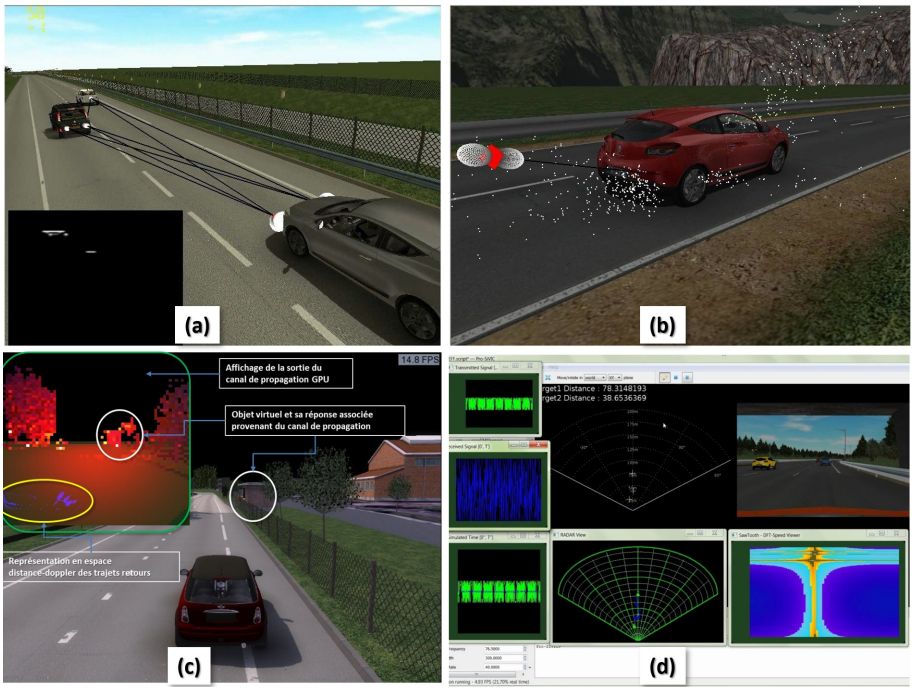


Figure 29: The 3 different levels of RADAR modelling implemented in ProSIVIC ®. (Source: UGE and ESI group)

In Figure 29, (a) presents 2 simplified models of RADAR. First one with Depth map, voting method with objects impacted by RADAR signal, multi-object tracking generating a Continental RADAR message, second one with ray-tracing (transmitted power, length of path and multiple path), transmitter/receiver modelling with antenna; (b) A more complex modelling with ray-tracing, RCS, and Transmitter/receiver modelling; (c) A physical modelling of the propagation channel (green box in the figure 26) an illumination of the environment allowing multi-reflections (waveguide effect), energy concentrations, scattering, absorption. This level of modelling also simulates the Doopler effect; (d) Realistic physical simulation of the sensor (electronic and signal processing part presented in orange boxes of the figure 26).

Depending on the objective, the simulation of the sensor can be more or less complex and detailed. We can therefore have simulation levels like those that exist on the Prosivic platform for example:

1. **Simplified physics-based model**: the model is simulating the Field Of View (FoV) in azimuth and elevation, the targets' distances and speeds without considering the radar signal processing block chain. The outputs of this sensor are ideal targets with appropriate distance and speed.

2. **Physics based radar**: this kind of model considers radar antenna diagram of single input single output (SISO) RADAR, Single-Input multiple Output SIMO, and multiple-input Multiple Output (MIMO) output and the bumper effect on radar. The simulation considers the Radar Cross Section (RCS) of targets to estimate electromagnetic wave energy and it includes the signal processing tool chain (signal modulation, Fourier transform and tracking). This type of sensor can perform real-time simulations depending on the scenario. The simulation outputs are typically those of a real radar (targets list) with appropriate distance, speed, angles, and energy reaching and departing from objects in the propagation channel

3. **High fidelity physics radar**: the model is a very complex model of the RADAR with more detailed modelled physics. It is dedicated to more advanced radar simulation to study physics phenomena such as the effects of the environment and the interference between multiple targets and other radars.

3.2.4.3 Camera technologies and model requirements

Camera model: simulates cameras for visual perception and computer vision tasks, including object recognition, lane detection, and traffic sign recognition. Of course, the models can be adapted to suit all different camera technologies (cyclop, infrared, RGB, fisheye, event-based camera, V2X cam edge computing...).

						Depth Information					
	Model	Baseline (mm)	HFOV (°)	VFOV (°)	FPS (Hz)	Range (m)	Img Res (MP)	Range (m)	Res (MP)	FPS (Hz)	Ref
Roboception	RC Visard 160	160	61 *	48 *	25	0.5–3	1.2	0.5–3	0.03–1.2	0.8–25	[40,41]
Carnegie Robotics®	MultiSense™ S7 [1]	70	80	49/80	30 max	-	2/4	0.4 min	0.5–2	7.5–30	[40,42,43]
	MultiSense™ S21B [1]	210	68–115	40–68	30 max	-	2/4	0.4 min	0.5–2	7.5–30	[40,44]
Ensenso	N35-606-16-BL	100	58	52	10	4 max	1.3		-		[40,45]
Framos	D435e	55	86	57	30	0.2–10	2	0.2 min	0.9	30	[40,46]
Nerian	Karmin3 [2]	50/100/250	82	67	7	-	3	0.23/0.45/ 1.14 min	2.7	-	[40,47]
Intel RealSense	D455	95	86	57	30	20 max	3	0.4 min	≤1	≤90	[40,48]
	D435	50	86	57	30	10 max	3	0.105 min	≤1	≤90	
	D415	55	65	40	30	10 max	3	0.16 min	≤1	≤90	
Flir®	Bumblebee2 [3]	120	66	-	48/20	-	0.3/0.8		-		[40,49]
	Bumblebee XB3 [3]	240	66	-	16	-	1.2				[50,51]

[1] HFOV, VFOV, image resolutions, image frame rates and depth information depend on the variant of focal length (optical lens geometry). [2] Specifications stated are in full resolution and monochrome, focusing on the standard 4 mm lens. [3] Offers either 2.5 mm, 3.8 mm or 6 mm lenses (specifications focus on 3.8 mm lens) but product no longer being produced or offered (discontinued). * A 6 mm lens has a HFOV of 43° and a VFOV of 33°.

Figure 30: General specifications of stereo cameras from various manufacturers. The parameters provides are horizontal field-of-view (HFOV); vertical field-of-view (VFOV); frames per second (FPS); image resolutions in megapixels (Img Res); depth resolutions (Res); depth frames per second (FPS) (Source: [5])

Creating a realistic simulation of a camera involves addressing various aspects to accurately replicate its behaviour and output. 2 sets of important requirements allowing to model a camera are provided in the following part. The first set of requirement is dedicated to the intrinsic modelling of the sensors:

- **Intrinsic Geometric Modelling** : Accurate representation of the camera's physical geometry, including lens characteristics, focal length, aperture size, sensor size, and lens distortion parameters. This ensures that virtual scenes are captured and projected realistically.
- **Intrinsic Sensor pixels matrix Simulation**: Modelling of the camera sensor's characteristics, such as pixel resolution, colour sensitivity, noise level, dynamic range, and pixel response function. This allows for the generation of realistic sensor data, including raw images and sensor noise.
- **Optical Effects Simulation**: Simulation of optical effects such as lens flare, glare, diffraction, vignetting, and depth of field. These effects add realism to rendered images and mimic the behaviour of real-world camera lenses.
- **Rendering Techniques**: Utilisation of advanced rendering techniques such as ray tracing or rasterization to accurately simulate the image formation process within the camera. This ensures that rendered images closely match the output of real cameras.
- **Camera Control**: Integration of camera control mechanisms to mimic real-world camera operations, including adjustments to exposure settings (e.g., shutter speed, aperture, ISO), focus distance, and white balance. This allows users to simulate different camera configurations and capture varied scenes.
- **Post-Processing Effects**: Implementation of post-processing effects such as tone mapping, colour grading, lens effects, and image stabilisation. These effects enhance the visual fidelity of rendered images and emulate the final output of real camera systems.

The second set of requirement is focused on the environment and extrinsic parameters:

- **Extrinsic Lighting Simulation**: Integration of lighting simulation techniques to accurately simulate how sun light and light sources interact with virtual scenes and objects. This includes simulating local and global illuminations, reflections, refractions, shadows (cast, catch, self-shadowing), and ambient occlusion (see figure 71) to achieve realistic lighting conditions. Ligth intensity can be encoded in HDR texture (see figure 35)
- **Material Properties**: Implementation of realistic material properties for objects within the virtual scene, including surface reflectance, texture mapping (seamless, procedural, moving textures), specular highlights, and roughness (bump mapping, ...). This contributes to the overall realism of rendered images captured by the simulated camera.
- **Dynamic Scene Interaction**: Support for dynamic scene interaction, enabling the simulation of moving objects (deformable mesh, articulated mesh parts, dynamic models and physical engine), changing lighting and weather conditions (rain, snow, fog, clouds, cloud of dust, ...), and evolving environments. This ensures that the simulated camera can capture realistic scenes with dynamic elements.

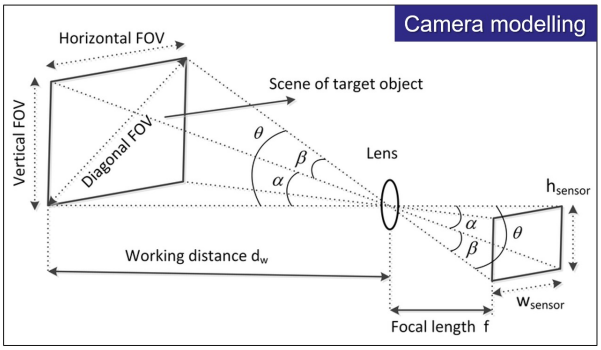


Figure 31: Camera model with the matrix size, the lens, the focal length, and the FOV

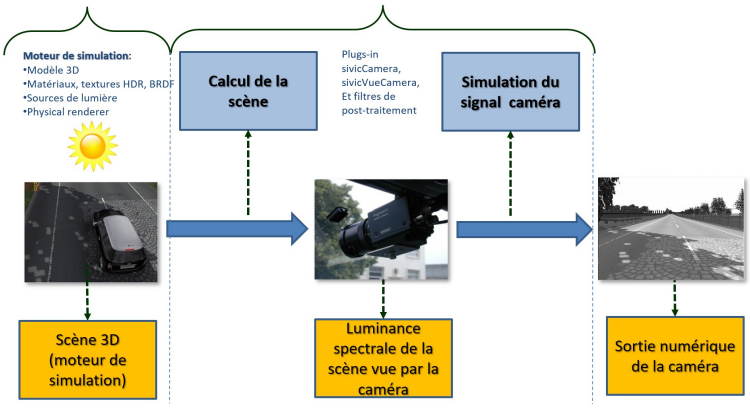


Figure 32: Camera functions in a virtual environment (source: UGE)

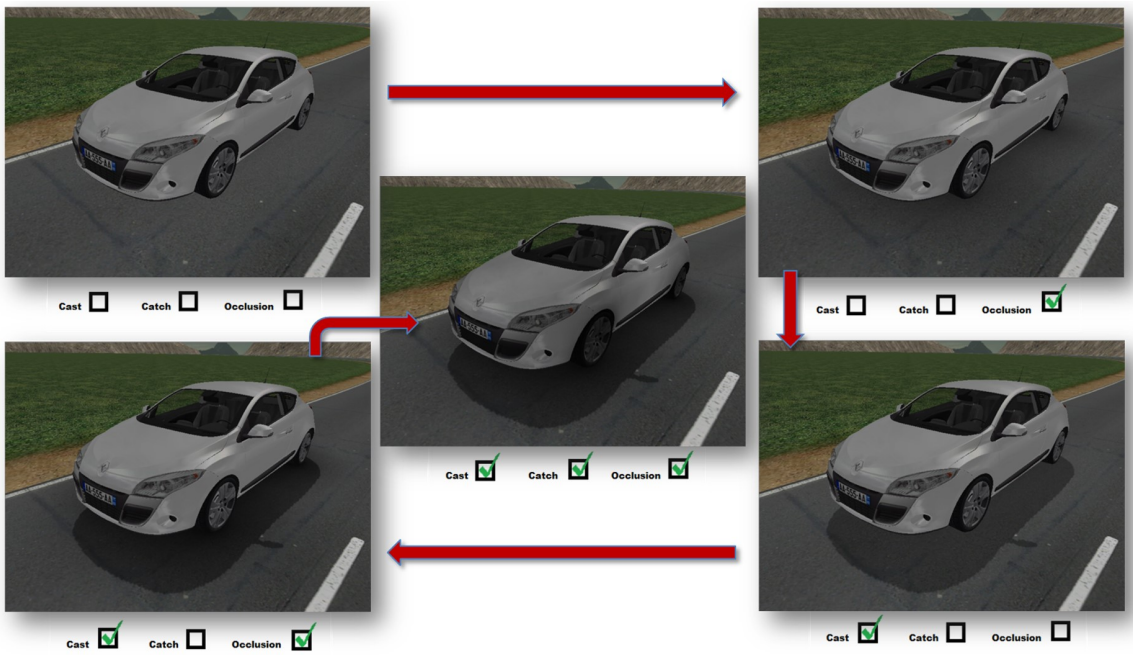


Figure 33: Different level of shadows management (catch, cast, occlusion, self-shadowing) used in Pro-SiVIC (source: UGE)

By fulfilling these requirements, a simulated camera model can provide a realistic rendering, enabling real ADAS and perception system to use virtual scenes with high fidelity and accuracy (environment perception (detection and recognition of objects, road features, situation, semantic aspects), AI model training, evaluation, and validation.

Figure 34: Material reflection on the car body, object, road surfaces implemented in Pro-SiVIC (source: UGE)

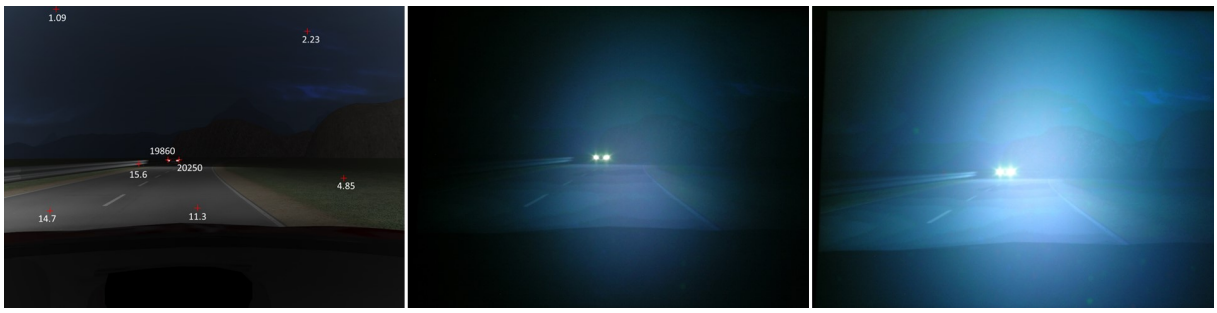


Figure 35: High Dynamic Range (HDR) texture for the ligth intensity encoding (in Pro-SiVIC). This mechanism is essential in order to generate light bluring and artefact on the camera rendering (source: UGE)

The Event camera or Neuromorphic camera has an operating different from the classical camera. The event-based camera does not produce pictures. The output of the camera is a set of vector with 4 components e = (t, x, y, p) for each single pixel of the image, where t is the time stamp of the event, (x, y) is the position of the pixel and p 0, 1 is the state of the pixel: p = 0 stands for a decreasing brightness and p = 1 stands for an increasing brightness. The pixel intensity is not an output of the camera. Another specificity of the event based camera is the asynchronicity. The event points are not output at the same time. The time between two output events is the order of two microsecond. Unlike the simulation of sensors such as cameras, LiDARs or RADAR, with which the ray-tracing technique allows for high simulation fidelity, the simulation of the event simulation needs a high frequency in the event simulation. So, in this context, simulate a neuromorphic camera involves addressing specific requirements to accurately model its unique characteristics and functionality. Here are the main requirements:

- **Intrinsic Sensor Modelling**: Develop a detailed model of the neuromorphic sensor array, capturing its spatial layout, pixel architecture, and sensitivity to light. This includes simulating the behaviour of individual pixels and their response dynamics.
- **Intrinsic Temporal Dynamics**: Incorporate temporal dynamics into the sensor model to simulate the asynchronous event-driven nature of neuromorphic cameras. This involves modelling pixel-level events, such as changes in brightness or motion, and their asynchronous capture and transmission.
- **Event Generation**: Implement algorithms to generate realistic events based on changes in the scene observed by the virtual neuromorphic camera. These events may include spikes in pixel activity corresponding to motion, contrast changes, or other visual features.
- **Spatial Resolution**: Define the spatial resolution of the simulated neuromorphic camera, considering factors such as pixel density, receptive field size, and spatial filtering characteristics. This ensures that the simulated output closely matches the spatial resolution of real neuromorphic sensors.
- **Dynamic Range**: Model the dynamic range of the neuromorphic camera, which determines its ability to capture both low and high-intensity events in the scene. This involves simulating the sensor's response to varying light levels and adjusting sensitivity accordingly.

- **Noise Characteristics**: Capture the noise characteristics inherent to neuromorphic sensors, including shot noise, readout noise, and temporal jitter. Accurate modelling of noise ensures that the simulated output matches the noise profile of real neuromorphic cameras.
- **Event Processing**: Implement algorithms for event processing and feature extraction, mimicking the on-chip processing capabilities of neuromorphic cameras. This may involve edge detection, motion estimation, or other forms of event-based processing.
- **Temporal Filtering**: Incorporate temporal filtering mechanisms to simulate the spatio-temporal integration properties of neuromorphic cameras. This includes modelling the time constants and temporal dynamics of individual pixels and their interactions.

By addressing these requirements, a simulated neuromorphic camera model can provide researchers and developers with a powerful tool for exploring the capabilities and potential applications of neuromorphic vision systems involving AI-based processing in diverse domains.

Simulating an **infrared (IR) camera** involves capturing the unique characteristics of IR imaging in a specific environment with material representing radiometric and temperature information. In this context, the environment modelling is different from RGB and Neuromorphic cameras. The main requirements for creating a realistic IR camera model are the following:

- **Radiometric Modelling**: Accurate modelling of radiometric properties to replicate how IR cameras detect and interpret thermal radiation. This includes modelling the relationship between temperature and emitted radiation, as well as sensor sensitivity and calibration.
- **Temperature Distribution**: Simulation of temperature distribution within the scene to generate realistic thermal images. This involves modelling heat transfer mechanisms, such as conduction, convection, and radiation, as well as environmental factors affecting temperature distribution.
- **Material Properties**: Incorporation of material properties affecting thermal radiation emission and absorption. Different materials have distinct emissivity and reflectivity characteristics, which influence how they appear in IR images. Accurate modelling of material properties enhances the realism of simulated scenes.
- **Atmospheric Effects**: Consideration of atmospheric effects impacting IR imaging, such as absorption, scattering, and atmospheric distortion. Modelling these effects allows for accurate simulation of long-range IR imaging scenarios, particularly in outdoor environments.
- **Sensor Characteristics**: Modelling of sensor parameters including spectral response, thermal sensitivity, noise characteristics, and dynamic range. This ensures that the simulated IR camera behaves realistically in capturing thermal images under various conditions.
- **Optical Effects**: Simulation of optical effects such as lens distortion, aperture diffraction, and optical aberrations. These effects influence the spatial resolution and image quality of IR cameras and should be accurately modelled for realism.

- **Dynamic Scene Simulation**: Integration of dynamic scene simulation to replicate changing thermal signatures and environmental conditions. This includes simulating moving objects, changing temperatures, and dynamic heat sources to create realistic thermal imaging scenarios.
- **Image Processing Algorithms**: Incorporation of image processing algorithms used in IR camera systems, such as noise reduction, image enhancement, and temperature mapping. These algorithms play a crucial role in improving image quality and extracting meaningful information from thermal images.
- **Integration with Environmental Models**: Integration with environmental modelling tools to simulate realistic thermal environments. This includes coupling with weather simulation models, building energy simulation tools, and thermal modelling software to generate accurate thermal scenes.

By fulfilling these requirements, a simulated IR camera model can provide valuable insights into thermal imaging applications across various domains, including surveillance, thermography, medical imaging, military reconnaissance, and automated driving in adverse conditions. Nevertheless, it is important to mention that a large number of technologies exist for IR sensors. These technologies can use different wave lengths (see figure 36) and architectures (multiple spectral, fast pixel, ...). The main wave lengths are the following:

- Short-Wave Infrared (SWIR) is generally defined as the light spectrum lying in the wavelength range 0.9 - 1.7 μm but which can be commonly extended to the range 0.7 - 2.5 μm.
- MWIR (Medium wave infrared): this is medium infrared, defined as the light spectrum located in the wavelength range 3 - 5 μm. It is in this part of the spectral band that the thermal image begins to form thanks to the thermal gradients present in the observed scene.
- LWIR (Long wave infrared): this is long infrared, defined as the light spectrum located in the wavelength range 7 - 14 μm. The LWIR camera is used to detect large distinct temperature differences. This sensor is the version currently used in AV.

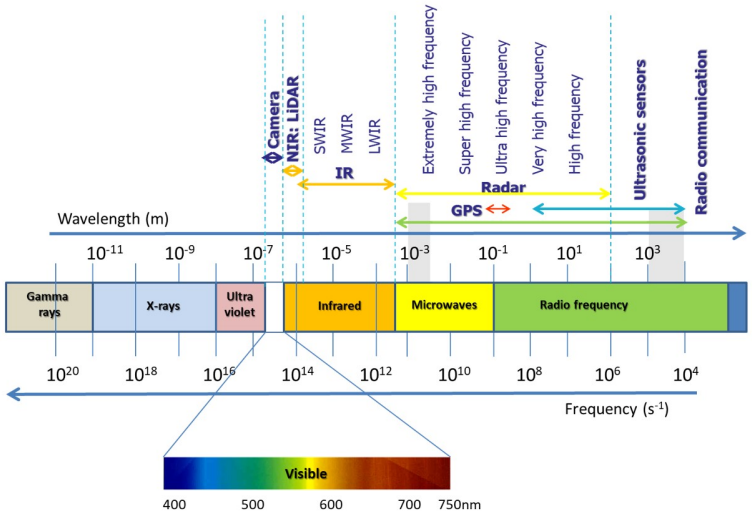


Figure 36: Different wave lengths for the different embedded sensing technologies(source: UGE)

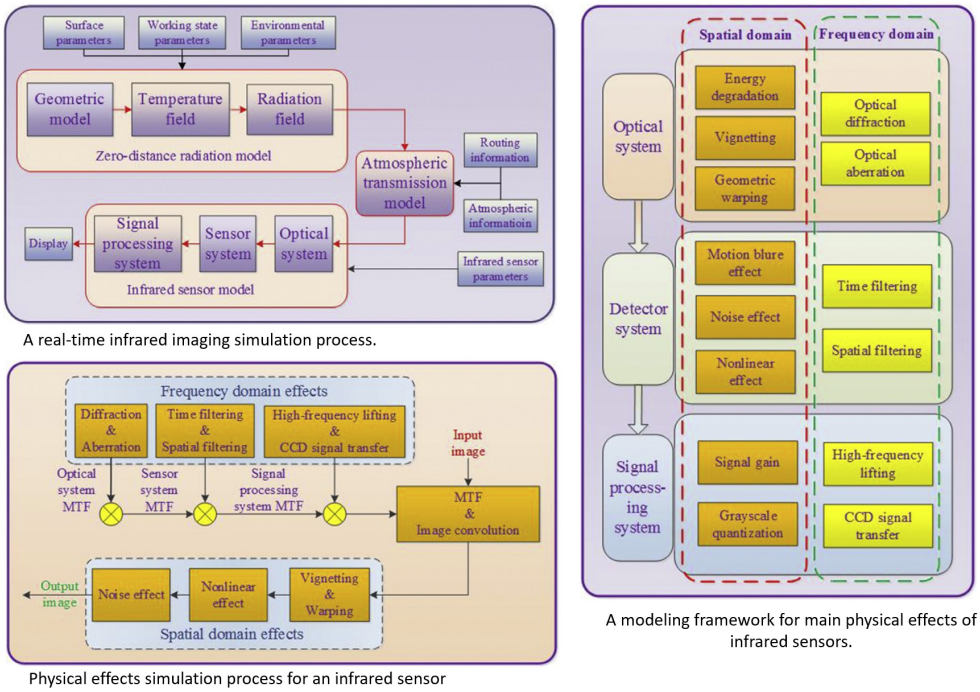


Figure 37: Modelling of an IR sensors (source: [7])

Figure 38: FLIR Thermal Sensing for ADAS

Figure 39: AdaSky's VIPER sensor with a 720p resolution by night

From a user's standpoint, infrared cameras operate much like regular video cameras, with the key difference being their detection of radiation in the long infrared range of the electromagnetic spectrum instead of measuring light from reflected objects. As long as there are no obstructions between the camera and the target, it can detect the temperature of the object. The primary advantage of thermal cameras in outdoor environments, such as road scenarios, is their independence from ambient light. Because they detect heat rather than visible light, they operate equally well at night and during the day. In fact, they often perform better at night due to the absence of sunlight, which enhances the contrast between heat-emitting and non-emitting objects. Living organisms are prominently visible in infrared cameras, as they all emit heat. Thus, identifying individuals in adverse weather conditions like rain, dust storms, fog, or smog is relatively straightforward with IR cameras. Vehicles also stand out as they generate heat. However, it's important to acknowledge some limitations of IR cameras:

- Reflections from water and glass can lead to misinterpretations, as these surfaces reflect infrared radiation, potentially creating false positives.
- IR cameras rely on contrast, so objects with similar temperatures to their surroundings may be challenging to distinguish.

- They don't provide distance information, making it difficult to discern between multiple objects if they have similar temperatures.
- Detection can be hindered if a person is wearing insulated winter clothing, as this reduces their heat signature.

In addition, some recent works focus their effort of promising new technologies like:

- High Dynamic Range (HDR) and high sensitivity camera: To prevent saturation in scenes with high contrast, High Dynamic Range (HDR) cameras are essential. These HDR cameras offer a greater number of intensity levels than the standard 8 bits (256 levels), often coded on 10, 12, or 16 bits. Another method involves capturing multiple images at varying exposure times successively, although this approach is only suitable for static or slow-motion scenes. The human eye achieves an even higher dynamic range thanks to its logarithmic intensity response. NIT, a company specialising in imaging technologies, offers a CMOS camera with a response close to logarithmic intensity. Similarly, event-based cameras also exhibit HDR capabilities. Sony recently introduced the IMX490 CMOS image sensor, designed specifically for automotive applications, boasting enhanced performance in low-light conditions. In nighttime or low-light scenarios, standard camera images often suffer from saturation in dark areas due to poor signal-to-noise ratio. Different cameras on the market exhibit varying performance levels in low-light conditions. One method for comparing their performance is using the absolute sensitivity threshold, defined in the EMVA1288 standard as the number of photons required to produce a signal equivalent to the observed noise. Sony image sensors consistently rank among the top performers in low-light sensitivity, as demonstrated in a comparison from 2019. Image intensification technology can enhance image contrast, with third-generation intensifiers offering gains up to 80,000. However, intensification is effective only in deep nighttime conditions and is susceptible to glare when dark and light areas are mixed. An alternative approach is working in the infrared spectral range instead of the visible range, as discussed in the following section.
- Fisheye camera: The choice of camera field of view is closely linked to the selection of the objective. Objectives vary widely in their field of view, ranging from a few degrees (zoom) to 180/200 degrees (fisheye). A larger field of view typically results in lower resolution, given a fixed number of pixels and assuming the same pixel sizes. In the human eye, sensor density is not uniform, and resolution decreases with distance from the main eye axis. This allows for a wide field of view of 170/175 degrees with good resolution near the main eye axis. Surprisingly, this advantageous property of the human eye has not yet been a source of inspiration for image sensor companies.
- Polarised camera: Light polarisation properties, though invisible to the naked eye, play a crucial role in understanding how light waves oscillate in space. There are four primary pure polarisation states: linear vertical polarisation, linear horizontal polarisation, circular polarisation, and unpolarised light. While the human eye cannot detect these properties, specialised cameras capable of measuring all four components of the polarisation state exist, such as those offered by Bossa Nova Tech. These advanced cameras utilise filters based on ferroelectric liquid crystals to selectively capture each polarisation component, albeit sequentially, which may introduce inaccuracies in capturing fast-moving objects.

However, recent advancements have led to the development of a new type of camera sensor capable of capturing linearly polarised and unpolarized components simultaneously, though it does not capture the circular component. This innovative design involves incorporating distinct linear polarisation filters in front of each pixel of the CMOS matrix, similar to color cameras, exemplified by Teledyne Dalsa cameras. While this approach provides an incomplete characterisation of the polarisation state, it offers the advantage of capturing all available components simultaneously. Such cameras have demonstrated utility in scenarios like visibility restoration in foggy conditions. There is ongoing exploration into extending the application of these cameras to phenomena such as snow and dust. Moreover, when light reflects off smooth surfaces like mirrors, the reflected light becomes polarised. This unique property suggests potential applications for polarisation cameras in mitigating specular reflections, enhancing contrast, and detecting wet or icy surfaces. However, the efficacy of this approach requires validation through extensive experimentation.

- Time gated imaging or ballistic photon: Time gated imaging or ballistic photon principle can be used to see through fog, rain, snow, dust and thus to go beyond visibility restoration by image processing. The idea is to illuminate the scene with a short flash and to collect only the photons that managed to make a round trip to a given distance. Several images are taken, with adequate tuning, to observe different depth slices of the scene.

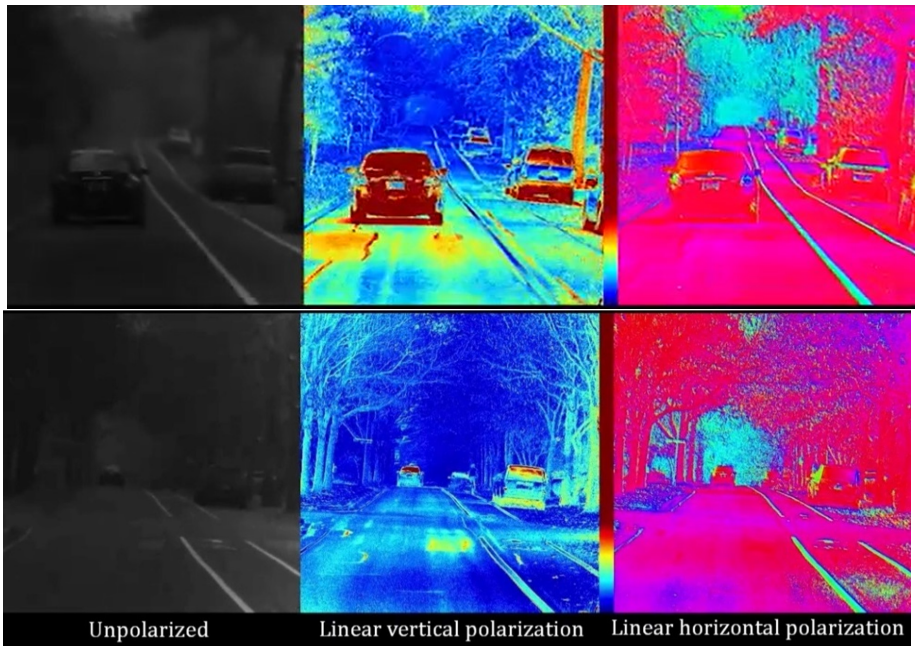


Figure 40: Overview of a polarised camera. left: classical camera; middle: linear vertical polarisation; right: linear horizontal polarisation (https://www.techbriefs.com/component/content/article/33184-new-polarization-camera-illuminates-foggy-streets-for-self-driving-cars)

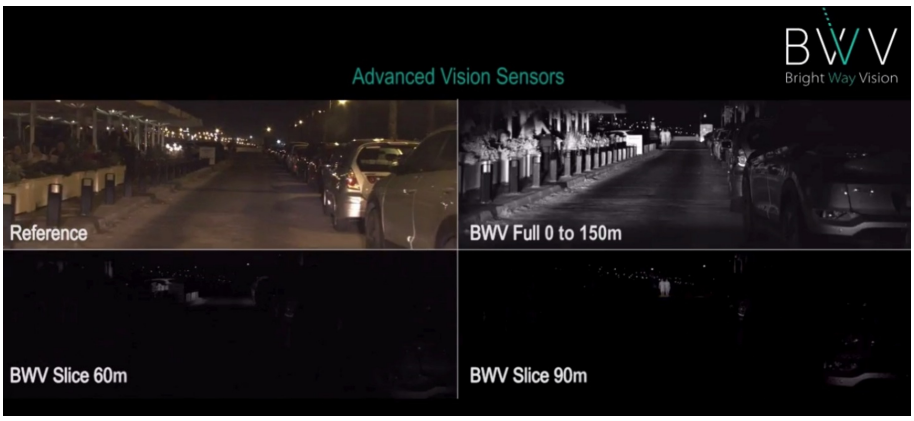


Figure 41: Time gated imaging or ballistic photon (https://www.brightwayvision.com/technology/)

New generations of sensors mixed several technologies in order to obtain bio-inspired sensors. It is the case for the bio-inspired Polarised event-based camera ([55]).

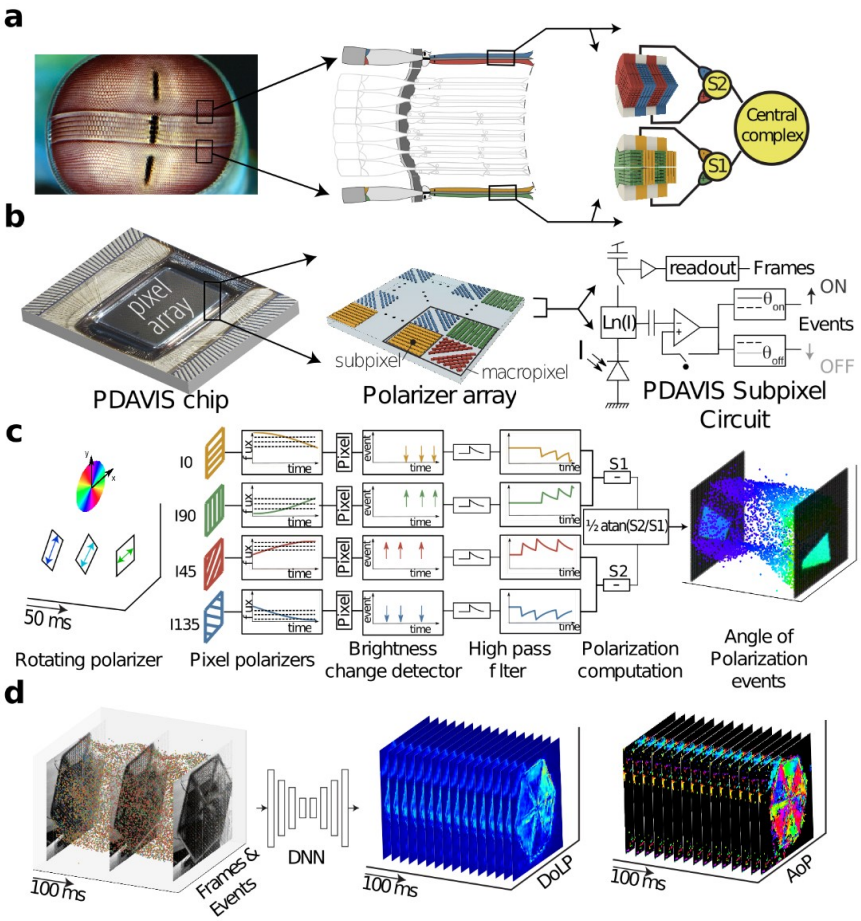


Figure 42: overview of the PDAVIS bio-inspired polarisation vision sensor. a: Polarisation vision in the mantis shrimp eye; b: The PDAVIS polarisation event camera; c: A rectangular rotating linear polarizer (left) generates a stream of brightness change events; d: A polarisation filter wheel is rotated in front of PDAVIS, which produces frames and events

This bio-inspired polarisation vision sensor (see figure 42) is inspired by the mantis shrimp's

eye. The mantis shrimp's eye has two sets of orthogonal microvilli capturing four linear polarisation states, which, coupled with logarithmic photo-receptors, enable effective predation in coral reefs. The PDAVIS polarisation event camera mimics this mechanism by integrating pixelated polarisation filters with a vision sensor, providing sustained pathway frames and transient pathway log-scale brightness change events. A rectangular rotating linear polarizer generates brightness change events from the PDAVIS macropixels, and a temporal filter computes the Angle of Polarisation (AoP), resulting in low-latency AoP events. Additionally, a polarisation filter wheel produces frames and events, and a Deep Neural Network (DNN) reconstructs Degree of Linear Polarisation (DoLP) and AoP from brightness change events at a higher rate than the camera's maximum frame rate.

The presentation of the unconventional cameras shows that the propagation channel and the environment modelling of a simulation platform becomes more and more multiple spectral with the needed to either implement additional material properties and specific textures, or to run in paralleled several dedicated rendering engine taking into account specific band of wave length.

3.2.5 Proprioceptive Sensor Models

3.2.5.1 GPS technologies and model requirements

GPS model : refers to a simulated representation of a Global Positioning System (GPS). A GPS model in a vehicle simulation includes the following aspects: satellite constellation simulation, signal propagation and reception, position calculation algorithms, error modelling, accuracy and uncertainty Estimation and integration with vehicle dynamics.

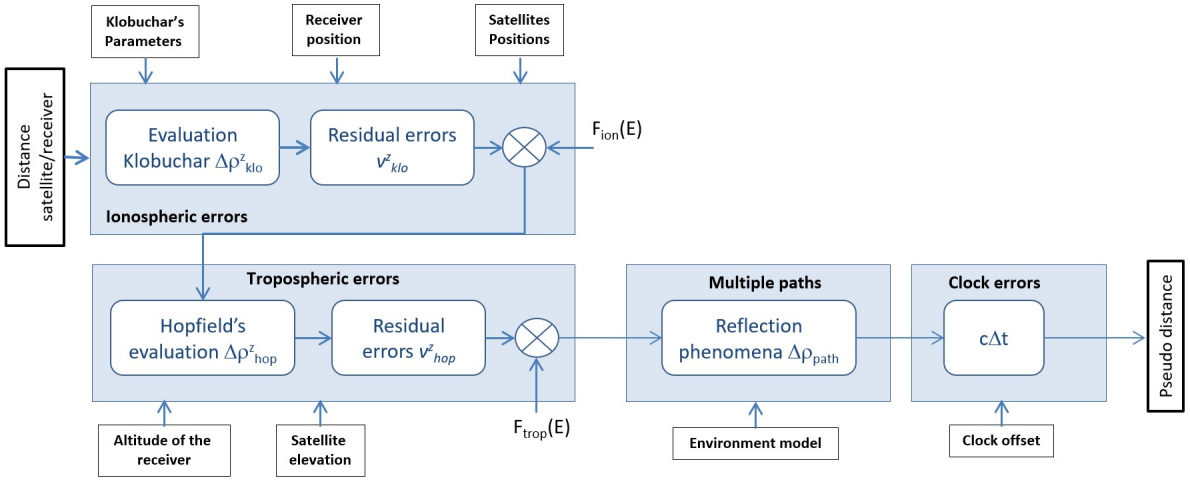


Figure 43: Diagram of the different disturbances on the GPS signal (functions involved in the Pro-SiVIC's GPS model) (Source: UGE)

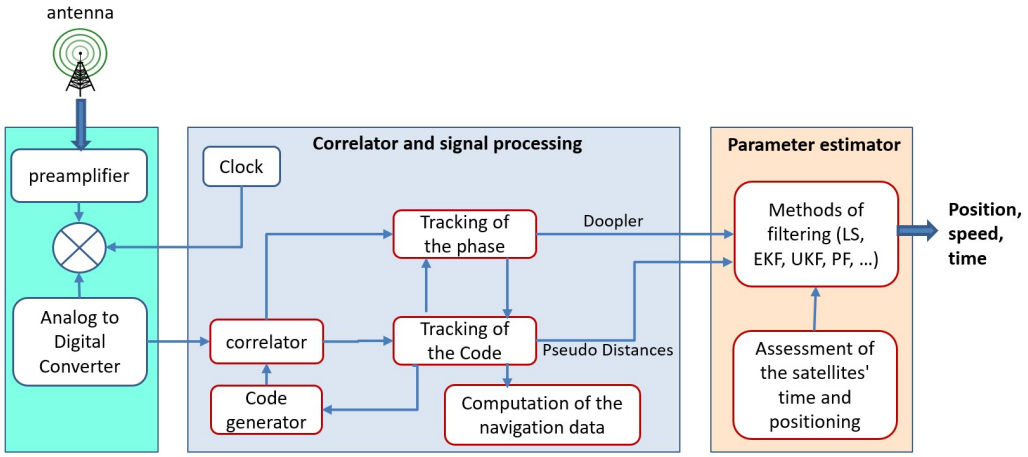


Figure 44: Diagram of the different functions involved in the GPS receiver(Source: UGE)

Developing a GPS simulation for autonomous driving entails addressing a multitude of high-level requirements to ensure the provision of accurate and realistic positioning information. Below are the key requirements to consider:

- **Realistic Geographical Data** : Incorporate high-quality and up-to-date geographical data, including maps, road networks, landmarks, and relevant infrastructure. This ensures the simulation accurately represents real-world driving scenarios and interactions with the GPS receiver.
- **Dynamic and Real-Time Positioning**: Simulate dynamic and real-time GPS positioning updates to reflect changes in the vehicle's location as it moves through the simulated environment. This involves computing pseudo-distances and generating NMEA frames to mimic the continuous and instantaneous updates necessary for accurate autonomous navigation.
- **Satellite Constellation Simulation**: Model a realistic satellite constellation, including the positions and movements of GPS satellites, to generate authentic GPS signals. This includes generating clock corrections and utilising ephemeral files to replicate the signals received by an actual GPS receiver.
- **Atmospheric Layers Modelling**: Simulate high-quality propagation of GPS signals through atmospheric layers, such as the ionospheric and tropospheric layers, considering their impact and interaction on signal transmission and pseudo-distance accuracy.
- **Accuracy and Precision Control**: Provide control over the accuracy and precision of the GPS simulation to mimic different real-world scenarios and GPS receiver capabilities. This allows testing under varying conditions, from high-precision scenarios to situations with lower GPS accuracy.
- **Signal Interference and Jamming Simulation**: Introduce scenarios with signal interference or jamming to assess the robustness of autonomous systems under challenging GPS conditions. Evaluate system performance in environments where GPS signals may be compromised.

- **Multi-Constellation Support**: Support the simulation of multiple satellite constellations (e.g., GPS, GLONASS, Galileo) to assess the performance of multi-constellation GNSS receivers. Reflect real-world scenarios where multiple satellite constellations are available for navigation.
- **Simulation of Urban Canyon Effects**: Model the effects of urban canyons, tall buildings, or tunnels on GPS signals to evaluate system performance in challenging urban environments. Replicate scenarios where line-of-sight to satellites is obstructed or reflections cause signal inaccuracies.
- **Dynamic Elevation Changes**: Simulate dynamic elevation changes, such as hills and valleys, to accurately represent the impact of terrain on GPS positioning. Ensure the system can handle changes in elevation for precise navigation.
- **Differential System of Positioning**: Simulate remote stations to provide positioning error for GPS correction (DGPS and GPS RTK). Mimic the complementary use of a reference utilising the same satellites to send positioning corrections via communication means.

The objective is to identify the essential modules, functions, and aspects required for a GPS model to achieve a high level of fidelity with real GPS data and behavior. The GPS model should encompass the following components:

- **Satellite Constellation**: Incorporate the satellite constellation at a specific date to accurately simulate satellite positions.
- **Atmospheric Layer Modelling**: Utilise atmospheric layer models, including the ionosphere and troposphere, to account for their effects on GPS signal propagation.
- **Low Altitude Propagation Channel**: Consider factors such as occlusion and multiple reflections in the low altitude propagation channel to accurately simulate signal transmission.
- **Computation of Pseudo-Distance**: Calculate pseudo-distances to simulate the distance between GPS satellites and the receiver.
- **NMEA Frame Generation**: Generate NMEA frames according to the NMEA 0183 standard to provide realistic GPS data output.

In practice, satellite positions can be calculated using ephemeris files obtained from the International GNSS Service website. These files allow for the calculation of satellite positions through Lagrangian interpolation or using orbital parameters. The accuracy of the trajectories depends on the precision of the ephemerides, which can vary based on the type of file used:

- **IGS**: ”precise” ephemerides (within 2 weeks)
- **IGR**: ”rapid” ephemerides (within 72 hours)
- **IGU**: ”ultra-rapid” ephemerides (within 24 hours)

The complexity of a GPS model can vary, ranging from simple models that introduce noise to the vehicle's position to more sophisticated models that accurately simulate satellite constellations, signal disturbances, and receiver functions. All these components are necessary to ensure the chosen model's accuracy and reliability.

In more complex models, considerations include:

- **Signal Disturbers**: Account for ionospheric and tropospheric errors, as well as reflection phenomena, to simulate real-world signal disturbances.
- **Time and Clock Errors**: Incorporate time and clock errors to mimic real GPS receiver behaviour.
- **Receiver Functions**: Model the various functions of the GPS receiver to accurately simulate its operation.

These functions collectively represent a complex process for modelling a high-fidelity GPS simulation.

In a realistic enough model, it is necessary to guarantee that we take into account the following requirements:

- **Atmospheric Layer Models**: Take into account and propose the implementation of models like ionospheric and tropospheric models affecting GPS signals. The ionosphere, located roughly 50 to 750 km above the Earth's surface, introduces delays in signal transmission due to solar radiation. On the other hand, the troposphere, the lower atmospheric layer, introduces refraction, and its effects need careful modelling.
- **Calculation of Pseudo-Distances**: Provide and develop algorithms for calculating pseudo-distances between satellites and the receiver.
- **Generation of NMEA Frames**: Ensure the accurate generation of NMEA frames, conforming to standards like NMEA 0183. The generated NMEA frames need to be compliant with real expected outputs and standards.
- **Multi-Reflection/Multi-Path Modelling**: Implement the modelling of multi-path effects in GPS signals, including signal attenuation and propagation delays.
- **Satellite Orbit and Clock Models**: Model the accuracy of satellite orbit and clock models used for satellite position and time corrections. The model needs to respect the known satellite positions and clock data provided by satellite agencies.
- **Receiver Dynamics and Sensitivity**: Take into account receiver dynamics and sensitivity to different signal strengths, frequencies, and environmental conditions. Model the receiver's ability to handle weak signal scenarios and signal interference.
- **Error Sources and Correction Models**: Implement the error sources such as clock biases, satellite ephemeris errors, and atmospheric delays. The model needs to respect the effectiveness of correction models like Differential GPS (DGPS) in reducing errors.
- **Signal Propagation and Signal Degradation Models**: Propose signal propagation models considering terrain, obstructions, and atmospheric effects. This model needs to mimic signal degradation due to foliage, buildings, or other environmental factors.

- **Real-Time Kinematic (RTK) and Precise Point Positioning (PPP)**: Generate high-precision positioning techniques like RTK and PPP, if implemented.
- **Integration with Vehicle Dynamics**: Develop models for the integration of GPS with vehicle dynamics and navigation algorithms.

3.2.5.2 INS technologies and model requirements

An Inertial Navigation System (INS) is a sensor involving 3 different complementary sensors: a motion sensors (accelerometers), a rotation sensors (gyroscopes or gyrometer), and a magnetometer. The INS allows to continuously calculate the position, orientation, and velocity of a moving object without external references such as GPS. This model includes inertial sensors models (accelerometers measure linear accelerations, while gyroscopes detect angular velocities) and can include also integration algorithms, error characteristics and calibration procedures.

Gyroscopes are essential devices mounted on a frame to detect angular velocity when the frame is in rotation. They come in various classes, each utilizing different physical principles and technologies. Gyroscopes can function independently or be integrated into more complex systems such as Gyrocompasses, Inertial Measurement Units (IMUs), Inertial Navigation Systems (INS), and Attitude Heading Reference Systems (AHRS).

The most commonly used classes of gyroscopes are mechanical gyroscopes, optical gyroscopes (including Fiber Optic Gyroscopes (FOGs) and Ring Laser Gyroscopes (RLGs)), and Micro-electromechanical system (MEMS) gyroscopes. One of the critical factors for all types of gyroscopes, as angular velocity sensors, is the accuracy in measuring angular velocity. Therefore, a key metric is the stability of the scale-factor. The scale factor denotes the sensitivity of the gyroscope, while accuracy, inversely related to sensitivity and accounting for measurement errors due to noise, can be quantified by parameters such as resolution (R) or Angle Random Walk (ARW) in RLGs. ARW correlates with the bandwidth (B) of the measurement system, defined as ARW = R/[60(B)]. A higher scale-factor stability results in reduced sensor errors but necessitates more advanced instruments and enhanced accuracy, leading to increased system costs. Consequently, the performance and costs of gyroscopes are directly influenced by the specific requirements of their applications. The principle of Gyroscope Technologies and their potential Applications is detailed in [8].

- **Mechanical Gyroscopes** : A mechanical gyroscope typically comprises the following components:
  - Spinning wheel mounted on two gimbals: This setup allows the gyroscope to undergo precession motions along two perpendicular directions.
  - Rigid frame with rotating bearings: The mechanical parts experiencing relative motion are prone to friction, resulting in measurement drifts over time. The primary objective in gyroscope design precessions to create a frictionless and perfectly balanced device. To reduce friction, high-precision bearings and specialized lubricants are employed. In critical applications, magnetic suspensions or fluid-suspended configurations are utilized.
  - Sensing systems (pick-offs): These systems are capable of detecting angular displacements between the adjacent gimbals and converting them into electrical sig-

nals using potentiometers, resolvers, or encoders. These signals serve as input for a computing unit.

- **Optical gyroscopes (including Fiber Optic Gyroscopes (FOGs) and Ring Laser Gyroscopes (RLGs))**: Optical gyroscopes function by detecting the difference in propagation time between beams traveling in opposite directions within closed or open optical paths. When there is a rotation-induced change in the path lengths, it creates a phase difference between the counter-propagating light beams. This phase difference is a manifestation of the Sagnac effect, which serves as the fundamental operating principle for all optical gyroscopes. Based on the measurement technique of the Sagnac effect, optical gyroscopes can be classified into two main types: active and passive architectures. In active configurations, the closed-loop optical path, such as the ring cavity, includes the optical source, forming a ring laser. These active configurations can be constructed using Bulk Optics or Integrated Optics technology, although commercial maturity has primarily been achieved with Bulk Optics solutions. Within the category of Ring Laser Gyros, various methods are employed to mitigate the lock-in effect, which occurs at low rotational rates, typically in the range of tens of degrees per hour. Lock-in can be reduced by introducing mechanical dither, magneto-optic biasing, or employing multiple optic frequencies configuration. In contrast, passive architectures involve an external optical source outside the closed optical loop, such as in the Interferometric Fiber Optic Gyroscope. Ring Laser Gyroscopes and Interferometric Fiber Optic Gyroscopes are the most prevalent types of optical gyroscopes, each offering distinct features in terms of size, weight, power requirements, performance, and cost.
- **Micro-electromechanical system (MEMS) gyroscopes**: MEMS gyroscopes typically employ a vibrating mechanical element as a sensing component to detect angular velocity. Unlike traditional gyroscopes, MEMS gyroscopes do not rely on rotating parts that necessitate bearings. This characteristic facilitates straightforward miniaturization and the utilization of manufacturing techniques commonly associated with MEMS devices. All MEMS gyroscopes utilizing vibrating elements operate based on the transfer of energy between two vibration modes induced by Coriolis acceleration. Coriolis acceleration, which is directly proportional to angular velocity, is an apparent acceleration observed within a rotating frame of reference. A review about MEMS and MOEMS Gyroscopes is given in [56].

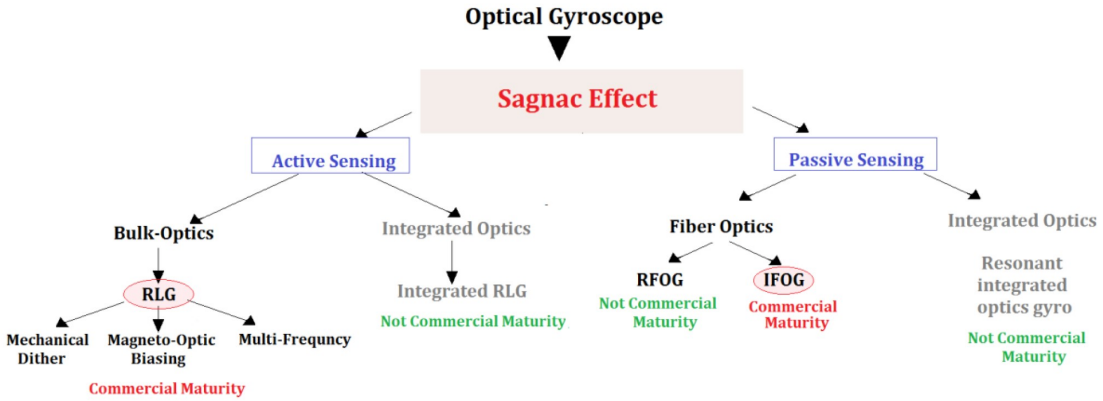


Figure 45: The different classes of optical gyroscopes ([8])

Modelling a gyrometer involves several key requirements to ensure accuracy, reliability, and effectiveness. Here are the main requirements:

- Understanding of Gyroscope Principles: A thorough comprehension of the underlying principles of gyroscope operation, including the physics involved in measuring angular velocity and the associated effects such as precession and drift.
- Selection of Gyroscope Type: Determine the specific type of gyroscope to be modelled, such as mechanical, optical, or MEMS-based gyroscopes, based on the application requirements and desired level of accuracy.
- Mathematical Modelling: Develop mathematical models that accurately represent the behaviour of the chosen gyroscope type, including equations describing its response to angular velocity inputs, environmental factors, and any inherent noise or errors.
- Calibration Parameters: Define calibration parameters to characterise the gyroscope's sensitivity, bias, scale factor, and other intrinsic properties, which are essential for accurately interpreting sensor readings and compensating for errors.
- Environmental Factors: Consider environmental factors such as temperature variations, vibration, magnetic interference, and external forces that may affect the gyroscope's performance, and incorporate appropriate compensation algorithms into the model.
- Integration with Other Sensors: If the gyroscope is part of a larger sensor array or inertial navigation system, ensure compatibility and seamless integration with other sensors such as accelerometers, magnetometers, and GPS receivers to provide comprehensive motion tracking capabilities.
- Error Modelling and Compensation: Develop methods for modelling and compensating for common sources of error in gyroscope measurements, including sensor noise, drift, cross-axis sensitivity, and dynamic errors arising from rapid motion or sudden changes in orientation.
- Computational Efficiency: Optimise the computational efficiency of the gyroscope model to minimise processing overhead and latency, especially in real-time applications where timely sensor data processing is critical.

By addressing these requirements, a comprehensive gyroscope model can be developed that accurately simulates the sensor's behavior and enables its effective integration into various applications, ranging from navigation and robotics to aerospace and automotive systems.

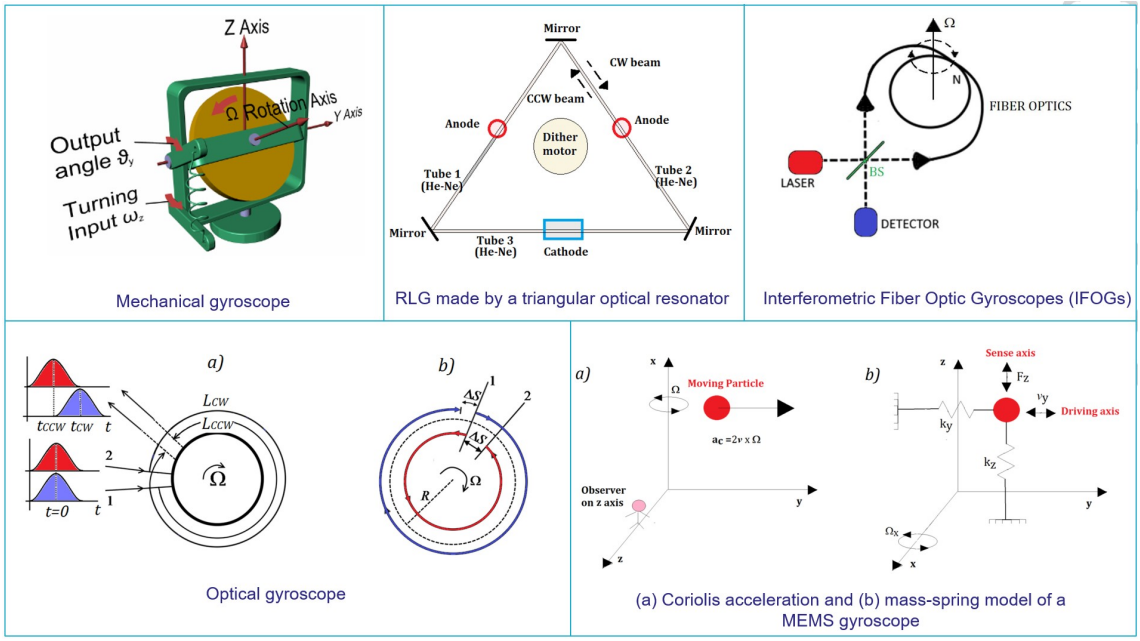


Figure 46: The different technologies of gyroscopes ([8])

Accelerometers are sensors designed to measure acceleration, which translates into changes in speed or direction. This encompasses various movements such as bumps, vibrations, sudden speed increases or decreases like during hard vehicle acceleration or braking, and forces indicative of turns taken too sharply or strong impacts. Typically, an accelerometer sensor comprises a known mass that is damped and held in place by elements like springs, which monitor the motion of the mass relative to the device. The deflection of these springs is then measured through methods such as piezoelectric voltage, capacitance, or optical means. A higher measured value indicates a greater deflection of the mass and hence a higher detected acceleration. Accelerometers provide valuable data by detecting voltages resulting from deflections in the elements holding the known mass, enabling measurement of a device's position and movement. This data is particularly useful when combined with information from other sensors like GPS. Accelerometers predominantly rely on Piezotronics systems, employing various transduction techniques including piezoelectric, piezoresistive, and capacitive methods.

- **Piezoelectric sensors** capitalise on the Piezoelectric Effect inherent in certain crystals. When stress is applied to the crystal due to force, negative and positive ions accumulate on opposite surfaces, producing a charge proportional to the applied force. This charge is then bled off through an amplifier to measure its amplitude, with the sensor's geometry determining its sensitivity to physical parameters like force, pressure, or acceleration.

- **Piezoresistive accelerometers** utilise metal strain gauges, piezoresistive silicon, or MEMS devices. A resistive material bonded to a cantilever beam undergoes bending under acceleration, altering its resistance and producing a change in output voltage proportional to acceleration.

- **Capacitive accelerometers** exploit the variation in distance between opposed plate capacitors, which changes proportionally with applied acceleration, thus altering capacitance. This variation is then converted into a voltage signal proportional to acceleration.

While piezoelectric accelerometers excel at measuring fast transient and periodic acceleration, piezoresistive ones can measure constant, transient, and periodic acceleration. Capacitive accelerometers are capable of measuring both constant and slow transient and periodic acceleration.

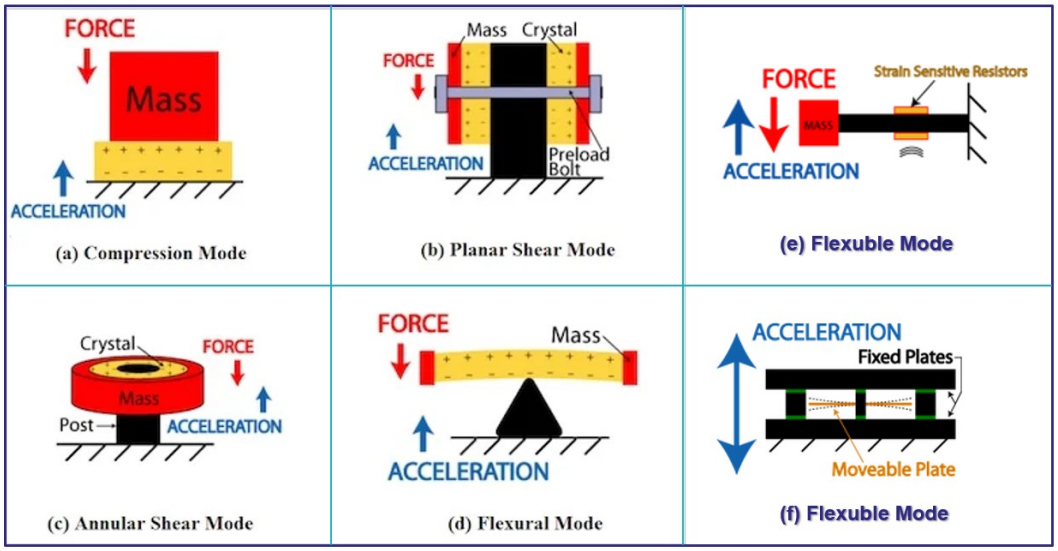


Figure 47: The different types of accelerometers technologies based on Piezotronics systems, employing various transduction techniques

Modeling an accelerometer requires attention to various factors to ensure accuracy, reliability, and relevance to the intended application. Here are the main requirements:

- Understanding of Accelerometer Principles: Gain a thorough understanding of the underlying principles of accelerometer operation, including the physical phenomena involved in measuring acceleration, such as inertial forces acting on a mass-spring system or changes in capacitance due to motion.
- Selection of Accelerometer Type: Determine the specific type of accelerometer to be modeled, such as piezoelectric, piezoresistive, capacitive, or MEMS-based accelerometers, based on the application requirements and desired level of accuracy.
- Mathematical Modeling: Develop mathematical models that accurately describe the behavior of the chosen accelerometer type, including equations governing its response to linear acceleration inputs, as well as any non-linearities, noise, or errors inherent in the sensor. The model should accurately estimate the vehicle's accelerations on the different axis with high accuracy taking into account possible vibration of the sensor support.
- Calibration Parameters: Define calibration parameters to characterize the accelerometer's sensitivity, bias, scale factor, and other intrinsic properties, which are necessary for accurately interpreting sensor readings and compensating for errors.
- Environmental Factors: Consider environmental factors such as temperature variations, gravitational effects, vibration, and electromagnetic interference that may affect the accelerometer's performance, and incorporate appropriate compensation algorithms into the model.

- Integration with Other Sensors: Ensure compatibility and seamless integration of the accelerometer model with other sensors such as gyroscopes, magnetometers, and GPS receivers to provide comprehensive motion tracking capabilities, especially in inertial navigation systems. The model should be scalable and adaptable to accommodate different types of configurations and positioning in the vehicles.

- Integration with Vehicle Dynamics: The accelerometer model should integrate seamlessly with the vehicle dynamics simulation, accounting for factors such as accelerations, decelerations on the different vehicle moving axis. When the vehicle applies a turn manoeuvre, a composition of the acceleration is shared on the different axis (x,y,z). When some specific behaviours of the vehicle happens, like aquaplaning, then the accelerometer has to reproduce The faithful measurement of this behavior.

- Error Modeling and Compensation: Develop methods for modeling and compensating for common sources of error in accelerometer measurements, including sensor noise, bias drift, cross-axis sensitivity, and dynamic errors arising from mechanical vibrations or shock. The simulation environment should accurately replicate real-world conditions, including variations in terrain, road surface quality (vibration effect on the tires, wheels, shock absorbers, and car body), and environmental factors like weather conditions (wind, tire grip on wet, snowy, or frozen road surfaces).

- Validation and Verification: Validate the accelerometer model against experimental data obtained from real-world tests or benchmark datasets to verify its accuracy, reliability, and consistency under various operating conditions.

- Computational Efficiency: Optimize the computational efficiency of the accelerometer model to minimize processing overhead and latency, particularly in real-time applications where timely sensor data processing is critical. The simulation should be computationally efficient to enable real-time or near-real-time performance at high frequency.

By addressing these requirements, a comprehensive accelerometer model can be developed that accurately simulates the sensor's behavior and enables its effective integration into various applications, such as motion analysis, vibration monitoring, structural health monitoring, and inertial navigation.

3.2.5.3 Odometric technologies and model requirements

Odometer model: an odometer model refers to a simulated representation of an odometer, the instrument used to measure the distance travelled by a vehicle.

the main requirements for providing an odometer model are:

- Accuracy and Precision: The model should accurately estimate the vehicle's position, distance travelled, and speed with high precision depending of the used sensor. From the figure 48, it is possible to see different resolution for the sensor.

- Integration with Vehicle Dynamics: The odometer model should integrate seamlessly with the vehicle dynamics simulation, accounting for factors such as acceleration, deceleration, and turning to provide realistic estimates of movement. In this context, the sensor have to reproduce the differential system allowing manage the wheel speeds in a turn manoeuvre (a wheel speed is lower than the opposite wheel).

- Dynamic Wheel Rotation : The model should accurately takes into account the rotational movement of the vehicle's wheels over time to determine the distance travelled. This imply the modelling of ABS effect or wheel blocking and sliding (with water and ice).
- Realistic Environment Representation: The simulation environment should accurately replicate real-world conditions, including variations in terrain, road surface quality, and environmental factors like weather conditions. This also means to be robust to Environmental Factors such as changes in road conditions, temperature, and humidity to maintain accuracy and reliability in the sensor behaviour and operating.
- Scalability and Flexibility: The model should be scalable and adaptable to accommodate different types of vehicles (size of the wheel for instance).
- Simulation Performance: The simulation should be computationally efficient to enable real-time or near-real-time performance at high frequency.

Several types of odometer are available for automotive application. The 2 main technologies are:

- Incremental Odometers: This sensor measures the number of wheel rotations during a period of time using incremental encoders. These encoders track the incremental rotations of the vehicle's wheels. The incremental odometer integrates the incremental wheel rotations to estimate the current state (distance, speed, position) based on a vehicle evolution model. It utilises two channels, A and B, to increase resolution and determine the direction of rotation (rising edge of channel A). If B = 0, the rotation direction is clockwise. If B = 1, the rotation direction is counterclockwise.
- Absolute Odometers: This type of odometers directly report the angular position to the controller.

This technology offers easy implementation, low cost, high measurement acquisition rate, autonomous operation, and good short-term accuracy. However, systematic errors can occur due to incorrect mechanical setup of the vehicle or sensors. Additionally, non-systematic errors are related to wheel-road contacts, such as random errors and dependence on road quality (irregularities, presence of ice), including wheel lock and slippage.

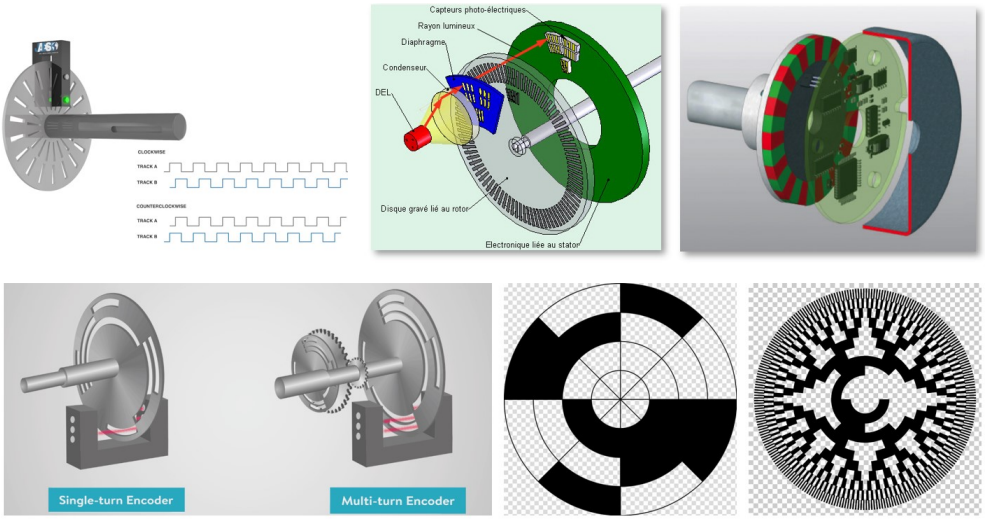


Figure 48: Different technologies of odometer: Incremental and absolute technologies

3.2.6 Communication technologies and model requirements

Sensors deployed on the infrastructure : as part of a system of systems, where the infrastructure plays an important role, we need to add models for all the sensors and remote equipment that communicate with the vehicle. In this context, V2X communication must also be modelled.

3.2.6.1 High level requirement for communication modelling

Simulating communication means for autonomous driving involves addressing various aspects to ensure the effectiveness and reliability of the communication systems. Here are 10 main high level requirements to consider:

- **Realistic Communication Protocols**: Simulate realistic communication protocols used in autonomous driving systems, such as V2X (Vehicle-to-Everything) communication, DSRC (Dedicated Short Range Communication), or cellular networks like 5G. This requires to take into account the 7 OSI layers has presented in figure 49.
- **Network Latency and Bandwidth Simulation: Requirement**: Simulate network latency and bandwidth constraints to replicate real-world communication challenges.
- **Dynamic Traffic and Environment Simulation**: Integrate dynamic traffic and environmental conditions to emulate realistic scenarios with complex situations, dense communication traffic. It is necessary to take into account communication systems under diverse situations, including heavy traffic, adverse weather, and complex road conditions which simulate issues like message loss or message collision.
- **Security and Privacy Testing**: Implement security and privacy testing scenarios to assess the resilience of communication systems against cyber threats and protect sensitive data. The communication simulation needs to give models and mechanisms which allows to test the safety and security aspects of autonomous vehicles against potential malicious activities. A presentation of the main classes of cyber attacks is given in the figure 50.

- **Interoperability Testing**: Validate the interoperability of communication systems by simulating interactions with different types of vehicles, infrastructure, and devices. Guarantee seamless communication between autonomous vehicles and their environment.
- **Scalability and Density Simulation**: Simulate scenarios with varying numbers of connected vehicles to assess scalability and network congestion. The proposed modelling need to allow to Understand how communication systems perform as the number of connected vehicles increases (message collisions, latencies, ...).
- **Reliability and Redundancy Assessment**: Provide models which allow to evaluate the reliability and redundancy mechanisms in communication systems. This level of simulation allows to test the communication failures or disruptions, and assess the impact on the safety of autonomous vehicles.
- **Integration with Sensor Data**: Integrate communication simulation with sensor data and perception data to mimic the holistic perception of the vehicle. This imply to simulate and to generate realistic communication messages of the facilities OSI layer like CAM (Cooperative Awarness Messages), DENM (Decentralized Event Notification Messages), CPM (Collective Perception Message), ... (see figure 50).
- **Edge Case Scenarios**: Simulate edge case scenarios, such as communication blackouts or intermittent connectivity, to assess system behaviour under extreme conditions.
- **Regulatory Compliance Simulation**: Incorporate simulation scenarios that comply with existing and anticipated regulations for autonomous vehicle communication.

By addressing these requirements in communication simulation for autonomous driving, it is possible to conduct comprehensive testing and validation of communication systems in a controlled environment before deploying them in real-world scenarios. In addition to these high level requirements, it is also mandatory to model hardware components of the communication system, and the propagation channel interactions and effects on the electromagnetic signal emitted and received by antennas. In this context, antenna modelling also is essential.

3.2.6.2 Communication protocol, strategies, Messages, and OSI layers

In order to provide an answer to the first requirement, it is necessary to take into account the 7 OSI layers. This is already done in the NS3 library for instance.

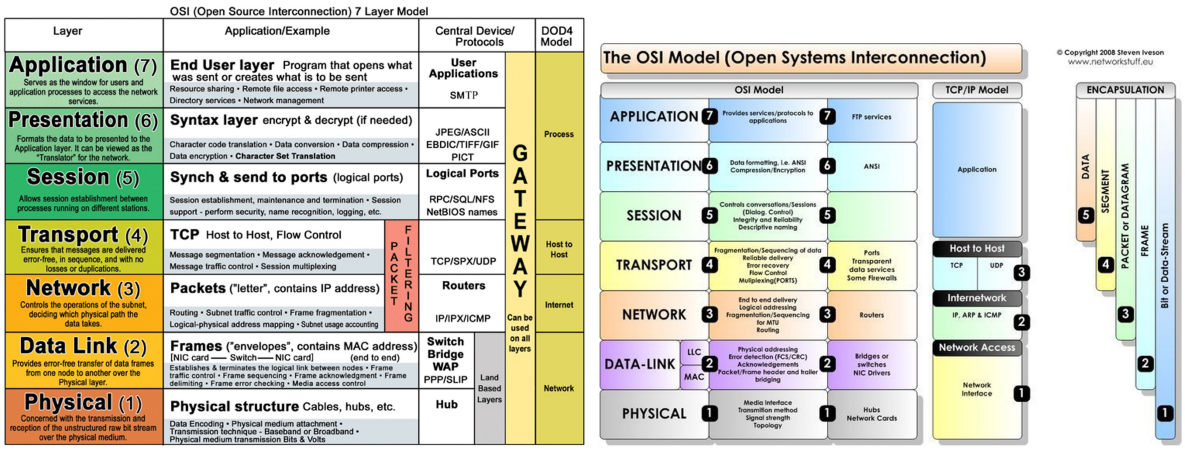


Figure 49: Overview of the OSI model and the fitting with TCP/IP.

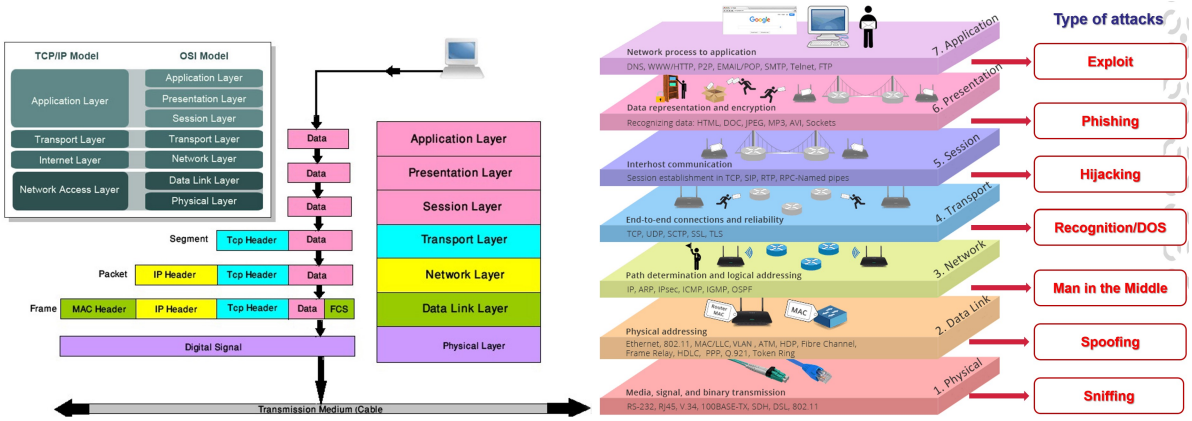


Figure 50: Overview of the different message format and possible cyber attacks by level of the OSI model

Phases of V2X application roadmap[2]	Message types[1]			Examples of applications based on the message types
	Europe	USA	Abbreviations explained	
Awareness driving	CAM, DENM	BSM	Cooperative Awareness message, Decentralized Environmental Notification Message, Basic Safety Message	Intersection Collision Warning Emergency Vehicle Warning Dangerous Situation Warning Stationary Vehicle Warning Traffic Jam warning Pre-/Postcrash Warning
	SPaT, MAP, IVI	SPaT, MAP, IVI	Signal Phase and Time, MAP message, In-Vehicle-Information message	Enabling Infrastructure-to-Vehicle Communication at e.g. traffic lights
	VAM	PSM	VRU Awareness Message, Personal Safety Message	VRU warning for (C-ITS) equipped Vulnerable Road Users
Sensing Driving / sensor sharing	CPM	CPM	Collective Perception Message	Overtaking Warning Extended Intersection Collision Warning Vulnerable Road User Warning for non-equipped VRU´s Cooperative Adaptive Cruise Control Long-term Road Works Warning Special Vehicle Prioritisation
Cooperative Driving with Coordinated maneuvering and cooperative automated driving	MCM, PCM	MCM, PCM	Maneuver Coordination Message, Platooning Control Message	(Static or dynamic) Platooning Area reservation Cooperative Merging Cooperative Lane Change Cooperative Overtaking

Figure 51: Relationship of V2X applications to message types to fit with V2X application roadmap. CAM means Cooperative Awareness Message, CPM is Collective Perception Message, VAM is VRU Awareness Message, PCM is Platooning Control Message, MCM Manoeuvre Coordination Message, DENM is Decentralised Environmental Notification Message

3.2.6.3 Emitter and receiver models

- **Transmission Power**: for instance the specifications of MK5 OBUs/RSUs Cohda wireless modules is from -10 dBm to 23.0dBm
- **Receiver Sensitivity**: for instance the specifications of MK5 OBUs/RSUs Cohda wireless modules is -97.0dBm at 6Mbps
- **Transmission latency (Ts)**: This quantification reflects the latency generated by the transmitter due to the packet size (PS) and the transmission rate (TR): Transmission latency $(Ts) = PS/TR$.
- **Theoretical maximum baud rate (throughput)**: The number of bits which can be sent by second. This quantity depends of the technology used.
- **MIMO technology**: At the end of 2009, the 802.11n standard offered a maximum communication speed of 150 Mbps. The arrival of MiMo (Multiple Inputs Multiple Outputs) with the 2x2 MiMo made it possible to reach 300 Mbps using 2 receiving antennas and 2 transmitting antennas, hence "2x2 MiMo". We then find the 3x3 MiMo (3 antennas in reception, 3 antennas in transmission) capable of offering up to 450 Mbps and the 4x4 MiMo (4 antennas in reception, 4 antennas in transmission) with speeds of up to 600Mbps in 2 ,4 and 5GHz. Note that all these flow rates are theoretical, and might be different from what can be observed in real deployments.

3.2.6.4 Antenna diagram and modelling

A large set of antenna exists to perform telecommunications. In this document, we will focus our study and evaluation protocol to antennas dedicated to V2V and V2X communications. In this study, an antenna will be represented by the following quantities as well as the antenna diagram and modelling:

- **Radiation diagram**: 3 types of emission are possible, either directional or bidirectional, or omnidirectional broadcastings. This information is a graphical representation of the signal emitted by the antenna.
- **Radiation angle**: This is the beam-width of the antenna expressed in terms of its horizontal and vertical degrees. This number indicates the coverage area where the radiation pattern is emitted.
- **The antenna gain**: This quantity is given in dBi. This is a power measurement that represents how efficiently the antenna converts electricity into radio waves. Gain can affect the direction in which the antenna operates. The higher the gain, the more directional the antenna. Antenna power doubles for every 3dBi. Decibel-isotropic (dBi) is a hypothetical reference point where an isotropic antenna transmits a signal in a perfect sphere. It should be noted that a perfect sphere is impossible to create, so 0dBi is a practically impossible number.
- **The frequency** : The transmission frequency on which the transmitted message is modulated (in GHz). For WiFi and 802.11p (dedicated to automotive applications), mainly 2 frequencies are possible: (i) around 2.4 GHz and (ii) around 5 GHz. The width of

each channel is 20MHz. For (i), this frequency offers 13 channels ranging from 2400 to 2483.5MHz, the width of each channel is therefore approximately 20 to 22MHz. For (ii), this frequency offers 22 channels, from number 32 to number 140, ranging from 5150MHz to 5710MHz. As with all terrestrial frequencies, lower frequencies carry further but high frequencies generally offer more bandwidth and allow higher data rates. The 2.4GHz and 5GHz bands are high frequencies (UHF or Ultra High frequency). IEEE 802.11p standard typically uses channels of 10 MHz bandwidth in the 5.9 GHz band (5.850–5.925 GHz). At present cellular operators are opting for 3 ranges of 5G frequencies which can be distinguished as Low, Mid and High band. Low band frequencies include frequencies around 600-700 MHz, while mid band frequencies are around 2-5 GHz and High band frequencies are in mmWave range of 28-46 GHz.

- **Channel width**: The different channel widths for the different frequencies and WiFi standard are given in the figure 52. 3 main channel widths are used: 20MHz, 40MHz, and 80MHz.
- **The effective range and coverage area**: The effective range represents the range of message transmission between 2 static antennas without external disturbances. It means the distance from a transmitter (sending a message) and a receiver receiving this message with enough power. The coverage is the physical area in which a signal can still be received and transmitted. In general, the more powerful an antenna is, the more coverage it provides. However, the more powerful an antenna is, the more directional the signal becomes. For WiFi systems applied in C-ITS, the range could reach in favourable conditions 600 to 700 meters (depending of the propagation channel and the relative speed between the 2 antennas). But real experiments, demonstrates the range rather varies between 200 and 600 meters.
- **WiFi standard**: WiFi 1 to WiFi 6.
- **Effective Isotropically Radiated Power (EIRP)**: Approximate the actual power output at the antennas.

Norme Wi-Fi	Lancement	Fréquence	Largeur de canal	Débit maximum théorique	MiMo	Portée	Nom de la norme
802.11	1997	2,4GHz	20MHz	21Mbps	Non	20m	-
802.11b	1999	2,4GHz	20MHz	11Mbps	Non	35m	WiFi 1
802.11a	1999	5GHz	20MHz	54Mbps	Oui	35m	WiFi 2
802.11g	2003	2,4GHz	20MHz	54Mbps	Oui	38m	WiFi 3
802.11n	2009	2,4 ou 5GHz	20 ou 40MHz	72,2-450Mbps	Oui (max 4 antennes 2x2 MiMo)	70m	WiFi 4
802.11ac (1ère vague)	2014	5GHz	20, 40 ou 80MHz	866,7Mbps	Oui (max 4 antennes 2x2 MiMo)	35m	WiFi 5
802.11ac (2ème vague)	2016	5GHz	20, 40 ou 80MHz	1,73Gbps	Oui (max 8 antennes 2x2 MiMo)	35m	WiFi 5
802.11ax	Fin 2019	2,4 ou 5GHz	20, 40 ou 80MHz	2,4Gbps	-		WiFi 6E

Figure 52: WiFi standard with the main parameters

3.2.6.5 Propagation channel

Modelling of the environment in order to take into account collusion and reflection Transmission modelling depending on relative speed between emitter and receiver, lost rate, delay of transmission, short range reflection A study about the representativeness and the KPIs to be used for the communication modelling evaluation has been addressed in BPI SINETIC project.

The main parameters to take into account concerning the effect of the propagation channel on the communication efficiency are:

- **Average range**
- **Average jitter**
- **Average packet loss rate**
- **Average bit rate (throughput)**
- **Loss Rate depending of the number of communication nodes**
- **Received signal strength indicator (RSSI)**
- **Network capacity**
- **Average Latency** : This metric represents the delay in the reception of a message. This delay is due to the packet size PS, the transmission rate TR, and the number of packets NP. The $Averagelatency = (NP - 1)PS/(2 * TR)$. To this quantity, it is necessary to add some external disturbers like multi-reflection, and degraded conditions.
- **End-to-end latency**: the quartiles of the time interval between sending a message and its reception by the recipient.
- **Channel Busy Ratio (CBR)**
- **Packet Delivery Ratio (PDR)**
- **Packet loss rate / reception rate**: the ratio of packets sent to those received.
- **Effective Throughput**: The effective throughput refers to the real amount of data received by all vehicles receiving application messages per unit of time.

In [57], the authors have proposed an analytical model allowing to evaluate the broadcasting performance on CCH in IEEE 802.11p/WAVE vehicular networks. This model explicitly accounts for the WAVE channel switching and computes packet delivery probability as a function of contention window size and number of vehicles.

In [58], the authors propose an recent analytical model for 5G network with mode 2 that estimates the packet loss rate and the network capacity taking into account the peculiarities of Mode 2 and, in contrast to the existing models, provides the accuracy required in the emerging V2X scenarios. This model can be used to find the optimal transmission parameters that maximise the network capacity and/or to select the required bandwidth.

In [59], the authors propose an interesting set of analytical models that capture the performance of vehicle-to-vehicle (V2V) communications using the IEEE 802.11p standard. The models consider detailed representations of propagation, interference effects, and the hidden

terminal problem. They quantify the Packet Delivery Ratio (PDR) based on the transmitter-receiver distance and provide analytical models for estimating the probabilities of the four types of packet errors that can be encountered in IEEE 802.11p transmissions. Each with distinct classifications:

- **SEN Error (Sensing Error)**: Occurs when a packet is lost due to being received with a signal power below the sensing power threshold, preventing the initiation of the decoding process.
- **RXB Error (Receiver Busy Error)**: Packet loss happens if the received signal power is above the sensing power threshold, and the receiver is occupied decoding another packet.
- **PRO Error (Propagation Error)**: Packet loss due to propagation effects occurs when the received signal power is higher than the sensing power threshold, but the Signal-to-Noise Ratio (SNR) is insufficient for successful decoding.
- **COL Error (Collision Error)**: Packet loss can result from interference and collisions with other vehicles if the received signal power is higher than the sensing power threshold, the radio interface is free, but an interfering packet arrives during decoding. Such errors are classified as COL if not categorised under SEN, RXB, or PRO errors.

A packet is correctly received if none of these 4 identified types of error occur. The PDR can then be expressed as a function of the probability of each type of transmission errors. Moreover, this work introduces an analytical model for accurately estimating the Channel Busy Ratio (CBR) metric, even in scenarios with high channel load. Validation through simulation demonstrates the models' reliability across various parameters such as traffic densities, transmission frequencies, power levels, data rates, and packet sizes. This recent work proposed detailed and rigorous models which could be efficiently applied in simulation platforms.

3.2.7 Control Systems technologies and model requirements

Represents the electronic control units (ECUs) and control algorithms responsible for vehicle stability, traction control, anti-lock braking systems (ABS), and other advanced driver assistance systems (ADAS).

3.2.8 Driver Behaviour technologies and model requirements

Simulates human drivers' behaviour requires to implement in the model some specific human skills and capabilities like the perception limits, the decision-making, the reaction time, and the driving style, which influences vehicle operation and response in the simulation. Compared to a camera, the human eye is a "poor quality" sensor. As we can observe in Figure 53, the part producing precise information as a camera would do is very small. It is necessary to take into account this essential constraint which impact strongly the performance of the human perception and risk assessment.

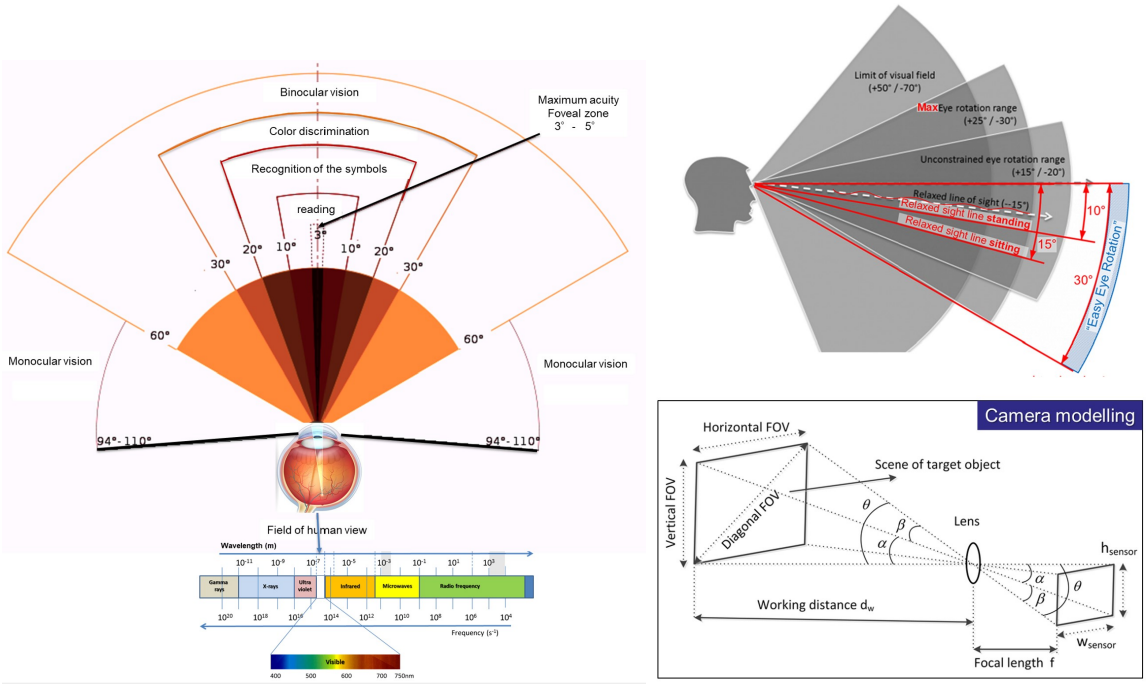


Figure 53: Modelling of the human eye with the different fields of view with their angles and functions. Bottom right provide the simple modelling of a RGB camera with the main parameters.

Creating a cognitive human driver simulation involves should consider various aspects of human behaviour, perception, decision-making, reasoning, action and interaction with the environment. Here are a set of requirements interesting for a cognitive modelling of a human driver:

- **Perception:** Simulation should model how drivers perceive their surroundings through visual and auditory senses. This aspect needs to take into account the eye model and the salient area detection by the driver. The eye capacities are represented by the figure ... and are really different from the camera capacities.
- **Attention**: Incorporate mechanisms for attention allocation, including focusing on relevant stimuli and filtering out distractions.
- **Memory**: Model short-term and long-term memory processes to simulate how drivers recall information, such as road signs, traffic rules, and past experiences.
- **Decision-making**: Simulate decision-making processes based on perceived information, past experiences, and situational factors.
- **Risk perception**: Include mechanisms for assessing and responding to perceived risks, such as hazards on the road, erratic driving behaviour, and adverse weather conditions.
- **Situation awareness**: Model the driver's understanding (with accuracy and certainty) of their environment, including the spatial layout of the road, nearby vehicles, pedestrians, and traffic signals.
- **Vehicle control**: Simulate the driver's control over vehicle speed, steering, braking, and acceleration, considering both manual and automated driving modes. This aspect takes

into account the capability of the driver to make several actions in same time or only one action by time. Capability to manage in same time a task additional to the driving task

- **Reaction time**: Incorporate realistic reaction times for responding to unexpected events, such as sudden braking by the vehicle ahead or obstacles on the road.
- **Driver physiology**: Model physiological factors that affect driving behaviour, such as fatigue, stress, and intoxication.
- **Emotions**: Consider the influence of emotions, such as frustration, anger, or excitement, on driving behaviour and decision-making. This aspect could change the risk perception and could modify the following risk-taking behaviour.
- **Risk-taking behaviour**: Model individual differences in risk tolerance and propensity for risky driving behaviours, such as speeding, tailgating, and aggressive manoeuvres. this aspect will impact the action of the driver (strong acceleration, braking, turn manoeuvre)
- **Adaptability**: Simulate the driver's ability to adapt to changing road conditions, traffic flow, and environmental factors.
- **Fatigue and drowsiness**: Model the effects of fatigue and drowsiness on driver performance, including decreased alertness, slower reaction times, and impaired decision-making. This aspect could be extended to the monotony and hypo-vigilance states.
- **Distraction**: Incorporate distractions inside and outside the vehicle, such as mobile phones, navigation systems, roadside advertisements, and scenery. This distraction will have an impact on the reaction time and the perception capacities (field of view shorter (angle and distance). This also leads to an inaccuracy and a uncertainty in the perception of the environment (worse object and situation positioning).
- **Anticipation**: Model the driver's ability to anticipate and to predict future events and plan actions accordingly, such as anticipating lane changes, turns, and merging manoeuvres. This means that the driver model is aware about specific situation no only in the first perception loop (vehicles and objets in the near perception field of view: closer vehicles surrounding the ego-vehicle)
- **Training and experience**: Consider the influence of driver training, experience, and skill level on driving behaviour and performance. This aspect implement the experience of young, middle age, and elder drivers.
- **Feedback and learning**: Provide feedback to drivers based on their actions and decisions, allowing for learning and improvement over time. This aspect allows to simulate the mechanism of training and reasoning of the driver in order to reach some specific goals: minimising of the risk, minimising of the energy consumption, ...
- **Traffic rules and regulations**: Ensure that drivers adhere to traffic laws, signals, and regulations governing speed limits, right-of-way, and road markings. A sportive behaviour will violate the rules of the traffic code: exceeding the speed limit, cutting a turn, overtaking manoeuvre in the right lane, etc.

- **Interaction with other road users**: Simulate interactions with pedestrians, cyclists, motorcyclists, and other drivers, including cooperative behaviours (strategies of negotiation), conflicts, and collisions.

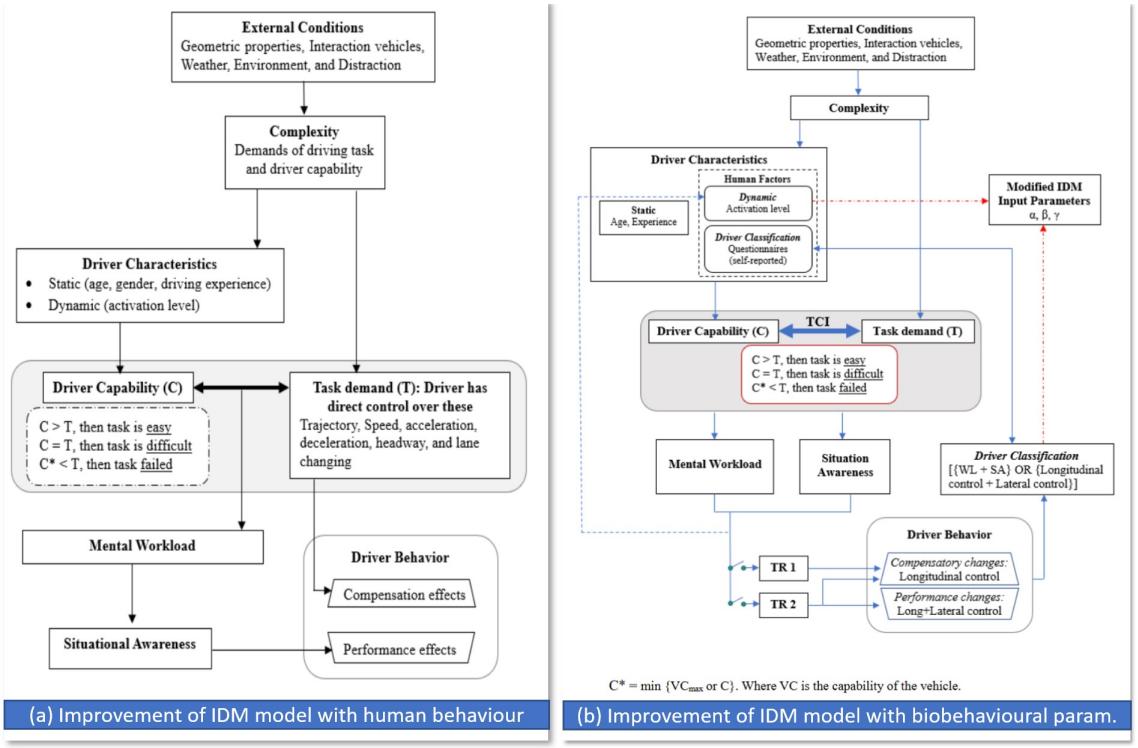


Figure 54: In (a), a framework developed by Hoogendoorn in 2012 to modify the IDM and involve some human behaviour characteristics. In (b), a theoretical framework for incorporating bio-behavioural human parameters.

In the figure 54 we can see some of this requirements with some additional parameters like the mental workload and situation awareness impacting the risk perception and the behaviour of the driver. Moreover, more specifically on reaction time aspect, a human information processing model has been previously proposed and shares the reaction time of a driver into three time steps:

- The perception time (Tp)
- The cognition time (Tc)
- The motor time (Tm)

The first two can be grouped together as one possible measurable step designated as perception–cognition time (Tpc = T p + Tc). [9] uses this method in a driving simulator experiment to generate probabilistic models on driver perception–response time among other things (see figure 55). They have conducted a driving simulator study with 37 male subjects (26 and 11 men aged 21–24 and 60–64 years old, respectively) to evaluate the perception, perception–cognition, and motor time during braking (among other things) for three groups of trained, untrained, and aged drivers. In their braking experiment, they divide the braking into two segments of Tp, time for perception, and Tm , time for motor action, e.g., for the execution of the brake by the driver (this Tp means the same Tpc from the first experiment). During perception time the vehicle is

still moving at a constant speed and during braking time, Tm , the vehicle speed reduces linearly to zero until the stopping point. The braking was done at 80 km/h on the onset of observing a random truck and the drivers were to complete the braking action immediately to achieve the shortest stopping distance.

	T_p (s)	T_{pc} (s)	T_m (s)
Trained subjects			
Median	0.473	0.586	2.404
Mean	0.481	0.599	2.663
Untrained subjects			
Median	0.512	0.729	2.440
Mean	0.523	0.717	2.379
Aged subjects			
Median	0.616	0.870	2.547
Mean	0.599	0.869	2.504

Figure 55: Estimated values of reaction time at 80 Km/h for 3 types of drivers in a driving simulator ([9])

Figure 56 presents the conceptual framework of the lane changing process. The main idea is that drivers will change lanes if they decide to respond to lane changing encouraging conditions whether they are mandatory lane changing conditions or discretionary lane changing conditions. The way the driver will respond is modelled using the random utility approach. Panel data is used to estimate the corresponding model.

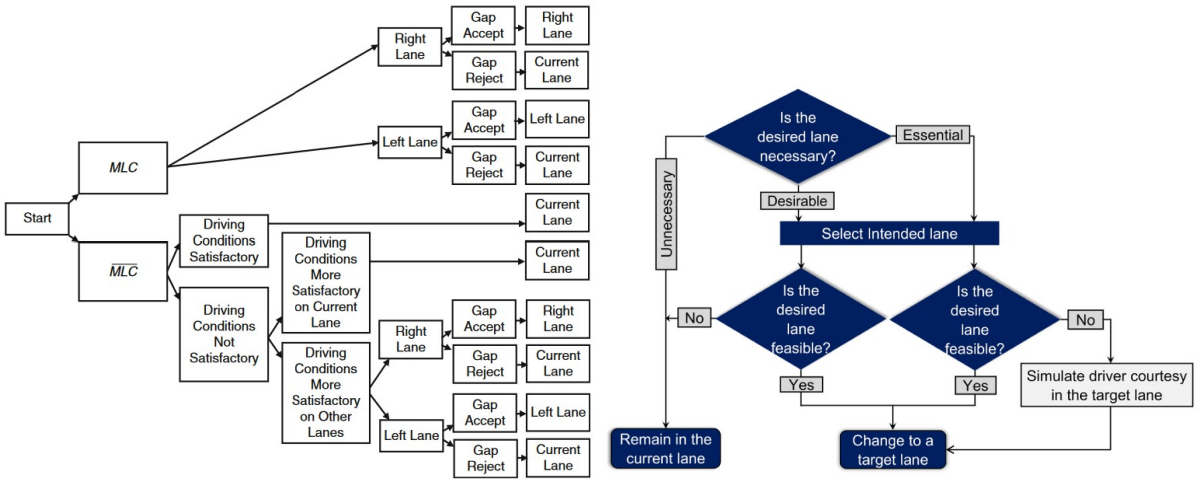


Figure 56: MITSIM's lane changing framework ([10]) and Lane Changing SITRAS model ([11])

The human driver embodies a sophisticated cognitive architecture that processes sensory information in real-time while operating a vehicle. This cognitive process involves a series of interconnected functions, including sensing, perception, memorisation, decision-making, response selection, and execution. Each of these functions is influenced by individual human characteristics, such as experience, attention, and emotional state.

In the context of driving, these cognitive processes manifest through various mental resources or behaviours, which can be categorised into knowledge-based, rule-based, and skill-based actions. These actions are intertwined with the hierarchical structure of driving tasks, as proposed by Michon or Donges. The driver seamlessly navigates between these levels of hier-

archy to adapt to the dynamic demands of the driving environment and achieve personal driving objectives.

At the core of this cognitive architecture is the driver's ability to apply automated driving actions that are suited to the prevailing traffic scenario and individual driving goals. This entails continuous monitoring of the surrounding environment, assessing road conditions, and aligning personal motivations with the demands of the road. Ultimately, the driver's cognitive processes facilitate safe and efficient navigation through the complexities of the driving environment.

An overview of this full and complex framework is given in and is presented in the figure 57.

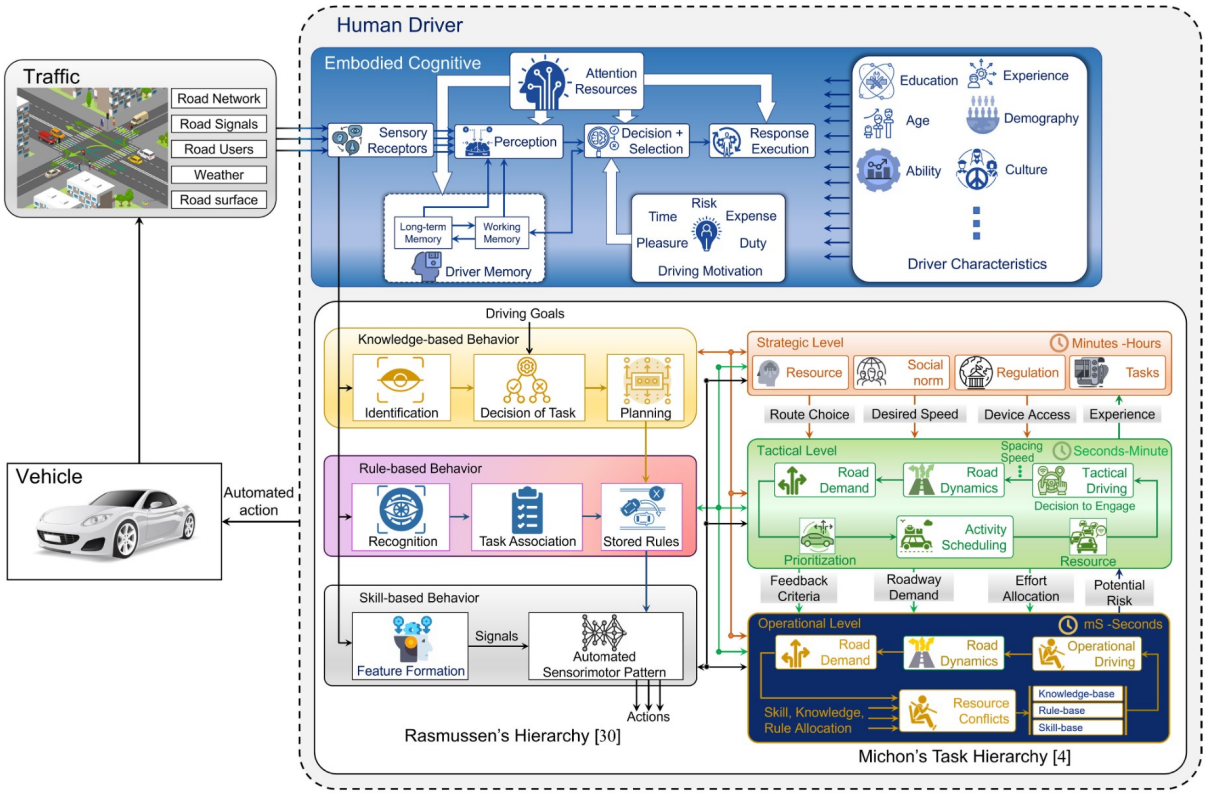


Figure 57: Driving Cycle and the Driver Modelling Conceptual Framework ([11])

No.	Human driver	AVs
1	Human perceptual capability.	Perception through the camera, lidar, and sensors.
2	Individual risk analysis and acceptance choice.	No choice or emotions towards a quantified risk.
3	Rely on daily mood, motive, attentiveness, and emotional load, etc.	Rely on sensors, actuators, complex machine learning algorithms, and processors.
4	Knowledge, skill, and rule-based driving mode.	Driving based on data-driven control algorithms.
5	React to emergencies within a second.	React to emergencies within 30 s due to algorithm constraints.
6	Expert drivers can identify and avoid dangerous drivers.	Don't respond to aggressive, dangerous drivers and do not counteract.

Figure 58: Major differences between human driver and autonomous vehicles. ([11])

3.2.8.1 Driving and driver modelling with Reinforcement Learning (RL)

RL is a decision-making algorithm based on Markov Decision Process (MDP), a stochastic control process where an agent learns to make decisions in an uncertain and complex environment. The environment E is essentially an MDP consisting of a set of states S, a set of action A, a set of reward R, and a state transition probability $P(s' \mid s, a)$, which denotes the result of action $a_t \epsilon A$ in the state $s_t \epsilon S$ resulting s' at time $t+1$. The agent (AV) performs a set of actions in the environment based on a policy π, receives a reward r, and transitions to a new state $s' \sim s_{t+1}$. Thus, the goal of the RL agent is to learn an optimal policy π^*, which maximises the long-term cumulative reward $R_t(s, a)$. The optimal policy can be learned from the available reward function through Deterministic Policy Gradient (DPG), Twin-Delayed deep deterministic policy gradient (TD3), Proximal Policy Optimisation (PPO), Trust Region Policy Optimisation (TRPO), Q-learning, State-Action-Reward-State-Action (SARSA) algorithms, etc. Figure 59 provides an overview and the main principles of RL, Inverse RL (IRL), Behavioural Cloning (BC), and Direct Policy Learning (DPL). In the sub figure a) about RL, the environment consists of transition probability of states and a known reward function, and at an instant time, the RL agent applies an action a on the environment and receive a reward r based on a policy $\pi(s, a)$. The final goal of the RL agent is to learn an optimal policy $\hat{\pi}$ to achieve the desired trajectory. In sub figure b), BC is an approach to imitate the expert's trajectory and employs a supervised learning approach to minimise a loss function to obtain the approximated policy that mimics the trajectories of the expert. In sub figure c), IRL is the inverse analogy of RL where the agent extracts the reward function that defines the intention of the expert driver by looking into the trajectories from a demonstration, and in sub figure d) DPL is the improved version of BC where the agent cross-validate the accuracy of the prediction with the expert driver during training.

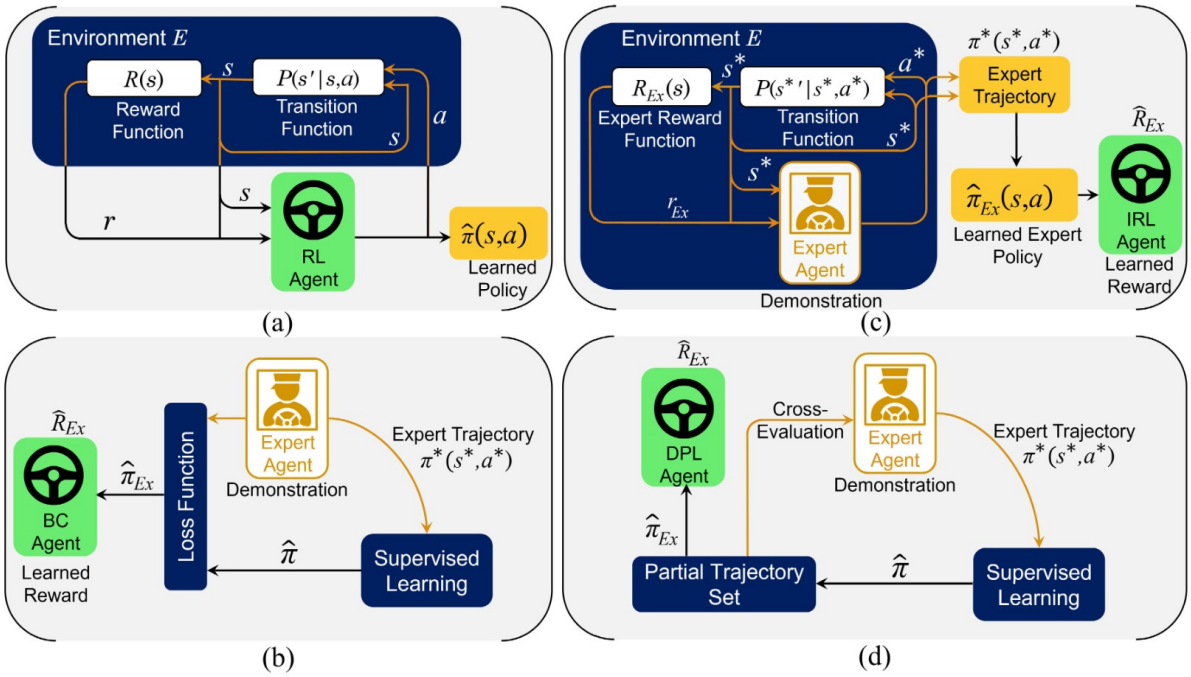


Figure 59: Major differences between human driver and autonomous vehicles. ([11])

3.2.9 Traffic technologies and model requirements

Simulates other vehicles, pedestrians, and entities interacting with the simulated vehicle. It includes models for vehicle movement, traffic patterns, and interactions with the environment.

Modelling the surrounding and obstacle vehicles around the Ego vehicle enables to explore relevant and usual dynamic scenarios.

Developing a realistic and high quality and performances traffic generator for autonomous driving involves meeting several critical high-level requirements to create realistic and diverse traffic scenarios. Here are the main requirements to consider:

- **Traffic Pattern Variety**: Generate diverse traffic patterns, including highway, urban, suburban, and mixed scenarios, to simulate real-world driving conditions. This implies to have Digital Twin and HD Maps of specific representative areas. The respect of this requirement is essential to provide a comprehensive set of scenarios for testing and validating autonomous driving systems.

- **Dynamic Vehicle Behaviour**: Simulate dynamic and realistic behaviours for different types of vehicles, including acceleration, deceleration, lane changes, and interactions with other road users. This requirement is important to evaluate the adaptability and responsiveness of autonomous systems in complex traffic situations.

- **Realistic Vehicle Types**: Include various vehicle types, such as cars, trucks, motorcycles, bicycles, and pedestrians, to reflect the diversity and the complexity of road users. This requirement allows to address the capability of autonomous systems to interact with and respond to different types of vehicles.

- **Traffic Density Control**: Enable control over traffic density to simulate both sparse and congested scenarios. This requirement is essential to evaluate system performance under varying traffic conditions, including peak hours and low-traffic situations.

- **Traffic Light and Sign Simulation**: Model realistic traffic light and road sign behaviour, including changes in signal timing, yellow light intervals, and adherence to traffic rules. This requirement will be used to assess the interaction of autonomous vehicles with traffic control infrastructure and generate specific risky and critical scenarios allowing to address interaction between AV and the other road users in intersection areas.

- **Pedestrian and Cyclist Behaviour**: Simulate realistic pedestrian and cyclist behaviours, including jaywalking, crossing at intersections, and interactions with vehicles. This type of scenarios allow to evaluate the ability of autonomous systems to navigate safely in the presence of vulnerable road users.

- **Adaptive Road Conditions**: Incorporate adaptive road conditions, such as changes in weather (rain, snow) and road surface conditions (dry, wet, slippery). This requirement is essential in order to guarantee a large coverage of the situations (environmental factors generating modifications and variations of the road environments) which could be encounter by AV.

- **Simulated Road Events**: Introduce mechanisms to simulate road events, such as accidents, construction zones, road work areas, temporary conditions (object falling on the

road surface) and detours, to assess the response and decision-making of autonomous vehicles. This requirement and the generation by the traffic generator of this events will give the possibility to assess the ability of autonomous systems to handle unexpected events and deviations from regular traffic conditions.

- **Scenario Customisation**: Allow users to customise specific traffic scenarios, including the introduction of specific vehicles, road configurations, and event triggers. The implementation of this requirement facilitate targeted testing for specific use cases and scenarios relevant to the development and validation process.

- **Scalability and Performance**: Ensure that the traffic generator can scale to simulate large-scale scenarios while maintaining computational efficiency. Application of this requirement allows to guarantee the capability to replicate realistic traffic conditions in a scalable manner to assess the scalability and performance of autonomous driving systems.

By meeting these requirements, a traffic generator for automated mobility (involving systems of systems, or AI-based systems) can provide a versatile and realistic environment for testing and validating autonomous systems in a wide range of scenarios.

3.2.9.1 Definition of the types of models and components involved

Traffic generation in simulation relies on several types of models aiming to mimic various components of the surrounding traffic composed of obstacle vehicles accurately. Some models focus on replicating the traffic flow dynamic with a microscopic description of the interactions between vehicles [60], while others aim at describing how a predetermined and nominal trajectory might be affected by human errors or approximations when driving a vehicle [61].

Usually, the distinction between models is developed as follows:

- **Random traffic generation:** Vehicles and their dynamic properties are randomly generated to randomly populate the environment around the Ego. With this easy-to-implement and basic approach, nothing ensures that the traffic dynamic is adequately reproduced since trajectories are predetermined and do not respond to traffic.

- **Microscopic traffic rule-based model:** Incorporating traffic rules and regulations regarding road signalization. Such models ensure a high representativeness level of the traffic flow dynamic for the obstacle vehicles populating the environment around the Ego. Among the set of of existing simulators, we can mention SUMO, VISSIM, AIMSUN Next, and SymuVia. Vrbanic et al. [62] compared three highly popular traffic flow simulators (SUMO, VISSIM, and AIMSUN Next) paired with telecommunication network simulators. They analyze the features related to the modeling process They conclude that traffic environments generated by AIMSUN Next are more suitable to achieve a traffic model with a low complexity, while VISSIM enables to cope with high complexity levels. Despite their distinction according to modeling options and configuration tools, simulator frameworks are mainly discriminated according to the natively hosted car-following models, especially those used to describe the vehicles' longitudinal movement. Mahapatra et al. [63] distinguish the following car-following approaches:

  - Analytical models, covering the following approaches:

    - Gazis-Herman-Rothery (GHR) approach [64]: this approach states that the driving behavior of vehicles mainly depends on speed differences between two successive vehicles and their free flow regime. This approach is hosted by MIT-SIM (MIcroscopic Traffic SIMulator) framework.
    - Safe distance [65] and psycho-physical [66] approach: this approach describes the car-following movements with the physical perspective, by including physical motion equations, based on a safe following distance. Accordingly, in such a model, a collision should be expected when the following vehicle infringes on the safe gap with its leader. This is one of the most popular car-following approaches implemented into microscopic traffic simulators. It encompasses a set of alternative models: Gipps'model [65] (by default in AIMSUN), Krauss models [67] (by default in SUMO)... Some were tested and compared by Brockfeld et al. [68]. This approach is hosted by the following frameworks: AIMSUN, CORSIM, CARSIM, SimTraffic, NETSIM, SUMO. Alternatively, VISSIM framework is mainly based on a psycho-physical approach, like Wiedemann models [66].
    - Intelligent Driver Model [69]: it is a time-continuous car-following model developed as an extension of Gipps' model [65] by Treiber [69] to cope with the losses of accuracies observed in the deterministic limits of Gipps' model.
    - Optimal Velocity approach: based on Bando et al. [70] works, this approach formulates an interesting description of the vehicle interaction using only the relative spacing and the desired velocity of the leader and follower. It becomes popular due to its simple formulation and its single-variable function.
  - Rule-based models, covering the following approaches:
    - Fuzzy-logic approach [71];
    - Cellular Automata approach [72]: this approach discretizes space and time into small cells (typically 7,5 meters long cells) occupied or not by a vehicle. Such models describe the transition dynamic from one cell to another, which is computationally efficient for large scale simulation but less realistic of microscopic movements.
    - Newell approach: this model, described by Newell [73], can be understood as a continuous cell-transmission model. This is the main car-following model hosted by the SymuVia simulator.

- **Behavioural models:** By representing various driving behaviours, this kind of traffic can have various outcomes and trigger diverse conditions. [61] develops a behavioural model enabling the reproduction of mistakes or misbehaviours of human drivers. It enables to finely reproduce the impact of human-driven obstacle vehicles (errors) on the Ego vehicle. Nevertheless, in opposition to the traffic rule, the trajectories of the obstacle vehicles do not respond dynamically to current traffic conditions and are completely determined at the beginning of the simulation

- **Manoeuvre-based model:** This kind of model focuses on specific scenarios to reproduce realistic simulations of complex traffic. Trajectories of the obstacle vehicles are accurately computed and predetermined to match some specific scenarios, usually identified as critical.

All of these models can be mixed to replicate the different types of behaviour with more or less randomness and diversity. Many subjects focus on the representativeness of such models by comparing them to real driving datasets. Adding Artificial Intelligence modules to such models can help increase realism, representativeness, and diversity of traffic generation. The figure 60 provides an overview of the main categories of car-following models for Human Driver simulation.

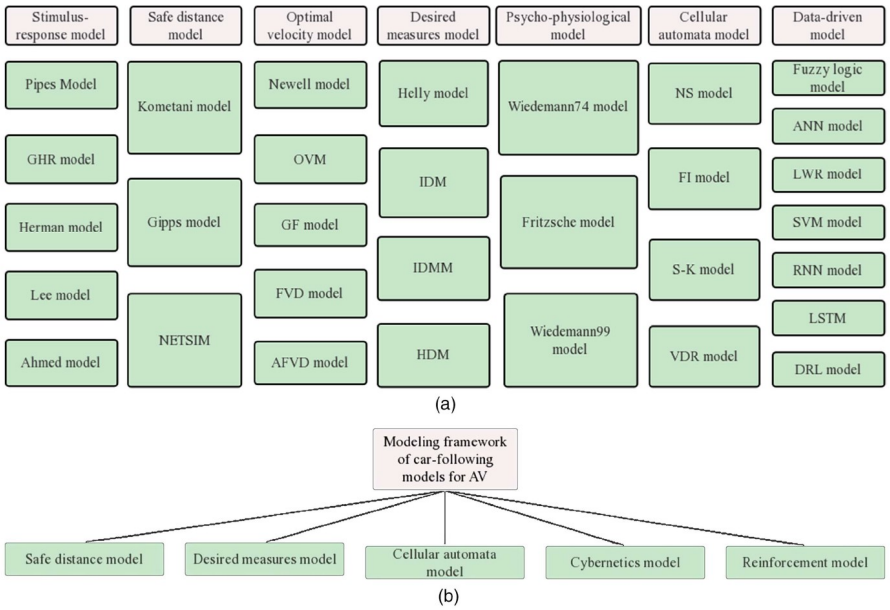


Figure 60: Summary of car-following models for HDVs and AVs: (a) development process of car-following models for HDVs; and (b) modelling framework of car-following (CF) models for AVs ([12])

3.2.10 Requirements for a Simulation Framework dedicated to Automated Mobility

Provides the infrastructure to integrate and manage different models, components, and simulations in a cohesive environment. This includes simulation rendering and physics engines.

A comprehensive simulation platform for automated vehicle evaluation and validation involving AI-based components requires careful consideration of various requirements to ensure its effectiveness, flexibility, and usability. Here's an extended overview of the main requirements expected for such a simulation platform:

- Usability: The platform should offer a user-friendly interface and intuitive interaction mechanisms to enable users to achieve their goals efficiently. Well-documented instructions, meaningful examples, and a supportive community enhance usability. Additionally, providing access to enrichments from existing ecosystems like CARLA, Pro-SiVIC, SCANeR, Pre-Scan, Car-maker, along with well-documented demonstrations for various use cases, further improves usability.

- Maintainability: Customization and continuous updates are essential for keeping the simulation platform relevant and adaptable to evolving needs. A modular and containerized architecture facilitates the seamless exchange or update of individual components. Open-source software eliminates potential constraints imposed by licensing regulations. Leveraging modular containerization enables continual updates of specific modules, with Docker images built within GitHub CI pipelines to enhance maintainability.

- Interoperability: The platform should support seamless integration with custom functions and external systems. This entails compatibility with standardized and custom interfaces, as well as containerized integration. While existing interfaces like ROS or OpenX simplify integration in existing platforms. The PRISSMA's distributed and multiple tools architecture further streamlines integration by leveraging container orchestration tools. This enables easy integration of complex software systems composed of multiple containers.

- Scalability and Automation: The ability to scale simulations and automate processes is crucial for efficient development and testing. PRISSMA's generic platform focuses on automating simulations using container orchestration tools and CI pipeline integrations. Sequential execution of multiple simulations within the data generation pipeline, along with support for parallel and distributed simulations facilitated by powerful orchestration tools, significantly enhances scalability compared to traditional approaches.

- Test Capability: Effective evaluation methods and metrics are essential for ensuring the reliability and safety of automated vehicle systems. The simulation platform should support a wide range of evaluation methods, including existing evaluation metrics and custom evaluation modules tailored to specific simulation use cases. This enables comprehensive testing and validation of automated vehicle systems within a simulated environment.

- Customization and Extensibility: The platform should allow users to customise simulation scenarios, vehicle models, environmental conditions, and other parameters to meet specific testing requirements. It should also support the integration of custom algorithms, sensor models, and control strategies to enable advanced research and development.

- Realism and Fidelity: Achieving a high level of realism in simulated environments is crucial for accurate evaluation of automated vehicle systems. The platform should provide realistic physics simulation, dynamic environmental conditions (such as weather and lighting), and detailed sensor models to replicate real-world scenarios as closely as possible.

- Data Management and Analysis: Efficient data management capabilities are essential for storing, managing, and analysing large volumes of simulation data generated during testing. The platform should support data logging, visualisation, and analysis tools to extract meaningful insights and facilitate decision-making.

- Safety and Security: Ensuring the safety and security of both the simulated environment and the simulation platform itself is mandatory in a ViL real/virtual simulation platform. The platform should incorporate robust safety measures to prevent unintended consequences of simulations, such as collisions or system failures. Additionally, it should adhere to security best practices to protect sensitive data and prevent unauthorised access.

- Scalable Infrastructure: As simulation requirements grow, the platform should offer scalable infrastructure options to accommodate increasing computational demands. This may include support for cloud-based deployment, distributed computing, and integration with high-performance computing (HPC) resources to scale simulations effectively.

By addressing these requirements, a simulation platform can provide a robust foundation for automated vehicle evaluation, offering usability, maintainability, interoperability, scalability,

automation, comprehensive test capabilities, and optimisation of autonomous driving systems in a simulated environment.

Nevertheless the last aspect about Scalable Infrastructure brings the necessity to propose some additional and essential requirements related to time management, synchronisation of models and tools, real-time operating capabilities, and ensuring the respect of component operating periods:

- Time Management and Synchronisation: The platform should provide accurate time management capabilities to ensure synchronization between different simulation components, models, and tools. It should support both simulated time and real-world time modes, allowing users to control the simulation timeline and ensure consistency across simulations.
- Real-Time Operating Environment: For applications requiring real-time simulation, the platform should offer real-time operating capabilities to ensure timely execution of simulation tasks. This includes minimising latency, optimising computational performance, and providing deterministic behaviour to meet real-time constraints.
- Synchronisation of Models and Tools: The platform should facilitate seamless integration and synchronisation of diverse simulation models, tools, and components. This includes synchronisation of vehicle dynamics models, sensor models, control algorithms, and external software interfaces to ensure accurate and coherent simulation outcomes.
- Respect of Component Operating Periods: To accurately replicate real-world conditions, the platform should respect the operating periods of individual components within the simulation. This includes modelling the lifecycle and operational characteristics of sensors, vehicles, and other system components, such as battery life, maintenance schedules, and wear and tear effects.
- Dynamic Time Scaling: The platform should support dynamic time scaling capabilities, allowing users to adjust the simulation speed in real-time based on specific testing requirements. This enables users to accelerate or decelerate simulation time to focus on critical events or extend simulation duration for long-term testing scenarios.
- Temporal Accuracy and Precision: Ensuring temporal accuracy and precision is essential for realistic simulation outcomes. The platform should provide high-fidelity timekeeping mechanisms, with support for fine-grained time resolution and synchronisation across distributed simulation nodes.

By incorporating these additional requirements, the simulation platform can offer enhanced capabilities for time-sensitive applications, ensuring accurate, synchronised, and real-time simulation of automated vehicle systems.

Finally, it is mandatory in the last stage of the simulation framework to add requirements dedicated to ground truth and reference generation, dataset generation, recording of data, and processes for analysis, evaluation, and validation of the results:

- Ground Truth and Reference Generation:
  - Ground Truth Annotation Tools: Develop tools for annotating ground truth data, including accurate labelling of object classes, semantic segmentation, instance segmentation, and geometric annotations (e.g., bounding boxes, keypoints) to create reference datasets.

  - Sensor Fusion Ground Truth: Generate ground truth data for sensor fusion algorithms by integrating data from multiple sensors (e.g., LIDAR, camera, radar) to create comprehensive reference data with precise object localisation and tracking information.
  - High-Fidelity Simulation Environments: Utilise high-fidelity simulation environments with realistic physics models, dynamic lighting conditions, and detailed scene geometry to generate ground truth data that closely resembles real-world scenarios.
  - Dynamic Environment Simulation: Simulate dynamic environmental factors, such as weather conditions, traffic flow, pedestrian behaviour, and road infrastructure changes, to create diverse and challenging scenarios for ground truth generation.

- Dataset Generation and Recording:

  - Scenario Generation Tools: Develop tools for generating diverse and representative scenarios, including urban, highway, rural, and off-road environments, with configurable parameters for traffic density, road conditions, weather, and time of day.
  - Sensor Data Recording: Implement mechanisms for recording sensor data streams, including raw sensor measurements, synchronised timestamps, and metadata annotations, in standardised formats (e.g., ROS bag files, HDF5) for dataset generation and offline analysis.
  - Data Augmentation Techniques: Incorporate data augmentation techniques, such as geometric transformations, color manipulations, occlusions, and sensor noise injection, to enhance dataset variability and robustness for training and validation purposes.
  - Scenario Metadata Annotation: Annotate scenario metadata, including scenario description, ground truth labels, environmental conditions, vehicle trajectories, and sensor configurations, to facilitate dataset organisation, retrieval, and analysis.

- Analysis, Evaluation, and Validation Processes:

  - Performance Metrics Definition: Define performance metrics and evaluation criteria for assessing automated driving algorithms, including object detection accuracy, tracking precision, localisation error, trajectory prediction accuracy, and scene understanding metrics.
  - Quantitative Analysis Tools: Develop tools for quantitative analysis of simulation results, including statistical analysis, performance benchmarking, and comparison against baseline algorithms or ground truth reference data to measure algorithm performance and identify areas for improvement.
  - Visual Inspection and Qualitative Analysis: Enable visual inspection and qualitative analysis of simulation results through interactive visualisation tools, 3D scene rendering, and playback of recorded sensor data to identify anomalies, errors, or unexpected behaviours in automated driving systems.
  - Validation Framework Integration: Integrate validation frameworks, such as the Continuous Integration/Continuous Deployment (CI/CD) pipeline, to automate the execution of validation tests, regression testing, and performance monitoring for iterative algorithm refinement and validation.

– Benchmarking and Standardisation: Establish benchmarks and standards for evaluating automated driving algorithms, datasets, and simulation frameworks, fostering collaboration, reproducibility, and fair comparison of results across research studies and industry implementations.

By adhering to these requirements, a simulation framework for automated vehicles can facilitate the generation of ground truth data, creation of representative datasets, recording of sensor data, and enable comprehensive analysis, evaluation, and validation of automated driving algorithms and systems.

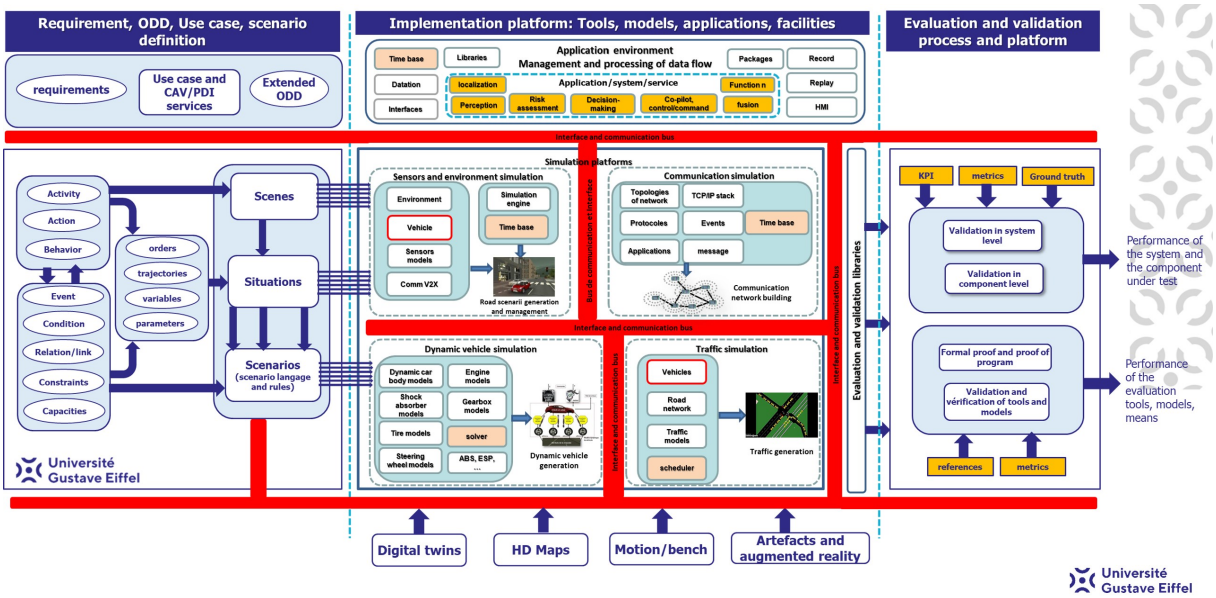


Figure 61: Overview of the full simulation framework for the PRISSMA's evaluation and validation methodology (source UGE)

3.2.11 User Interface and Visualisation Tools requirement

The main requirement for a user interface and visualisation tool are:

- **User-friendly Interface** : The interface should be intuitive and easy to use, allowing users to interact with the simulation environment without extensive training. The interface need to provide common functionalities like object, scene, and scenario loading, saving, reloading.
- **Customizability**: Users should have the ability to customise the interface layout, settings, and features according to their preferences and specific simulation requirements. It means Modular and adaptive User friendy interface. A User Preferences Management function could provide options for users to customise their interface preferences, such as theme selection, language settings, and default layout configurations. Allow users to save and restore personalised settings across sessions for a consistent and personalised user experience.
- **Real-time Feedback**: The interface should provide real-time feedback on the simulation status, data updates, and any user inputs to ensure smooth interaction.

- **Visual Representation**: Utilise clear and visually appealing graphics, including charts, graphs, and 3D models, to represent simulation data and scenarios effectively.
- **Interactivity**: Allow users to interact with simulation elements, such as adjusting parameters, initiating actions, and controlling simulations, through the interface. It also means peripherals management and updating in real time during the simulation running, and "script language" (Python, LUA, ...) allowing to modify in real time the operating and the behaviour of the simulation, the objects and components involve in a scenario. The menu needs to implement mechanism of completion (access to the previous commands or proposition of possible commands from only the first characters of a command). Last aspect of interactivity concerns the capability to handle and move the current state and configuration of a 3D object (position, rotation, homothety) with the using of a peripheral (mouse, joystick, keyboard, ...) in the working window by a selection-clicking-moving mechanism.
- **Data Visualisation**: Provide various visualisation tools and techniques to interpret complex simulation data, including heatmaps, histograms, and animations. It also means a set of available functional windows and layer mechanism in order to have in same time a working space and window, some windows for the sensors and components data display, and windows for references and ground truth display. Windows with the main parameters of downloadable elements and components (3D object, lights, vehicles, pedestrians, weather conditions, ...)
- **Multi-platform Compatibility**: Ensure compatibility across different platforms and devices, including desktop computers, tablets, and mobile devices, to facilitate access for users.
- **Accessibility**: Design the interface to be accessible to users with disabilities, incorporating features such as screen readers and keyboard shortcuts.
- **Performance Optimisation**: Optimise interface performance to minimise latency and response times, especially in large-scale simulations with extensive data processing. It also means a mechanism allowing to manage efficiently with event management the writing of scripting commands during the simulation process.
- **Error Handling**: Implement robust error handling mechanisms to detect and report errors, guiding users with troubleshooting steps and preventing data corruption.
- **Security**: Incorporate security measures to protect sensitive simulation data, including user authentication, data encryption, and access control.
- **Documentation and Help Resources**: Provide comprehensive documentation, tutorials, and help resources within the interface to assist users in navigating and utilising simulation features effectively.
- **Integration Capability**: Support integration with external systems, databases, and software tools to enable seamless data exchange and interoperability.
- **Scalability**: Ensure that the interface can scale up to accommodate large datasets, complex simulations, and increasing user demands without compromising performance.

- **Feedback Mechanism**: Include mechanisms for users to provide feedback, suggestions, and bug reports directly within the interface, facilitating continuous improvement and user engagement.

Here are the additional requirements aiming to enhance the usability, performance, and flexibility of the Human Machine Interface in a simulation environment.

- **Specific Display Modes**:
  - Resolution Control: Allow users to adjust the resolution of the interface to suit their display preferences and optimize performance.
  - Full-Screen Mode: Provide an option for full-screen mode to maximise the visibility of simulation data and eliminate distractions. This mode is essential if we want to obtain immersive aspect with a real human in the loop (driver, passenger, road users).
  - Low Rendering Mode: Offer a low rendering mode for users with lower-end devices or limited graphics capabilities, ensuring smooth operation without sacrificing essential features. This mode is also usefulness in specific topics involving only no visual aspect (vehicle dynamic and vehicle control)
- **Operating System Compatibility**: Ensure compatibility with various operating systems, including Windows, macOS, and Linux, to accommodate a diverse user base. Support for different versions of operating systems and provide regular updates to maintain compatibility with the latest releases. It is usefulness for multiple platforms simulation framework like Symuvia (Linux or Windows), NS3(Linux), Pro-SiVIC (Windows), RTMaps (Windows or Linux), ROS (Linux) architecture. This requirement involved a user friendly management of multiple OS interfaces and Virtual Machine management.
- **Event Management, logging and analysing**: Implement event handling mechanisms to capture user interactions, system events, and simulation events effectively. Allow users to define custom events, triggers, and actions within the interface to enhance simulation control and flexibility. Enable logging of simulation events, user interactions, and system activities for later analysis and troubleshooting. Offer tools for analyzing event logs, identifying patterns, and extracting insights to improve simulation design and performance.
- **Time and task Management**: Provide tools for managing simulation time, including pausing, resuming, and adjusting the simulation speed. Enable synchronisation with real-time clocks or external time references for accurate timekeeping in simulations. Support for scheduling and automation of simulation events based on specific time intervals or triggers. Implement task scheduling and prioritization features to manage concurrent tasks, background processes, and resource allocation effectively. Allow users to track the progress of ongoing tasks, prioritize critical activities, and allocate resources based on priority levels.
- **Performance Monitoring**: Include performance monitoring features to track resource utilisation, such as CPU, memory, and GPU usage, during simulation execution. Provide real-time feedback on performance metrics and optimisation recommendations to ensure efficient utilisation of system resources.

- **Concurrency Control**: Support concurrent access to simulation data and resources by multiple users, ensuring data consistency and integrity. Implement locking mechanisms, transaction management, and conflict resolution strategies to handle concurrent updates and prevent data corruption. This aspect is essential for distributed simulation using cloud functionalities and multiple user mechanism.

In order to have a user friendly simulation environment, it is necessary to have a set of HMI functionalities. In order to have a good overview about these needed function, we will take in consideration the current HMI functions in the software and tools bring in PRISSMA by the partners. For instance, in Pro-SIVIC, a generic graphical user interface is proposed (illustrated in Figure 62). This HMI is divided into the following user interface elements:

- 1: The simulation view. This is the most important element, that will allow you to visualize a scenario when you edit it, as well as a running simulation;
- 2. The menu bar, that enables to perform important actions on the scenario, the view, the simulation, and to access help and configuration options;
- 3. The toolbar, which gives quick access to some menu bar functions that are used extensively. The toolbar itself is divided into three subsections:
  - The scenario toolbar provides shortcuts to load, save, or discard the current scene and begin a new scenario;
  - The object manipulation toolbar allows to create and delete objects, as well as editing their position, rotation, and scale; you can also choose the coordinate system and the 2D plane in which to move and orient the object;
  - Finally, the simulation toolbar allows to play, pause or stop the simulation, as well as enabling/disabling the scenario editing mode
- 4. The scene objects panel lists all the items contained in the simulation scenario, be it the sensors, the lighting elements, or the scenery (buildings, vegetation, sky);
- 5. When selecting an object by clicking on it either in the scene objects panel or in the simulation view, its properties will be displayed in the object configuration panel, to visualise and edit them in order to tweak the object behaviour; the effects can then be viewed in real-time in the simulation view;
- 6. The object catalogue contains a list of preconfigured scenario elements (sensors or scenery items for example) that can be added to the current scenario by double-clicking on an item;
- 7. The console panel is targeted for advanced users; it allows to manipulate the scenario and the objects using the Pro-SiVIC scripting language; it provides the same functionality as the graphical user interface; albeit in some particular cases, specific actions can currently only be accomplished using the console.

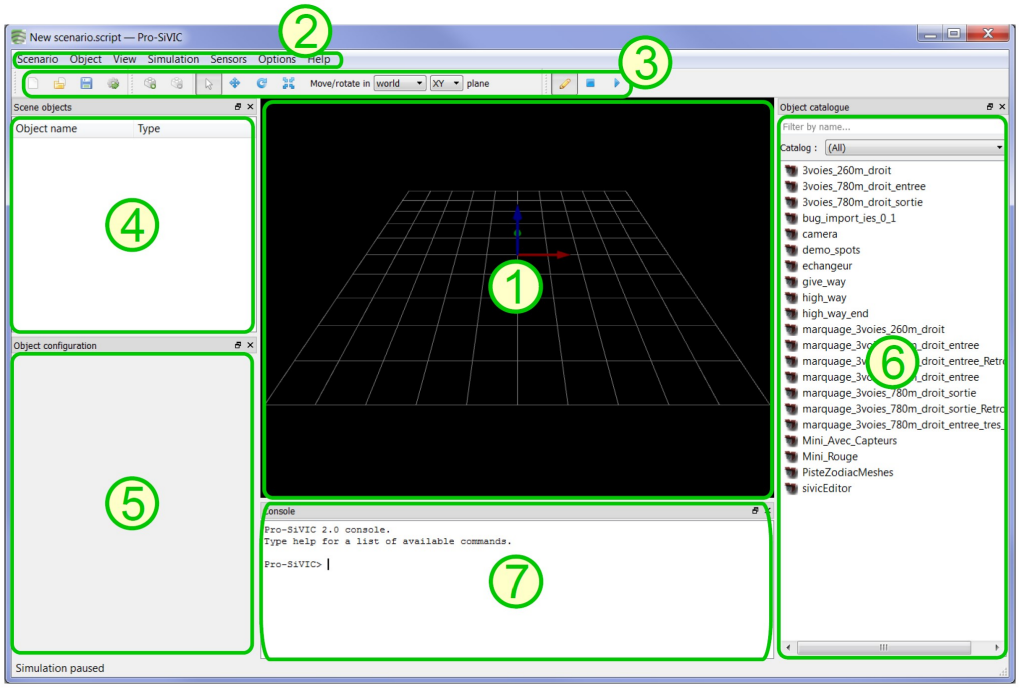


Figure 62: Pro-SiVIC user interface (source ESI group)

Pro-SiVIC offers too some additional function in the user interface like:

- The peripheral management and updating
- The trajectory generation and management for object without dynamic model
- The commands to manage the simulation process (start, stop, pause): Once the building of the scenario is finished, the next step is to run the simulation. The Edit mode toolbar button is then usable to disable the Scenario Designer-specific views (Figure 63. From this interface the following functions are available: the Play button to play and stop the simulation, and the Pause button can Pause the simulation while it is running. The Stop button will reset the simulation environment and reload the default state (before the simulation was launched), so that it can be modified or launched again.

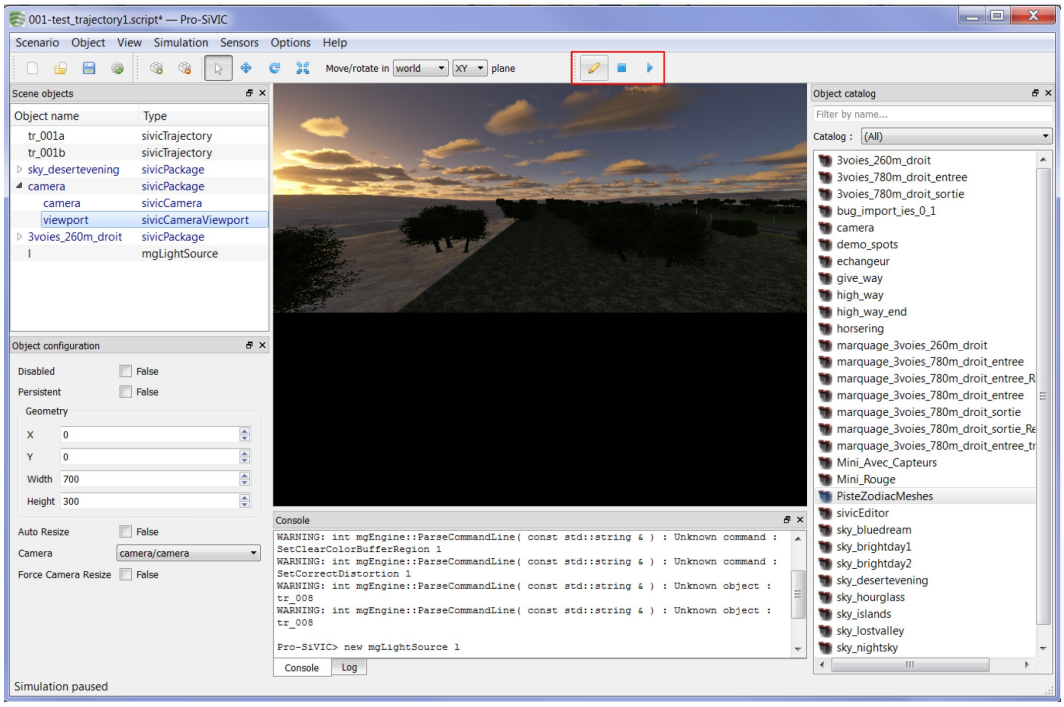


Figure 63: Pro-SiVIC: The simulation toolbar allows to enable/disable the editing tools, and control the simulation flow user interface (source ESI group)

These HMI elements are mandatory for the real-time using of the simulation platform. In a upstream stage, it is necessary to propose the same type of HMI but for the scenario definition, management, and generation (see HMI of MOSAR). Some interesting applications like SCANeR Studio (from AVS) or ROAD Runner (from MathWork) propose user friendly interfaces allowing to generate efficient virtual environment with road networks, road signage, and dynamic objects (ego-vehicle and surrounding vehicle).

3.2.12 Event management and requirement

Here are the 15 main requirements for event management in a simulation environment dedicated to evaluation and validation of automated mobility:

- Event Definition: Provide a mechanism to define various types of events (spatial, temporal, semantic) that can occur within the simulation, such as user interactions, system events, sensor readings, and environmental changes.
- Event Scheduling: Enable users and components to schedule events at specific simulation time points or relative to other events, allowing for precise control over the timing and sequencing of simulation activities.
- Event Triggers: Define triggers or conditions that initiate the execution of events, including temporal triggers (time-based), state triggers (condition-based), and external triggers (input-based).
- Event Handlers: Implement event handlers or callbacks to execute predefined actions in response to triggered events, such as updating simulation parameters, modifying object states, or generating new events.

- Event Prioritisation: Support prioritisation of events to ensure that critical or time-sensitive events are processed with higher priority, minimising delays and ensuring timely responses in dynamic simulation environments.
- Event Queuing: Maintain an event queue or buffer to store scheduled events in chronological order, facilitating efficient event processing and ensuring proper sequencing of event execution.
- Event Dispatching: Dispatch scheduled events from the event queue to their respective event handlers for execution, considering event priorities and dependencies to maintain consistency and coherence in simulation behaviour.
- Event Propagation: Enable propagation of events across simulation components and subsystems, allowing events to trigger cascading effects and interactions between different elements of the simulation environment.
- Event Logging: Log event occurrences, timestamps, and associated metadata to facilitate post-simulation analysis, debugging, and performance monitoring, providing insights into simulation dynamics and behaviour.
- Event-Based Modelling: Support event-driven modelling paradigms to capture complex system behaviours and interactions through the specification of discrete events, transitions, and state changes.
- Event Coordination: Coordinate events between concurrent simulation processes, distributed simulation nodes, or interconnected simulation modules to maintain synchronisation and consistency across the simulation environment.
- Event Interception: Allow for the interception and modification of events before their execution, enabling advanced event processing techniques such as event filtering, transformation, and augmentation.
- Event Abstraction: Abstract event handling mechanisms to encapsulate complex event processing logic and promote modular, reusable, and maintainable simulation models, components, and libraries.
- Event Visualisation: Provide visualisation tools and techniques to represent event occurrences, sequences, and dependencies graphically, aiding in the comprehension and analysis of simulation dynamics and behaviour.
- Event-Based Control: Enable event-based control strategies to regulate simulation activities, trigger system interventions, and adapt simulation parameters dynamically based on evolving simulation conditions and external stimuli.

These requirements aim to ensure effective event management in a simulation environment, encompassing event definition, scheduling, handling, prioritisation, queuing, dispatching, logging, modelling, coordination, interception, abstraction, visualisation, and control.

3.3 Digital Twin, Digital Shadow, and Digital Model requirements

3.3.1 Digital Twin, Shadow, and Model: Definition and differences

- **Digital Model**: A Digital Model refers to a computer-generated representation of a physical object, system, or environment. Unlike Digital Twins and Digital Shadows, which focus on replicating the behaviour and characteristics of real-world entities, a Digital Model primarily emphasises the visual and geometric aspects of the object or system. The main aspects handle by the Digital Model could be the following:
  - Geometric Representation: At its core, a Digital Model consists of geometric data that describes the shape, structure, and spatial relationships of the physical entity. This may include 3D models, CAD drawings, surface meshes, point clouds, or other geometric representations.
  - Visual Rendering: Digital Models often incorporate textures, colours, materials, and lighting effects to enhance visual realism and accuracy. They aim to closely resemble the physical appearance of the object or environment, making them valuable for visualisation and communication purposes.
  - Static Representation: Unlike Digital Twins and Digital Shadows, which may involve dynamic or time-varying data, Digital Models typically provide a static representation of the physical entity at a specific point in time. They capture a snapshot of the object's geometry and appearance but do not simulate its behaviour or interactions.
  - Accuracy and Detail: The level of detail and accuracy in a Digital Model can vary depending on its intended purpose and the available data sources. High-fidelity models may include intricate geometric details, while low-fidelity models may prioritise computational efficiency and performance.
  - Integration with Other Technologies: Digital Models can be integrated with other technologies, such as simulation engines, virtual reality (VR) systems, augmented reality (AR) applications, and geographic information systems (GIS), to create immersive and interactive experiences.
  - Lifecycle Management: Throughout the lifecycle of a physical object or system, Digital Models may undergo updates, revisions, and enhancements to reflect changes in design, construction, operation, or maintenance. Effective version control and data management practices are essential for maintaining the integrity and accuracy of Digital Models over time. But these updates of the Digital Model will come from human operator and will provide a new version of the virtual model without time link.
- **Digital Shadow**: A Digital Shadow is a virtual representation or simulation of the behaviour, interactions, and characteristics of a physical object, system, or environment. Unlike a Digital Twin, which typically reflects the real-time state and condition of its physical counterpart, a Digital Shadow focuses more on historical or predictive analysis rather than real-time monitoring. Digital Shadows complement Digital Twins by providing deeper insights, analysis, and foresight into the behaviour and performance of physical systems. The main aspects handle by the Digital Model could be the following:

  - Historical Data Analysis: A Digital Shadow is often based on historical data collected from sensors, historical databases, operational systems, and other sources. This data provides insights into past behaviours, trends, and patterns associated with the physical entity.
  - Predictive Modelling: In addition to historical data, a Digital Shadow may incorporate predictive models and simulations to forecast future behaviours, performance, or outcomes. These models can help anticipate potential issues, optimize operations, and support decision-making.
  - Scenario Analysis: Digital Shadows enable scenario analysis and what-if simulations to evaluate the potential impact of different conditions, events, or actions on the physical system. This capability allows organisations to assess risk, plan for contingencies, and optimize strategies. Digital Shadow allows to apply experimental plan with a large set of scenarios with initial conditions. For a scenario and an initial condition, it is possible to apply variations among a set of identified parameters and variables considered as important and impacting the system behaviour.
  - Offline Simulation: Unlike Digital Twins, which often involve real-time data integration and monitoring, Digital Shadows are typically used for offline simulation and analysis. They provide a static or semi-dynamic representation of the physical entity, focusing on historical or hypothetical scenarios rather than current conditions.
  - Performance Monitoring: While Digital Shadows may not offer real-time monitoring capabilities, they can still provide performance metrics, KPIs, and historical trends to assess the effectiveness and efficiency of the physical system over time.

- **Digital Twin**: A Digital Twin is a virtual representation of a physical object, system, or process. It's not just a static 3D model; it's a dynamic, data-driven model that reflects the real-time status, behaviour, and condition of its physical counterpart. Its added value comes from multiple sources of independently used information and data stream, brought back together in order to get ecosystemic view and 3D representation model, playing on various settings.

  - Virtual Representation: A Digital Twin provides a digital counterpart of a physical entity, which can range from individual components, such as a machine part or a sensor, to entire systems, like a manufacturing plant or a smart city.
  - Real-Time Data Integration: It incorporates real-time data streams from sensors, IoT devices, and other sources to reflect the current state and behaviour of the physical entity. This data includes information about performance, operating conditions, environmental factors, mandatory / legislation, geolocated information and more.
  - Physics-Based Modelling: It employs physics-based models or simulations to accurately replicate the behaviour and interactions of the physical entity. These models allow for predictive analysis, what-if scenarios, and optimisation strategies. The main methods are Data-driven approaches involve using AI and machine learning algorithms to develop digital twins based on data collected from sensors and other data acquisition systems, Physics-based modelling involves using mathematical equations to model the physical behaviour of a system, and Hybrid approaches

involve integrating data-driven and physics-based modelling techniques to develop digital twins.

- Interconnectedness: A Digital Twin is often part of a networked ecosystem, where multiple Digital Twins interact with each other, as well as with external real systems.
- Lifecycle Perspective: It spans the entire lifecycle of the physical entity, from design and development to operation, maintenance, and decommissioning. This holistic view facilitates continuous improvement, predictive maintenance, and sustainability initiatives.
- Remote Monitoring and Control: Digital Twins enable remote monitoring and control capabilities, allowing users to visualise, analyse, and manage the physical entity from anywhere in the world. This capability is particularly valuable for complex, distributed systems and assets. This aspect fit with the ImPACT 3D platform developed by UGE and which has for objective to interconnect 3 facilities (A real automated vehicle, an immersive and dynamic 6DoF platform, and a street crossing simulator (for pedestrian) in the same real and virtual environment).
- Feedback Loop: A key aspect of Digital Twins is the feedback loop between the virtual and physical realms. Insights gained from analysing the digital twin can inform decision-making and actions in the physical world, leading to continuous improvement and optimisation.

If we try to summarise these 3 levels of Digital representation and modelling, the lower level is the Digital Model with a 3D modelling of an object, component, environment with shape, structure, and spatial relationships of the physical entity. But the Digital Model does not involve time aspects and dynamic modelling. But the Digital Model is useful for Digital Shadows and Digital Twin. It is the core module.

Then the Digital Shadow involves, from past observations of the system and from knowledge about its operating an behaviours, simulation models of the behaviour, interactions, and characteristics of a physical object. This includes dynamic models. But this level of modelling is without a link and real-time information about the real system working in same time. Moreover, the modification and updating of the Digital Shadow need possible intervention of a human operator.

The Digital Twin concerns the more complex and exact-faithful-accurate-representative level of modelling. It is interconnected in real time with the real system and allows to collect information from the real system, and provide to the real system data and observation about its behaviour and current state. In this condition, the Digital Twin could use the data coming from the real system in order to modify and to update its parameters and its configuration.

Even if these 3 level of modelling are complementary, it is important to accurately highly their main differences:

- Purpose:
  - Digital Twins: Primarily used to simulate, monitor, and optimize the behaviour and performance of physical assets or systems in real-time.
  - Digital Shadows: Focus on collecting and analysing data from physical entities to create a digital representation that mirrors their behaviour and characteristics.
  - Digital Models: Emphasise the visual and geometric aspects of physical entities, providing static representations for visualisation, design, and analysis purposes.

- Dynamic Behaviour:
  - Digital Twins: Incorporate dynamic, real-time data and simulation capabilities to emulate the behaviour and interactions of physical assets or systems.
  - Digital Shadows: Capture and analyse historical or real-time data to create a digital representation that reflects the past or present behaviour of physical entities.
  - Digital Models: Provide static representations of physical entities' geometry and appearance, lacking dynamic behaviour or predictive capabilities.
- Level of Detail:
  - Digital Twins: Often include detailed models and simulations that accurately represent the physical asset's behaviour, components, and interactions.
  - Digital Shadows: Focus on capturing and analysing data at a high level to identify patterns, trends, and anomalies in the behaviour of physical entities.
  - Digital Models: Represent the visual and geometric aspects of physical entities, ranging from high-fidelity models with intricate details to low-fidelity models with simplified geometry.
- Predictive Capabilities:
  - Digital Twins: Used for predictive maintenance, performance optimisation, and scenario analysis based on real-time data and simulation results.
  - Digital Shadows: Provide insights and historical analysis to support decision-making but lack the predictive capabilities of Digital Twins.
  - Digital Models: Primarily used for visualisation, design, and communication purposes, lacking predictive or analytical capabilities.
- Real-Time Interaction:
  - Digital Twins: Allow for real-time interaction and control of physical assets or systems based on sensor data and simulation feedback.
  - Digital Shadows: Provide historical or near-real-time insights into the behaviour and performance of physical entities but do not support real-time interaction.
  - Digital Models: Typically used for passive visualisation and analysis, with limited or no support for real-time interaction or feedback.
- Data Sources:
  - Digital Twins: Integrate real-time sensor data, operational data, and external inputs to simulate and monitor the behaviour of physical assets or systems.
  - Digital Shadows: Collect and analyse data from various sources, such as sensors, IoT devices, databases, and external sources, to create a digital representation of physical entities' behaviour.
  - Digital Models: Rely on data from design specifications, CAD models, survey data, or other sources to create a visual representation of physical entities' geometry and appearance.

- Complexity and Scale:
  - Digital Twins: Often involve complex, high-fidelity simulations of large-scale systems or assets, requiring advanced modelling and computational capabilities.
  - Digital Shadows: Focus on analysing data from multiple sources to create a holistic view of the behaviour of physical entities, which may vary in complexity and scale.
  - Digital Models: Can range from simple geometric representations to complex, detailed models, depending on the level of detail required for visualisation and analysis.
- Operational Focus:
  - Digital Twins: Primarily used for operational purposes, such as monitoring, control, and optimisation of physical assets or systems in real-time.
  - Digital Shadows: Provide insights and analysis to support decision-making and planning processes, focusing on understanding past and present behaviour rather than real-time operations.
  - Digital Models: Used for design, visualisation, and communication purposes, with a focus on representing the visual and geometric aspects of physical entities.
- Domain-Specific Applications:
  - Digital Twins: Widely used in domains such as manufacturing, energy, healthcare, transportation, and smart cities for asset management, predictive maintenance, and process optimization.
  - Digital Shadows: Applied in various industries for data analytics, risk assessment, anomaly detection, and historical analysis of physical entities' behavior.
  - Digital Models: Find applications in architecture, engineering, entertainment, and product design for visualization, prototyping, and design analysis purposes.
- Integration with Physical World:
  - Digital Twins: Actively interact with and control physical assets or systems based on real-time data and simulation feedback, bridging the gap between the physical and digital worlds.
  - Digital Shadows: Provide insights and analysis based on data collected from physical entities, serving as a digital representation of their behavior and characteristics.
  - Digital Models: Represent the visual and geometric aspects of physical entities in a digital format, serving as a tool for visualization, analysis, and communication but lacking direct interaction with the physical world.

3.3.2 What means Digital Twin for Automated Vehicle and Mobility Means

Digital Twin and Digital Shadow of urban area, highway or test track are a comprehensive virtual replica of the full system involving environment (the World), infrastructure, vehicles and embedded systems and components, road users and human aspects (Drivers, pedestrian, VRU ...), traffic generation and management. The Digital Twin consists to runs this Digital

Shadows with the real system in real time in order to assess the possible optimisation of the system under test and to take into account critical situations until crashes in the Digital Twin. This corresponds to ViL approach.

The environment and the Digital Model encompassing various dimensions such as geography, road surface and road configuration, buildings, vegetation (garden, forest, trees, bushes ...). The mobility service and traffic management involve information and models about transportation networks. Unlike the HD Map which are specialised maps tailored for autonomous vehicle navigation, The DM and more accurately the BIM aspects and purpose depends on the application but generally it is to provide a dynamic and holistic representation of the urban landscape, enabling stakeholders to visualise, analyse, and simulate complex urban systems. In the Connected and Automated Vehicle simulation and on-board sensors perception framework, the DS aims to simulate not only the complex urban and transportation system but also the physical principles governing the systems, systems of systems, objects dynamics, sensors operating as it is presented and shown in the figure 64. Digital Twin is not only a Digital Shadow or Digital Model, but the capacity to mimic all the dynamic and physical aspects of the systems with the spatial, physical, temporal, and material dimensions. A Digital Twin provides the capability to predict and to anticipate the future states, evolution, behaviours, actions, activities, degradation of each sub-part of the whole mobility system. Data for the BIM aspect (Digital Shadow) of DT typically comes from a variety of sources, including aerial and satellite imagery, LIDAR scans, ground surveys, IoT sensors, social media feeds, and administrative records. These diverse datasets are integrated and processed to create a comprehensive model of the urban, highway or testing track's landscape.

From a requirement point of view, Digital Twin, Digital Shadow, and Digital Model need to address at least these main aspects:

- **Data Integration and Collection**: The digital twin must be capable of integrating data from various sources, including crowd-sourcing, IoT sensors, operational systems, and historical databases. A generic methodology addressing this aspect is provided in the next section.

- **Real-time Data Processing**: It should process data in real-time to provide up-to-date insights and support timely decision-making. Moreover in the framework of a ViL system involving human in the loop, this real time aspect is essential in order to obtain a full immersive system with a high level of acceptability by the human(s).

- **Modelling and Simulation**: The digital twin should include accurate models that represent the physical system(s) or asset being simulated. For mobility systems and automated mobility, all these different complex and interacting systems are synthesised in the figure 64 and enumerated in the previous paragraph of this section.

- **High-Fidelity Representation**: It must accurately replicate the behaviour and characteristics of the physical counterpart to ensure realistic simulations and predictions. This requirement imply the capability to verify, to evaluate, and to validate the level of modelling and fidelity of the different models. It is not only linked to the vehicles, sensors, traffic, PDI modelling but also the 3D and physical modelling of the environment (Digital Model), the ligth sources, and the disturbances existing in this environment (center parts of the figure 64). These 3D rendering aspects are essential in order to provide physical and representative data for instance to the sensors like cameras, LiDARs, RADARs,

and Infra-Red cameras. In this context, a methodology and a set of scores are needed in order to evaluate the level of fidelity and to provide a ”label” about the quality of the synthetic data. A set methodology has been proposed in PRISSMA for the assessment of the synthetic images quality and fidelity ([74])

- **Scalability**: The digital twin should be scalable to accommodate changes in the complexity and scope of the physical system or asset. It is the case for the different levels of evaluation and validation in the road eco-system and more specifically for the automated mobility.
- **Predictive Analytics**: The digital twin should leverage predictive analytics techniques to anticipate future behaviour, performance, and potential issues. This aspect is essential in order to manage the operating safety, the failure of some components and sub-systems, or the reaction and operating in case of misuses (improper use). Also it is the case for intentional (cyber attacks) or non intentional attacks.
- **Historical Data Analysis**: It should enable the analysis of historical data to identify trends, patterns, and insights that can inform decision-making and improve performance.
- **Lifecycle Management**: The digital twin should support the entire lifecycle of the physical system or asset, from design and development to operation, maintenance, and decommissioning. In the framework of automated mobility, this mean to have the capability to take into account the lifecycle of the main components of the service. Components could be the vehicle (lifecycle of batteries impacting the efficiency and operating capability of the service), the road side and embedded sensors (reliability and performances degradation during the lifetime), the actuators, ...
- **Feedback Loop**: It should incorporate a feedback loop mechanism to continuously update and refine models based on real-world observations and feedback. This mechanism is essential in order to take into account the modification of the infrastructures and environment during the service lifetime.

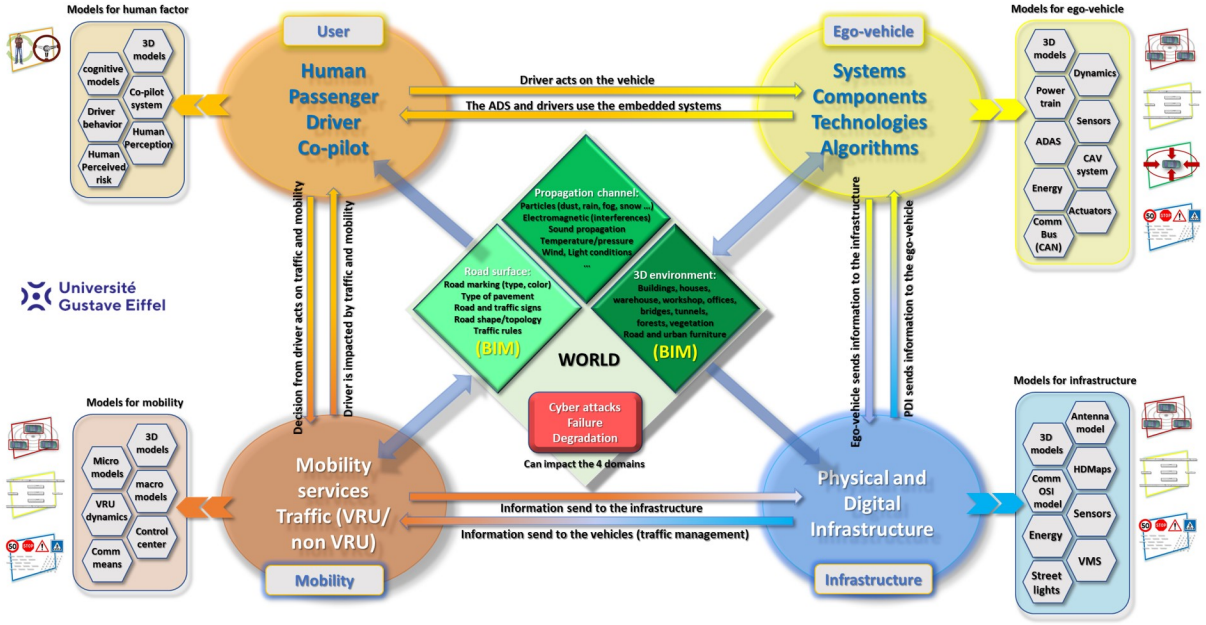


Figure 64: Taxonomy of the Digital Shadows for Connected and Automated Mobility means (source: UGE).

In [13], a High-level view of DT lifecycle is given (see figure 65) and a terminology and some definitions are proposed about Digital Twin.

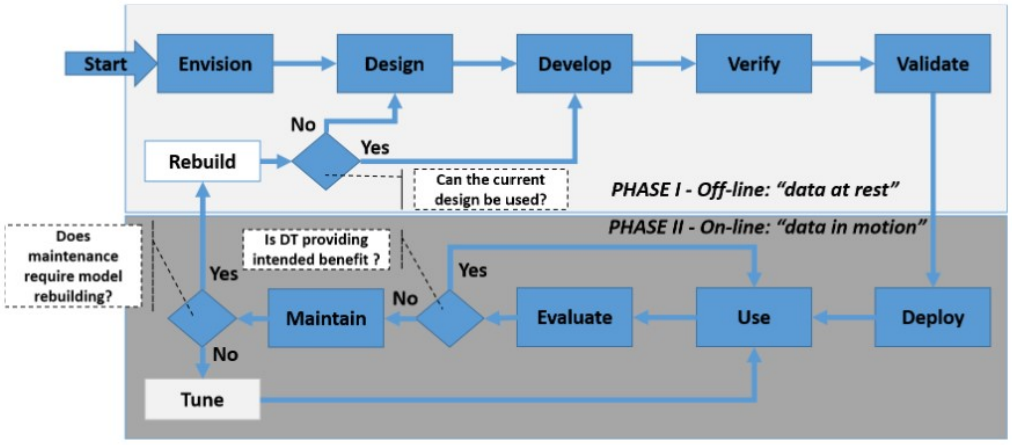


Figure 65: High-level view of DT lifecycle (source: [13]).

3.3.3 Generic methodology for the Digital Shadow and Digital Model development

A generic methodology for the generation of a Digital Twin typically involves several key steps and functions. Here's an outline of the main components:

- **Define Objectives and Scope**: Clearly define the objectives of creating the digital twin and identify the scope of the physical system or asset to be modelled. In the case of PRISSMA, the objectives and scope are define by the different POCs but addresses systems of systems and AI-based systems evaluation and validation for automated mobility.

Different environment have been identified (open road like Paris2Connect, and controlled environments like UTAC, Transpolis, and Satory test tracks).

- **Data Collection and Integration**: These steps involve gathering data from various sources such as LiDAR, GPS RTK, cameras, IoT sensors, operational systems, historical databases, and manual inputs. The collected data is then integrated and preprocessed to ensure consistency, quality, and compatibility with the digital twin environment. For Digital Shadow, this process corresponds to the first group of functions (orange boxes) in Figure **??**. The outputs of this step primarily focus on generating a high-fidelity 3D environment with vehicles and UAVs equipped with high-resolution LiDAR and cameras. The resulting model includes millions of points and faces along with photogrammetric images, where one pixel could represent a couple of square centimetres (centre of the figure 64). It serves as a foundational model rather than the final model required for real-time operation. For the other parts (Digital Model and Digital Twin (4 surrounding domains of the figure 64), benches, human expertise, observation facilities, and theoretical and physical knowledge are essential.

- **Modelling and Simulation**: Develop models that accurately represent the physical system or asset, including its structure, behaviour, and interactions with the environment. In automated mobility, these models are represented and presented in the four domains (User, Ego-Vehicle, Infrastructure, and Mobility) of the figure 64. Some needed models to develop are provided for each domain. Use simulation techniques to validate and refine the models, ensuring they accurately capture the dynamics and the high fidelity behaviour of the real-world system.

  - **Digital Model Development**: From the high resolution 3D model, a set of 3D lighter (but representative) models are extracted or generated (meshes, material, textures, ...). This stage is provided in the figure 66 by the green and cyan boxes. The data of the Digital Shadow may also include in addition to the spatial information, sensor data, environmental conditions, and operational parameters. Cleanse, preprocess, and transform the collected 3D data to ensure consistency, accuracy, and compatibility with the digital twin environment and a real-time operating. This may involve data reduction, sharing, filtering, noise reduction, calibration, alignment, and normalisation.
  - **Digital Shadow Development**: A Digital Shadow relies on historical data from sensors, databases, and operational systems to understand past behaviours and trends. It utilises physics-based models or simulations to accurately replicate physical entity interactions and behaviours, enabling predictive analysis and optimisation. Like the Digital Twin, methods used to develop dynamic and physical models include data-driven approaches using AI, physics-based modelling with mathematical equations, and hybrid approaches integrating both techniques for digital twin development. Nevertheless, Digital Shadow stay a Model and software in the loop approach without real-time links with the real system.
  - **Digital Twin Development**: Implement the digital twin environment, including software platforms, databases, and communication infrastructure. Integrate the developed models into the digital twin framework, ensuring interoperability and scalability. This stage involves to develop mathematical models, algorithms, and simu-

lations to represent the behaviour, dynamics, and interactions of the physical entities or systems. This includes creating 3D geometric models, physics-based models, control algorithms, and scenario simulations. Integrate the processed data and simulation models into a cohesive digital twin framework. Merge the diverse data streams and models to create a comprehensive representation of the physical entities or systems.

- **Deployment and Integration**: Deploy the digital twin into operational environments, ensuring seamless integration with existing systems and processes.This part, for a sub part of Digital Shadow, corresponds to the last stage (blue box) of the figure 66

- **Feedback and Continuous Improvement**: Establish feedback loops to capture insights from users and real-world observations, incorporating them into the digital twin to improve accuracy and reliability. Continuously update and refine the digital twin based on new data, changes in the physical system, and evolving requirements.

- **Maintenance and Support**: Establish procedures for ongoing maintenance and support of the digital twin, including software updates, troubleshooting, and performance optimisation. Monitor the performance and effectiveness of the digital twin, making adjustments as necessary to ensure it continues to meet the objectives and requirements. Provide continuous monitoring and updating with new data, insights, and improvements. This iterative process ensures that the digital shadows and by extension the digital twin remains accurate, up-to-date, and relevant to its physical counterpart.

- **Validation and Calibration**: Validate the digital shadow against real-world observations and experimental data. Calibrate the simulation models and parameters to ensure accuracy and reliability in capturing the behaviour of the physical counterpart.

Additional functionalities could be provided for the using and updating of Digital Shadows like:

- Real-time Monitoring and Control: Establish mechanisms for real-time monitoring of the physical system, collecting sensor data, and updating the digital twin accordingly. Implement control algorithms to enable remote control and management of the physical system through the digital twin interface. Analytics and Predictive Maintenance:

- Apply data analytics techniques to analyse historical and real-time data, identifying trends, patterns, and anomalies. Use predictive analytics to forecast future behaviour, performance, and potential issues, enabling proactive maintenance and optimisation.

- Visualisation and User Interface: Provide tools and interfaces for visualising and interacting with the digital shadow. This may include 3D visualisation platforms, graphical user interfaces, virtual reality environments, and augmented reality applications. Provide dashboards, reports, and customise views to address specific needs of different users and use cases in terms of analysing, understanding, and interpretation of data.

- Security and Privacy: Implement robust security measures to protect sensitive data and prevent unauthorised access or tampering. Ensure compliance with relevant privacy regulations and standards to safeguard the confidentiality and integrity of the data.

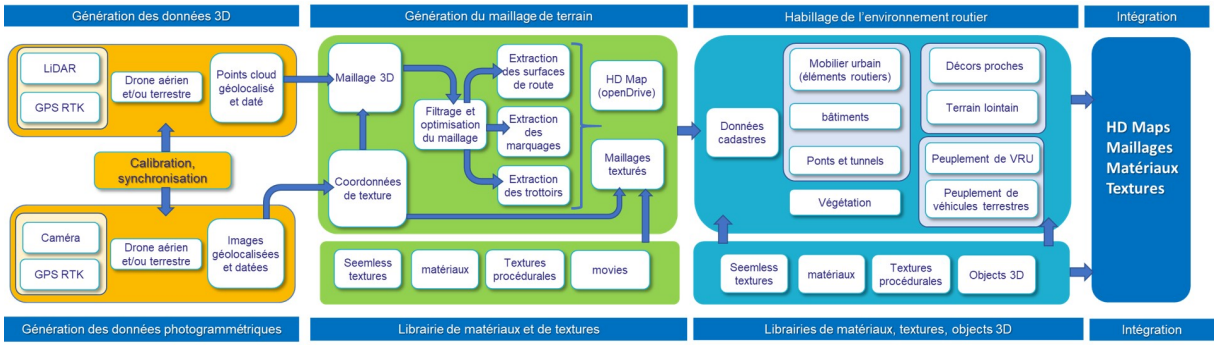


Figure 66: Process for the generation of Digital Model (source: UGE).

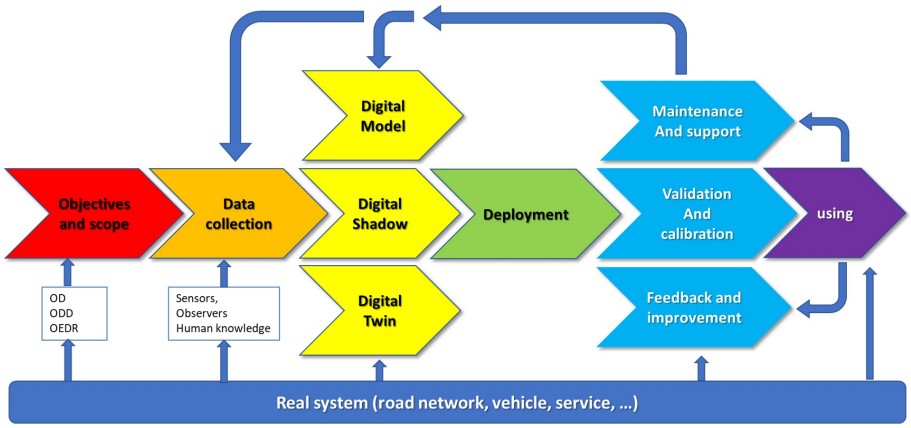


Figure 67: Digital Twin process of development (source: UGE).

3.3.4 Some Digital Shadows developed in the framework of PRISSMA or associated projects

3.3.4.1 Satory test tracks

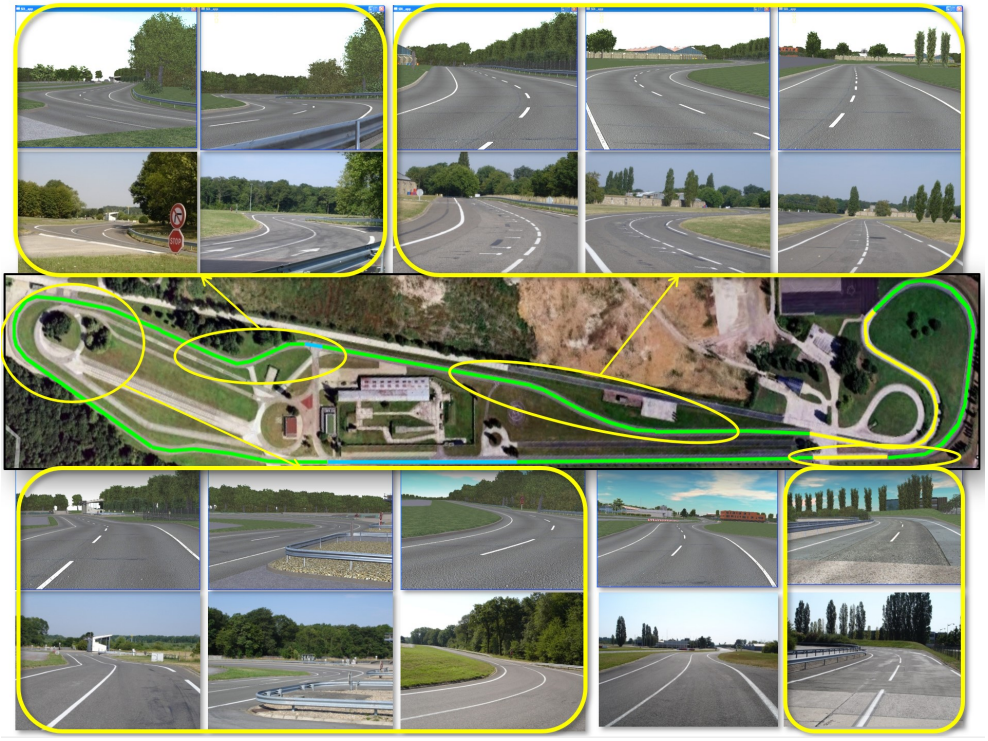


Figure 68: Digital Model developed for Satory test track (source: UGE).

3.3.4.2 Transpolis test tracks

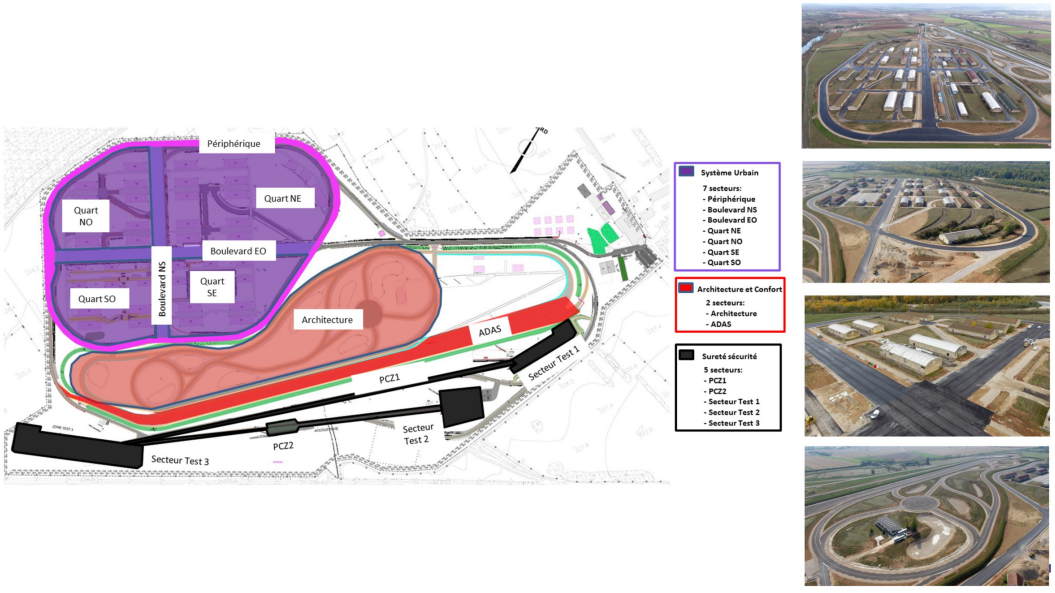


Figure 69: View of the Transpolis test tracks.

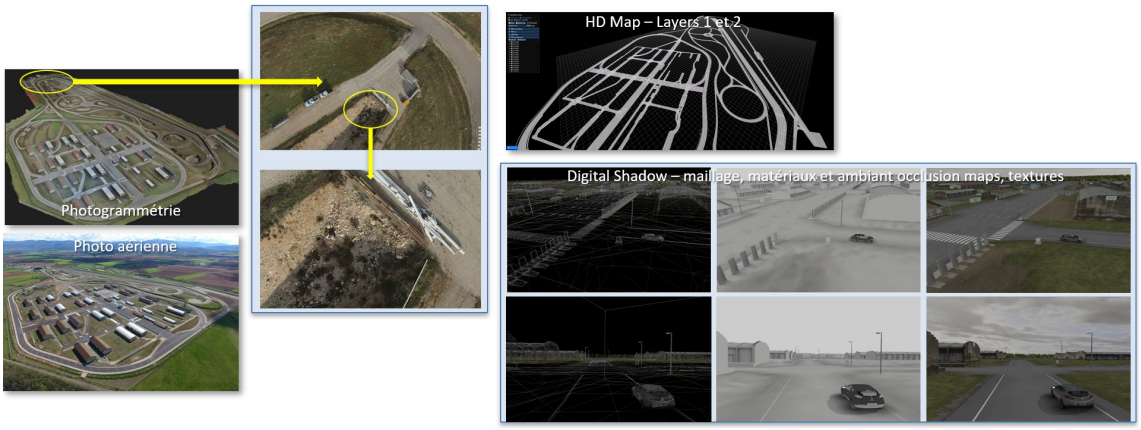


Figure 71: Digital Model and HD Maps (Transpolis), a long and resource consuming procedure (source: UGE).

3.3.4.3 Paris2Connect open area

Figure 72: Digital Model in progress for the Paris2Connect Use case in PRISSMA (source: UGE and VALEO).

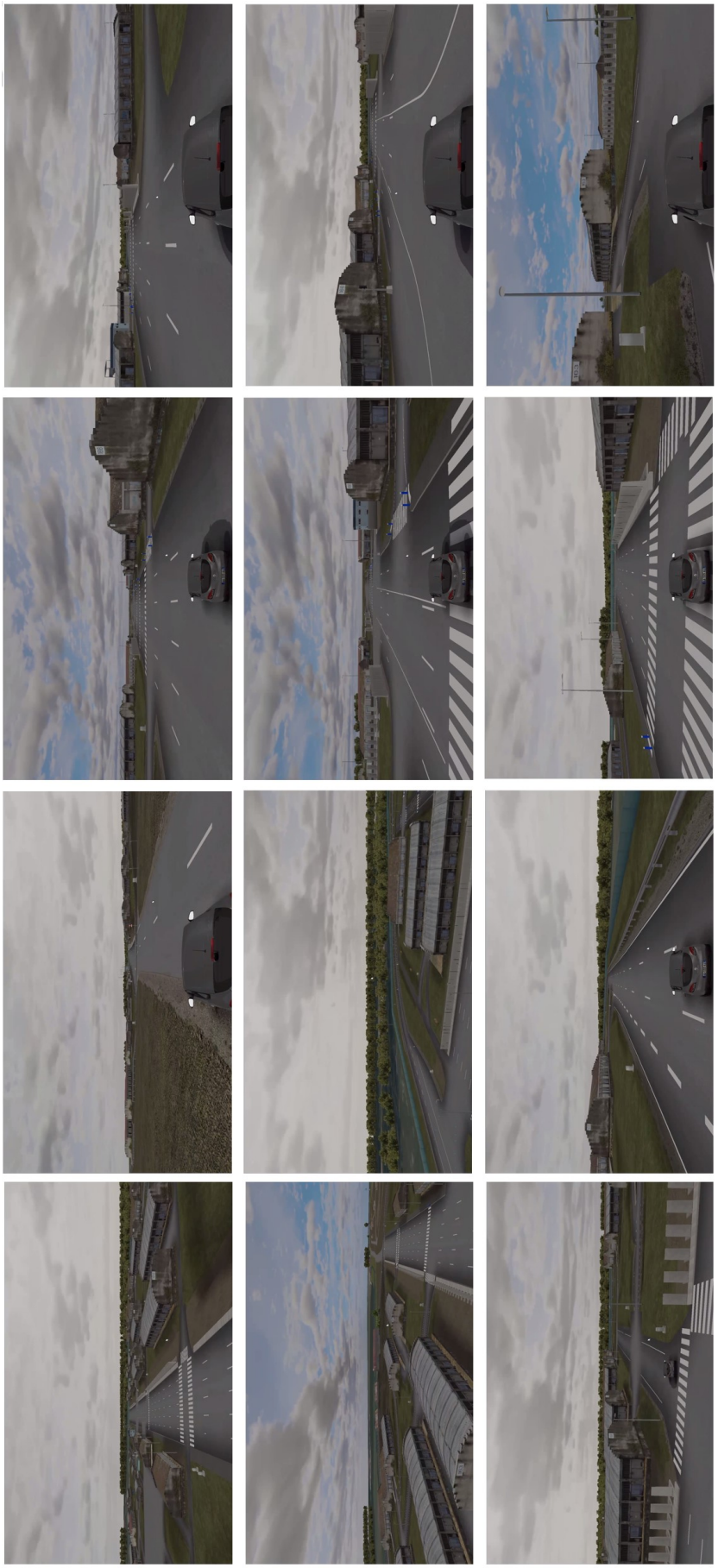


Figure 70: Digital Model developed for Transpolis test track (source: UGE).

Figure 73: Digital Model and Ambient Occlusion Map, a mandatory rendering mechanism in order to improve significantly the image fidelity (source: UGE and VALEO).

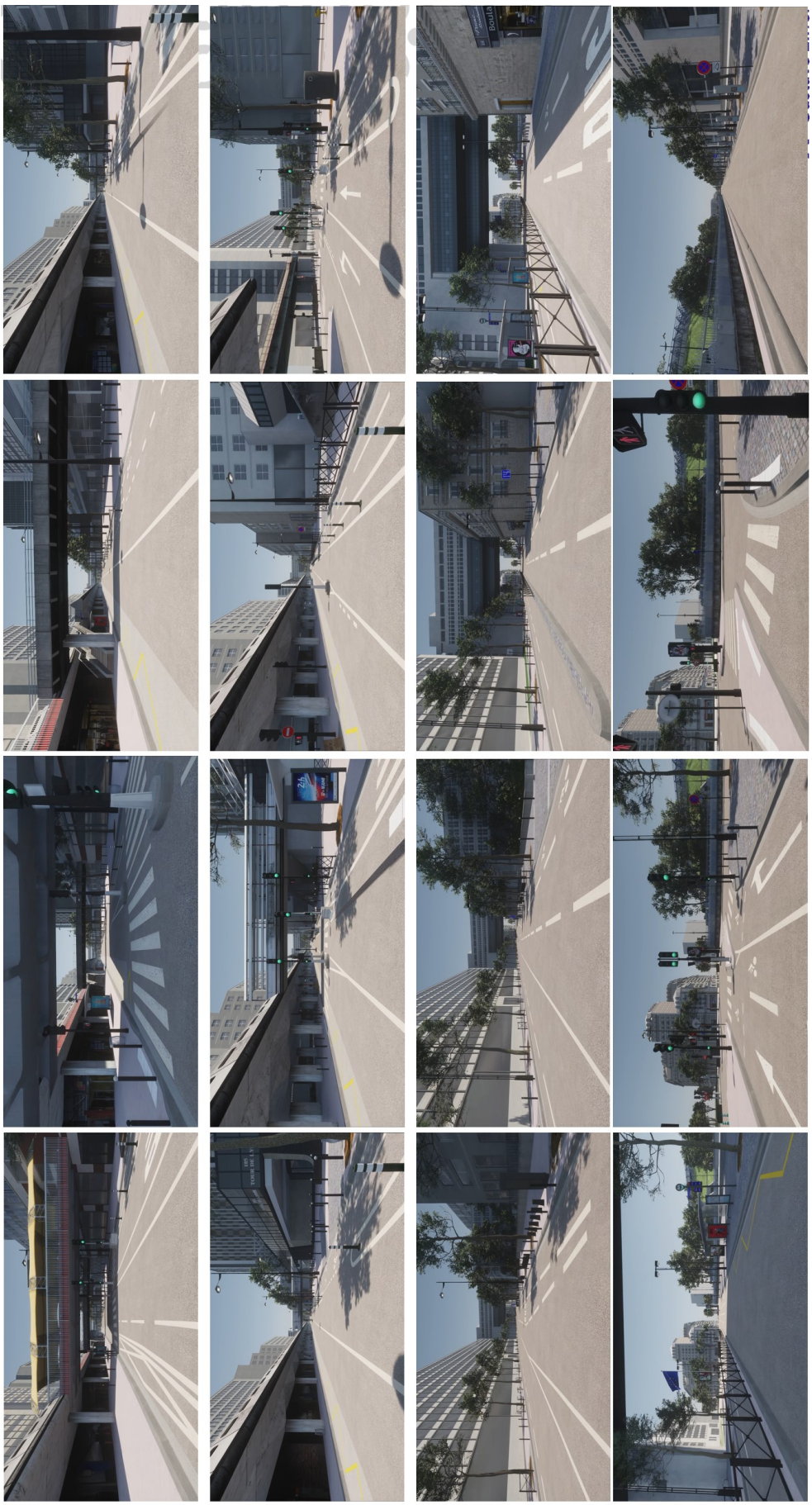


Figure 74: Screenshot of the Digital Model usable in the Digital Shadows Paris2Connect (source: UGE and VALEO).

3.4 PRISSMA's Platforms, systems and simulation engine

3.4.1 Definition of existing platforms

Different platforms exist as a virtual test environment for Connected and Automated Vehicles. These platforms can be separated into different types that will be described in the next paragraphs. But a non exhaustive overview for 2024 is available in [14] and presented in figure 75.

Category	Simulator	Released Year	Active Maintenance	Open-source	Related Tasks
Traffic flow simulator	PTV Vissim [31]	1992	√	-	①
	Paramics [32]	1994	√	-	①
	Aimsun [46]	1999	√	-	① ②
	SUMO [37]	2001	√	√	① ②
	POLARIS [47]	2016	√	-	① ②
	Flow [48]	2017	-	√	①
	CityFlow [49]	2019	-	√	①
Sensory data simulator	Sim4CV [50]	2016	-	-	③ ⑦
	AirSim [38]	2017	-	√	③ ④
	Parallel Domain [51]	2017	?	-	③
	SVL [44]	2018	-	√	③ ④
	UniSim [52]	2023	√	-	③
Driving policy simulator	TORCS [53]	1997	-	√	⑤
	VDrift [54]	2011	√	√	⑤
	CarRacing [55]	2016	√	√	④
	Udacity [56]	2016	-	√	⑤
	CommonRoad [39]	2017	√	√	④
	highway-env [45]	2018	√	√	④
	MACAD [57]	2019	?	√	④
	BARK [58]	2020	?	√	④
	DriverGym [59]	2020	-	√	④
	SMARTS [60]	2020	√	√	④ ⑤
	SUMMIT [61]	2020	?	√	④ ⑤
	DI-Drive [62]	2021	?	√	⑤
	L2R [63]	2021	√	√	⑤
	MetaDrive [64]	2021	√	√	④ ⑤
	NuPlan [8]	2021	√	√	④
	InterSim [65]	2022	√	√	④
	Nocturne [66]	2022	?	√	④
	TBSim [67]	2023	√	√	④
	Waymax [9]	2023	√	√	④
Vehicle dynamics simulator	CarSim [34]	1996	√	-	⑥ ⑦
	Webots [68]	1998	√	√	⑥ ⑦
	CarMaker [35]	1999	√	-	③ ⑥ ⑦
	Gazebo [69]	2002	?	√	⑥ ⑦
	VI-CarRealTime [41]	2009	√	-	⑥ ⑦
	Matlab [70]	2018	√	-	⑥ ⑦
Comprehensive simulator	SCANeR Studio [71]	1990	√	-	③ ④ ⑤ ⑥ ⑦
	Virtual Test Drive [72]	1998	√	-	③ ④ ⑤ ⑥ ⑦
	PreScan [40]	2006	√	-	③ ④ ⑤ ⑥ ⑦
	rFpro [36]	2007	√	-	③ ⑤ ⑥ ⑦
	CARLA [7]	2016	√	√	③ ④ ⑤ ⑦
	DeepDrive [73]	2018	?	√	③ ④ ⑤ ⑥
	Nvidia Drive Sim [74]	2020	√	-	③ ⑤ ⑥ ⑦
	Vista [75]	2020	√	√	③ ⑤
	VI-WorldSim [41]	2020	√	-	③ ④ ⑤ ⑥ ⑦

Figure 75: Main simulation tools and platforms. The available functions are: 1- traffic control design; 2- v2x communication; 3- sensor data processing; 4- driving policy design; 5- end-to-end driving policy design; 5- vehicle dynamics optimisation; 7- vehicle control (source: [14]).

Features		Simulator							
		CARLA	AirSim	DeepDrive	LGSVL	NVIDIA Drive	rFpro	MATLAB	Gazebo
General	Licence	Open-Source Windows and Linux	Open-Source Windows and Linux	Open Source Windows and Linux	Open-Source Windows and Linux	Commercial Windows and Linux	Commercial Windows and Linux	Commercial Windows and Linux	Open-Source Windows and Linux
	Portability	Unreal Engine	Unreal Engine and Unity	Unreal Engine	Unity	Unreal Engine	U	Unreal Engine	DART
	Physics Engine	Python	C++, Python, Java	C++, Python	Python	Python	U	MATLAB	C++, Python
	Scripting Languages								
Environmental	Urban Driving	Town	Town, City	Road Track	City	City, Harbor	Town, City, Road Track	N	Road Track
	Off-Road	N	Forest, Mountain	N	N	N	N	N	N
	Actors–Human	Y	N	N	Y	N/A	Y	Y	Y
	Actors–Cars	Y	Y	Y	Y	N/A	Y	Y	Y
	Weather Conditions	Y	Y	Y	Y	Y	Y	N	N
Sensors	RGB	Y	Y	Y	Y	Y	Y	Y	Y
	Depth	Y	Y	Y	Y	N/A	Y	N	N
	Thermal	N	Y	N	N	N/A	N/A	N	Y
	LiDAR	Y	Y	N	Y	Y	Y	Y	Y
	RADAR	Y	N	N	Y	Y	Y	Y	Y
Output Training Labels	Semantic Segmentation	Y	Y	N	Y	Y	Y	Y	Y
	2D Bounding Box	Y	N	N	Y	Y	N/A	Y	Y
	3D Bounding Box	Y	N	Y	Y	N/A	N/A	Y	Y

Legend: U: Unknown, Y: Yes, N: No.

Figure 76: The main commercial and open-source Automated Driving Simulation Platforms ([15])

Simulator	Open source	Vehicle dynamic		X-in-the-loop				Interface to other simulators	Reference
		customization	soft/rigid	MiL	SiL	HiL	ViL		
MATLAB/SIMULINK [340]	×	✓	–	✓	✓	✓	–	CARSIM, CARMAKER, PRESCAN, GAZEBO, CARLA, RFPRO, VTD, COGNATA, ADAMS PRO-SIVIC	[162, 163, 248] [188, 204, 277] [153, 218, 257] [195, 198]
CARSIM [341]	×	✓	rigid	✓	✓	✓	–	MATLAB/SIMULINK, RFPRO, NVIDIA DRIVE SIM, VTD, PRO-SIVIC, DONKEY CAR	[195, 257]
VISSIM [281]	×	×	–	–	✓	✓	✓	CARLA, VTD, PRESCAN, CARMAKER, RFPRO, SUMO	[288, 289]
SUMO [280]	✓	×	–	–	✓	✓	✓	CARLA, VISSIM, COGNATA, RFPRO	[194, 242, 277] [283]
WEBOTS [342]	✓	–	rigid	–	✓	–	–	–	[247, 292]
VTD [343]	×	✓	–	✓	✓	✓	✓	CARSIM, MATLAB/SIMULINK, ADAMS, VISSIM, RFPRO	[260, 308]
GAZEBO [344]	✓	–	rigid	–	✓	✓	–	MATLAB/SIMULINK, ADAMS	[254, 271, 275]
PRESCAN [303]	×	–	rigid	✓	✓	✓	✓	MATLAB/SIMULINK, VISSIM, PRO-SIVIC	[166, 188, 306] [195, 302]
BEAMNG [25]	✓	✓	soft	✓	✓	✓	✓	–	[189, 193, 293] [190, 200, 293] [191, 192]
CARLA [345]	✓	×	rigid	✓	✓	✓	✓	CARSIM, VISSIM, SUMO, MATLAB/SIMULINK	[101, 171, 346]
AIRSIM [347]	✓	×	rigid	–	✓	✓	–	–	–
RFPRO [348]	×	✓	rigid	–	✓	✓	–	CARSIM, MATLAB/SIMULINK, CARMAKER, VISSIM, VTD, SUMO	–
COGNATA [349]	×	✓	–	✓	✓	✓	-	MATLAB/SIMULINK, SUMO	–
NVIDIA DRIVE SIM [350]	✓	✓	–	–	✓	✓	✓	CARMAKER, CARSIM	–
LGSVL [351]	✓	×	–	✓	✓	✓	–	–	[72, 140, 196] [235, 252, 305]
SCANER STUDIO [352]	×	✓	soft/rigid	✓	✓	✓	✓	–	[164]
ADAMS [353]	×	✓	rigid	-	✓	✓	–	GAZEBO, MATLAB/SIMULINK, VTD	–
CARMAKER [222]	×	✓	rigid	✓	✓	✓	✓	MATLAB/SIMULINK, VISSIM, RFPRO, NVIDIA DRIVE SIM	[214, 277, 288] [218, 221, 289] [198, 221, 308]
PRO-SIVIC [354]	×	✓	–	✓	✓	✓	✓	MATLAB/SIMULINK, CARSIM, PRESCAN	[302]
DONKEY CAR [282]	✓	×	–	–	✓	✓	✓	MATLAB/SIMULINK	[279]

Figure 77: The main commercial and open-source Simulation Platforms for Automated Driving Systems Testing ([16])

In the PRISSMA framework, a set of existing tools, models, and platforms are existing (coming from a part of the partners) and have been used. Among these simulation softwares, we can mention:

- MOSAR from SystemX: scenario definition and generation
- Optimized FiFo DDS and DDsL for tools interfacing
- SCANeR Studio from AVS for vehicle modelling, sensors and environment simulation (road network, road environment, weather conditions). Used in the UTAC POC
- Pro-SiVIC from ESI group and UGE for vehicles and pedestrian modelling, highly realistic sensors models and environment simulation (road network, road environment, weather conditions)
- CarMaker from IPG and used in the VALEO POC
- CARLA, an open source simulation platform based on Unreal Engine 5 and a physical engine.
- ANSYS' toolchain used in the VALEO POC
- NS3 for the communication modelling
- SiVIC MobiCoop for the communication and propagation channel modeling
- SUMO for the traffic management and generation
- Symuvia from UGE and interconnected with Pro-SiVIC in order to generate dense traffic with interaction between the highly realistic vehicle dynamic model (ego-vehicle) and the vehicles managed by Symuvia. The dynamic of vehicles managed by Symuvia and in strong interaction with the SiVIC ego-vehicle allows to obtain realistic scenario and behaviour.
- RTMaps, Matlab, Simulink, ROS for the "applications" environments

3.4.1.1 Definition of existing platform for ADAS and CAV prototyping, test, and evaluation

In UGE, we have proposed in PRISSMA a generic framework and architecture for the simulation of road situation involving the different aspect presented for the generation of Digital Twin (see figure 78)

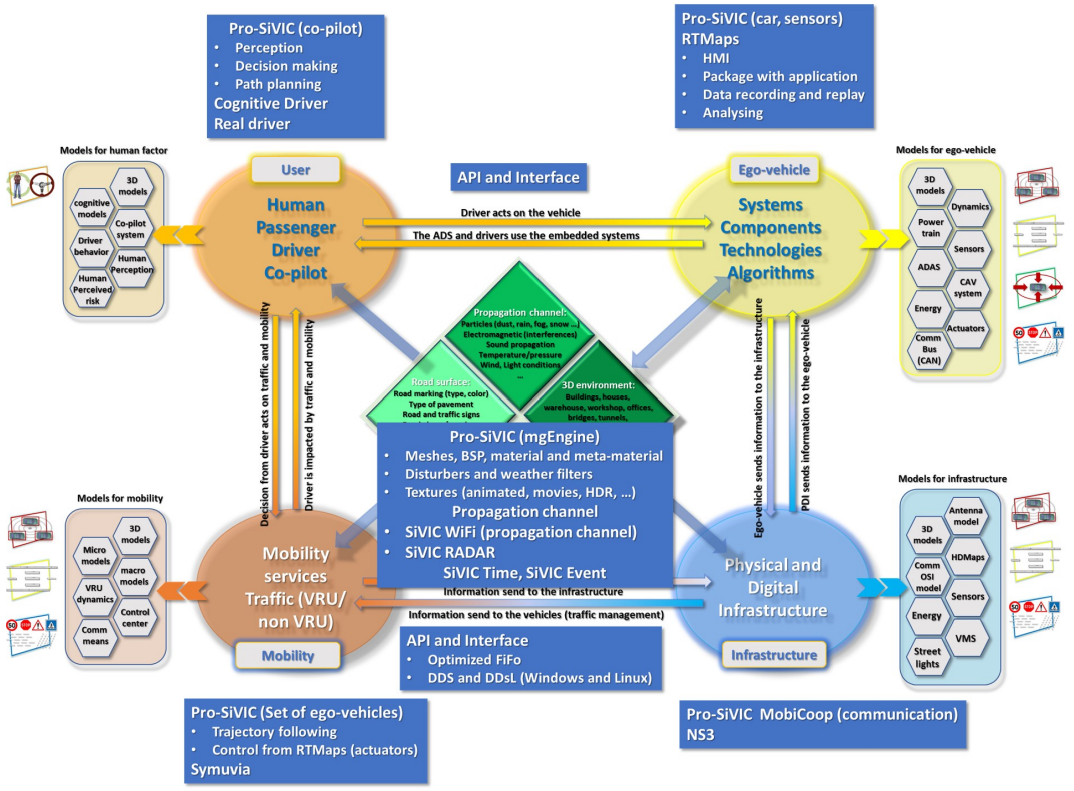


Figure 78: Generic simulation framework for Connected and Automated Mobility means (source: UGE).

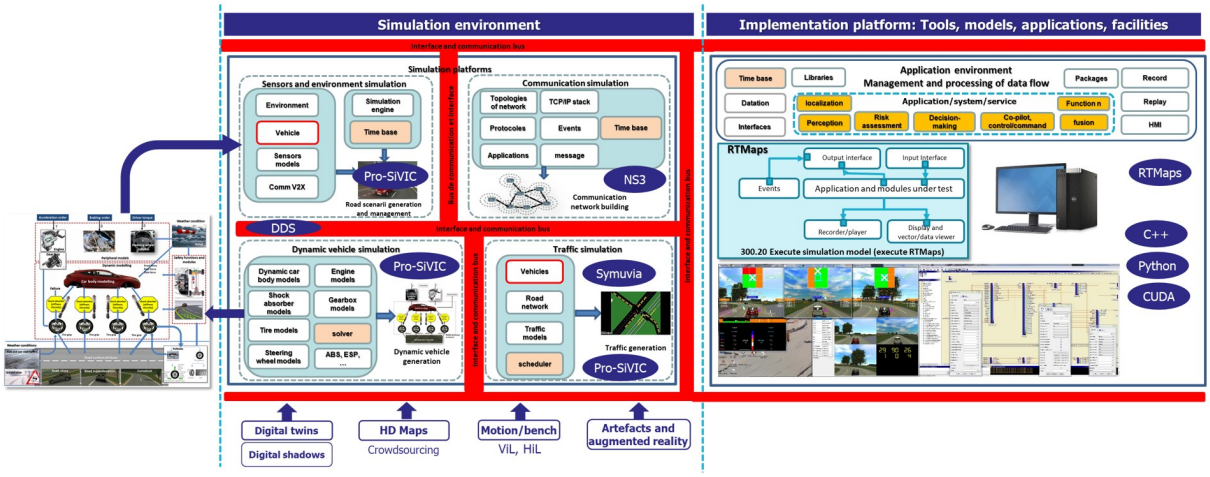


Figure 79: Generic simulation architecture proposed by UGE for Connected and Automated Mobility means (source: UGE).

3.4.1.2 Definition of existing platform for vehicle dynamics

For vehicle dynamics, they can be seen as two types : multi-body dynamics model that represent the mechanical systems of the vehicle, including shock absorbers, chassis, steering,

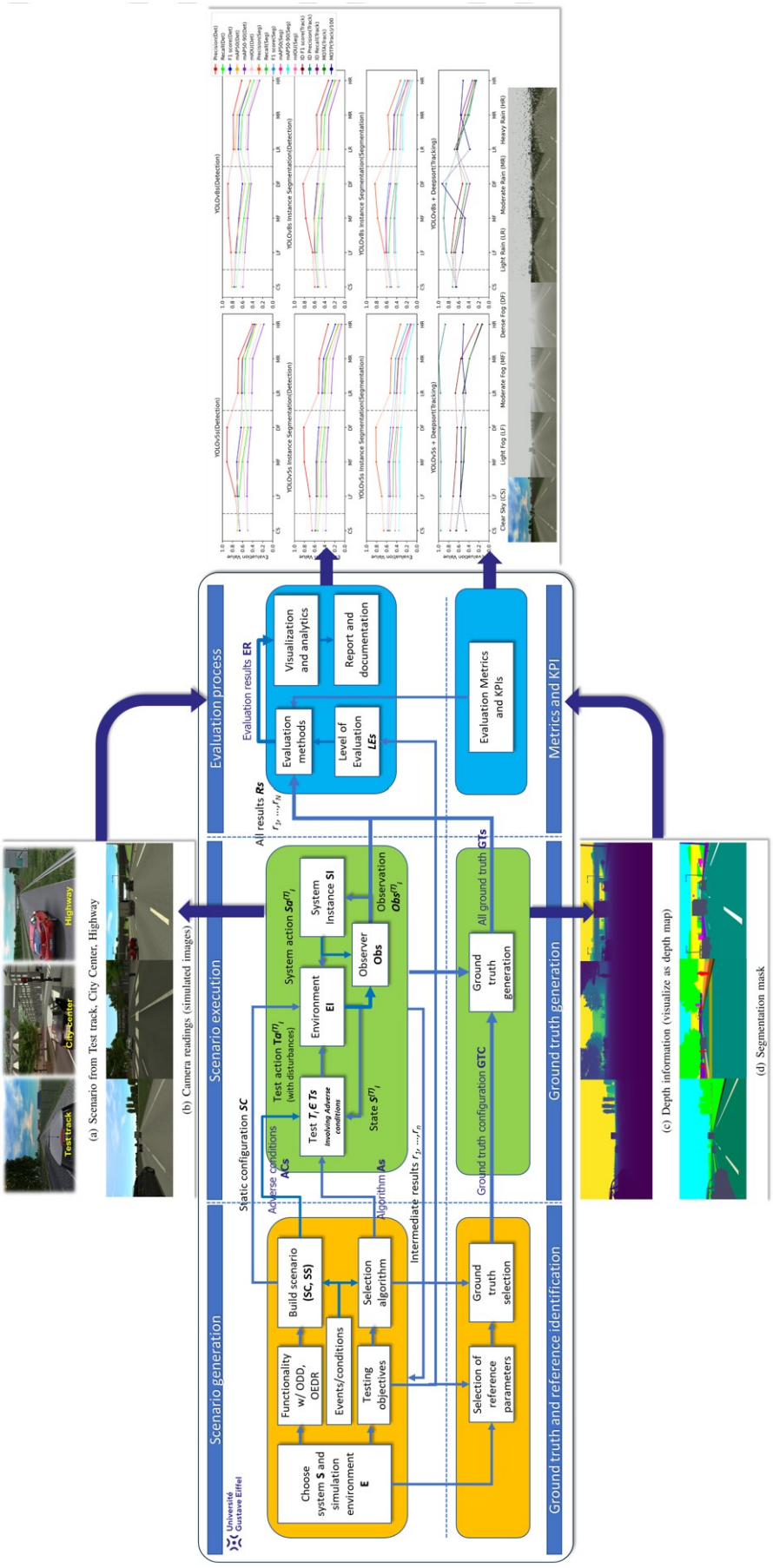


Figure 80: Generic evaluation framework proposed by UGE for Connected and Automated Mobility means (source: UGE).

tires, and drivetrain. It simulates the movement, forces, and interactions between these components based on Newtonian physics principles. Otherwise powertrain model simulates the engine, transmission, and other powertrain components, accounting for torque, gear ratios, fuel consumption, and performance characteristics.

Reference(s)	Name of mobility generator	Active development	Release	License	Map	Traffic model	Network simulator
28–30	SUMO	Y	2021	Open Source	Real and User Defined	Microscopic Mesoscopic	NS-2, NS-3, OMNeT++
31–33	MATSim	Y	2021	Open Source	Real and User Defined	Microscopic	N/A
34	DTALite	Y	2021	Open Source	Real	Mesoscopic	N/A
35,36	SMARTS	Y	2020	Open Source	Real and User Defined	Microscopic	N/A
21,37,38	PARAMICS	Y	2020	Commercial	Real and User Defined	Microscopic	NS-2, OMNeT++
32,38	MovSim	Y	2018	Open Source	Built-In	Microscopic	N/A
21,27,39	VISSIM	Y	2016	Commercial	Real and User Defined	Microscopic Mesoscopic	NS-2, QualNet
40	VNEtIntSim	N	2015	Open Source	Real and User Defined	Microscopic	Integration OPNET
38	Traffisim	N	2014	Open Source	Real and User Defined	Microscopic	N/A
41	CityMob	N	2009	Open Source	Built-In	Microscopic Macroscopic	NS-2
41	FreeSim	N	2008	Open Source	Real	Microscopic Macroscopic	N/A
41	STRAW	N	2007	Open Source	Built-In	Microscopic	NS-2, SWANS
41	Vanet-MobiSim	N	2007	Open Source	Real and User Defined	Microscopic	NS-2, QualNEt, OMNeT++, GloMoSim

Y = Supported, N = Not Supported

Figure 81: Traffic simulators.

3.4.2 Definition of existing simulation engine

3.4.3 Definition of time management

In the PRISSMA's simulation frameworks, we have addressed a large set of requirements about time management. The main time management processes and mechanisms are the following:

- **Graphical Engine Running Time** : Ensure efficient utilisation of the graphical engine's running time to render simulation visuals without significant lag or delays. Optimize rendering processes to maintain a smooth frame rate, especially in scenarios with complex scenes or high object counts. In Pro-SiVIC, the graphical engine called mgEngine allows to manage a simulation with a frequency up to 1000 Hz. This period of 1 ms is essential to allows the using of solvers and complex dynamic models (vehicles). Moreover, this high frequency operating allows to simulate a large category of sensor technologies.
  - Real-Time Simulation Capability: Provide real-time simulation capabilities to synchronise simulation time with real-world time for applications requiring time-critical responses, such as training simulations, control systems, and virtual prototyping. Ensure low-latency communication between simulation components and external systems to minimize delays and maintain responsiveness.

  - Time Scaling and Compression: Support time scaling and compression techniques to accelerate or decelerate simulation time, allowing users to observe long-term trends or fast-forward through repetitive phases of the simulation. Maintain synchronisation between scaled time and real time to ensure that simulation outcomes remain consistent and meaningful.
  - Clock Resolution and Granularity: Provide configurable clock resolution and granularity options to balance computational overhead with temporal accuracy, allowing users to tailor time management settings to their specific simulation requirements.
  - Temporal Consistency Across Components: Ensure temporal consistency across simulation components, including graphical rendering, physics simulation, sensor emulation, and control algorithms, to maintain a cohesive and synchronised simulation experience. Coordinate time management strategies and synchronisation mechanisms to minimise temporal discrepancies and ensure seamless integration of diverse simulation elements.

  In Pro-SiVIC, these mechanisms are managed in the graphical engine and can be fix in a configuration file used during the start of the simulation software.

- **Sensors and events modes and time-stamping**:

  - Sensors' Time Modes: Provide configurable time modes for sensors to simulate real-world sensor behaviours accurately, including sampling rates, refresh intervals, and synchronisation with the simulation clock. Support asynchronous sensor operation to mimic real-world sensor networks and varying data acquisition rates. In Pro-SiVIC, this mechanism is managed by the sivicRecorder plug-in. The different sensors' time modes are off, on, record, RTMaps, DDS, Network and the available period mechanism is presented in the figure 82. It is interesting to mention that each sensor has their own time and each specific mode also can have their own period.
  - Time-Stamped Data Logging: Support time-stamped data logging capabilities to record simulation events, sensor readings, and system states with precise temporal information. Enable users to analyse simulation data over time, correlate events, and identify causal relationships for in-depth analysis and validation.

- **Dynamic Vehicle Constraints**: Implement dynamic vehicle constraints to model realistic vehicle behaviour under changing conditions, such as acceleration, deceleration, turning radius, and speed limits. Ensure that vehicle dynamics are synchronised with the simulation clock and updated dynamically based on external factors, such as road conditions and traffic congestion. In Pro-SiVIC, all vehicles use a complex dynamic model. This means that we need to have a simulation engine allowing to guarantee a "world" frequency between 500 and 1000 Hz. It is the case in SiVIC. In order to run with a lower frequency, it is mandatory to modify the solver and to implement a solver with a variable step mechanism and with a 4 or 5 order level.

- **Event Management**: Enable precise event scheduling and execution mechanisms to trigger actions, behaviors, and state changes at specific simulation time points. Support event-driven programming paradigms for handling asynchronous events, including sensor data arrival, user interactions, and system notifications. In Pro-SiVIC, a large set of event are available but two of these events are dedicated to the time (time point from the starting

time of the simulation, and period). Moreover the event plug-in and mechanism is also considered as a source of information and can be recorded like a sensor.

- **Loop and interpolation Mechanism**: Implement a robust simulation loop mechanism to control the flow of simulation time and the number of loop for either a scenario or a specific component of the simulation. This mechanism include 2 modes: 0 (for infinite number of loop), and n (for n loops). In addition to this easy mechanism, it is possible to add an event script and event variables to a loop. In this context, each time a loop is over, the associated event is triggered and the next loop is started with the content of the event script and the new value for the event variable. Moreover, the loop mechanism can be associated with the interpolation mechanism which allow to provide a trajectory for an object with a set of Points of Interest (position, orientation) and a time step to respect between each points. So, these mechanisms handle initialisation, time stepping, and termination conditions. These mechanisms Ensure that the simulation loop and the interpolation iterate at a consistent and predictable rate to maintain temporal accuracy and reproducibility.

- **Simulation Pause and Resume**: Enable users to pause and resume simulation execution at any point to inspect intermediate states, analyze data, or make adjustments. Provide mechanisms for saving and restoring simulation states to facilitate seamless transitions between paused and active states.

- **Time Synchronisation with External Systems**: Facilitate time synchronisation between the simulation environment and external systems, such as hardware-in-the-loop (HIL) setups, networked simulations, and distributed computing platforms. Ensure accurate alignment of simulation time with external time references to maintain coherence and consistency across interconnected systems. In Pro-SiVIC, this function is handle by the plug-in sivicTime and allows to send a ”master time” to external platforms like RTMaps, Matlab, ROS. In RTMaps, this time coming from Pro-SiVIC allows to synchronise the module running in a RTMaps diagram. RTMaps becomes the time ”client” and Pro-SIVIC the time ”server”. From the RTMaps side, 2 times are handle, the first one is the TimeOfIssue (date of data generation by the sensor) and the second one the TimeStamp (time of availability by the processing component). Generally, the TimeOfIssue is lower than the TimeStamp.

- **Temporal Accuracy and Precision**: Ensure temporal accuracy and precision in simulation time management to minimize time drift, jitter, and synchronization errors. Employ high-resolution timekeeping mechanisms and numerical integration techniques to maintain temporal fidelity across simulation components.

- **Time-Dependent Simulation Effects**: Model time-dependent simulation effects, such as dynamic weather patterns, diurnal variations, seasonal changes, and time-of-day effects, to simulate realistic environmental conditions and their impact on system behaviour. In Pro-SIVIC, this aspect is managed by the event, loop, and interpolation mechanisms.

- **Distributed Time Management**: Facilitate distributed time management across multiple simulation nodes or instances to coordinate synchronised simulations, parallel processing, and distributed computing tasks. Implement mechanisms for clock synchronisation, message time-stamping, and event coordination to ensure temporal coherence in distributed

simulation environments. This functionality is not yet fully functional in Pro-SiVIC but it is in progress in the ImPACT 3D platform (interconnection of ImPACT 3D AV, ImPACT 3D VR&Motion, and the pedestrian crossing street simulator). In order to manage efficiently this Distributed Time Management, NTPs and PTP modules will be implemented.

These requirements aim to ensure effective time management in a simulation environment, encompassing various aspects such as graphical rendering, sensor behavior, dynamic vehicle dynamics, event handling, and synchronisation with external systems.

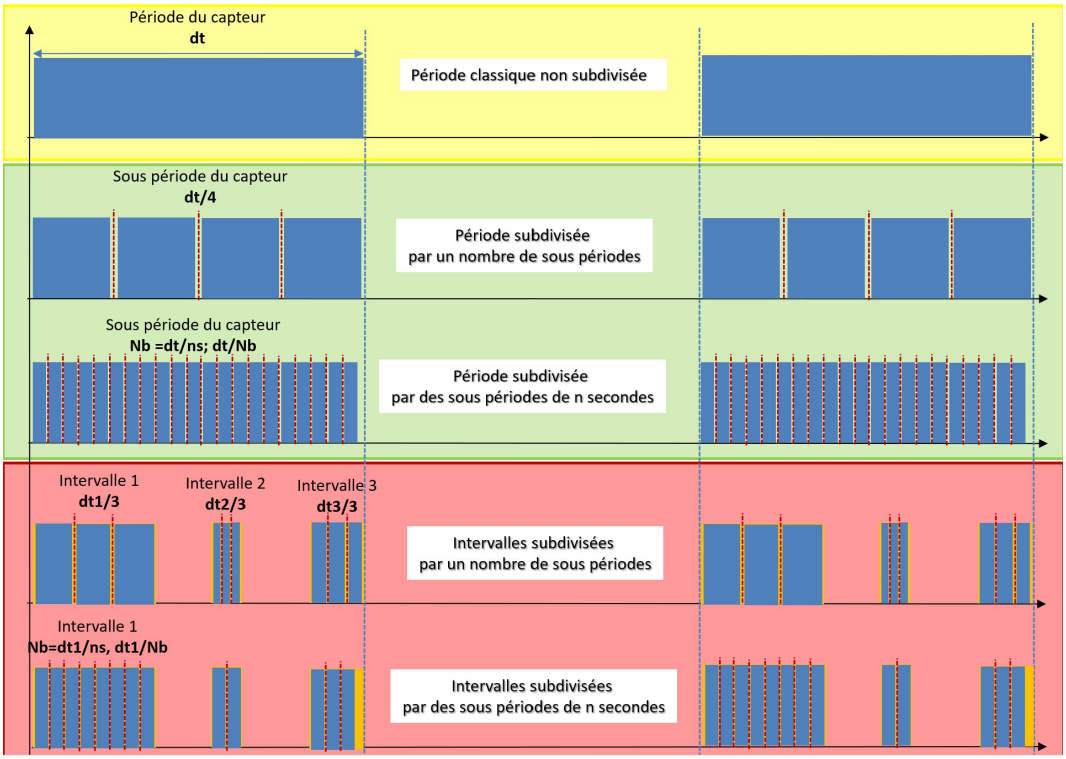


Figure 82: Overview of the different time mechanisms and time period available in Pro-SiVIC and managed by the plug-in sivicReorder (source UGE)

3.4.4 Definition of event mechanisms and event engine

In PRISSMA's generic framework, several event mechanisms have been used. These mechanisms strongly depend on the used platform. For instance in Pro-SiVIC, the implemented mechanism of events management respect a part of the previously detailed requirements.

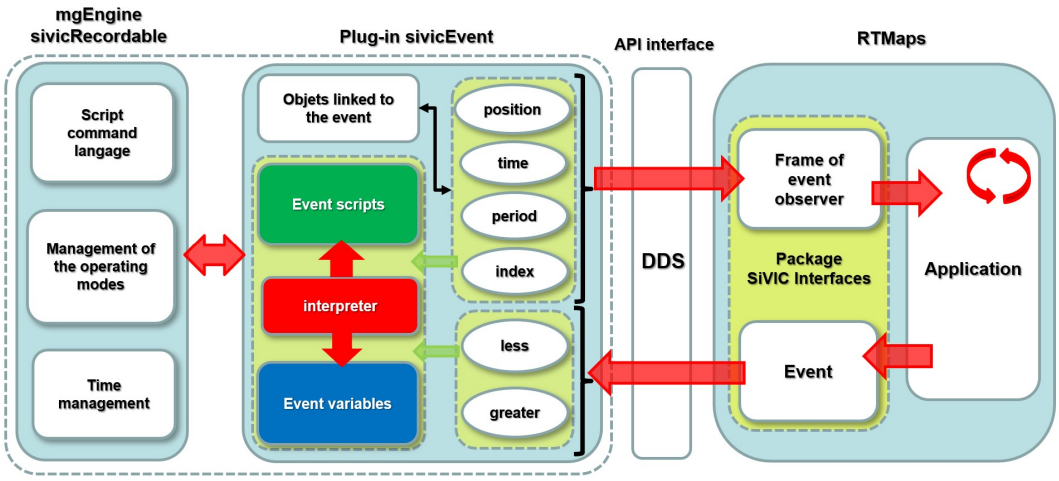


Figure 83: Overview of event management in Pro-SIVIC (source UGE)

3.4.4.1 Pro-SiVIC event management

- Event Definition: Pro-SiVIC proposes and provide a mechanism (plug-in sivicEvent) to define various types of events (spatial, temporal, semantic) that can occur within the simulation, such as objects interactions, system events, sensor behaviours and states, and environmental changes (weather conditions).

- Event Scheduling: Enable users and components to schedule events at specific simulation time points or relative to other events, allowing for precise control over the timing and sequencing of simulation activities. This scheduling is handle with the graphical engine and the sivicEvent plug-in.

- Event Triggers: Define triggers or conditions that initiate the execution of events, including temporal triggers (time-based), state triggers (condition-based), and external triggers (input-based). This aspect concerns the so-called DDS event managed from external application and environment like RTMaps. In this context, we have 2 possible triggers: either an event variable or value (coming from a component or the output of a module (perception, localisation, ...) is less or greater than a threshold value (this threshold can be updated following some constraints by an event script).

- Event Handlers: Implement event handlers or callbacks to execute predefined actions in response to triggered events, such as updating simulation parameters, modifying object states, or generating new events. In Pro-SiVIC, this aspect is managed by event script. Each event can implement several event script. In an event script (see figure 85 we can set a set of Pro-SiVIC script command with an ID put at the beginning of each command line. If the ID takes the value 0 then it means this command need to by applied each time the event is triggered. If ID takes the value n then it means the command will be applied at the nth triggering of the event.

- Event Queuing: Maintain an event queue or buffer to store scheduled events in chronological order, facilitating efficient event processing and ensuring proper sequencing of event execution. In Pro-SiVIC, each event can access to the pointer of the list of event. This means that an interaction and a combination of events could be implemented if needed.

- Event Propagation: Enable propagation of events across simulation components and subsystems, allowing events to trigger cascading effects and interactions between different elements of the simulation environment. This aspect is managed by the event script. Indeed, when a event occurs, then an event script is executed with inside a set of script commands impacting and modifying several components-objects-states. With the ID value, different sub event script can be executed depending of the current situation.

- Event Logging: Log event occurrences, timestamps, and associated metadata to facilitate post-simulation analysis, debugging, and performance monitoring, providing insights into simulation dynamics and behaviour. In pro-SiVIC, an event is also considered as an information source. This means, an event can be managed as a sensor with its own operating frequency and at each time, a data frame will be generated with the current state of the event. This mechanism is mandatory because it provides a ground truth about the event triggering an action or a behaviour change.

- Event Coordination: Coordinate events between concurrent simulation processes, distributed simulation nodes, or interconnected simulation modules to maintain synchronisation and consistency across the simulation environment. This mechanism is operational with DDS event and semantic events managed from third party application and software.

- Event Interception: Allow for the interception and modification of events before their execution, enabling advanced event processing techniques such as event filtering, transformation, and augmentation. This modification is available in Pro-SiVIC by modifying the content of an event script or an event variable, or an event threshold. Moreover, depending of specific situation and condition, an event can be restart and take its initial configuration. As an event has operating mode similar to a sensor, it is also possible to fix the mode of the event to ”off”.

- Event Visualisation: Provide visualisation tools and techniques to represent event occurrences, sequences, and dependencies graphically, aiding in the comprehension and analysis of simulation dynamics and behaviour. This functionality is available because the sivicEvent plug-in generate a frame (observer) with the information provided in the figure 84

- Event-Based Control: Enable event-based control strategies to regulate simulation activities, trigger system interventions, and adapt simulation parameters dynamically based on evolving simulation conditions and external stimuli. In Pro-SiVIC, this mechanism is managed by DDS event. In the current application, the mechanism is used and implemented in RTMaps in order to trigger a set of actions, modifications, behaviour from the ”application” environment.

oSimuTime	<MAPS::Float>	SimuTime (ms).
oIteration	<MAPS::Float>	Current iteration.
oTypeOfEvent	<MAPS::Float>	Type of event.
oTime	<MAPS::Float>	Time (ms).
oDeltaTime	<MAPS::Float>	Delta time (ms).
oTimePeriodEvent	<MAPS::Float>	Time period event (ms).
oIndice	<MAPS::Float>	Indice.
oIndiceEvent	<MAPS::Float>	Indice event.
oPositionX	<MAPS::Float>	Position x (m).
oPositionY	<MAPS::Float>	Position y (m).
oPositionZ	<MAPS::Float>	Position z (m).
oPositionEventX	<MAPS::Float>	Event position x (m).
oPositionEventY	<MAPS::Float>	Event position y (m).
oPositionEventZ	<MAPS::Float>	Event position z (m).
oDistance	<MAPS::Float>	Current distance (m).
oThreshold	<MAPS::Float>	Distance threshold (m).

Figure 84: Output frame generated by the event observer (source UGE)

```
# if we have a 0 value, one executes at each time
# else the index determines which command lines must be executed.

# The condition has been reach a first time.
# The command lines with the index value 1 are executed.
1 ambient                        .7 .7 .7
1 pedestrian1.SetGlobalPosition 556.7000 158.00000 -0.0500000
1 pedestrian1.SetGlobalRotation 0.000000 0.000000 0.000000
1 pedestrian1.SetCastShadows    1
1 pedestrian1.SetMesh           pedestrian1
1 pedestrian1.SetCatchShadows   0
1 box.SetGlobalPosition         14.7000 9.00000 0.000000

# The condition has been reached a second time.
# The set of script command with the index 2 are executed.
2 ambient                        .6 .6 .6
2 pedestrian1.SetGlobalPosition 16.7000 9.00000 0.000000
2 pedestrian2.SetGlobalPosition 6.7000 9.00000 0.000000
2 puppet2.SetMode               1
2 puppet2.SetIndice             385
2 car2.SetIndice                1800

# The condition has been reached a third time.
# The set of script command with the index 3 are executed.
3 ambient                        .5 .5 .5

# The script commands with the index 0 are executed for each
# call of the event script.
0 car.GetIndiceEnd
0 car.SetIndice                 1800
```

Figure 85: Example of event script. The command script with the first ID character with 0 value means these 2 command lines will be executed for each trigger of this event. The 7 command lines with the ID equal to 1 mean these command line will be executed at the first event trigger (source UGE)

3.4.4.2 Explanation of the sivicEvent functionalities

In order to be able to dynamically manage events relating to a set of criteria such as time, period, position, distance and others, an event management mechanism has been integrated into Pro-SiVIC. These events and their management are carried out using the sivicEvent plug-in. The main purpose of this mechanism is to be able to enrich scenarios using modifications and conditional triggers. Each type of event is constructed generically. An event is triggered (execution of an event script) only when a condition is met by a ”positionable” object or a mobile object such as a vehicle or pedestrian. For example, if the ”timesivic” criterion is selected and a period of 500ms is set, then the event script will be called and executed after an elapsed time period of 500ms. In the case of a “position” type event, the event script will be called and executed when an object or a vehicle chosen as the source of the event is within a fixed radius.

The events that can be managed are currently of the “position”, “inter distance between objects”, “trajectory index”, “date”, “time interval”, “larger” and “smaller” type (outdoor mode managed from an RTMaps type application).

- position : the reference object is close to the defined position.
- distance : the reference object is at a distance from another vehicle.
- index : the reference vehicle or pedestrian is at a trajectory index.
- time : the reference vehicle is on a date.
- timesivic : the reference object has operated for a time interval.
- ddsless : conditional value less than a threshold
- ddsgreater : conditional value greater than a threshold

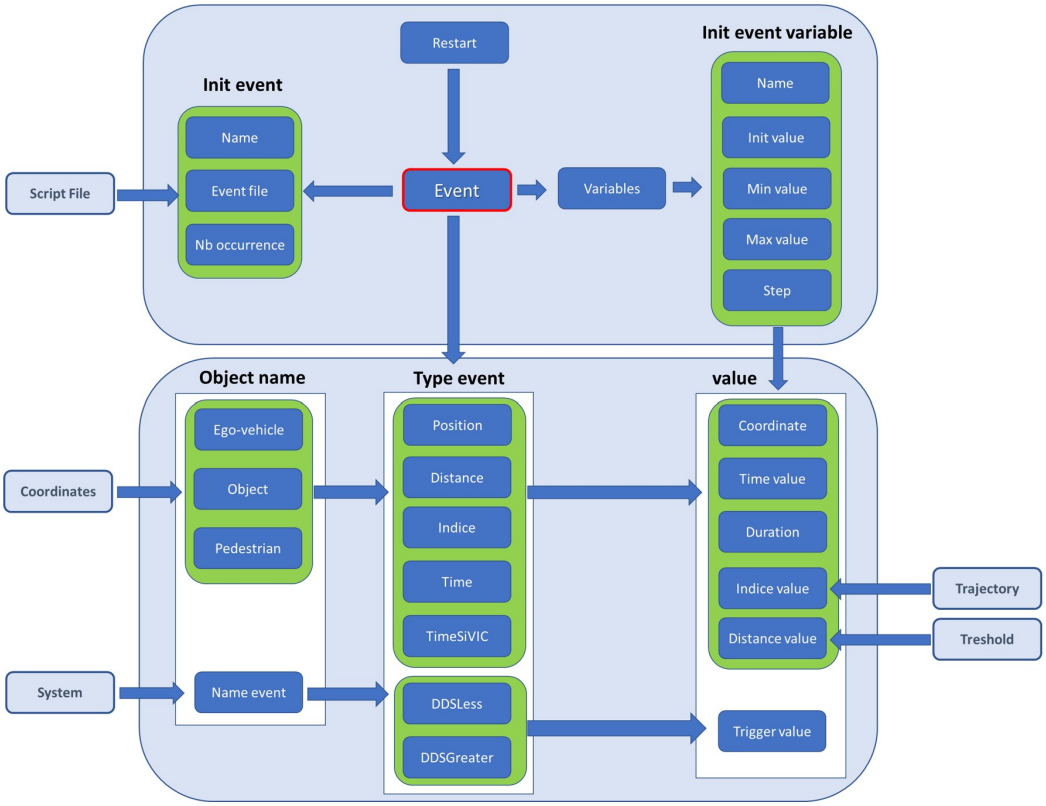


Figure 86: Overview of the event management mechanism implemented in Pro-SiVIC (source UGE)

A number of iterations can be defined in order to specify the set of commands to be carried out following the realisation of a specific event. This sivicEvent plug-in inherits from the operating mode and time management plug-in (sivicRecordable). This implies that it accesses the same mode of operation as all the plug-ins available in the Pro-SiVIC platform. Thus, to disable the management of an event management plug-in (sivicEvent), the Off mode will be selected. To activate the taking into account of an event, the On mode will be activated.

The main event script commands are:

- *new sivicEvent e* : create a new event call e
- *e.SetNbIteration < nb iteration>* : number of event occurrence
- *e.GetNbIteration* : get the number of occurrence
- *e.SetEvent <object_name><type_event><value>* : configuration of the type of event. <object_name> is the identifier of the ”positionable” object or the reference vehicle that will serve as the source for triggering an event.
- *e.SetDistanceTreshold <value>* : In both position and distance modes, the distance threshold allowing an event to be triggered can be modified using the following command
- *e.GetDistanceTreshold* : allows you to know at any time the value of the threshold distance

In order to reset an event, the ReStart (e.ReStart) command can be used. Of course, several events with same or different types can be done. Moreover, an event can act on another one.

In order to simplify the use of events and to be able to manage automatic modifications of the attributes of an object, an event variable management mechanism is implemented. The declaration of an event variable is done through its identifier preceded by a $, its initial value, the lower and upper bounds of the range of possible values, and the increment step of the variable. Once a variable is declared, it can be used in the event script to override the value of an attribute of a script command. In order to be able to set an event variable, the following command is used:

- *SetVariable $<name><init_value><min_value><max_value><step>*

$name determines the name of the variable. The $ identifies the declaration of a variable. init_value sets the initial value to be used. min_value sets the lower bound of the range of possible values. max_value sets the upper limit of the range of possible values. step gives the incremental step of the variable.

Once a variable is declared, it can be used in the event script to override the value of an attribute of a script command. An event variable can be modified and handle with the following commands:

- *RemoveVariable $<name>* : delete an existing event variable
- *GetNumberVariables*
- *GetVariable $<name>* : give the current configuration of an event variable
- *GetVariables*

3.4.5 Definition of peripheral access and management

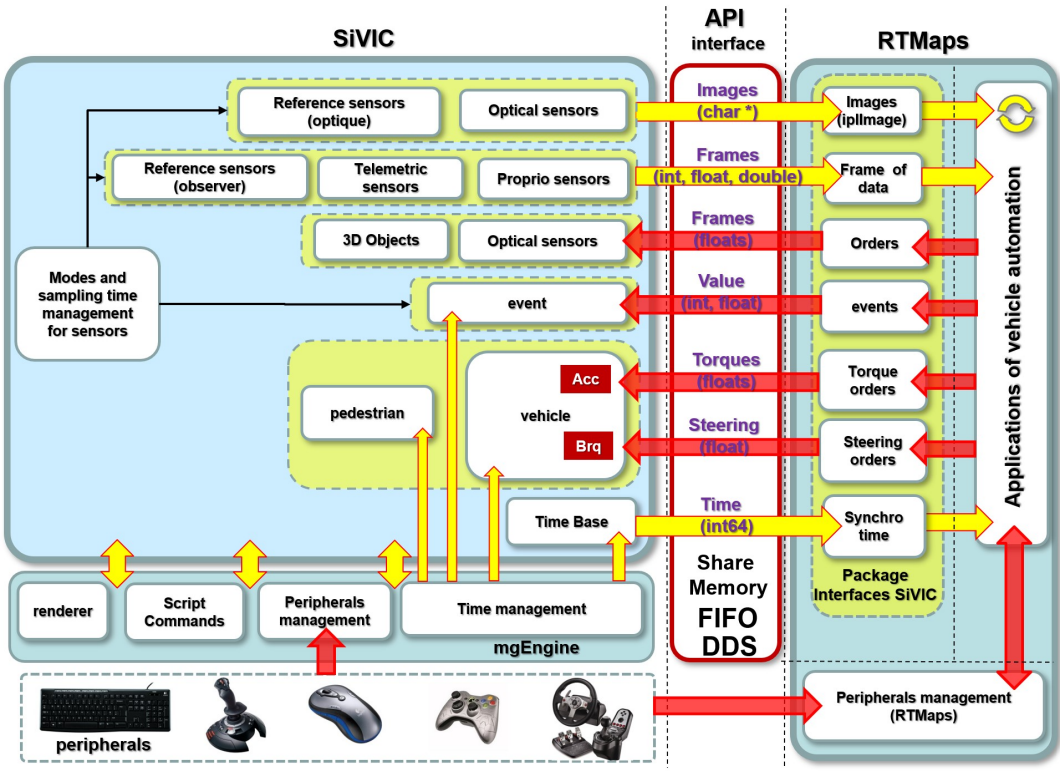


Figure 87: An overview of the main data flow and peripheral management (Source UGE)

3.4.6 Definition of ground truth and references types, and generation

Automated vehicles rely heavily on perception systems to interpret their surroundings and generate decision and information for path planning module allowing the operating of automated driving. However, developing and validating these perception systems require extensive testing and evaluation in simulated environments before real-world deployment. Ground truth data, which provides accurate and reliable annotations of the environment, is essential for both training and validating perception algorithms. Without ground truth data, it is challenging to assess the performance of these AI-based algorithms objectively.

In PRISSMA and with Pro-SIVIC, we have proposed to develop a generic framework for the definition, generation, execution of scenario, the definition, generation of DataSet, et the evaluation and validation stages. This framework is called SiVIC-ADVeRSce. The 2 stages are given by the figure 88 and figure 89. In both cases, the ground truth identification, configuration, generation, and using are essential stages in both the DataSets production and the evaluation and validation stage.

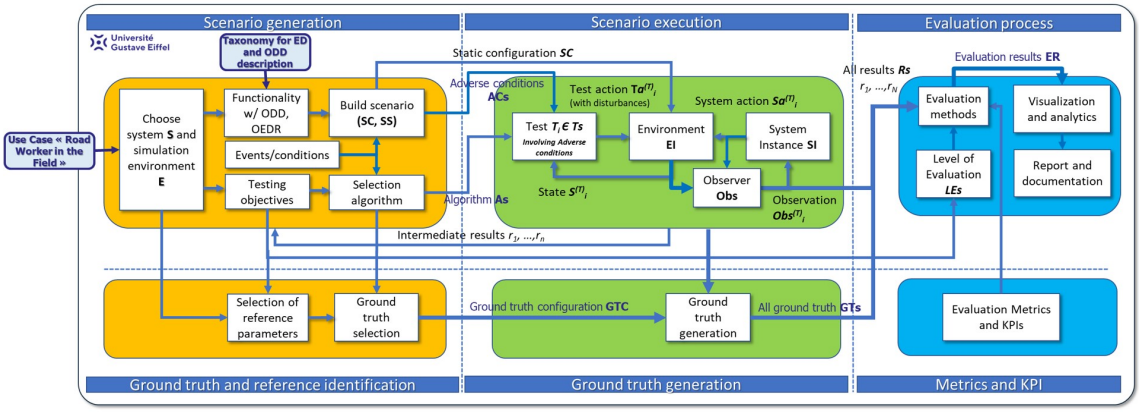


Figure 88: Generic Conceptual Framework of SiVIC-ADVeRSce: The scenario definition, management, execution (Source UGE)

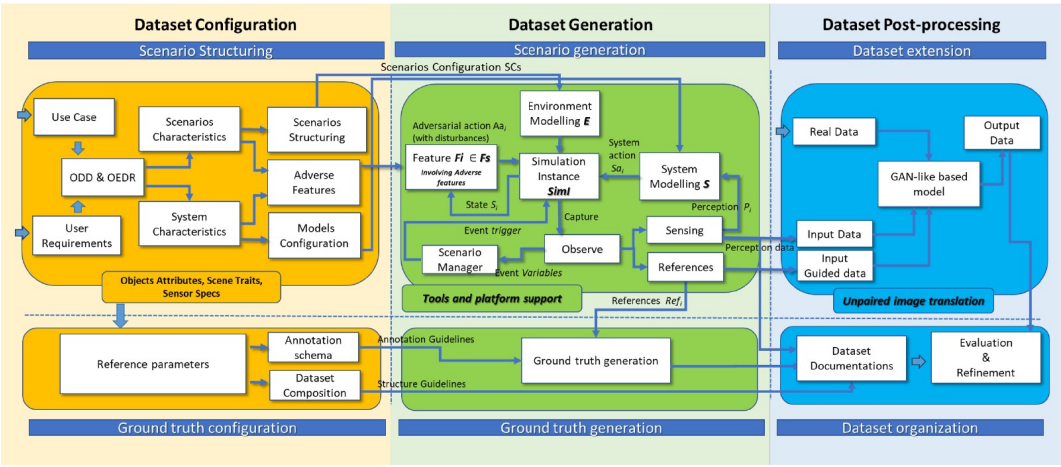


Figure 89: Generic Conceptual Framework of SiVIC-ADVeRSce: The Dataset definition, generation, and post processing (Source UGE)

During the simulation process of Pro-SiVIC, perception data and ground truth data are gathered using specific plug-in in Pro-SiVIC and some module in RTMaps through the data-sharing mechanism developed to apply an efficient interconnection between several applications either on the same computer or remote. For instance, as illustrated in Figure 90, simulated image frames produced in Pro-SiVIC are captured and stored by the sensor module defined in RTMaps to construct the dataset. Various mechanisms embedded within Pro-SiVIC are employed to generate the reference data. One method involves altering the rendering texture of objects and the environment (such as vehicles, pedestrians, lanes, roads, buildings, etc.), resulting in the creation of segmentation masks (depicted in Figure 90) that are then collected as part of the reference data.

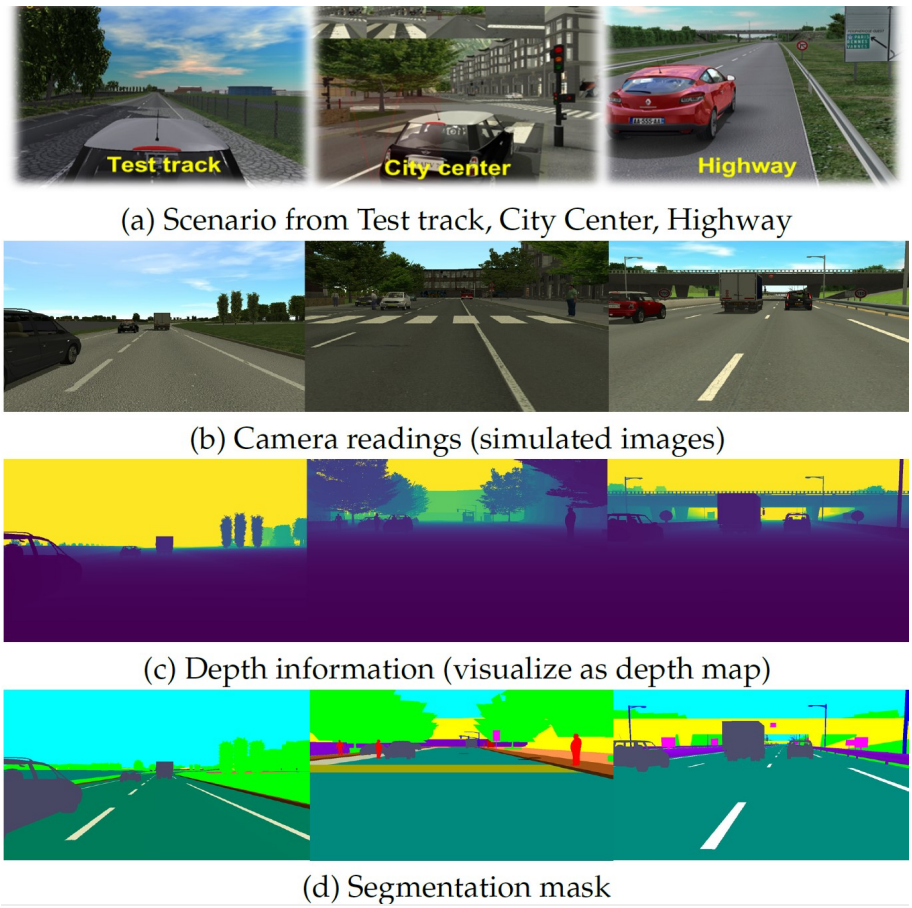


Figure 90: Generic Conceptual Framework of SiVIC-ADVeRSce: Depth Map and segmentation generated by Pro-SIVIC (Source UGE)

In addition to visibility-based mechanisms, a specific mechanism known as the ”observer” in Pro-SiVIC facilitates the real-time generation of the state vector of different objects in the scene, including vehicles, pedestrians, static objects, and road configurations. Notably, the depth matrix (visualised in Figure 90) of sensors can also be captured as reference data, which can contribute to refining annotation and improving the dataset.

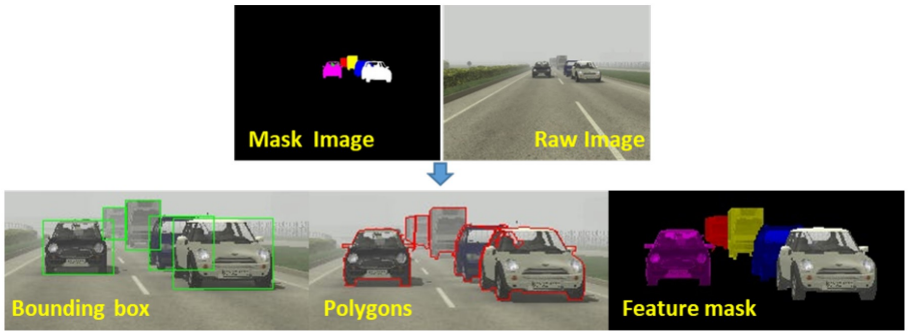


Figure 91: Generic Conceptual Framework of SiVIC-ADVeRSce: Generation of a set of annotation (Source UGE)

As defined in the conceptual framework, the annotation labels are generated based on the configured annotation schema from the upstream layer. In the implemented SiVIC-ADVeRSce

framework, by leveraging various mechanisms for reference data generation, multiple annotation schema possibilities have been provided, each corresponding to different functional aspects of perception. These annotation schemas are primarily categorised into object, semantic, and temporal domains, enabling comprehensive annotation of multi-data modalities. Within object annotations, the goal is to obtain precise annotation of objects using bounding boxes, polygons, and pixel-level masks. Figure 91 illustrates the different object annotations in the implemented framework. The second type of annotation implemented in SiVIC-ADVeRSce, namely semantic annotations, encompasses the entire perception data and allows for the extraction of coherent sub-segments or regions, assigning meaningful labels to each segment based on its semantic content. Furthermore, SiVIC-ADVeRSce extends its annotation schema to include the temporal aspect, enabling the annotation of timestamps, events, and temporal segments. This feature aligns with the virtual timestep in Pro-SiVIC, ensuring accuracy and consistency throughout the annotation process.

3.5 PRISSMA's Sensor models (LNE, UGE, CEREMA, AVS, TRANSPOLIS)

Sensor models come in 3 levels of fidelity, Level 1 ideal model, Level 2 Real-time physics and Level 3 Full Physics.

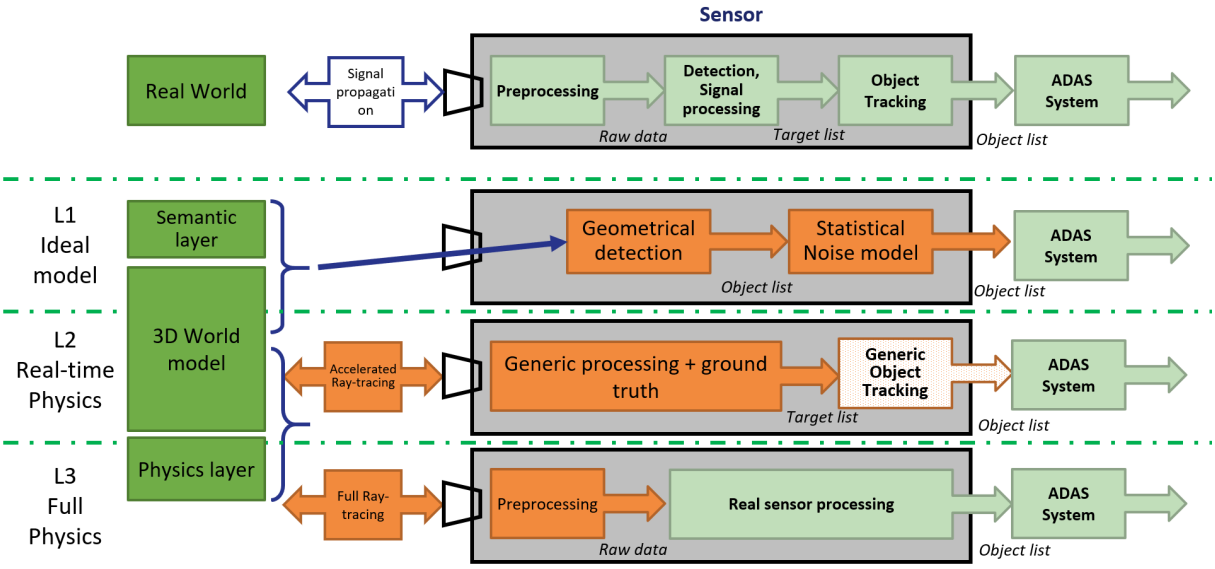


Figure 92: Different levels of sensor modelling (Source AVS)

Level 1 is the ideal representation of the real world sensor. It does not simulate the signal propagation or any of the internal processing inside the sensor. It only uses the 3D World model and the Semantic layer. It applies simple geometrical detection to verify which objects can be detected and then uses the semantic layer to compute the object list.

Level 2 Real time physics models a re able to perform real-time simulation under certain conditions. This model does not need sensor processing. It is used in the case :

- no need for physic effects in the signal propagation
- Have no access to the real processing Signal Processing

- Only feed the ADAS system or access the data at an intermediate level

Level 2 sensor are similar to level 3 sensors however differ on two points:

- Relies on an accelerated ray tracing
- Applies a generic processing that is able to deliver data at a higher level

Level 3 Full physics sensor aims to deliver super-realistic raw data that can feed the real sensor processing. It is based on a Full Physics model and focuses on signal propagation, and uses the 3D World model and the physics layer.

The main goal is to be realistic on:

- Full ray-tracing
- Signal Processing
- The interaction of the signal with the World model and the Physics layer

Full ray-tracing is applied to simulate how the waves propagate in the scene and arrive at the entry of the sensor. Preprocessing step is then simulated to be able to deliver the raw data. The sensor processing is needed to be able to reconstruct the object list. Without it, only raw data is available that cannot be directly fed to the ADAS system.

3.5.1 Definition of the Camera models and components involved

A camera model is capable of simulating a real camera as it would be attached to a vehicle. This sensor generally will have two types of outputs: in image of the scene, and conveniently information about the road marking.

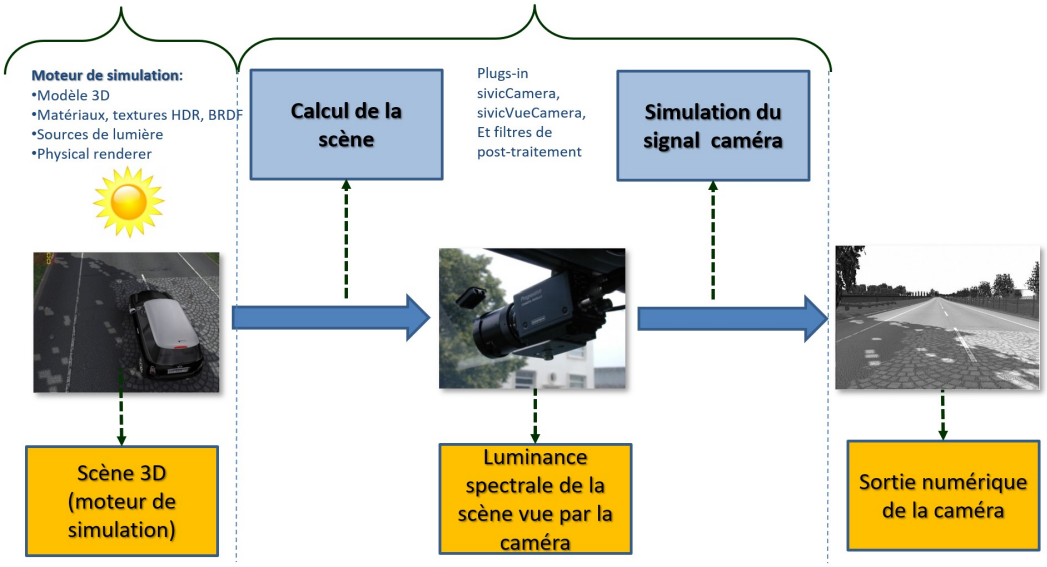


Figure 93: Overview of the Camera modules in Pro-SiVIC platform (source UGE)

Camera models can be used to simulate for instance a park assist camera, by displaying the parts of the scene that would be viewed by the rear camera. Using a 3D visual rendering engine

such as Unreal Engine in the simulator, ray tracing and physics based rendering materials can be implemented into the scene to have a more realistic output image. Camera models also have the ability to add an optical lens distortion effect that provides a more realistic output image of the scene. The distortion effects can be applied by defining the optical properties of the lens, described by 3 values ; θ First value the input angle in degrees from the camera sensor optical axis, Y the second value in millimetres the distance of the projected point from the optical centre in the distorted image and the third value YO the distance in millimetre of the projected point from the optical centre in the original undistorted image.

$$YO = tan(\theta) * \text{focal length}$$

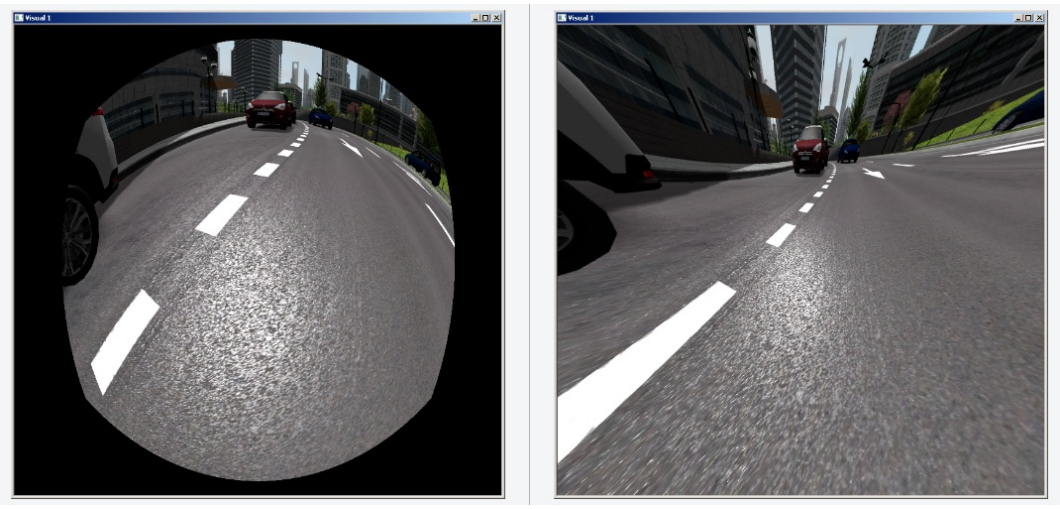


Figure 94: Different types of distortion available on the output image (Source AVS)

It is also possible to modify the properties of the output image in order to have a resulting image that is much more similar to the real camera. Many properties such as the Depth of field, Bloom, Chromatic aberration, white balance and image effects can be added to add a certain depth and realism to the output image.

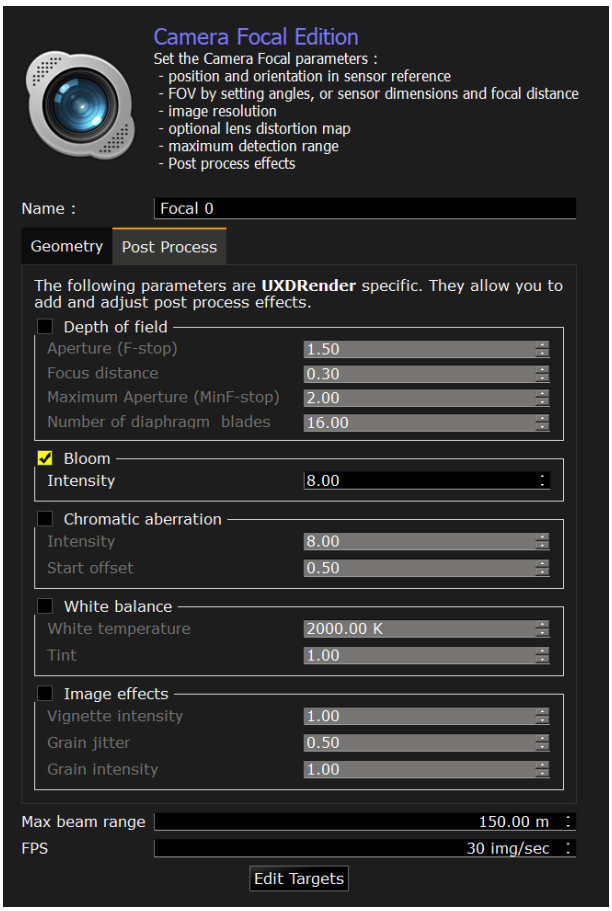


Figure 95: Post processing options available via the Camera Sensor module (Source AVS)

Virtual Camera Sensors are also able to simulate detection of objects and lines, and send over this information to be manipulated by and external ADAS systems attached to the simulator. However it must be take into account the these detection are considered perfect detection from the perfect sensors and thus contain no influence of perturbation, weather influences, message interference's. All the detection are as if the camera is able to detect 100% of all the objects on the scene. However it is also possible to choose the types of objects to be detected such as detection of only objects, vehicles or pedestrians. Bounding box information is also provided. Another possibility is to send the video flux to your program of choice, and executing your detection program, computing the decision and rebroadcasting that information back to the simulator to integrate this information to the other systems (vehicle processes etc..).

Figure 96: Perfect camera sensor and object detection (Source AVS)

In pro-SiVIC, the modelling of the camera is mainly based on the using of a filter mechanism (see figure 97) and proposed a large set of filters allowing to obtain a physical behaviour very close to the intrinsic operating of a real camera (see figure 98)

- Cameras RGB
- Event-based camera
- IR
- Fisheye
- Cyclop

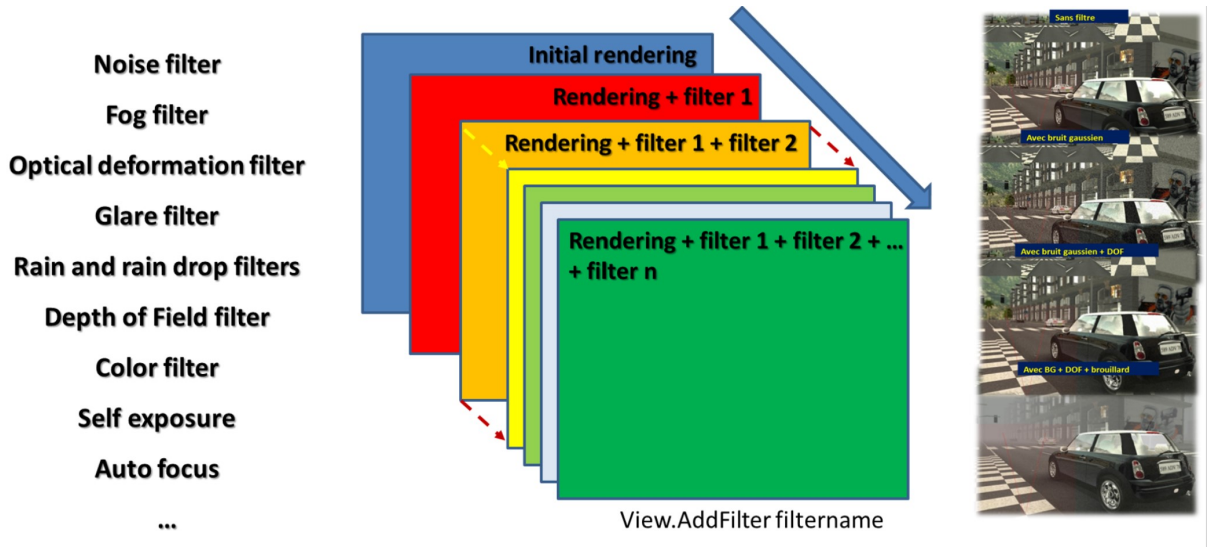


Figure 97: Modelling of the Pro-SiVIC camera with the filter mechanism. In this framework, it is possible to add a large set of filter in order to obtain a set of relevant disturbances on the final image generated by the virtual camera. (Source UGE)

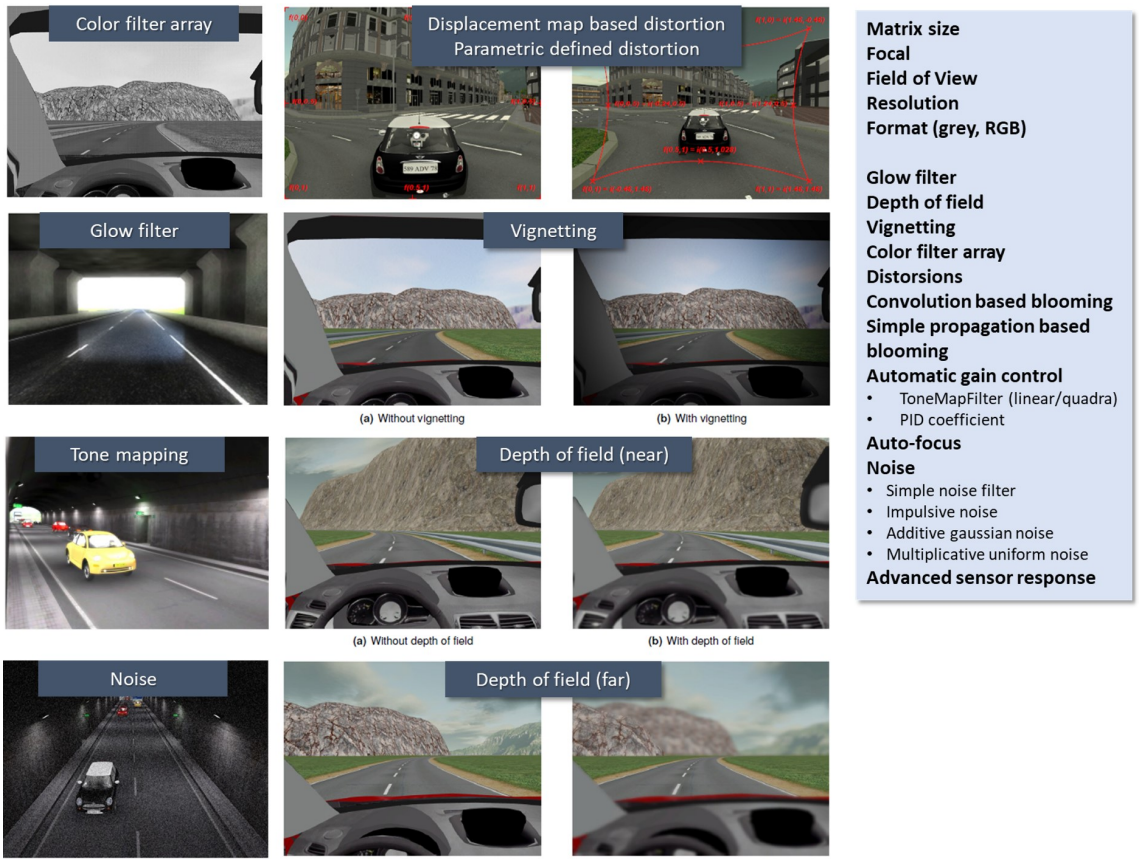


Figure 98: Modelling of the Pro-SiVIC camera with the main parameters and the different filters apply to the images generated by the renderer. The intrinsic result are simular than a large set of real cameras. (Source UGE and ESI group)

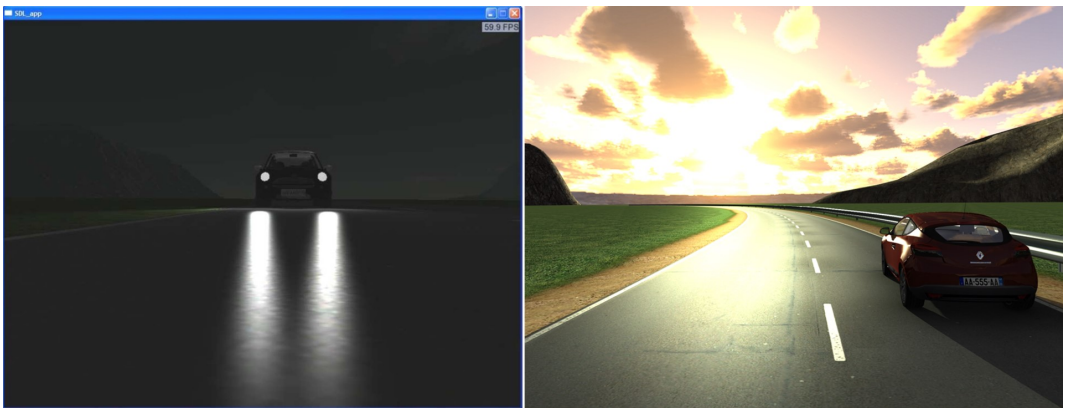


Figure 99: Modelling of the Pro-SiVIC camera with a pixelic rendering taking into account ligth sources and reflection of ligth effects. (Source UGE and ESI group)

Figure 100: Behaviour of the Pro-SiVIC camera by night with different configuration and performances level. This set of scenario with different camera dynamics shows the impact on the obstacle detection (a truck) stopped on the right lane (Source ESI group)

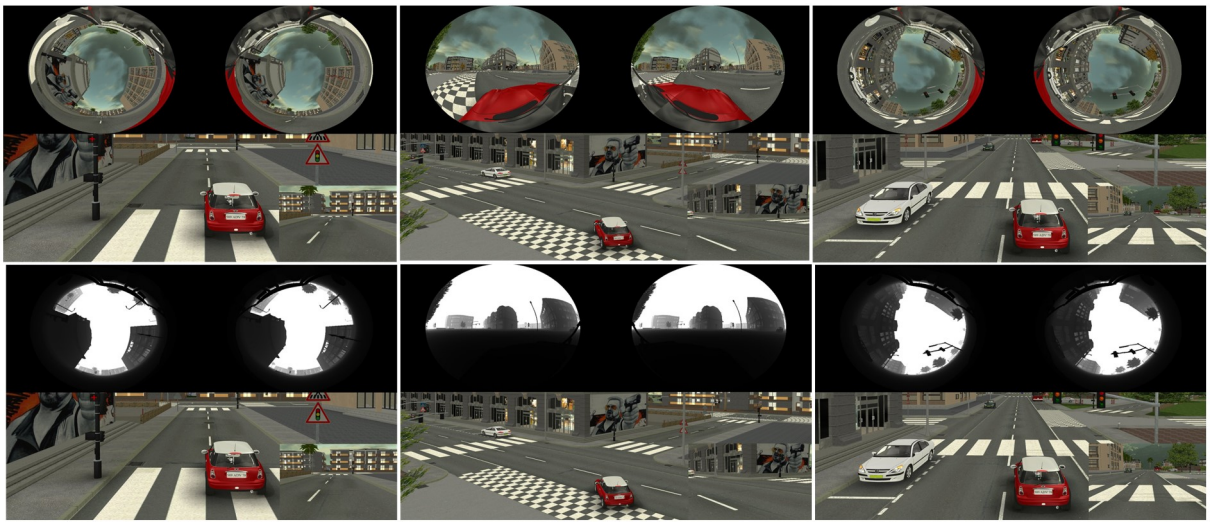


Figure 101: Modelling of the fisheye camera in the Pro-SiVIC platform (Source UGE)

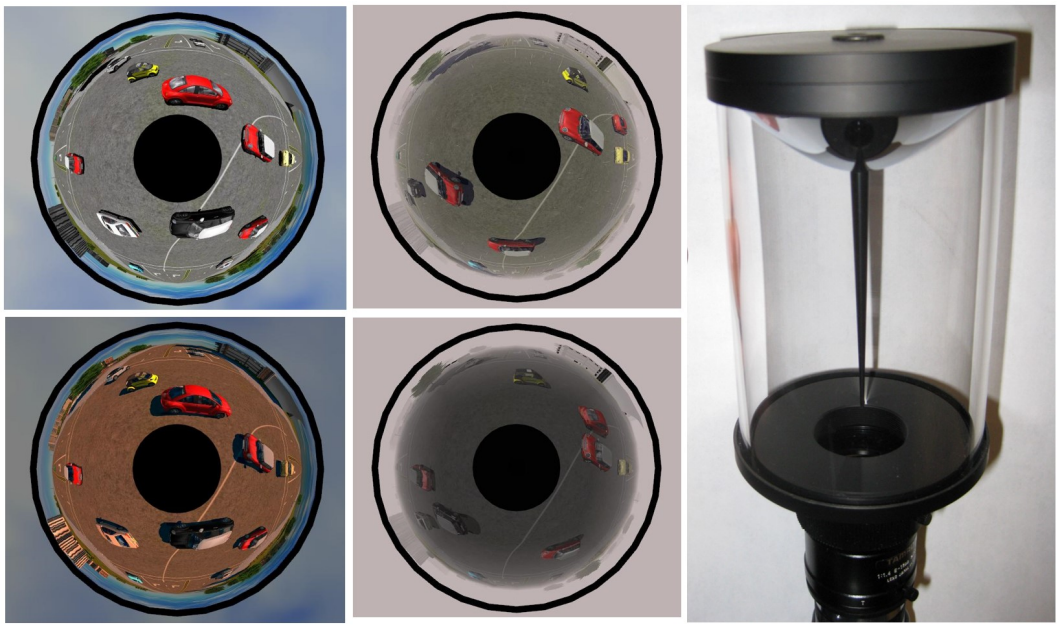


Figure 102: Physical modelling of the cyclop (omnidirectional) camera in the Pro-SiVIC platform (Source UGE)

3.5.2 Definition of the RADAR models and components involved

Radar sensor can detect mobile and infrastructure targets located in the sensor detection area. The module then send information about each detected target such as its type, name, absolute and relative positions, speeds etc.. In the case of a level 1 RADAR, no RCS information is needed.

In the case of Physcal Radars, the radar simulation is based on the material components and the tools to apply materials to the object (asset creation) A ray-tracing engine is used to resolve the asymptotic formulation of Maxwell equation to take into account the physical optics and geometrical optics to simulate the wave propagation.

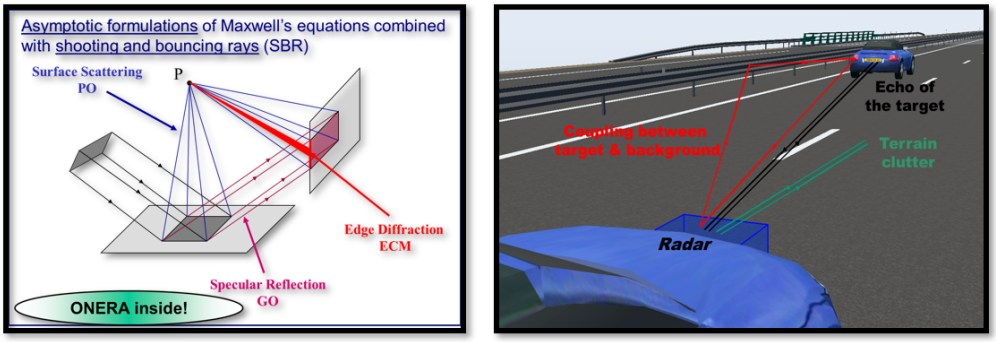


Figure 103: An example of how the rays are computed. In reality, millions of rays are computed to contribute to the processing (source AVS)

Ray tracing is computed to take into account all the multiple reflection and the interaction the exist between the target and the environment. Dependant on the level of modelling of the environment, this coupling is simulated.

In the image below, we can observe an urban scene. In the scene, metallic barriers, a bridge and building are modelled.

Figure 104: Simulation environment (source AVS)

If only a few rays are launched, a clear separation between the targets in the environment can be observed. In reality however, many multiple reflections/contributions occur between all the elements of the scene. This phenomenon makes it difficult to separate the target from the environment, and causes detection errors/imprecision from the detection/recognition algorithms. To evaluate the performance of the detection algorithms, these phenomenons will be simulated since real raw signals normally exhibit this complexity.

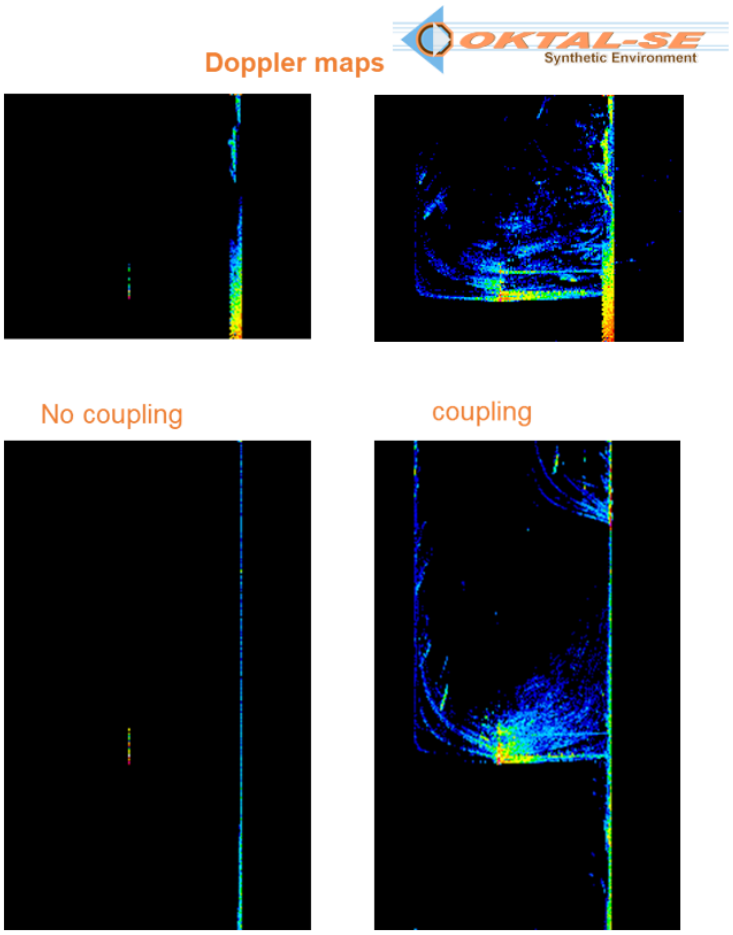


Figure 105: Coupling of the environment and the effects on the signal (source AVS)

EM Ray-tracing features are able to compute near fields and far fields high frequency scattering due to metallic and/or dielectric materials. This includes:

- Reflection
- Transmission
- Scattering
- Diffraction of electromagnetic wave interaction

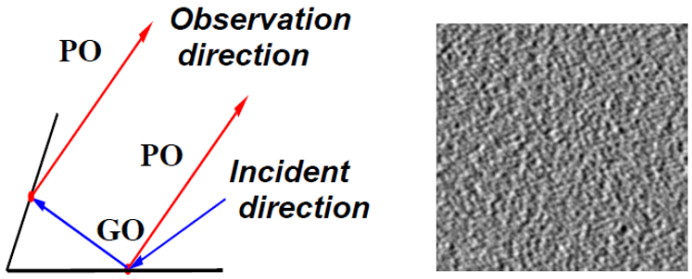


Figure 106: EM Ray tracing (source AVS)

When a ray is launched and created reflections, obtained are:

- Geometrical Optics (in blue): only consider specular reflection (just simple reflection)
- Physical Optics (in red): emission of energy scattered back to the observer when you have reflection

Difference between level 2 and Level 3 Radar simulation models

For both Level 2 and Level 3 models, the user will need to build and model their scenes, including the terrain and 3D Objects. The scenario is the defined to set the behaviour of the dynamic objects.

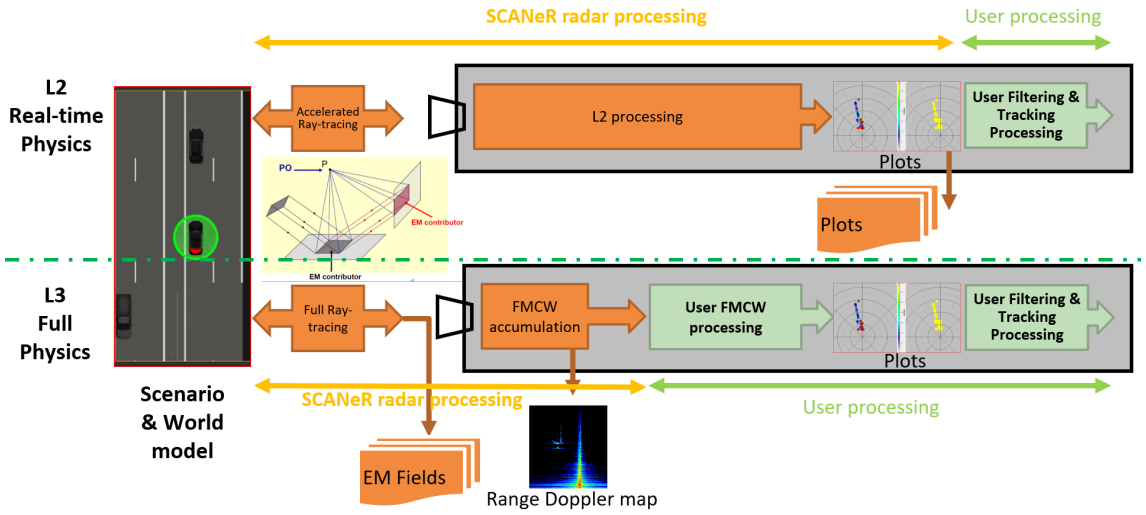


Figure 107: Difference in L2 and L3 radar model simulation (source AVS)

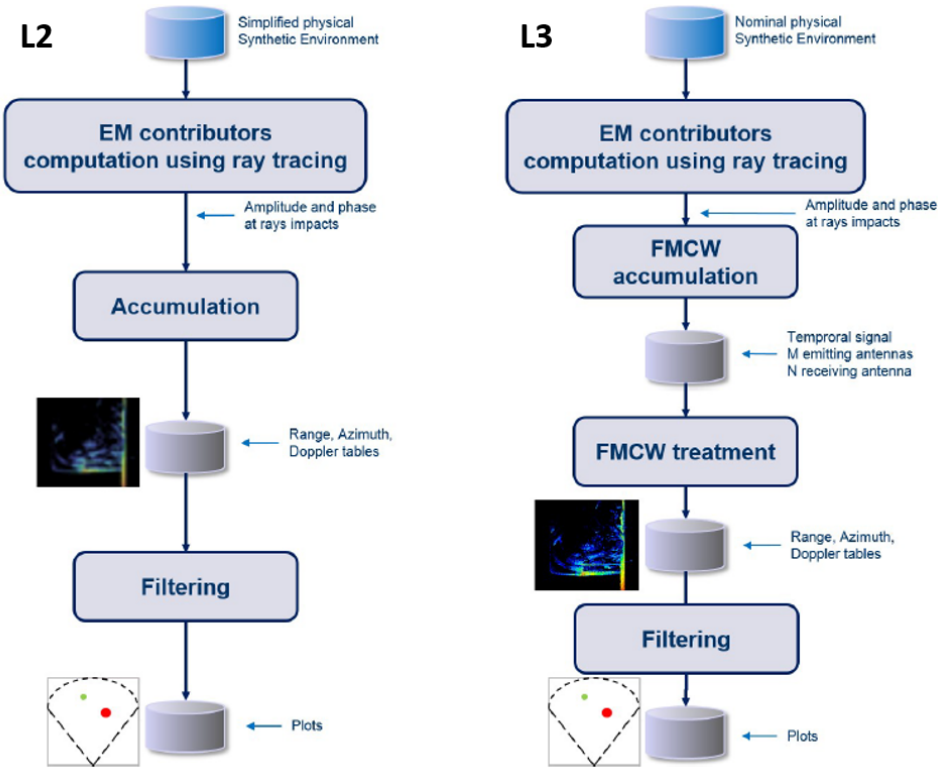


Figure 108: Difference in L2 and L3 radar model simulation (source AVS)

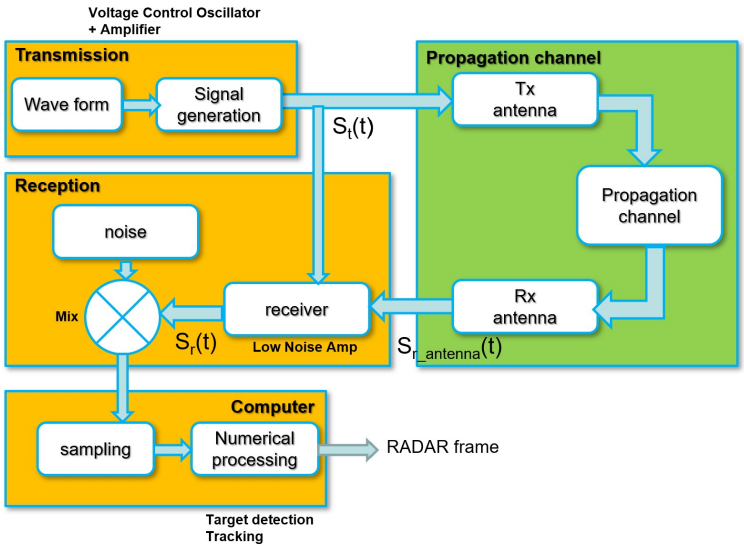


Figure 109: Overview of the RADAR modules (source UGE)

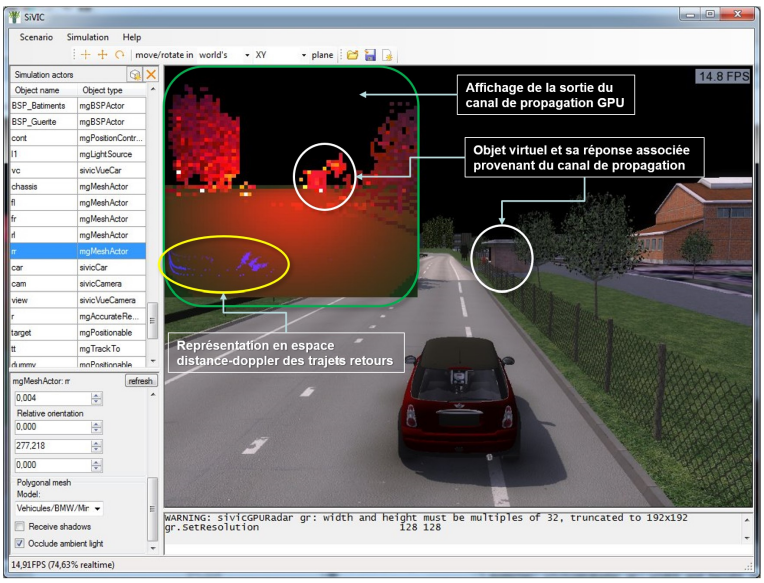


Figure 110: Overview of the RADAR modules (source UGE)

3.5.3 Definition of the LIDAR models and components involved

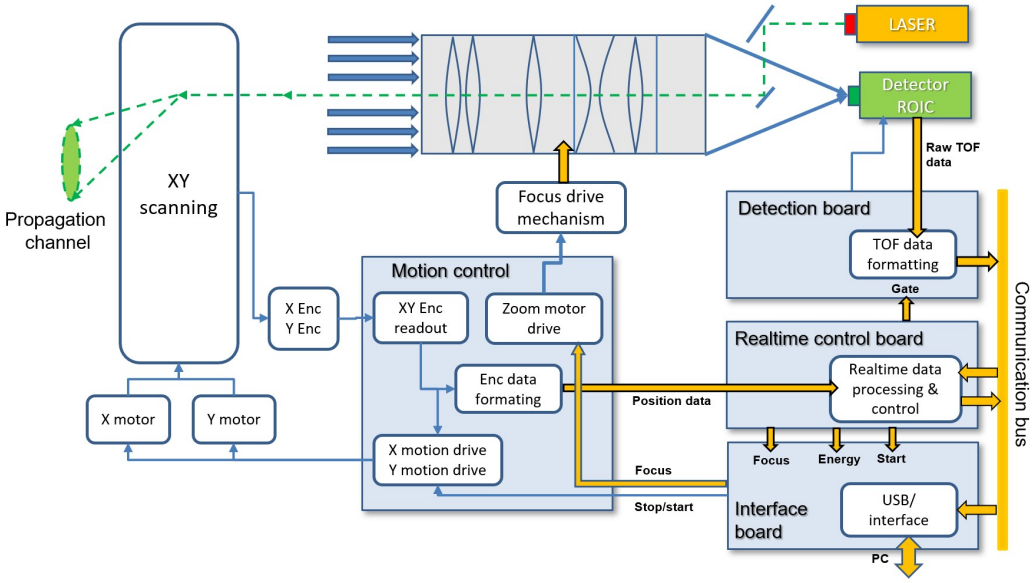


Figure 111: Overview of the LIDAR models and components involved (source UGE)

LIDAR models can simulate different kinds of LIDAR, aimed to compute the distance between the sensor and 3D targets of the synthetic environment. It does so by executing a picking in the 3D environment, detecting everything that has a 3D representation: Vehicles, pedestrians, objects, houses, etc.

e.g : In the Paris2Connect infrastructure, for example, LIDARs enable us to scan a very precise area and identify a whole range of road users (pedestrians, cyclists, scooters, cars, lorries, etc.).

This precision enables us to identify in real time a range of hazards inherent in the interaction between these different users.

This data can be used as a basis for LIDAR models in particular, and can be cross-referenced in more or less dynamic models within digital twins, for example. These digital twins will also enable us to simulate a number of different variables and environments in order to carry out stress tests on LIDARs in particular.

An interface to observe the graphical results is available, however normally raw data is available to be used in real time or post-processing.

Figure 112: Visual representation of the LIDAR Model in the 3D environment (source AVS)

Different parameters of the LIDAR can be set, such as the rotation Speed, the amount of points, maximum distance detected, rotation direction of the LIDAR and the pattern of the LIDAR. For the Physical LIDAR, more parameters can be set:

- Intensity of each beam in W.sr-1
- Shape of the beam weather a Perfect ray or Conic Ray
- Aperture of the conic ray
- Spot Shape

Level 2 LIDARs also include the effects of echos.

It is possible to add the effects of perturbations via post-processing of the raw data. This can be used to inject a more realistic data into the ADAS system to be able to fully test the performance of the detection algorithm.

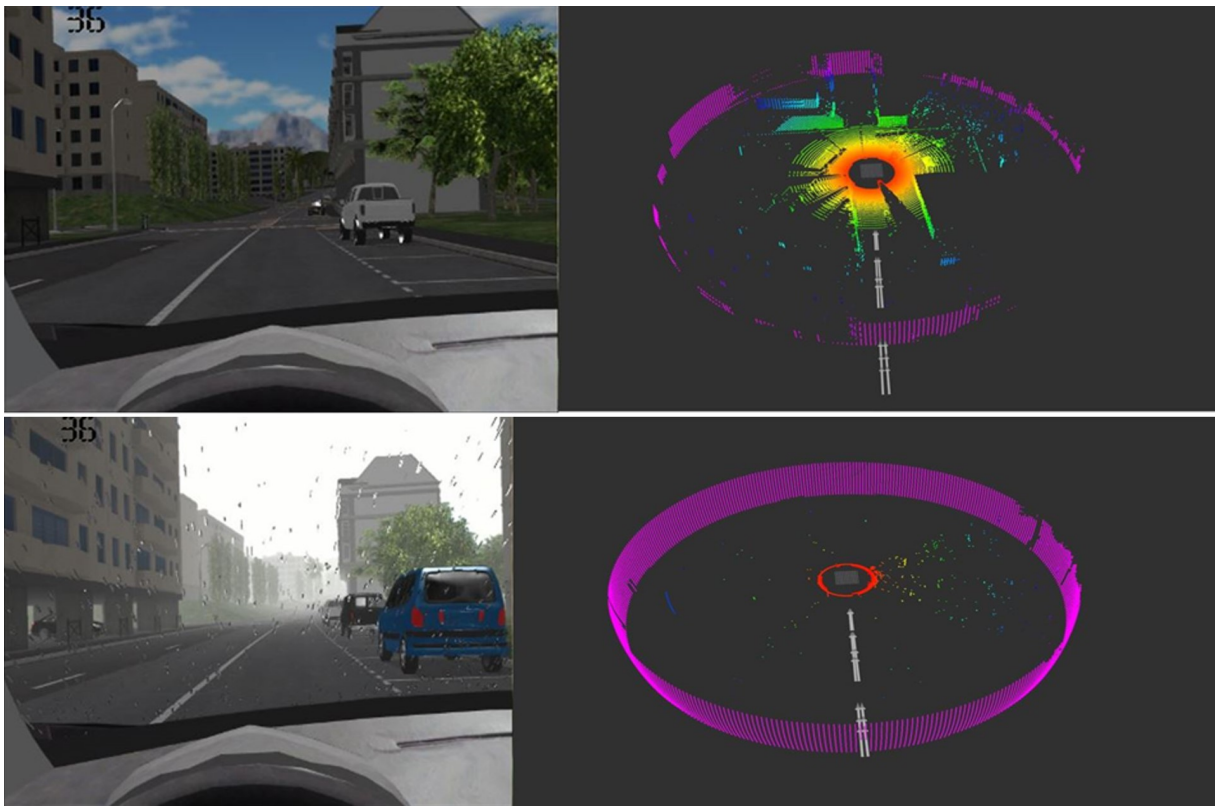


Figure 113: Implementation of a multiple mayer LiDAR model in Pro-SiVIC with degraded weather conditions (source: ESI group)

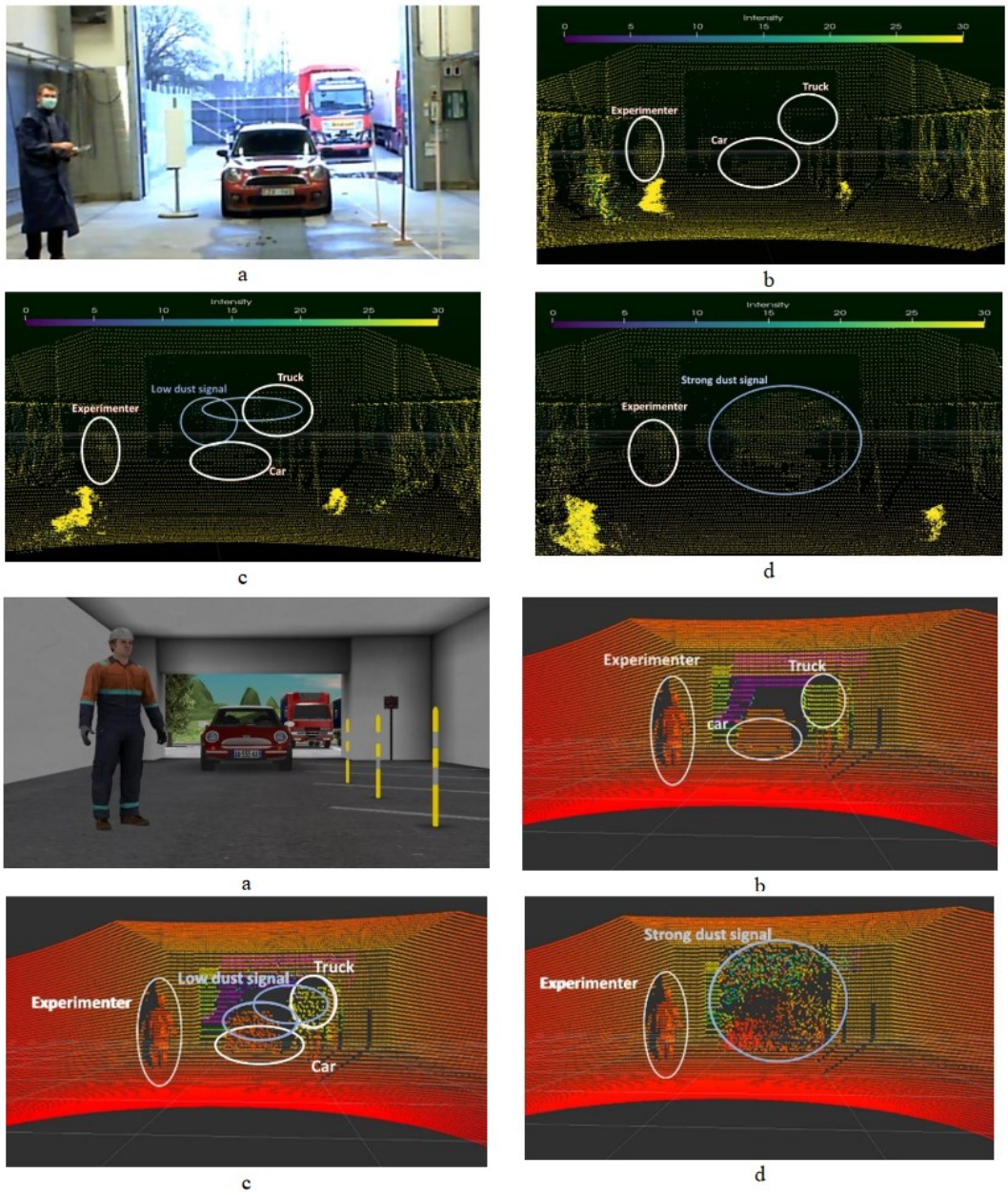


Figure 114: Implementation of a multiple layer LiDAR model in Pro-SiVIC with dust cloud in the atmosphere. Comparison with real LiDAR. On the top, experiment environment and Ouster OS-1 LiDAR data. a) environment, b) LiDAR reference image, c) low dust cloud density disturbance, d) strong dust density disturbance. On the bottom, Simulation of experiment environment and simulation of LiDAR data in PROSIVIC. a) environment, b) LiDAR reference image, c) low dust cloud density disturbance, d) strong dust density disturbance (source: ESI group)

3.5.4 Definition of the GPS models and components involved

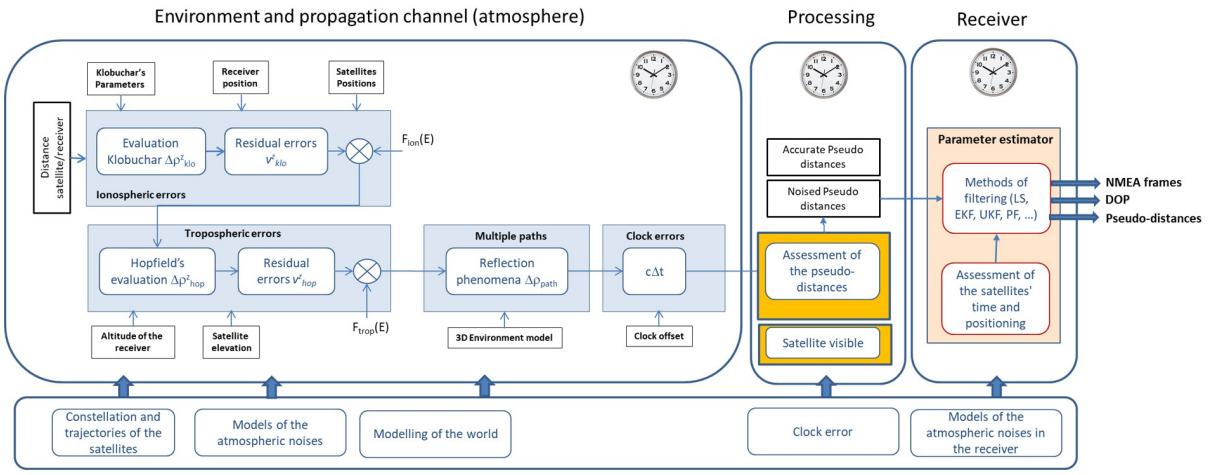


Figure 115: Overview of the GPS modules implemented in Pro-SiVIC (source UGE)

The GPS simulation module implemented in Pro-SiVIC and used in PRISSMA respect the requirements identified for GPS modeling and ensure the accuracy, reliability, and high level of performance of the GPS model. The model has been evaluated and validated in real condition in the Satory test track in same time than a set of real GPS receivers. The GPS model provides a high level of fidelity with the data and the behaviour of a real GPS. This verification and validation process involved the satellites constellation at a specific date, the use of atmospheric layer model with the associated models (ionosphere, troposphere), the low altitude propagation channel (occlusion, multiple reflection), the computation of the pseudo-distance, and finally the capability to generate NMEA frame defined in the standard NMEA 0183. For instance the figure 53 and and 54 show the result obtained in Pro-SiVIC for the GPS simulation. We can see that NMEA frames are generated and the coordinates (WG84) generated from the latitude/longitude/altitude extracted from this NMEA frame provides a realistic positioning (projection of the WG84 coordinated in Google Earth). In this modelling, the simulation of the satellite constellation consists of calculating the position of all satellites for a given date. Thanks to the downloadable satellite ephemeris files on the International GNSS Service website, it is possible to calculate the satellite positions either by performing Lagrangian interpolation or by using the orbital parameters derived from these ephemerides. Of course, in practice, the trajectories of satellites (and therefore their positions) are affected by errors. These errors are observed by ground control systems and used to correct the satellite's trajectory relative to its theoretical trajectory. In the simulator, the only trajectory errors will depend solely on the accuracy of the ephemerides. We would like to remind you that three types of ephemeris files exist and can be used depending on the desired precision:

- IGS: "precise" satellite ephemerides (within 2 weeks)
- IGR: "rapid" satellite ephemerides (within 72 hours)
- IGU: "ultra-rapid" satellite ephemerides (within 24 hours

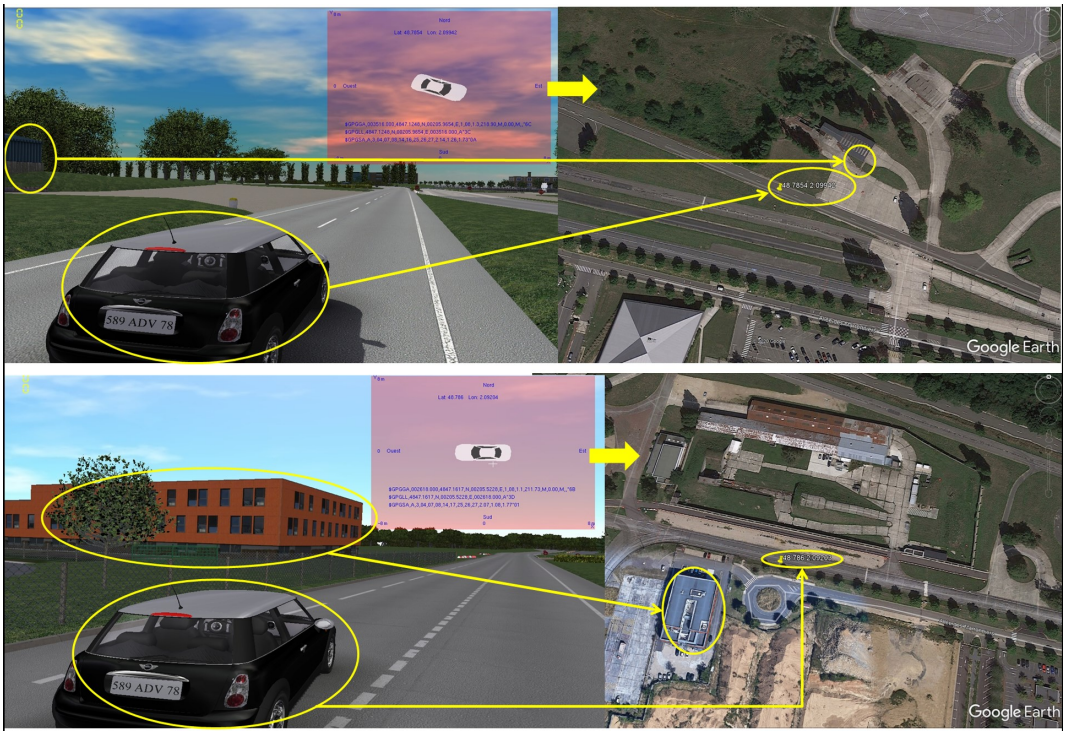


Figure 116: Result of NMEA frame generation and projection in Google Earth

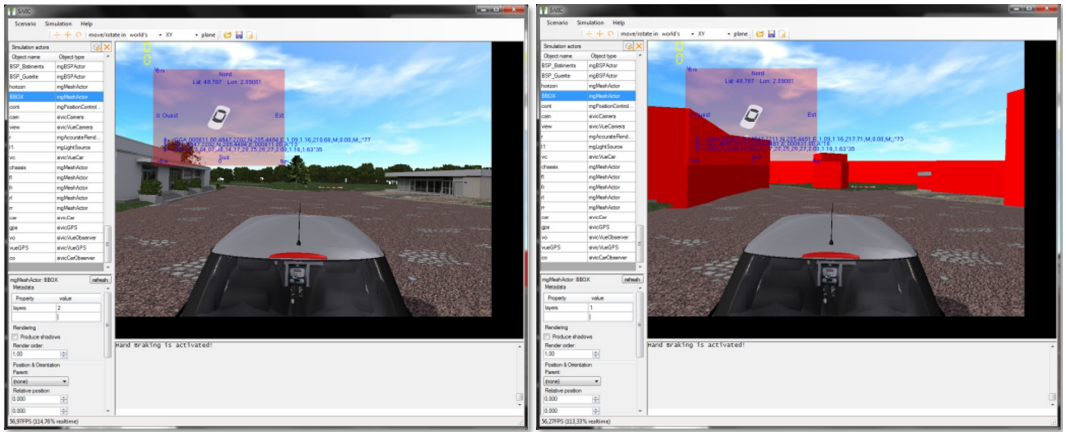


Figure 117: Modelling of the environment in order to take into account ground disturbances (multiple reflection and occlusion)

The complexity of a GPS model can vary greatly. In all cases, it is necessary to validate all the components of the chosen model. For instance, either the model of the GPS is very simple and apply a noise on a positioning generated by the car position in the simulation environment, or the model could be more realistic, complex, and respect a high level of fidelity with the real system. In this second condition, it will be necessary to take into account the satellite constellation, the different signal disturbers (ionospheric and tropospheric errors, reflection phenomena), the time and clock errors (see figure 115), and the different function of the receiver (see figure 115). All these functions represent a complex process for the modelling of a high level GPS simulation. The integration of the GPS model relies mainly on two components and the use of a GPS simulation library designed by LASMEA as part of the FUI eMOTIVE project. The first

component is dedicated to the actual simulation of the GPS, while the second component handles the display of the results produced by the GPS simulation. The following figure presents the functionality of displaying GPS data as a compass, with the needle representing the vehicle equipped with the GPS. This display includes cardinal points, a scale (the ”range”), the calculated position (the ”white cross”), as well as 3 NMEA frames. The proper functioning of the GPS receiver in SiVIC requires configuring the component with four priority commands:

- Associate a Lambert projection type.
- Geolocate the graphical object representing the track within this projection framework.
- Define a date and time.
- Attach the sensor to a vehicle.

Figure 118: Display mode for the GPS simulation in the pro-SiVIC platform (Source: UGE)

In addition to the model proposed in Pro-SiVIC and used in PRISSMA, some other complex and realistic models have been recently proposed. For instance, in [17], a GNSS model is proposed for urban area with the modelling of the main issues encountered by real GNSS. From the figure 119 it is possible to have a list of effects to check in a high fidelity GNSS simulation in order to validate it:

- Effect of diffracted and reflected signal impacting the pseudo distance assessment
- Open-Sky C/N_0
- Multipath effect and noise
- Doppler shift effect
- Satellite clock bias
- Receiver clock bias
- Ionospheric delay
- Tropospheric delay
- Elevation-based tropospheric delay variance

All these aspects and parameters will have a significant impact on the pseudo range assessment. The simulation like with the real GNSS system need to provide a physical twin of this effects.

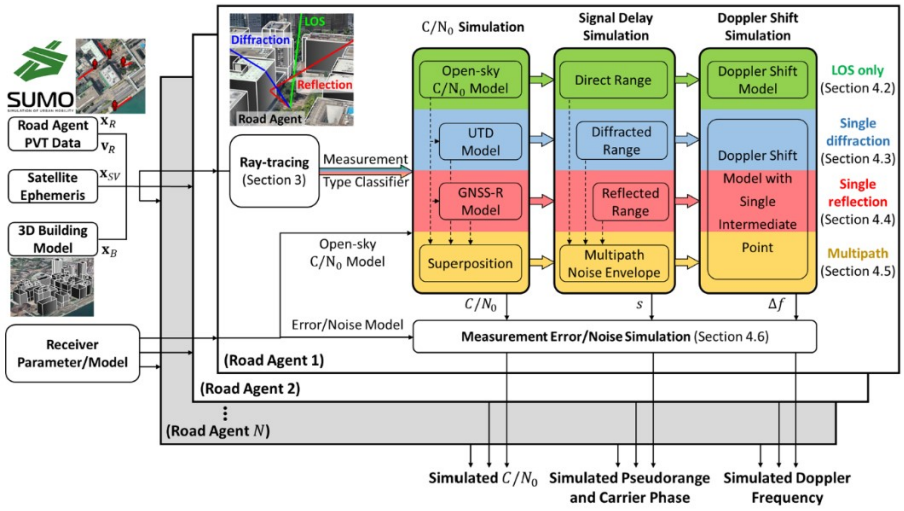


Figure 119: Diagram of realistic GNSS simulation in urban multi-agent context (GNSS RUMS) [17]

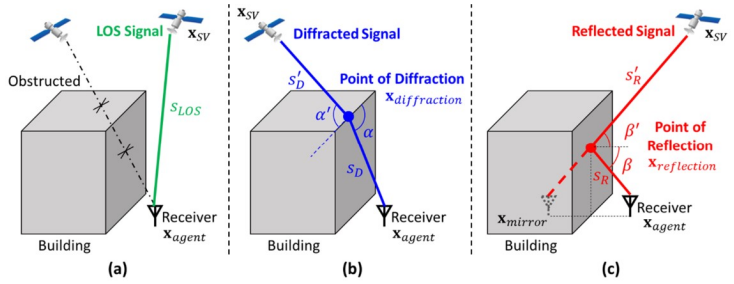


Figure 120: Main effect encountered in urban areas including (a) the LOS signal; (b) the diffracted signal; (c) the reflected signal [17]

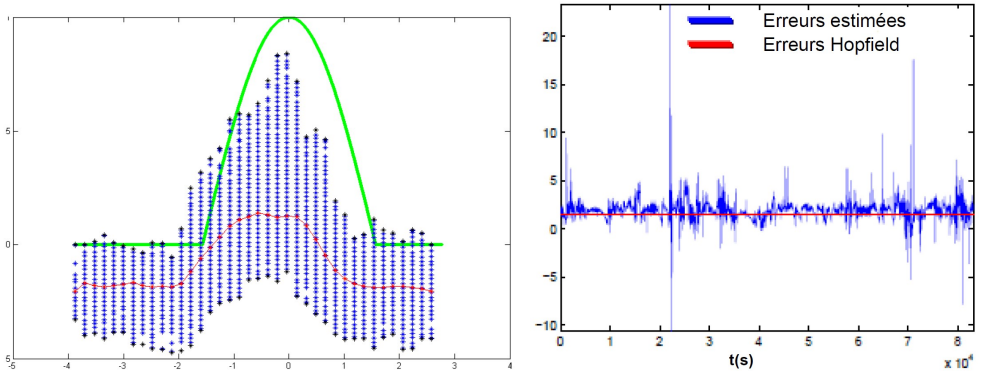


Figure 121: Left: Ionospheric normalized errors over 24 hours. In green, errors assessed by the Klobuchar model disseminated by the GPS system. In blue, EGNOS corrections obtained over several months. In red, a random and continuous representation of EGNOS corrections. Righ: Tropospheric errors over 24 hours. In blue, the measurements obtained with a UBLOX receiver. In red, the evaluation of these errors by the Hopfield model.

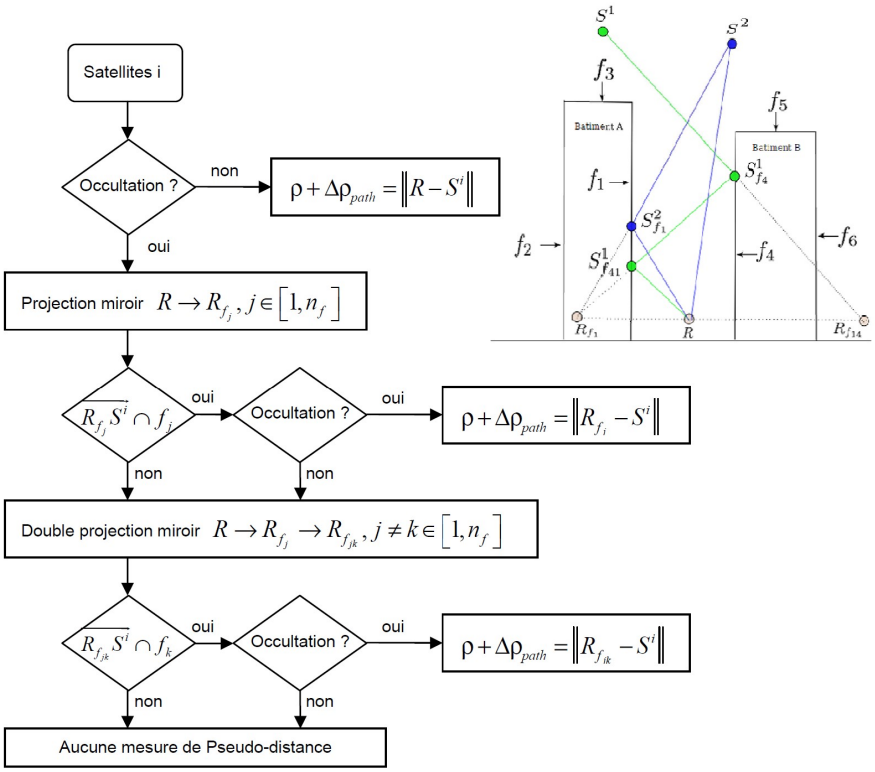


Figure 122: Calculations of possible pseudo-distances based on multi-paths.

As already mentioned, the complexity of a GPS model can vary greatly. The more complex and step-numbered the GPS simulation is, the more difficult it will be to validate from end-to-end.

In fact, we have described in the previous pages (and in general terms) a simulation model (that of Pro-Sivic) which starts from the satellites, their orbital and clock parameters, the atmospheric crossing, and the multipath. From there, Pro-Sivic runs, step by step, a GPS solver using perturbed pseudo-ranges to generate solutions.

Another approach has been adopted in the COST action SaPPART [75] and [76] in its appendix. It is more directly that of simulating solutions without going through ranging, propagation, a GPS solver, etc. And simulating solutions is ultimately a learning process and can be regarded as such. It is a data science approach, rather than a physical model (such as that of Pro-Sivic).

Furthermore, the method which makes it possible to validate that a simulation is faithful to the truth is not simple. Examples and illustrations have been given previously but the question remains whether or not it is enough? As far as we know, this question is still a field of research.

About the comparison metrics between simulation and truth, SaPPART proposed examining the cumulative distribution functions (CDFs) AND the auto-correlations (ACs) of both. Actually, the temporal shape of the GPS positioning error matters for navigation and driving process later.

The figures 123 and 124 examine the resemblance between models and real position errors, in dynamic conditions in Paris centre, with three different models to benchmark. Both CDF and AC are examined. Note that along-track and cross-track positioning errors are simulated, because it is well known, in urban environments, that streets make canyons and cause plane error distribution being non-isotropic.

To conclude, research investigations are still on-going relatively to the question of simulating GPS and GNSS in general. With one pending question remaining opened about how much data one should collect, in real world, to be confident with the simulated data and not miss possible behaviour of a GPS receiver which may rarely happen.

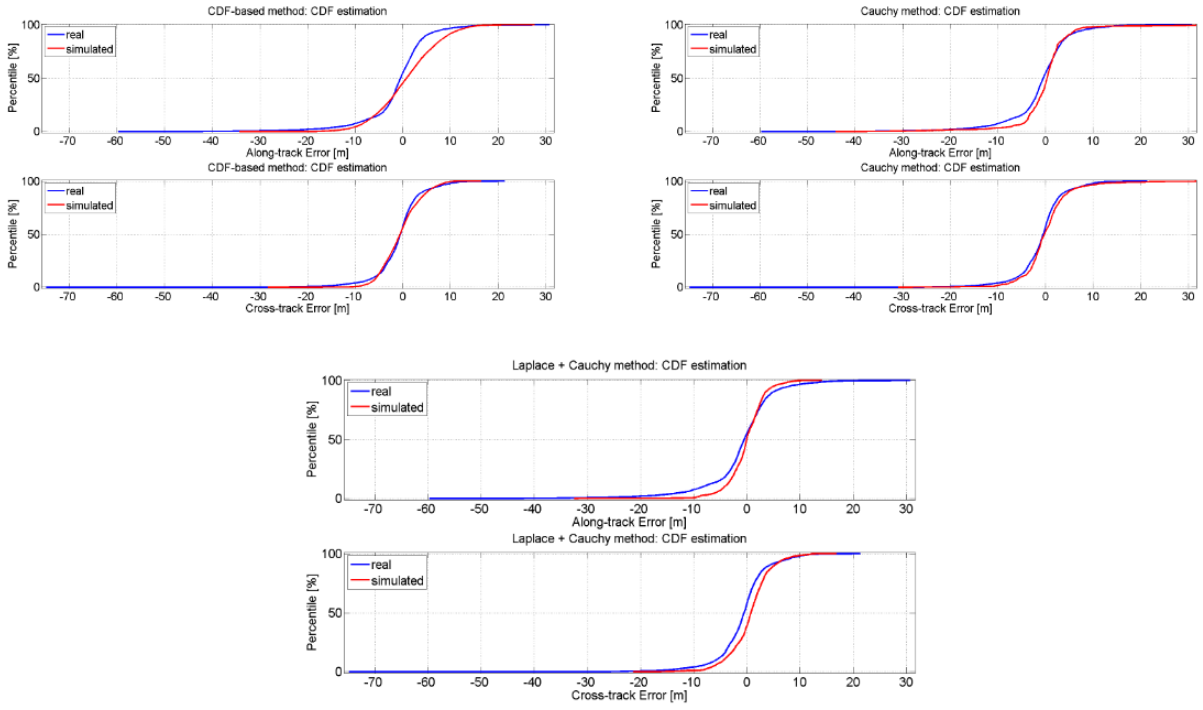


Figure 123: Actual and simulated along-track and cross-track CDFs for CDF-based, Cauchy and Laplace-Cauchy models for the Paris data set

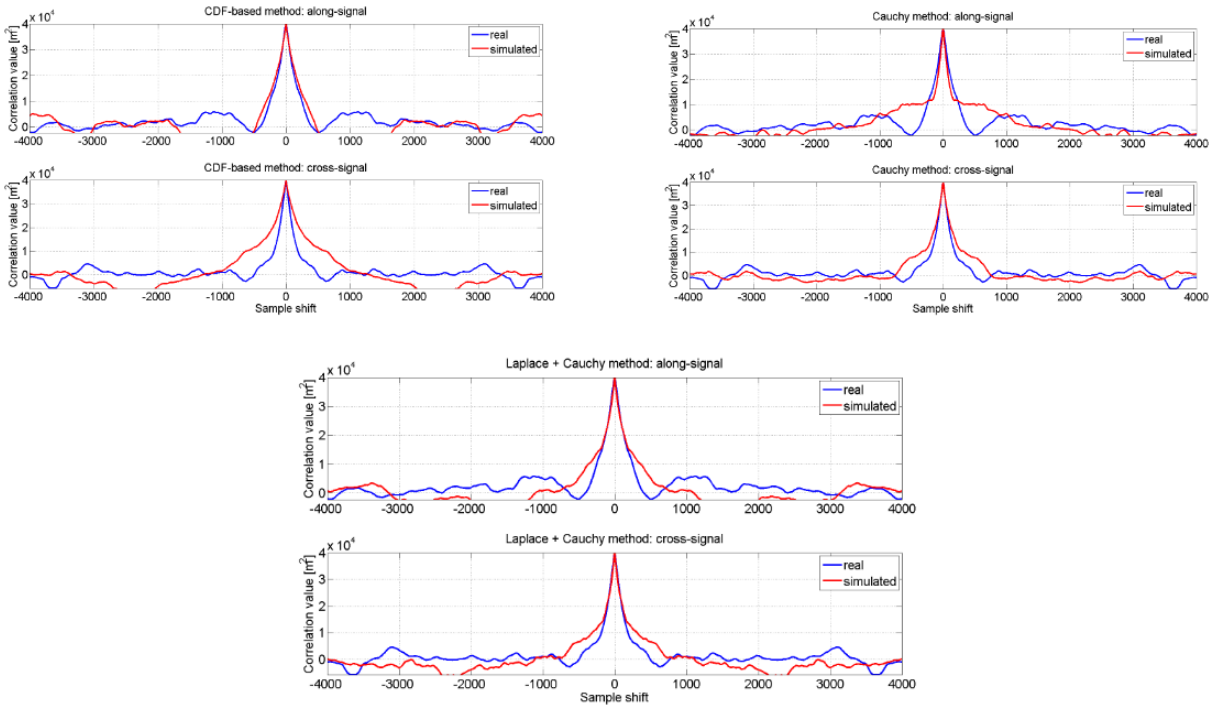


Figure 124: Actual and simulated along-track and cross-track auto-correlation functions for CDF-based, Cauchy and Laplace-Cauchy models for the Paris data set

3.5.5 Definition of the proprioceptive sensors: INS, Odometer

Most proprioceptive and reference sensors need to be attached to a target object, so they can obtain information needed for their processing. This object is set with the Object property, but only compatible objects can be set for each type of sensor. Data collected by the sensor can be exported as a list of decimal values, in a specific order described in each sensor's section. In some cases, these data are also shown in the object configuration panel as read-only properties that update regularly. Position and rotation vectors exported by these sensors are global. All vectors have (x; y; z) components. Rotation is performed in the direct sense around the normalized rotation vector with an angle defined by the norm of the rotation vector (in rad).

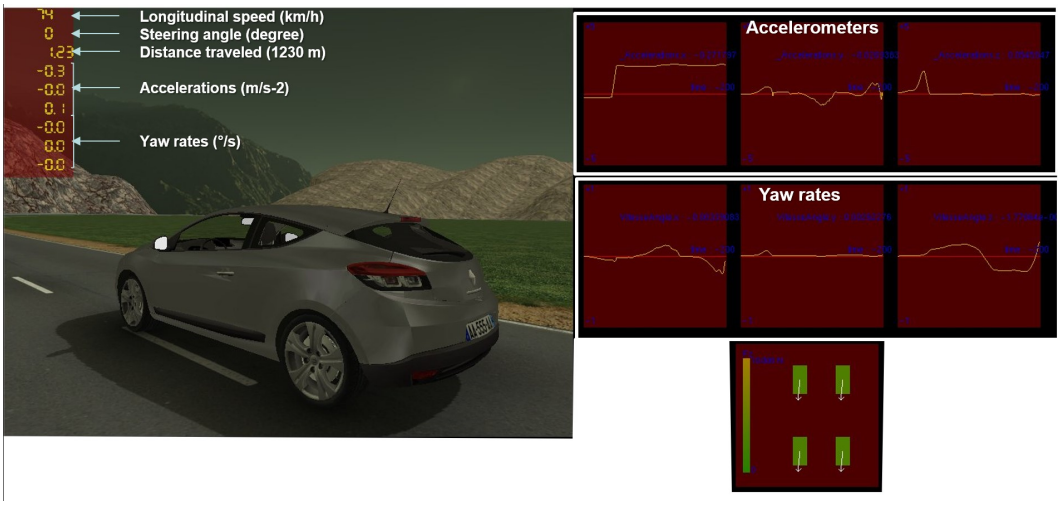


Figure 125: Overview of the proprioceptive sensors embedded in the SiVIC's ego-vehicle (source UGE)

3.5.5.1 Odometer

This sensor provide the distance covered by a car is usually obtained thanks to an odometer. Pro-SiVIC offers a simple yet realistic odometer model sivicOdometer by using wheel rotation data rather than real travelled distance that is usually unknown. This model is indeed based on a coder mounted upon one of the car wheels. By attaching an odometer to your car, it will use wheel data as well as its rotation velocity to compute the number of pulses between two instants. Odometer needs to have an operating period, the name of the vehicle and the wheel. In order to fix the resolution and the accuracy of the sensor, it needs too the number of TIC by wheel round. This sensor can take into account the sensor failure and trouble like the wheel sliding, the wheel blocking, and the wheel speed differential (if several sensors are put on the different wheels). Computed values are exported in the following order:

- 0: distance since simulation start (in m)
- 1: current speed (in m/s)
- 2: number of tics since simulation start

3.5.5.2 Inertial Navigation System

Pro-SiVIC provides an inertial navigation system simple model through the plugin named sivicINS, which can be attached to any object. This plugin provides acceleration as well as rotation speed. Initialisation process is rather simple, and does not require specific commands except the attachment to a mobile object. For this sensor, we compute the angular speed, the speed composition, the absolute speed, the acceleration composition, and finally the absolute acceleration. This means that this INS sensor can be attached to every object in the environment (vehicle, pedestrian, motorcycle, bicycle, or a object (ballon for instance). The outputs are the angles (x,y,z), the yaw rates, and the accelerations Another sensor is available, simulating an Inertial Measurement Unit. It is named sivicIMU and also produces acceleration and rotation speeds, but can only be attached to cars. The IMU sensor will use the dynamic state of the vehicle in the state vector computed by the solver and the differential equations of the vehicle dynamics. Computed values are exported in the following order:

- (0,1,2): rotation vector
- (3,4,5): angular speed (in rad/s)
- (6,7,8): acceleration (in m/s2)

3.5.5.3 Viewer of the odometric data

In pro-SiVIC, several ways are offered to display the real-time result of the odometric sensors. The fist way is a DataViewer (top left of the figure 125 with a set of digit (it is possible to tune the size, the color, the back ground, the accuracy (number of digits). The over way consists to use ”oscillocopes” (right part of the figure 125. It is also possible to display in a bird view the force vectors applied to each wheels. The last way consists to use the sivicCarObserver plug-in providing a reference sensor with the main important parameters and variable of the vehicle (40 parameters).

3.6 Mobile dynamics

The majority of the dynamic models need to use of ODE solver. Several solver are available following the requirement (computation period, complexity of the model, ...). Often, the used solver is the Runge–Kutta with different order (2,3,4,5) with sometime a variable step.

There also exist the Callas (coupled with ground adhesion limit) model that is a dynamic model for trucks, bus, car, motor-sport, tractor and military vehicles.

3.6.1 Definition of the Vehicles models and components involved

3.6.1.1 Vehicle dynamics modelling in CALLAS (AVS)

The Callas model can be divided in three main areas:

- Interface
- Calculation
- Data processing and the results

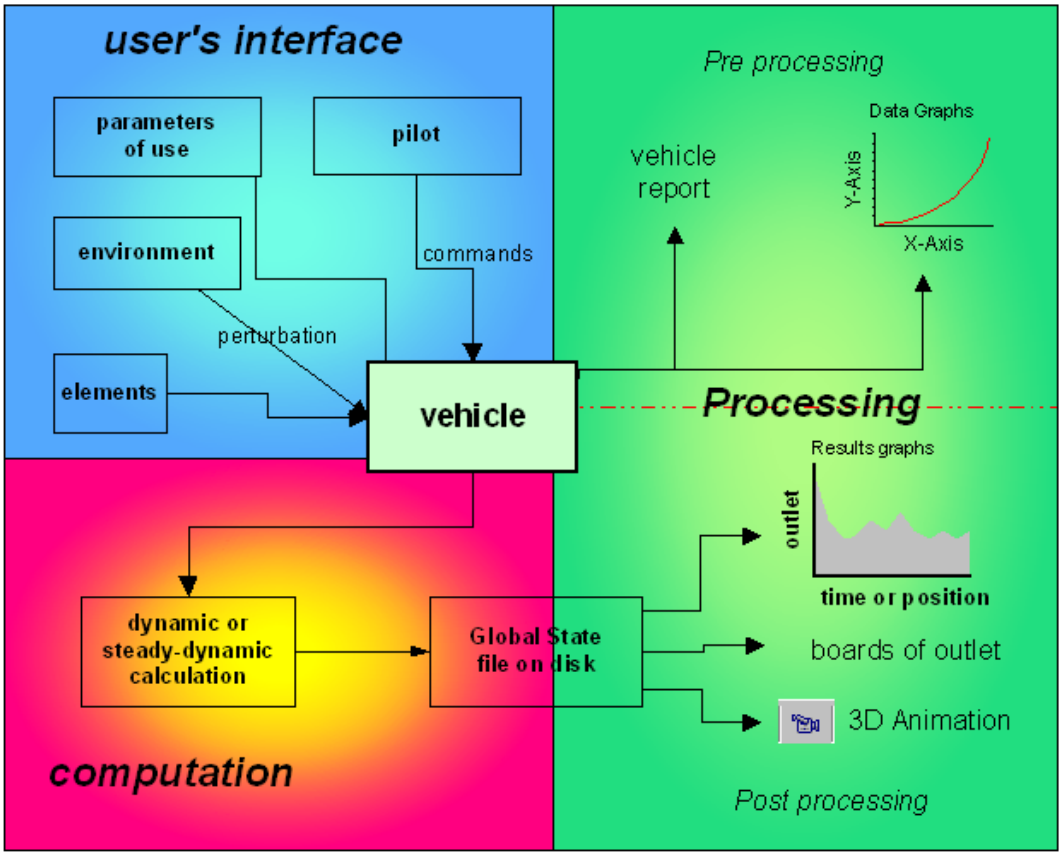


Figure 126: The three main areas represented in the graph (source AVS)

5 Principles govern the reference frames and signs used:

- Input data and vehicle analysis
  - Whatever is known to be linked to the chassis, is defined in the “architect” reference frame
  - All sign are kept positive for all data where the sign or direction is obvious (Wheel loads, drag, etc..)
  - Data relative to a static setting is expressed in the set-up sheet conventions (camber, etc.)
  - Whatever is linked to the full car’s assembly is defined in the easy check condition.
- Results variables: Everything is defined in the framework of the car’s dynamic,no reference being made to the inside or the outside.

Another model is a simple car model that is comprised of only:

- bi-axle (2 wheels by axle)
- position of the vehicle is computed with 1 road picking
- terrain following
- engine

- transmission
- braking
- steering

Simple models have vehicle tires, suspension and steering models that are not sufficiently detailed to have an appropriate response to subtle things such as rumble strips. The limitations of the simple model is that there is no component model (no physical model of engine), no roll movement, vehicles starts with the engine on, only one point of road picking and infinite grip (lateral speed is always +/- 0)

Vehicle models also consist of a 3D model, where certain parts of the vehicle have to be well named in order for the visual engine to recognise it and assigned the correct movement/material to it. For example the four wheels and axles have to be correctly names to ensure that the wheels turn, and mirrors also ensure that a reflection can be generated on the material. left and right dashboard should also be integrated via switches in order to be able to alternate between the two in the different traffic situations.

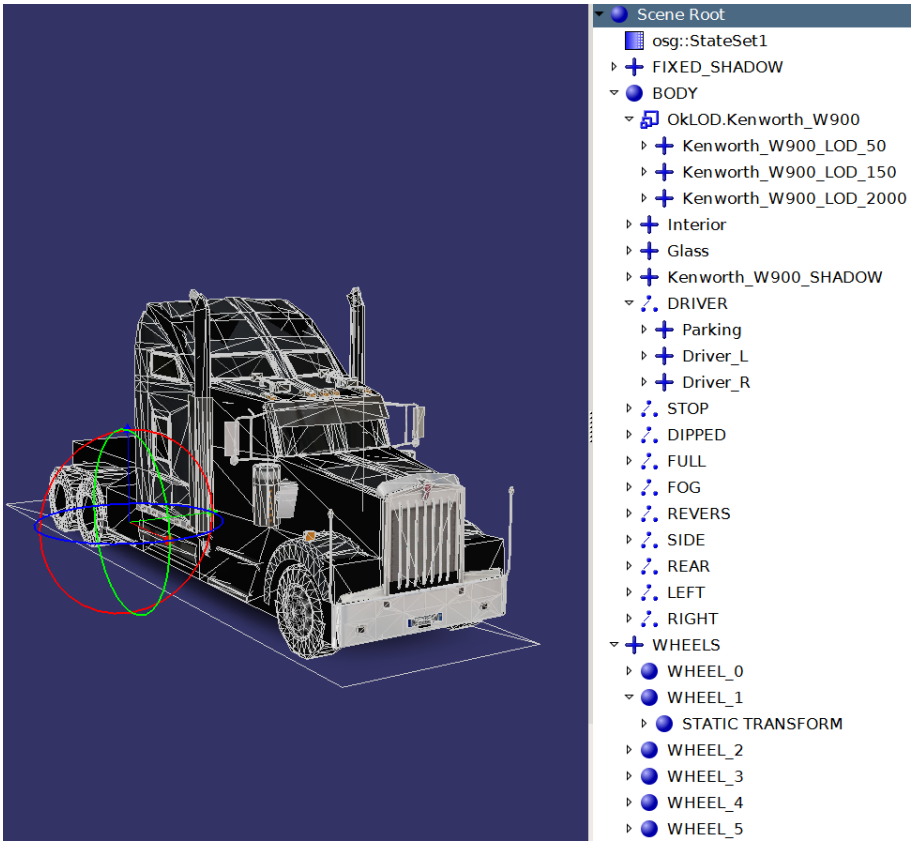


Figure 127: Hierarchy of the creation of a 3D vehicle model (source AVS)

3.6.1.2 Vehicle dynamics modelling in Pro-SiVIC (UGE and ESI Group)

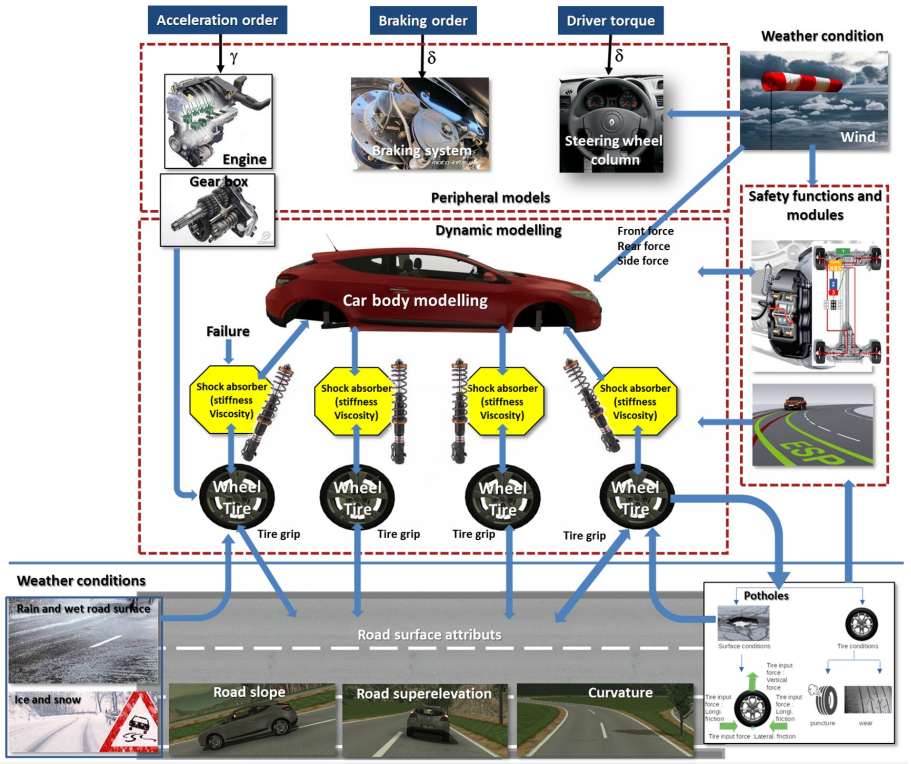


Figure 128: Overview of the vehicle modules with the interaction with environment and road surfaces (source UGE)

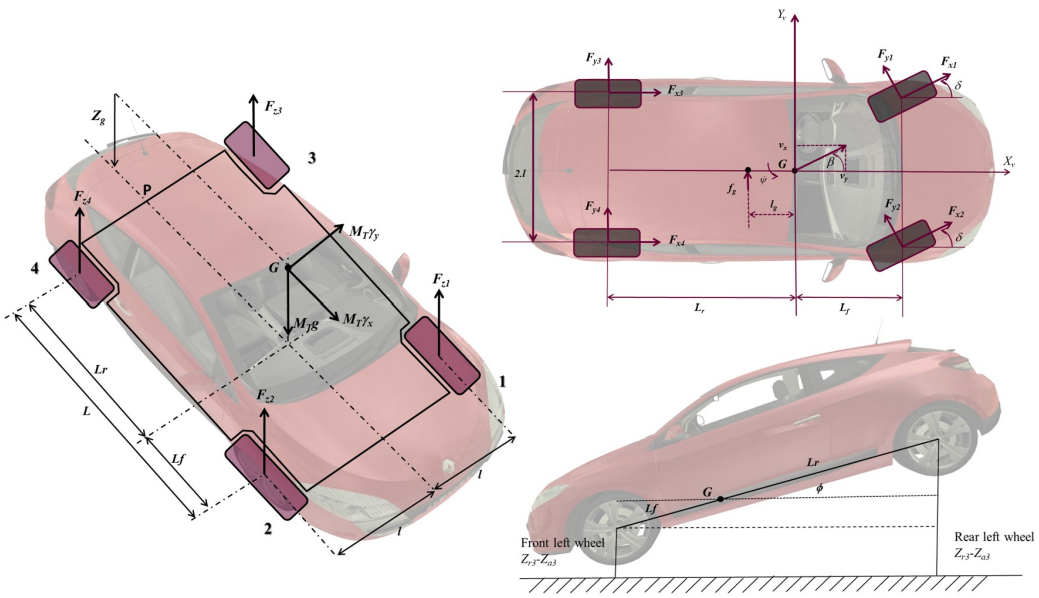


Figure 129: Overview of the car body parameters took into account in Pro-SiVIC (Source: UGE)

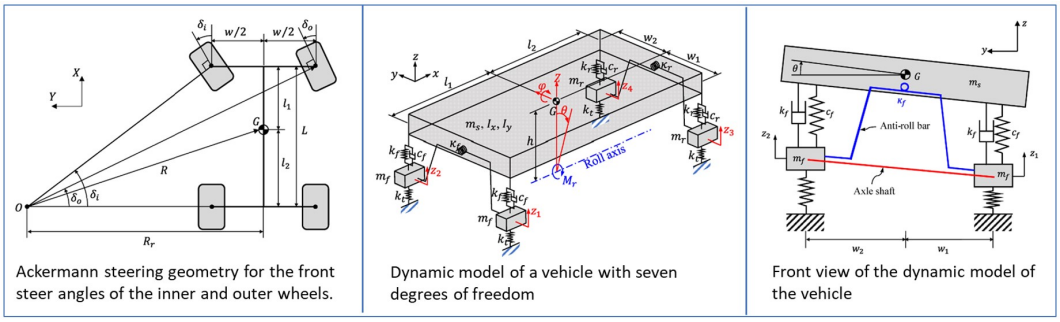


Figure 130: Overview of the vehicle dynamic model with ackermann streering geometry, the seven degrees of freedom, and the lateral dynamic ([18])

In order to reproduce sufficient vehicle behaviour to reproduce the effects of its dynamics on the on-board sensors (proprioceptive or exteroceptive), the complex model (figure 128) developed by S. Glaser in his PhD. thesis has been implemented in Pro-SiVIC. This model is available via the sivicCar plugin. This model take into account to model the chassis, the shock absorbers as well as the tires (deformation and tire/road contact) and the wheels. In order to take into account the contact between the tires and the road, we use the ray tracing engine and we apply a set of ray tracing to each wheel. This makes it possible to precisely determine the distance between each wheel and the ground. This mechanism makes it possible to guarantee physically realistic operation (driving on the sidewalk, taking a speed bump, etc.). During the last decade, this model has greatly improved with the addition of a thermal engine model (with engine mapping) and a gearbox (manual and automatic). In addition we added a steering column with consideration of control, driver, and self-alignment torques. We have also integrated the handbrake and taking into account the aerodynamic coefficient to apply the appropriate forces in the event of a gust of wind. Finally, a fuel tank was added with a fuel consumption sensor. The wheel vibration due to the granularity of the road surface and a failure on the tire and shock absorber is proposed. The dynamic model runs between 500 and 1000 Hz and uses a Runge–Kutta solver. In order to control the vehicle model, a set of mode are proposed:

- Mode 0: It is the ”user” mode. A real human can drive the car with the main peripherals (keyboard, mouse, joysticj, logitech steering wheel and pedals)
- Mode 1: It is the ”trajectory” following. This mode needs a HD Maps (OpenDrive, trk file) and a trajectory file
- Mode 2: It is the ”controller” mode. This mode also needs a HD Maps (OpenDrive, trk file) and a trajectory file but the model apply a controler in order to converge toward the reference trajectory with the respect of the physical constraints of the vehicle. This means in an strong curvature and with a high speed, the vehicle will have a realistic behavioir and will make a lane departure.
- Mode 3: It is the mode ”RTMaps”. The vehicle will be controlled by orders coming from RTMaps
- Mode 4: it is the mode ”trajectory generation and replay”. This mode record the current driving as a trajectory and this file could be replay in future simulation step.
- Mode 5: it is the mode ”Matlab”.

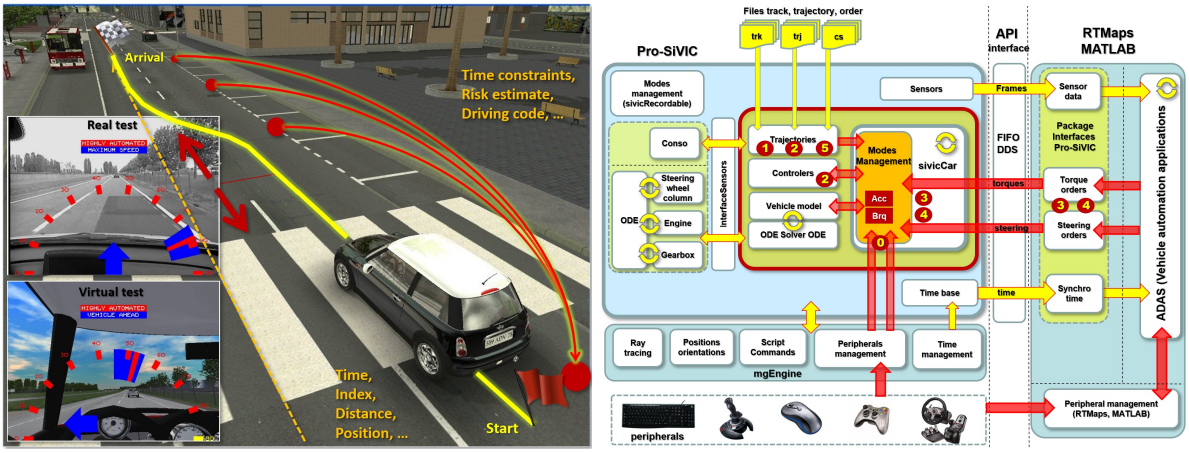


Figure 131: Control modes for sivicCar plug-in and trajectory following mode in Pro-SiVIC with the main parameters (Source: UGE)

3.6.1.3 Other existing complex vehicle modelling

Existing simulation platforms involving high-level vehicle dynamic modelling are AMESIM from Siemens and CarMaker from IPG. In these 2 types of models, the different functions need to be validated with real bench.

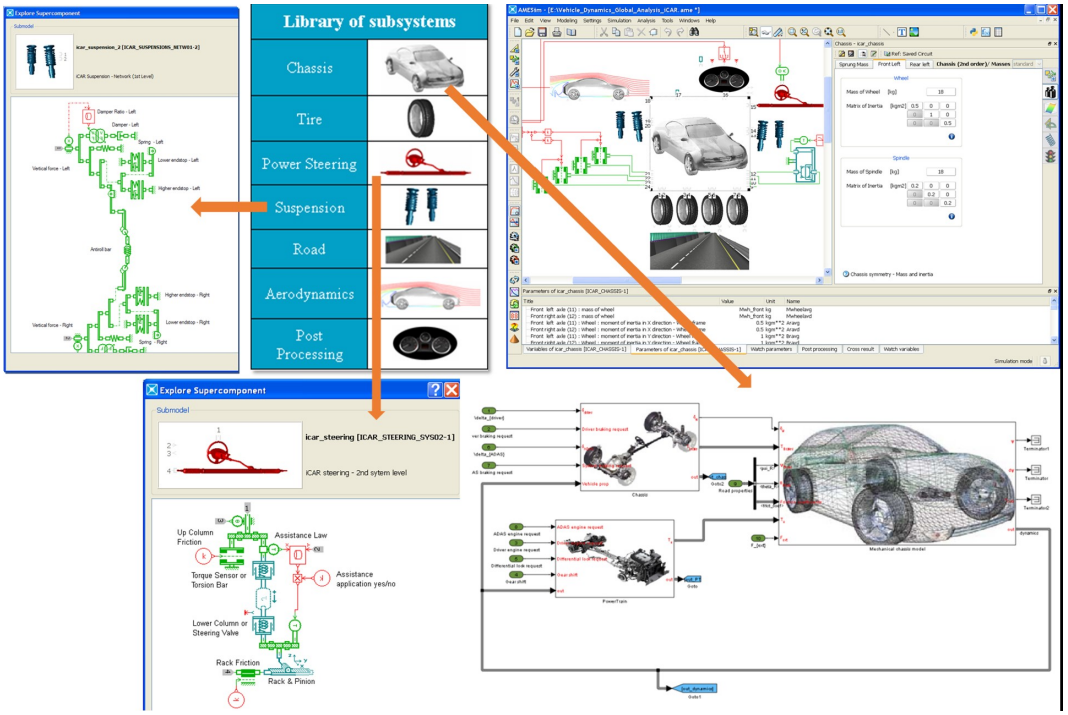


Figure 132: AMESim (SIEMENS) full vehicle model running on HiL platform [19]

In 2023, Graz University of Technology and IPG have proposed a framework for the Validation of Automated Driving Function Based on the Apollo Platform. This platform vehicle-in-the-Loop Testbed involving IPG product for dynamic modelling of vehicle.

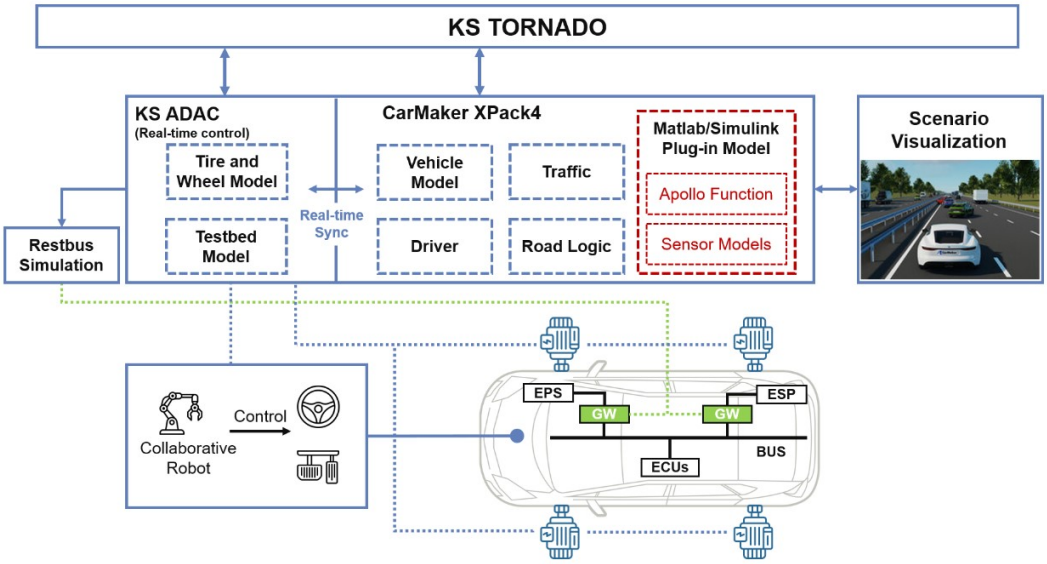


Figure 133: vehicle-in-the-Loop Testbed with Apollo platform and IPG CarMaker model [20]

3.6.2 Definition of the trucks models and components involved

About the truck and bus models, several categories of vehicle exist. For the bus, mini-bus, and light truck, the car modelling presented in the previous section is good enough. Nevertheless in the framework of articulate truck with a 4x2 tractor and a standard semitrailer with 3 axes, or a tractor (the cabin) and several trailers (platoon with physical link), it is mandatory to use a complementary physical engine allowing to manege the link and forces between the cabin and the trailers.

Figure 134: Overview of bus, light truck, and shuttle categories

For the 4x2 tractor and a standard semitrailer model, originally developed by Sayers and Riley (1996) and commonly referred as the a ”full model”, it comprises eight rigid bodies representing the sprung masses of the tractor, trailer, and six axles. While the first axle (Axle 1) features single tires on each side, all other axles (Axle 2, 3, 4, 5, and 6) have dual tires per

side, resulting in a total of 22 tires for the complete tractor semi-trailer configuration. These rigid bodies are interconnected by joints and forces/moments to accurately capture the vehicle's behaviour.

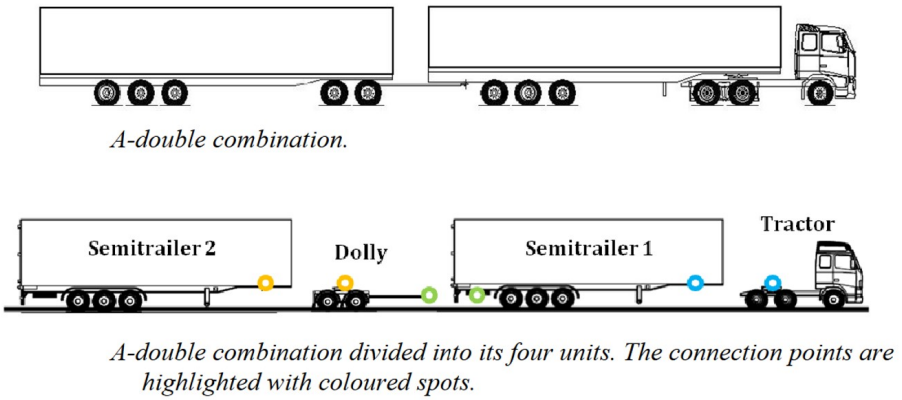


Figure 135: Overview of trucks with a tractor, multiple trailers and dolly ([21])

The tractor has six degrees of freedom (DOF), allowing movement in all three translational and rotational axes, whereas the trailer has three kinematic degrees of freedom due to the hitch constraint, allowing three rotations but no translation with respect to the tractor. The axles can translate along the body-fixed z-axis (jounce) and rotate around the x-axis (roll) with respect to the tractor or trailer, while the wheels can rotate around the y-axis (pitch) of the axles. Consequently, the model encompasses a total of 33 rigid body degrees of freedom as follows:

- Tractor: 6 DOF (3 translational and 3 rotational)
- Trailer: 3 DOF (3 rotational)
- 6 Axles: 2 DOF each (roll and jounce)
- 12 Wheels: 1 DOF each (spin)

In addition to the kinematic constraints, various forces and moments act between the rigid bodies at specific points. Suspension forces (F S) act between the axle and the sprung mass, while tire forces (F T) act between the tire and the inertial frame to support the weight and facilitate vehicle movement. The model comprises 122 forces and moments, including:

- 12 suspension spring forces and dampers between axle and tractor or trailer
- 66 tire forces (22 vertical, 22 longitudinal, and 22 lateral)
- Aerodynamic drag force between tractor and inertial frame
- 22 tire aligning moments in yaw DOF
- 6 axle moments in roll DOF
- 3 hitch moments (1 yaw, 1 pitch, and 1 roll) between tractor and trailer

The full model is constructed by synthesising these rigid bodies and forces/moments, resulting in 91 states and approximately 120 parameters.

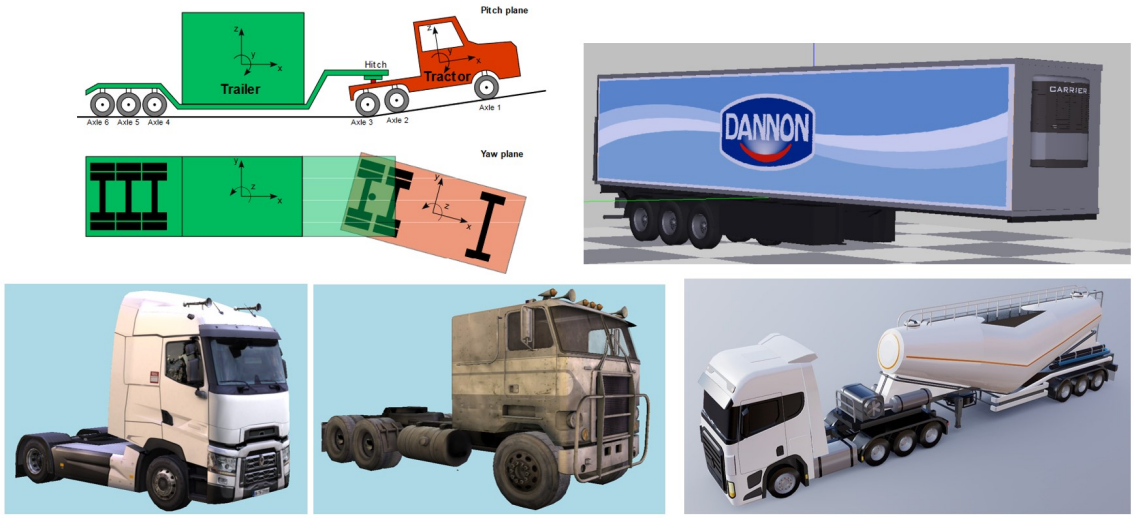


Figure 136: Overview of tractor and semitrailer categories

In [77], a new generation of truck modelling is proposed with rear steering wheels (see figure 137).

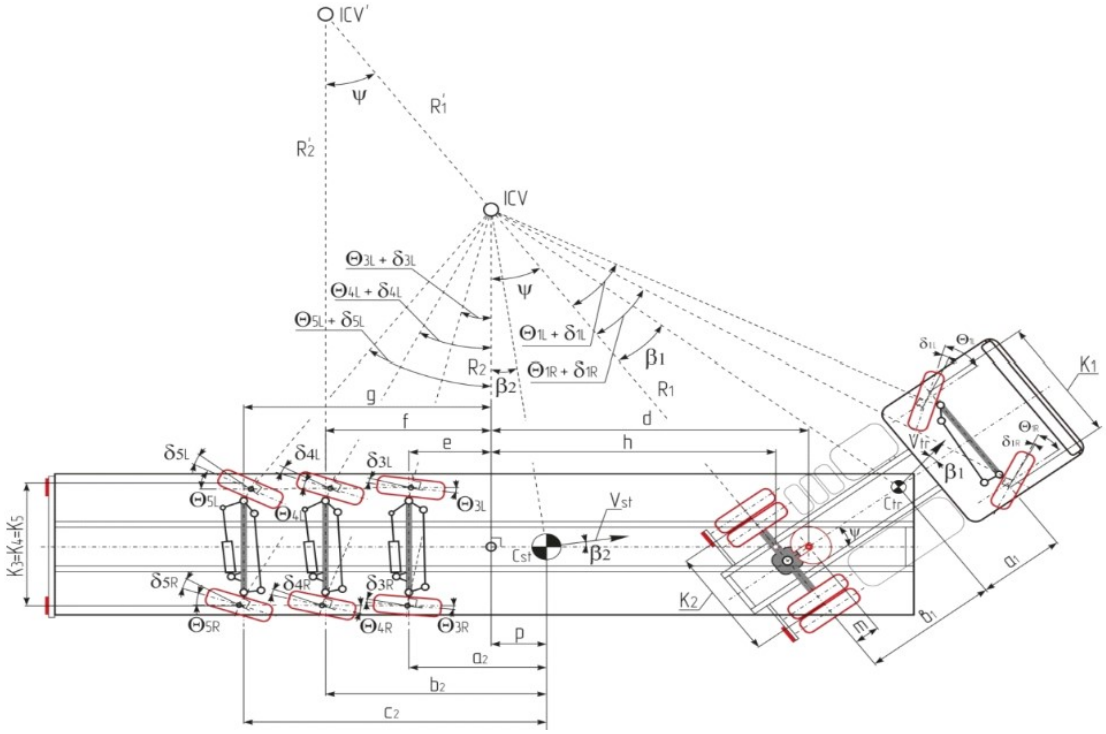


Figure 137: Scheme for determination of the theoretically required turning angles of the semitrailer wheels depending on the truck articulation angle

3.6.3 Definition of the Motorcycle, bicycle, scooter models and components involved

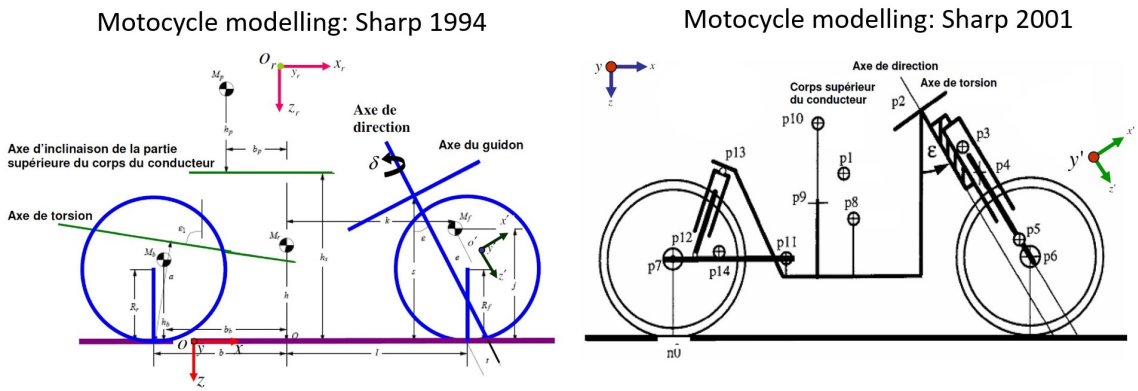


Figure 138: Overview of the Sharp motorcycle models (source UGE)

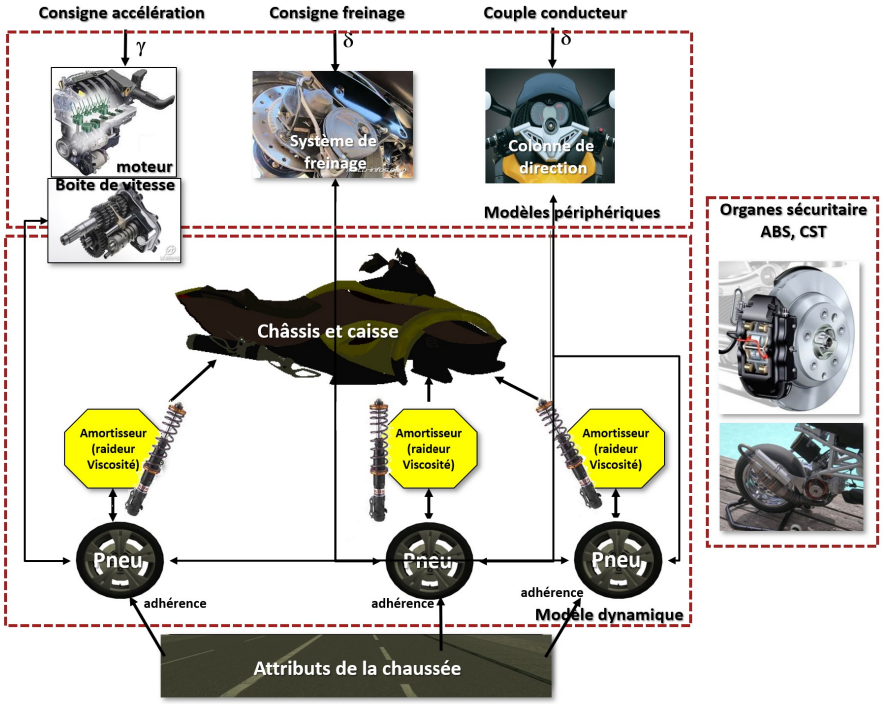


Figure 139: Overview of the motorcycle modules: Spyder model with 3 wheels and frontal engine (source UGE)

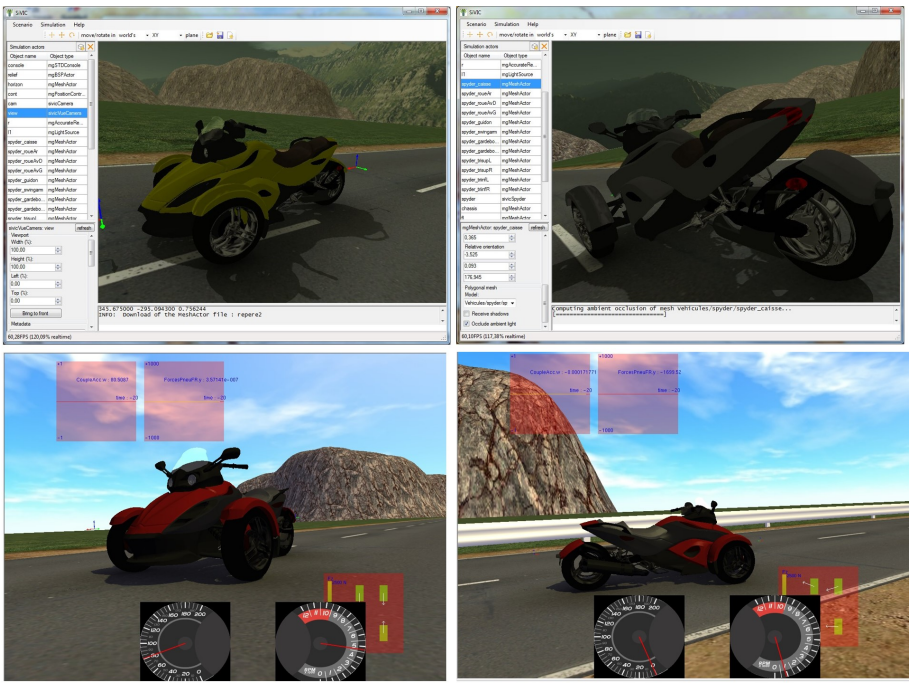


Figure 140: Overview of the Spyder model developed with CTA (Univ Sherbrooke and Bombardier) in Pro-SiVIC (source UGE)

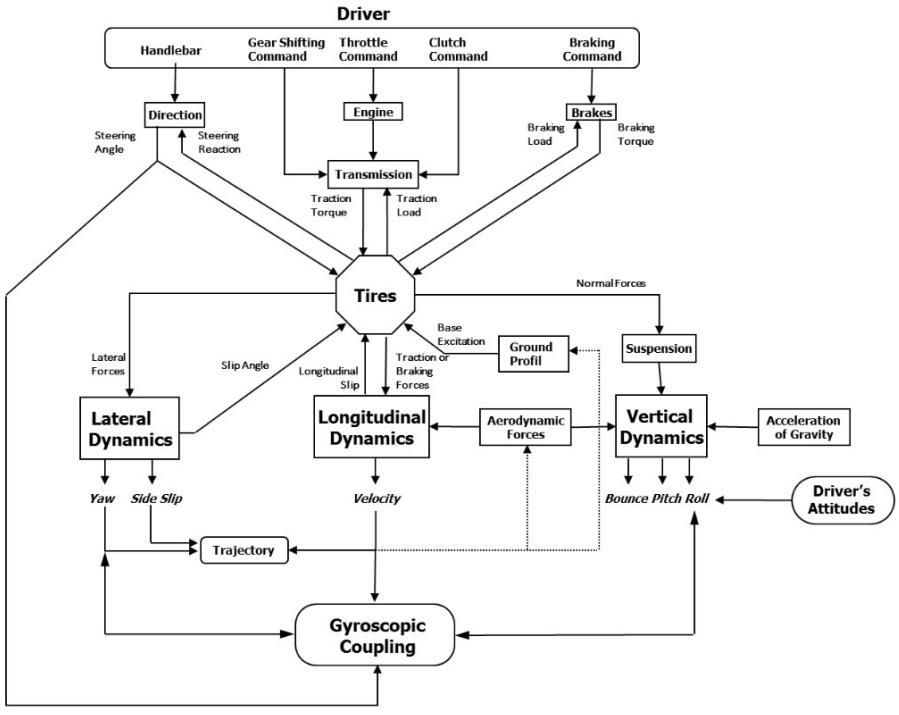
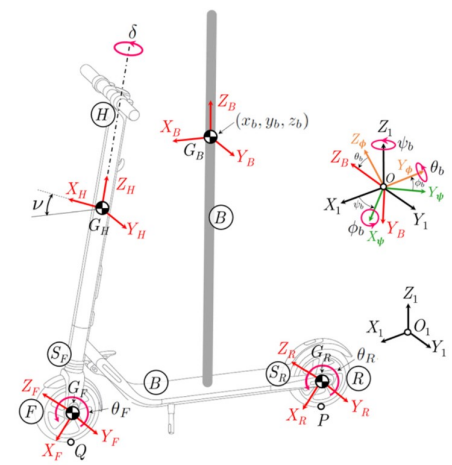
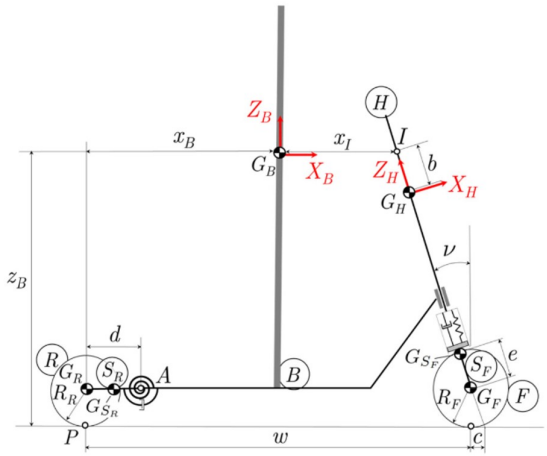


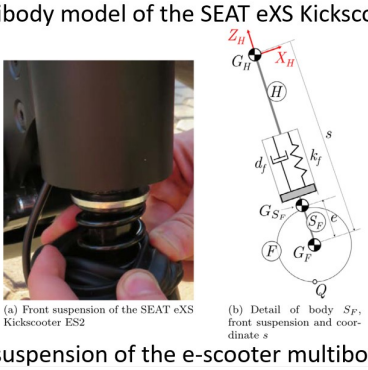
Figure 141: Ground Vehicle as a Dynamic System. Subsystems Interaction in a Motorcycle



Multibody model of the SEAT eXS Kickscooter ES2:



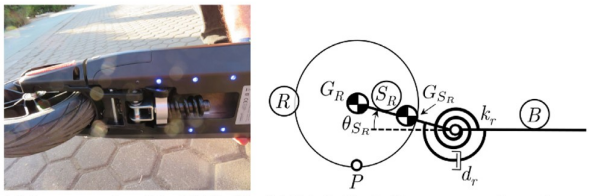
Lateral view of the e-scooter multibody model with the main geometric parameters.



(a) Front suspension of the SEAT eXS Kickscooter ES2

(b) Detail of body S_F, front suspension and coordinate s

Front suspension of the e-scooter multibody model



(a) Rear suspension of the SEAT eXS Kickscooter ES2

(b) Detail of body S_R, rear suspension and coordinate θ_{S_R}

Rear suspension of the e-scooter multibody model

Figure 142: Scooter dynamic modelling with both rear and front shock absorbers ([22])

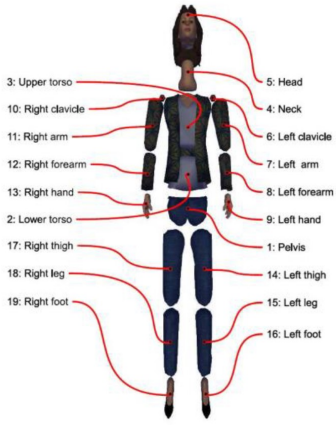
3.6.4 Definition of the Pedestrian models and components involved

In Pro-SiVIC, the pedestrian models consist of a 3D definition and configuration file (see figure 145) and it's associated animation file. The pedestrian normally have at least three types of basic animations:

- A stop idle animation: the pedestrian keep a static position
- A walking animation: the pedestrian apply a walking motion with a control similar to the vehicle (order in angle and in acceleration or torque). At this moment, the pedestrian has 2 specific modes: 0 for user control, and 1 for trajectory following.
- A run animation: Crossing the max walk speed, the model switch to a run mode.

In addition to these 3 motion modes, we have implemented orders allowing to manage the different part of the body like the head. This option is very useful in order to management of an object tracking by the eyes, or some head movement due to a specific event or situation. Moreover, the pedestrian uses the ray tracing engine in order to management collision with obstacles or or to go up and down a sidewalk.

The integration of these animation are done via a configuration file and the name of the file should reflect as specified in the configuration file.



Nom	Numéro
vertebra 1	1
vertebra 2	2
vertebra 3	3
vertebra 4	4
vertebra 5	5
left clavicle	6
right clavicle	10
left shoulder	7
right shoulder	11
left elbow	8
right elbow	12
left wrist	9
right wrist	13
left hip	14
right hip	17
left knee	15
right knee	18
left peg	16
right peg	19

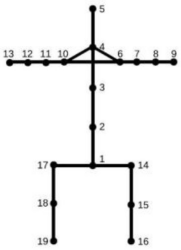


Figure 143: Graphic parts and points of articulation for pedestrian movement (Source: UGE)

Figure 144: Some different dynamic pedestrian types (Source: UGE)

```
Parts-Path:                Pedestrians/pantin-Woman/
Parts-Orientation: 0  0  90  #Euler angles to make it stand in OXZ plane
Root-Orientation:  0  0   0  #Euler angles to make it stand in OXZ plane
Scale-Factor:              0.017

Texture-File:              oakqrtrt.tga
#Texture-File:             ReflectionsGrises.tga
Use-Env-Map:               false

Curves-File:               walking.cset
Angle-Scale-Factor:        0.85
Z-Comp-Factor:             0.030012

Max-Walk-Speed:    2.0
Mass:                      70.0
Radius:                    0.4
Height:                    0.8

Number-Of-Parts:   19
Parts-Definition:

HIPS               hipsmain              0 0    -66     ROOT                0   0   66
RIGHT_UPPER_LEG    right_upper_legmain   0 4    -51     HIPS                0  -4  -15
RIGHT_LOWER_LEG    right_lower_legmain   0 4    -27     RIGHT_UPPER_LEG     0   0  -24
RIGHT_FOOT         right_footmain        0 4    -3      RIGHT_LOWER_LEG     0   0  -24
LEFT_UPPER_LEG     left_upper_legmain    0 -4   -51     HIPS                0   4  -15
LEFT_LOWER_LEG     left_lower_legmain    0 -4   -27     LEFT_UPPER_LEG      0   0  -24
LEFT_FOOT          left_footmain         0 -4   -3      LEFT_LOWER_LEG      0   0  -24
LOWER_TORSO        lower_torsomain       0 0    -66     HIPS                0   0    0
UPPER_TORSO        upper_torsomain       0 0    -66     LOWER_TORSO         0   0    0
NECK               neckmain              0 0    -87.5   UPPER_TORSO         0   0  1.5
HEAD               headmain              0 0    -87.5   NECK                0   0    0
RIGHT_CLAVICLE     right_claviclemain    0 10   0       UPPER_TORSO         0 -15   15
RIGHT_UPPER_ARM    right_upper_armmain   0 15   -81     RIGHT_CLAVICLE      0   0    0
RIGHT_LOWER_ARM    right_lower_armmain   0 15   -63     RIGHT_UPPER_ARM     0   0  -18
RIGHT_HAND         right_handmain        0 15   -48     RIGHT_LOWER_ARM     0   0  -15
LEFT_CLAVICLE      left_claviclemain     0 -10  0       UPPER_TORSO         0  15   15
LEFT_UPPER_ARM     left_upper_armmain    0 -15  -81     LEFT_CLAVICLE       0   0    0
LEFT_LOWER_ARM     left_lower_armmain    0 -15  -63     LEFT_UPPER_ARM      0   0  -18
LEFT_HAND          left_handmain         0 -15  -48     LEFT_LOWER_ARM      0   0  -15
```

Figure 145: Graphical definition and configuration for a pedestrian (Source: UGE)

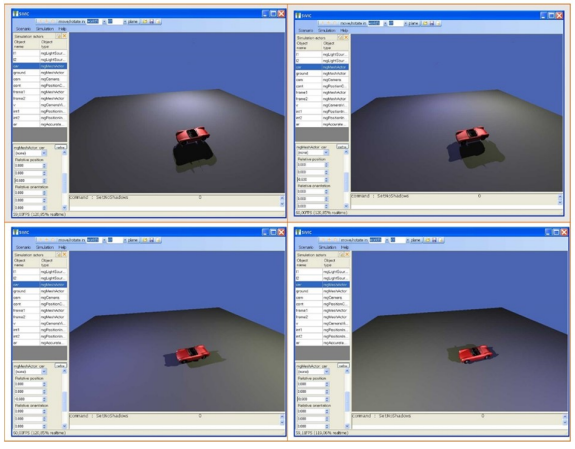
3.6.5 Definition of the other dynamic objects models and components involved

The main dynamic objects are the ones defined in the previous section. Nevertheless in the real environment, a large set of static objects could have, during a time period, a dynamic behaviour. It could be the case of a balloon, a fallen box, a paper, a group of leave ... In this condition and without dynamic models for these objects, it is mandatory to provide some mechanisms allowing to move these objects and to apply a specific trajectory. It is for this reason that 2 plug-in have been developed. The first one is mgPositionInterpolator and the second one is sivicTracking.

3.6.5.1 How to define a trajectory for a static component-element-object of Pro-SiVIC

The script-only mgPositionInterpolator tool allows defining simple trajectories that can be applied onto objects with no dynamic model. The operation of this module is very simple. Its mechanism is to define a set of waypoints for an object and assign an execution period. The object will then do its movement along the list of waypoints using an interpolation function with splines. The commands to create and configure a mgPositionInterpolator are described in the following table. There are two ways of creating a trajectory. The first is live recording positions from the simulation at given period using the StartRecord command. This requires controlling the object during the simulation from other sources, like another type of trajectory, user inputs with a mgPositionController, or third-party software interfaces. The other way of creating a trajectory is to manually add keyframes with the AddKeyFrame command. This method is typically used outside Pro-SiVIC, by manually editing a script file such as illustrated in Figure146. In this example, the movement of the 2 light sources will produce 2 different shadows of the car.

ApplyTo	<object>	Use positionable object as a target
SetPeriod	<t>	Set the time period for recording and playing (seconds between keyframes)
StartRecord		Start recording keyframes from target's movement
RecordOnce		Add one keyframe at target's current position
AddKeyFrame	<x> <y> <z> <r_x> <r_y> <r_z>	Add one keyframe at given position (x,y,z) and rotation (r_x, r_y, r_z)
SetLoop	<nb>	Enable play loop if nb!=0 (0 by default)
Play	<i>	Start playing from keyframe i (0 by default)
Stop		Stop playing or recording



```
# declaration of 2 light sources
new mgLightSourceI1
new mgLightSourceI2

# declaration of interpolators
new mgPositionInterpolator int1
new mgPositionInterpolator int2

# Object int1 with type mgPositionInterpolator
int1.SetPeriod   0.5
int1.SetLoop     0
int1.ApplyTo     I1
int1.AddKeyFrame    -7.52595 10.4517 9.89779 -15.4835 102.191 213.456
int1.AddKeyFrame    -12.7121 2.06879 9.89779 -81.8172 135.546 226.346
int1.AddKeyFrame    -10.4517 -7.52595 9.89779 -214.529 158.074 153.594
int1.AddKeyFrame    -2.06879 -12.7121 9.89779 -284.49 70.3217 -106.849
int1.AddKeyFrame    7.52595 -10.4517 9.89779 -175.674 -26.6174 -222.115
int1.AddKeyFrame    12.7121 -2.06879 9.89779 -96.4214 -58.2011 -219.085
int1.AddKeyFrame    10.4517 7.52596 9.89779 -47.8269 -64.908 -189.009
int1.AddKeyFrame    2.06879 12.7121 9.89779 -15.2056 -61.5151 -151.218

# Object int2 with type mgPositionInterpolator
int2.SetPeriod   0.5
int2.SetLoop     0
int2.ApplyTo     I2
int2.AddKeyFrame    -10.9104 -3.70592 6.70586 39.7163 -30.6472 -63.0534
int2.AddKeyFrame    -10.3353 5.09435 6.70586 27.6271 -48.1534 -104.72
int2.AddKeyFrame    -3.70592 10.9104 6.70586 8.32497 -64.5905 -145.134
int2.AddKeyFrame    5.09435 10.3353 6.70586 -21.407 -79.0321 -182.717
int2.AddKeyFrame    10.9104 3.70592 6.70586 -68.6626 -88.981 -212.958
int2.AddKeyFrame    10.3353 -5.09435 6.70586 -149.65 -85.8589 -219.034
int2.AddKeyFrame    3.70592 -10.9104 6.70586 -278.082 -35.8415 -126.463
int2.AddKeyFrame    -5.09435 -10.3353 6.70586 -272.405 73.7847 123.084

int1.Play
int2.Play
```

Figure 146: mgPositionInterpolator plug-in allowing to define a trajectory with position and orientation for a static object. The example presents 2 generated trajectories for 2 light sources lighting a static car (source UGE)

3.6.5.2 How to modify the spatial configuration of objects and sensors from third part application

In order to be able to modify the spatial configuration of objects during operation and from another application (RTMaps), the sivicTracking plug-in is available. This plug-in allows to manage a landmark to which you can attach a graphic object or a sensor (sensor position) and modify it from a third part application (outside Pro-SiVIC). On the Pro-SiVIC side, using this module is very simple. Simply create an object with the type sivicTracking and then activate the use of this object using the following script command SetActivated ¡0—1¿ (Allows activation of mark configuration control).

For other script commands, they correspond to all “positionable” and “recordable” commands. Consequently, like all sensors, this module has several operating modes (Off, On, Record, RTMaps, DDS, etc.). Its preferred mode of operation is, of course RTMaps, to control its position and angles from remote application. The object to be monitored must then be attached to this landmark using the script command <object>.MakeChildOf <sivicTracking object> . From RTMaps, the module used is LivicPositionRotationSimulation2 contained in the RTMaps_Livic_Simulator package (API for interconnection between Pro-SiVIC and RTMaps). In this RTMaps module, it is possible to set an initial position and orientation. It is also possible to choose the type of transformation that can be performed (Translation, Rotation, Translation and Rotation). The script in figure 147 shows the configuration of the controlled object (beetlejaune) from RTMaps and the object landmark. In reality, it is the track that will really be controlled. If in RTMaps we develop a module generating a continuous trajectory then the object in Pro-SiVIC will follow this trajectory.

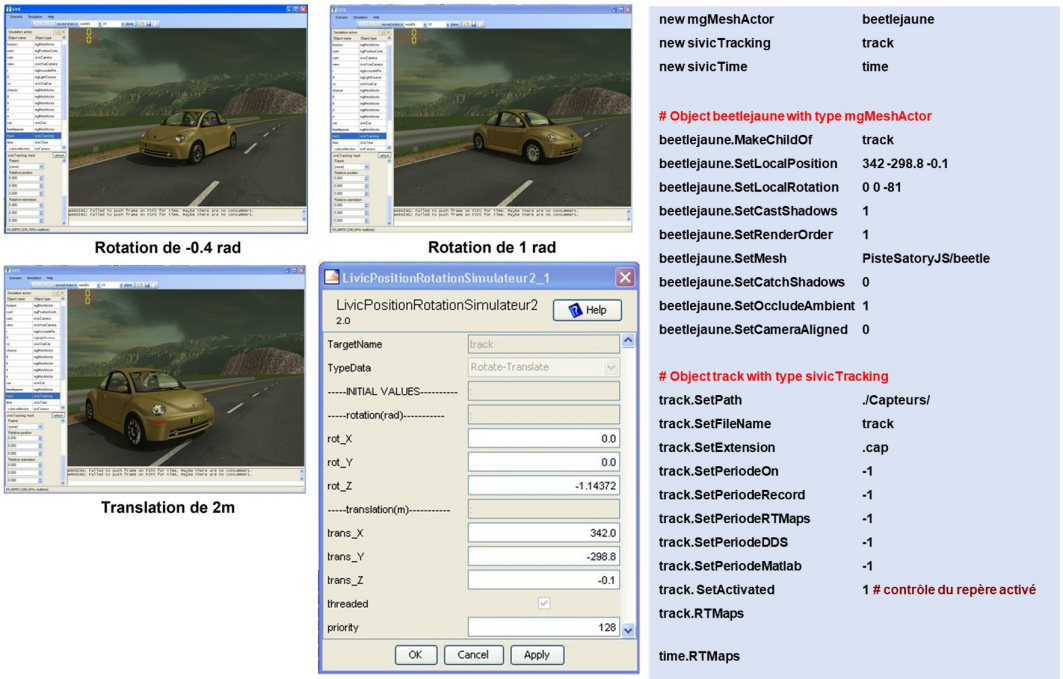


Figure 147: sivicTracking plug-in and RTMaps package allowing to define a new position and orientation for a static object or a sensor. The example presents several configuration sent to a vehicle in Pro-SIVIC (source UGE)

3.7 Communication means and network

3.7.1 Existing libraries and platforms

Numerous communication model and communication frameworks have been developed in the previous decade in order to address the scenarios involving communication means. The figure 148 proposes a list of these network simulators.

Network Simulator 3 (NS-3) is an open-source project distributed under the GNU GPLv2 license, developed almost exclusively in C++ with the option to use Python. NS-3 does not have a graphical user interface. Despite its name similarity, it is not the new version of the NS-2 simulation software but a separate version. The development of this new simulator was motivated by a quest for improving the realism of simulation models tailored to communication. This simulator offers a wide variety of simulation models, mainly focused on layers 1 and 2 of the OSI model (Wifi, WiMax, LTE). However, it allows simulating network nodes (routers, switches, or computers). It also enables simulating the behaviour of an Ethernet card or WiFi (802.11). Additionally, it allows modelling the propagation channel (optical fiber, Ethernet cable, wave propagation in the atmosphere, etc.) as well as various communication protocols. Its modular architecture allows for easy implementation of new modules for use in custom simulations (development of dynamic clustering strategies, routing strategies, communication energy management, etc.). The available models in the simulator are developed by the user community, and their number is continually growing. Indeed, NS-3 has a very large community of researchers contributing to its development. Many studies on VANETs over the past decade have been conducted using NS-3. It also features a real-time synchronizer that allows it to be integrated with real hardware (Simulation in the Loop) in simulation scenarios, as well as a mechanism for saving in PCAP format to keep a record of simulation execution for analysis software such as Wireshark. It also allows the simulation of a large number of devices. However, since this simulator is based on the use of models, some limitations of this approach need to be considered. Indeed, models are not perfect and may contain inaccuracies regarding specific scenarios or behaviours. NS-3, being based on a wide variety of models that are constantly expanding, requires caution regarding the results of these simulations. Similarly, even a validated model will always have differences from real results because a model cannot predict errors (such as human errors) during manipulations. Therefore, it is important to consider that the results of the simulation using the model provide accurate results for a specific case. It is also important to note that this simulator operates under Linux. An installation for Windows is available but requires the use of the Windows Subsystem for Linux (WSL).

OMNeT++ is a simulation framework primarily used for simulating networks. It is written in C++ and is compatible with Windows, Linux, and MacOS operating systems. It is an open-source software that can be used freely for research and studies but becomes paid for commercial use. The platform includes a graphical environment called TKENV and also allows for module development in Java and C. The architecture of the OMNeT++ simulator is based on a module hierarchy that communicates with each other through links that have their own properties. These elementary modules can be grouped together to form larger modules, making its development relatively easy. Regarding the network part: OMNeT++ can use different network frameworks such as the Mobility Framework, INET, INETMANET, OverSim, etc. Each of these frameworks has its own functionalities. For example, INETMANET is specialised for mobile ad-hoc networks. This framework is a fork of the INET framework and aims to include

MANET routing protocols that are not currently included in the INET framework. Especially the protocols:

- AODV - Ad Hoc On-Demand Distance Vector
- DYMO—Dynamic MANET On-demand
- OLSR—Optimized Link State Routing

This wide choice of frameworks and the simplicity of creating new modules make it a preferred platform because it is very flexible, allowing simulation for VANETs. Moreover, its very precise time management makes it easily integrable within a physical simulation platform. However, although OMNeT has a graphical interface and means to record scenario execution, leading to the generation of raw data files, the platform lacks analysis tools for the results and may require the user to develop their own scripts. It is also interesting to note that NS-3 seems to have more models available than OMNeT.

The network simulator **OPNET Modeler** is used to simulate the behavior and performance of any type of network. This software is developed in C and is proprietary, owned by the American company OPNET Technologies Inc. The license is obtained either by paying the usage rights or, in the context of research, by having a partnership between the university and the company. The main advantage of this simulator is its wide variety of ready-to-use protocol and hardware models. However, one of the major drawbacks of this software, due to its proprietary license, is that it does not allow for the implementation of new protocols or new hardware. Additionally, OPNET is not as popular as the two platforms presented earlier. It is important to note that in recent years, this platform seems to have no new developments.

In addition, merging communication and traffic simulator, iTETRIS proposes an interesting and efficient solution.

iTETRIS is a project for a large-scale V2X communication simulation platform for real-time traffic management applications within a European framework. The main objective of this project is to evaluate mobility systems and cooperative and communicative services (http://www.ict-itetris.eu/). The architecture of this project aims to provide a real-time closed-loop architecture, consisting of three distinct blocks: the SUMO traffic simulator, the NS-3 communication simulator, and the iTETRIS Control System (iCS), which allows coupling and controlling interactions between SUMO and NS-3 and also provides a user interface to the iTETRIS platform. This platform allows the creation of realistic traffic scenarios with the goal of evaluating various traffic management strategies. These realistic scenarios rely on real-time information exchange between vehicles (V2V communication) as well as with the existing infrastructure (V2I communication) to help improve traffic management. However, within the scope of this project and to account for the operational characteristics of the environment, this V2V-V2I system, based on cooperative vehicles, still needs to be defined and optimised. Recently, at the European level, standardisation of various communication protocols for Collective Perception Messages (CPM) has been proposed by the European Telecommunications Standards Institute (ETSI). This standardisation should also be integrated within this platform. In summary, this European project must address four challenges:

- Being an open-source simulation platform for road traffic and wireless communications.

- Creating large-scale scenarios.
- Realising a realistic simulation of V2V and V2I communications.
- Intelligent Transportation System (ITS) application based on dynamic, distributed, and autonomous cooperative systems.

The great strength of this project is its ability to integrate new modules into its ICS, allowing platform developers to add new features. Indeed, the ICS has its own interface, which also makes it easy to evolve the platform with future standards established by the ETSI.

The **VNetIntSim** project is the concept of using two distinct simulators to create a simulation platform suitable for simulating Vehicular Ad Hoc Networks (VANET). This platform is based on two components, the OPNET communication simulator and the INTEGRATION traffic simulator. The principle of the platform is to leverage the advantages of each of the two simulators. The OPNET simulator was chosen for the reliability of its simulation (simulation results have been tested for real-world cases). INTEGRATION, on the other hand, was selected because it allows simulating a large number of vehicles simultaneously while maintaining control over time management during a simulation of approximately 100 ms. This precision allows for a finer analysis of acceleration, lane changes, etc. To enable communication between these two simulators, the platform has two options. The first option is to use shared memory on the same machine running both simulators. This option would optimize data exchange between the two simulators but is limited by the machine's power, both in terms of available computing power and memory. The other method chosen for this project is to use the TCP protocol to send information between the simulators over a network. This method is more flexible as it allows the use of multiple machines to run both parts of the platform. However, it may encounter network-related issues such as latency, congestion, and reliability. The main strength of this platform is its accuracy in simulating large-scale traffic and communications. However, the architecture of this platform also suffers from a performance issue because the resources used by the platform increase exponentially with the number of simulated vehicles, thereby increasing simulation time. This platform can also be further improved by adding the impact of eco-friendly vehicles or congestion avoidance to optimize the simulations produced.

Veins combines the functionalities of the SUMO traffic simulator and the OMNET++ network simulator. This platform is open-source. It relies on the models implemented in the network and traffic simulators to perform the most realistic simulation of VANETs possible. As both simulators are developed in C++, it was possible to modify their core to integrate the two simulators. This integration involves implementing modules in each simulator to exchange information in the form of commands via the TCP protocol. The main advantage of this platform is the speed of setting up VANET scenarios. Indeed, a set of scenarios are already implemented and well-documented. However, implementing a new scenario is more complicated if it requires models that are not implemented in the platform and requires significant development work.

Reference(s)	Name of network simulators	Active development	Release	License	802.11p Support	Architecture Language	Simulation Language
42,43	OPNET	Y	2021	Commercial	Y	C++	C++ OTCL
33,42,44	NS-3	Y	2021	Open Source	Y	C++ Python	C++ Python
42	OMNeT++	Y	2020	Open Source	Y	C++	C++
33,42	QualNet	Y	2019	Commercial	Y	C++	C++
42,45	NS-2	N	2011	Open Source	Y	C++	C++ OTCL
33,42	JiST/SWANS	N	2005	Open Source	N	JAVA	JAVA
40,45	GloMoSim	N	2000	Open Source	N	C	C

Y = Supported, N = Not Supported

Figure 148: Network simulators.

Even if some references are given about LTE and 5G protocols, the main protocol studied and used will be the WiFi 802.11p dedicated to CAV. This media is a medium-to-short range system (¡1km) and is a part from the WAVE framework (with IEEE 1609). In order to develop this communication service, we propose to separate the telecommunication system for V2X in a set of sub parts (sub modules) involving Communication protocol, strategies, and OSI layers (Open Systems Interconnections), Emitter and receiver models, antenna diagram modelling, and Propagation channel modelling.

in [23], the CARTERY simulation framework is created for CAV prototyping and evaluation. This framework integrates three different well known simulators, namely SUMO, Carla, and OMNet++, for simulating the traffic, physical environment, and communication network, respectively.

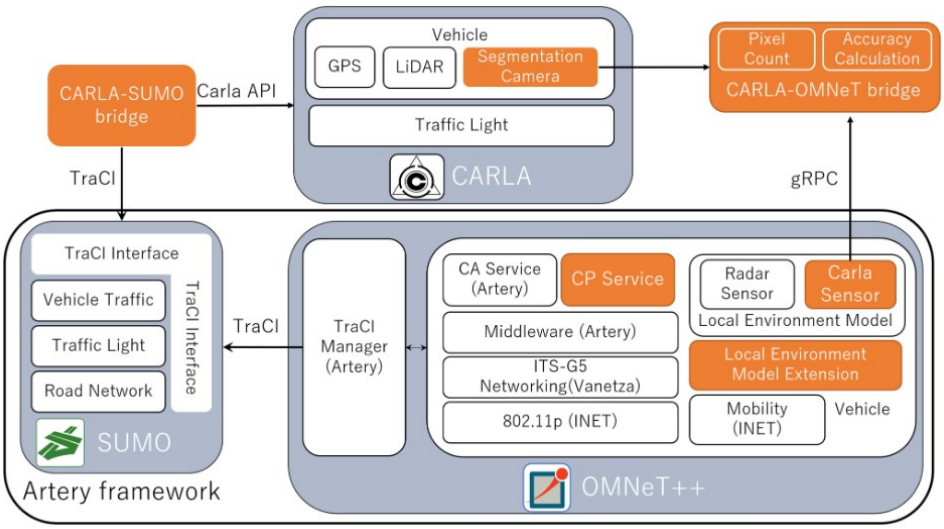


Figure 149: CARTERY simulation framework with CARLA, SUMO, and OMNET++ for CAV prototyping ([23])

3.7.2 Implementation in Pro-SiVIC

From the figure 150, we can see the shared architecture used by UGE in order to mimic the communication systems and strategies. RTMaps, like in the real prototype, allows to implement the real applications with the real processing modules needed for the deployment of CAVs. In this context, the different types of message (CAM, CPM, DNEM, ...) are received in RTMaps by using a module (RTMaps package for SiVIC Interfaces) similar to the one used in real condition. After the processing of the data, the new message is sent by the same mechanism toward NS3. NS3 is a well known and efficient framework for the simulation of the main communication protocols. The interconnection between Pro-SiVIC and NS3 provide the mobility information and the simulation of the propagation channel needed by NS3 in order to build a dedicated graph of possible communication nodes and paths.

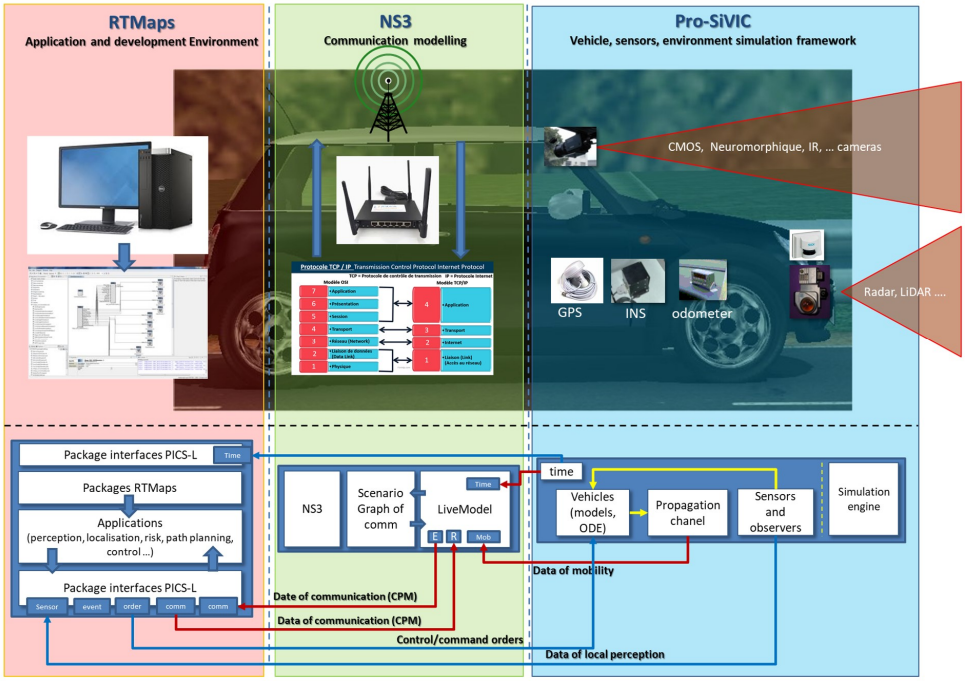


Figure 150: Communication architecture proposed by UGE and using an interconnected framework between Pro-SiVIC, NS3, and RTMaps.

In order to implement the wave propagation in the atmosphere, we have developed an experimental plan (describe in deliverable 2.7) with 2 vehicles equipped with 802.11p with a large range of relative speeds and with vehicle moving away and coming together scenarios (see figure 151). From this dataset, a set of 802.11p models have been proposed. These models (polynomial, semi-logistical, and semi-linear) are defined for a point-to-point connection and communication (2 nodes) in a motorway configuration (straight road or road with a very low curvature). The proposed set of functions of Frame Loss probability are function of the inter-distance and the relative speed between 2 communication nodes. These models have for constraint to reproduce the experimental data and have the capability to generate new plausible data. Nevertheless, these models are accurate and representative for motorway but not enough generic for all rural and suburban area. Moreover these models provide an approximation of the lower OSI layers in a small-to-medium-sized network (no complex topology). The real dataset have been used in order to assess the models parameters. The latency aspect is only based on

the frame's size. The 3 models are presented in the figure 152.

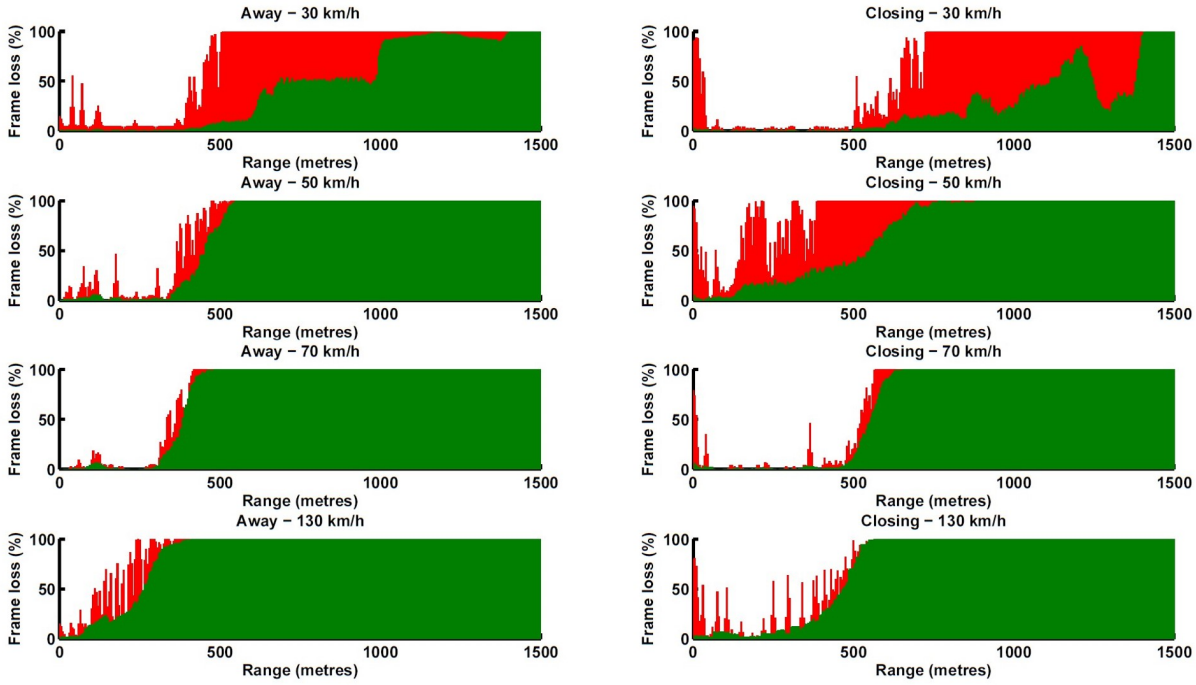


Figure 151: Detailed frame loss measurements for 30, 50, 70, and 130 km/h (5 metres intervals); red is the maximum value and green the average

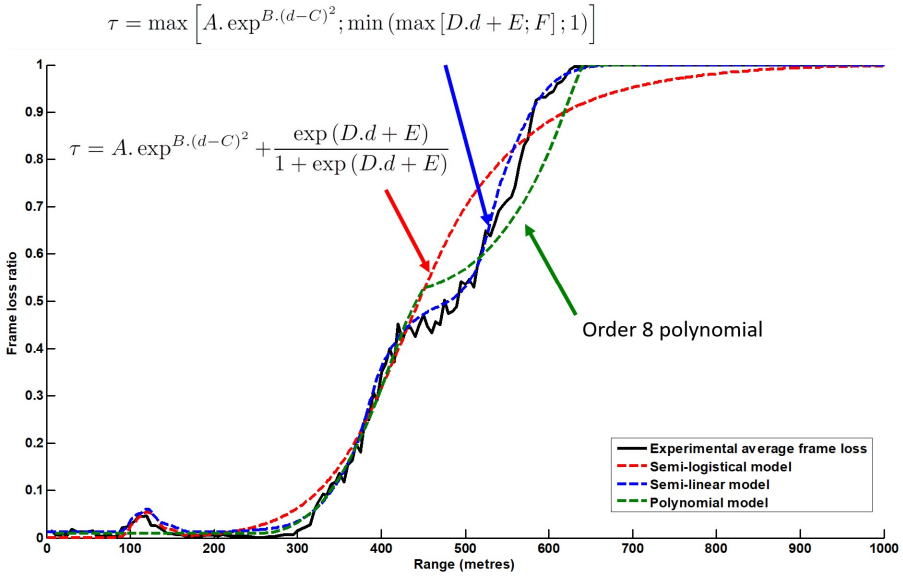


Figure 152: The 3 models of 802.11p communication standard proposed by UGE with the motorway dataset.

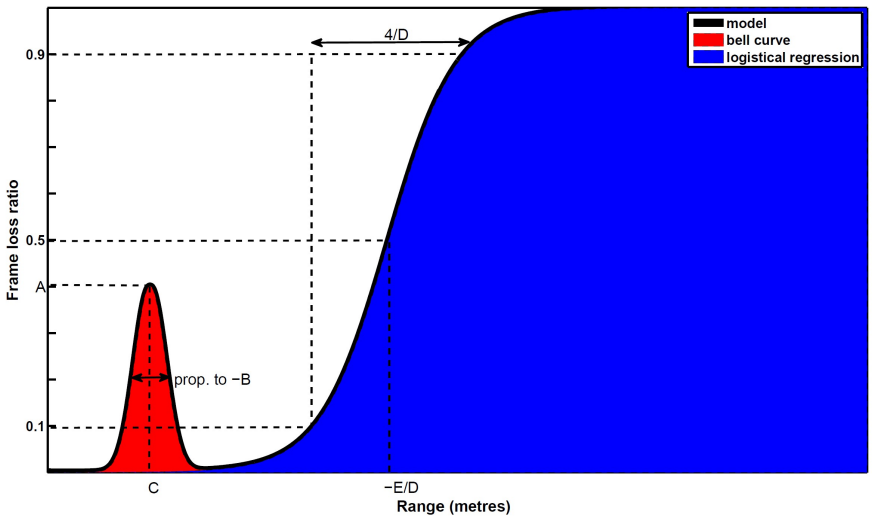


Figure 153: Decomposition of a frame loss profile with its parameters. This profile is share in 2 parts, the red one for the short range reflection interference given a significant loss of signal, the blue part representing the communication profile without interferences

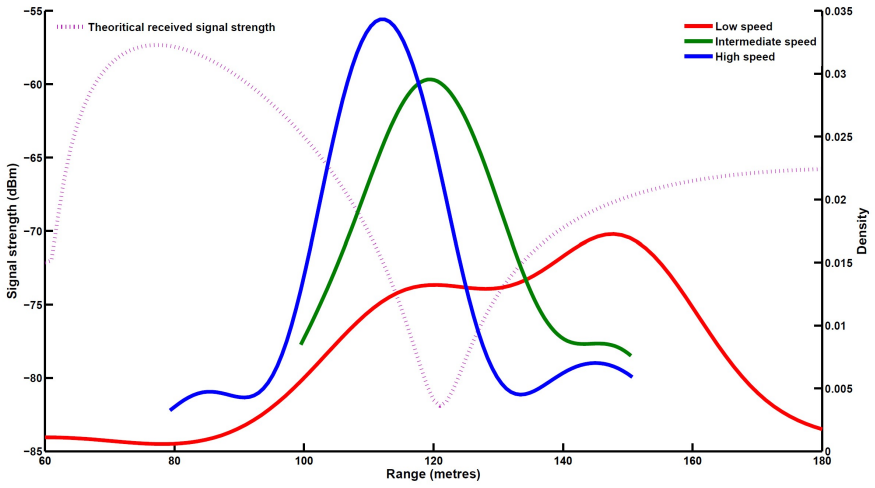


Figure 154: Distribution of parameter C (the central distance of strongest ground reflection interferences) for the three speed classes (right axis), compared to the received signal strength theoretical value (left axis)

For the logistical model, d is the distance between the emitter and receptor; and $A, B, ..., E$ are parameters estimated from empirical data. τ is the addition of two models, as illustrated in figure 153. Term $A.\exp B.(d-C)^2$ represents the frame loss area corresponding to the strongest ground reflection interferences, centred at distance C. At this point the ground-reflected signal is strong enough to cancel out a large proportion of the incoming direct signal's energy, pushing a proportion of frames under the reception threshold of the chipset; the frame loss corresponding to this proportion is represented by A. The width of the bell curve is in proportion to B; note that B is always negative. The model assumes that no counter-measure is applied to reduce the frame loss induced by interferences at C. The Friis transmission equation, modified to account for ground reflections, can be used to theoretically confirm the value of C. Assuming a dry concrete ground and realistic antenna heights, the Friss equation yields the received signal

lowest strength at a distance of 120 metres. The probability densities for the location of C are shown in figure 154, together with the theoretical value computed with the Friss equation. Factors such as speed, vehicle body shape, altitude profile of the track and pitch variations, explain that C is not always recorded at the same distance. Term $\exp(D.d + E)/(1 + \exp(D.d + E))$ is a logistical regression where the log-odds of τ is modelled linearly as a function of distance d. This term represents the progressive increase of frame loss as the received signal strength decreases. D and E by themselves have no direct physical meaning; however, the ratio $-E/D$ corresponds to the distance from the emitter at which the average frame loss passes over 50 %. Similarly, $4/D$ expresses the distance between the 10 % and 90 % frame loss thresholds. This profile is very interesting because it could be used both as a ground truth in order to validate other propagation channel models, or as a propagation channel in a communication simulation model. The implementation of this model in Pro-SiVIC is presented in the figure **??** with a platoon of 7 vehicles using a "lobe" model for antenna, a generation of a graph of possible paths between antenna, and with the 802.11p propagation channel model.

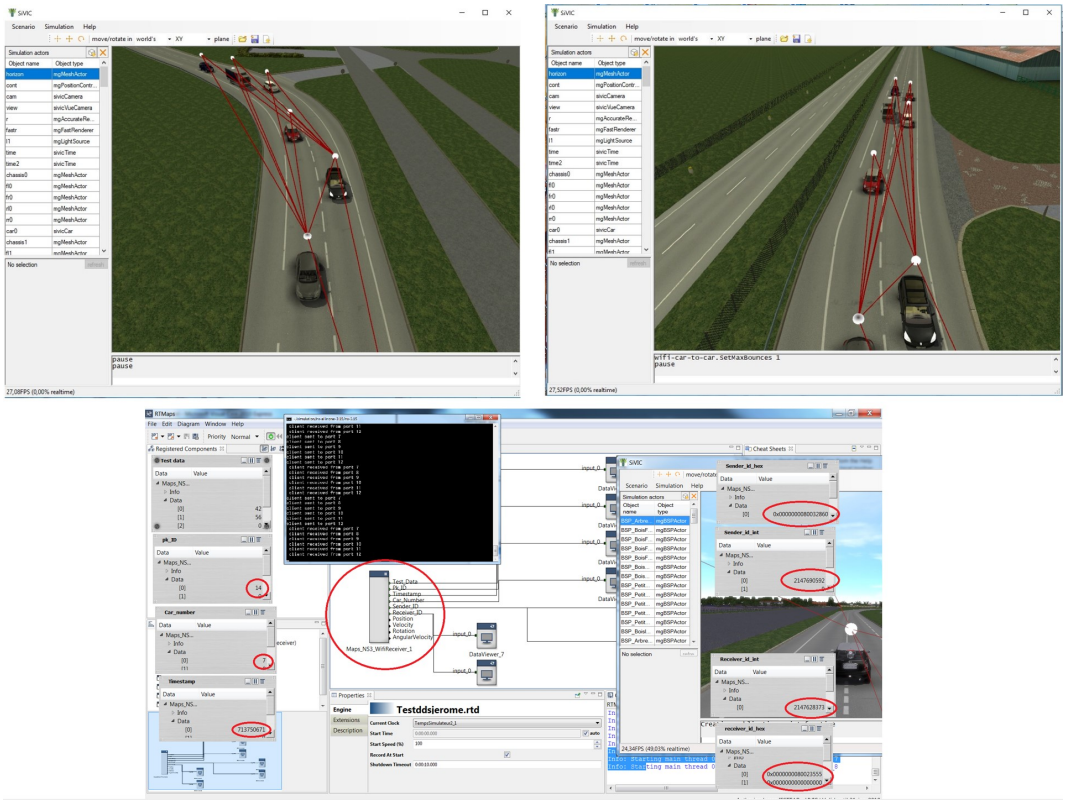


Figure 155: Overview of the communication results in 1 hop with propagation channel, emitter, receiver, and antenna diagram (source UGE)

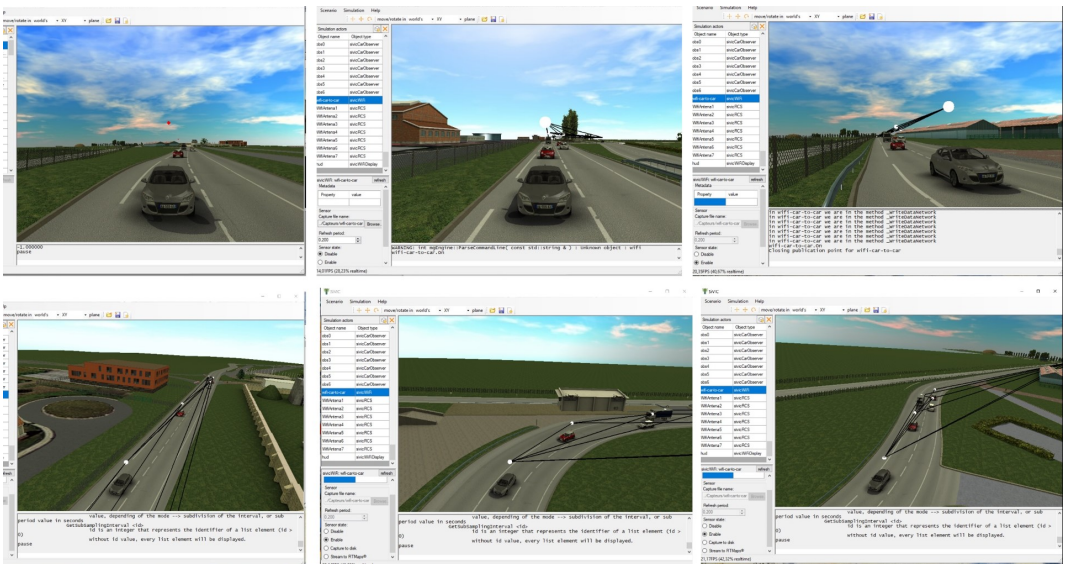


Figure 156: Overview of the communication results with propagation channel in 1 hop (source UGE)

3.8 Traffic generation

Modelling the surrounding and obstacle vehicles around the Ego vehicle enables to explore relevant and usual dynamic scenarios.

Developing a realistic and high quality and performances traffic generator for autonomous driving involves meeting several critical high-level requirements to create realistic and diverse traffic scenarios. Here are 10 main requirements to consider:

- **Traffic Pattern Variety**: Generate diverse traffic patterns, including highway, urban, suburban, and mixed scenarios, to simulate real-world driving conditions. This implies to have Digital Twin and HD Maps of specific representative areas. The respect of this requirement is essential to provide a comprehensive set of scenarios for testing and validating autonomous driving systems.

- **Dynamic Vehicle Behaviour**: Simulate dynamic and realistic behaviours for different types of vehicles, including acceleration, deceleration, lane changes, and interactions with other road users. This requirement is important to evaluate the adaptability and responsiveness of autonomous systems in complex traffic situations.

- **Realistic Vehicle Types**: Include various vehicle types, such as cars, trucks, motorcycles, bicycles, and pedestrians, to reflect the diversity and the complexity of road users. This requirement allows to address the capability of autonomous systems to interact with and respond to different types of vehicles.

- **Traffic Density Control**: Enable control over traffic density to simulate both sparse and congested scenarios. This requirement is essential to evaluate system performance under varying traffic conditions, including peak hours and low-traffic situations.

- **Traffic Light and Sign Simulation**: Model realistic traffic light and road sign behaviour, including changes in signal timing, yellow light intervals, and adherence to traffic rules. This requirement will be used to assess the interaction of autonomous vehicles with traffic

control infrastructure and generate specific risky and critical scenarios allowing to address interaction between AV and the other road users in intersection areas.

- **Pedestrian and Cyclist Behaviour**: Simulate realistic pedestrian and cyclist behaviours, including jaywalking, crossing at intersections, and interactions with vehicles. This type of scenarios allow to evaluate the ability of autonomous systems to navigate safely in the presence of vulnerable road users.

- **Adaptive Road Conditions**: Incorporate adaptive road conditions, such as changes in weather (rain, snow) and road surface conditions (dry, wet, slippery). This requirement is essential in order to guarantee a large coverage of the situations (environmental factors generating modifications and variations of the road environments) which could be encounter by AV.

- **Simulated Road Events**: Introduce mechanisms to simulate road events, such as accidents, construction zones, road work areas, temporary conditions (object falling on the road surface) and detours, to assess the response and decision-making of autonomous vehicles. This requirement and the generation by the traffic generator of this events will give the possibility to assess the ability of autonomous systems to handle unexpected events and deviations from regular traffic conditions.

- **Scenario Customisation**: Allow users to customise specific traffic scenarios, including the introduction of specific vehicles, road configurations, and event triggers. The implementation of this requirement facilitate targeted testing for specific use cases and scenarios relevant to the development and validation process.

- **Scalability and Performance**: Ensure that the traffic generator can scale to simulate large-scale scenarios while maintaining computational efficiency. Application of this requirement allows to guarantee the capability to replicate realistic traffic conditions in a scalable manner to assess the scalability and performance of autonomous driving systems.

By meeting these requirements, a traffic generator for automated mobility (involving systems of systems, or AI-based systems) can provide a versatile and realistic environment for testing and validating autonomous systems in a wide range of scenarios.

3.8.0.1 Definition of the platform for traffic simulation and management: EPICAM (UGE)

A traffic platform simulates other vehicles, pedestrians, and entities interacting with the simulated vehicle. It includes models for vehicle movement, traffic patterns, and interactions with the environment.

In UGE, a new platform has been developed from intern funding. This platform is called EPiCAM. The EPiCAM software platform (Digital Evaluation of Connected and Automated Mobility Services) aims to support the testing, deployment, and evaluation of automated driving and cooperative mobility. The main originality of this platform lies in the coupling of physical sites (experimental sites, living labs) and digital tools (simulation tools). This phygital (Physical and Digital) platform aims to combine digital tools and physical sites to form a digital twin in the validation and pre-certification phase of Connected and Automated Vehicles (CAVs). This

platform is part of a Hardware in the Loop (HIL), Model in the Loop (MIL), and Software in the Loop (SIL) approach.

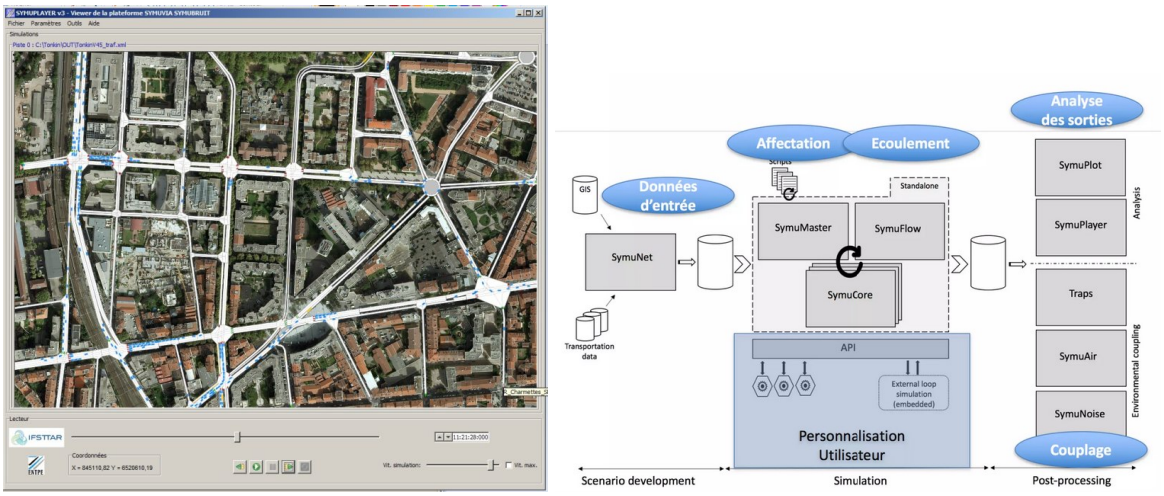


Figure 157: Symuvia, an open-source platform for traffic generation (Source: UGE, https://fr.slideshare.net/FabMob/fiche-symuvia-v5)

The EPiCAM software platform combines dynamic traffic simulation with simulation of perception and communication subsystems present in CAVs (and their associated components) to recreate an environment where multiple CAVs operate in augmented reality on a real site (e.g., Transpolis, Satory, Paris2Connect, ZEHNS). All manoeuvres of real CAVs are integrated into the same virtual ecosystem to ensure interaction with vehicles controlled by the traffic simulator. This software platform coupling traffic simulation (SymuVia), sensors, environment, vulnerable road users, and the vehicle itself (SiVIC) is generic and interoperable for use on future generations of Gustave Eiffel University's HIL/MIL/SIL platforms. The EPiCAM platform is presented in the form of a Data Distribution System (DDSL) offering communication channels and topics to SymuVia (see figure 157) and Pro-SiVIC. While Pro-SiVIC manages the EGO vehicle and its static environment, SymuVia provides dynamics for obstacle vehicles. The SymuEpicam.exe module, associated with SymuVia, as well as the sivicEpicam.dll plugin for Pro-SiVIC, then publish the appropriate information on topics and subscribe to topics to update their sets of controlled vehicles at each time step. The architecture of the EPiCAM platform thus consists of a set of 3 modules:

- EpicamLauncher.exe: an executable module for launching and instantiating the different modules of the platform (for example, 1 SymuVia module and 1 Pro-SiVIC module in a typical configuration).
- SymuEpicam.exe: an executable module responsible for simulating obstacle vehicles with SymuVia, ensuring, in particular: connection to the DDS bus for publishing information related to obstacle vehicles. Usually, SymuVia reproduces the movement of vehicles in lanes with instantaneous lane changes corresponding to pipe changes. To overcome this lack of realism during 3D rendering, a "ghost" module has been introduced to allow SymuVia to reproduce the impact of a smooth lateral lane change.
- sivicEpicam.dll: a module integrated into Pro-SiVIC and dedicated to the use of Pro-SiVIC within the EPiCAM platform. This plugin ensures, in particular: the dynamics

of the ego vehicle, which it publishes on the DDS, the creation of digital objects associated with obstacle vehicles (The plugin collects information related to obstacle vehicles, then converts them to estimate the positioning of the chassis and wheels of obstacle vehicles). The projection of information carried by the ”ghost” module of SymuVia allows the reproduction of smooth lane changes.

In both, Pro-SiVIC and Symuvia, the road network is provided with an OpenDrive file from the Satory or Transpolis test tracks.

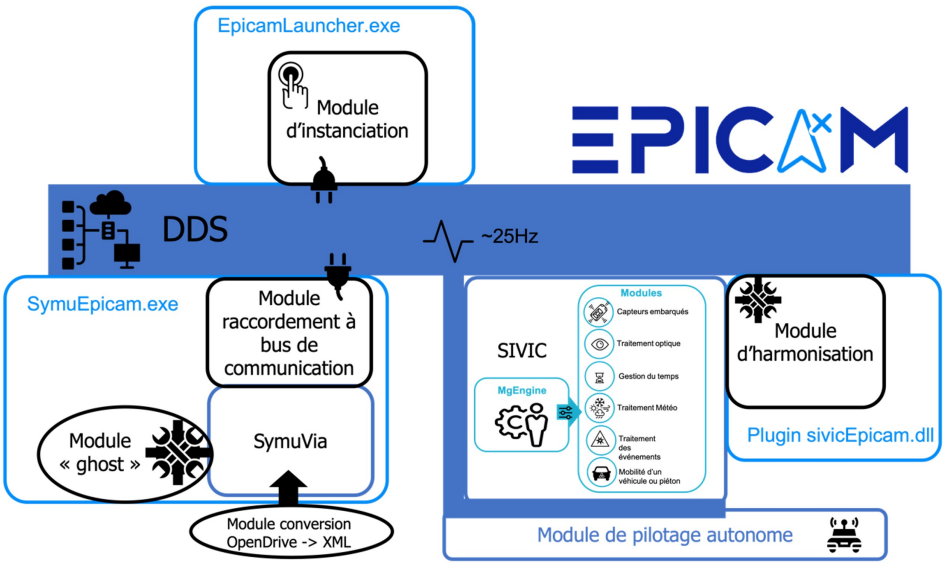


Figure 158: Current architecture of EPICAM platform (SiVIC+Symuvia) (Source: UGE)

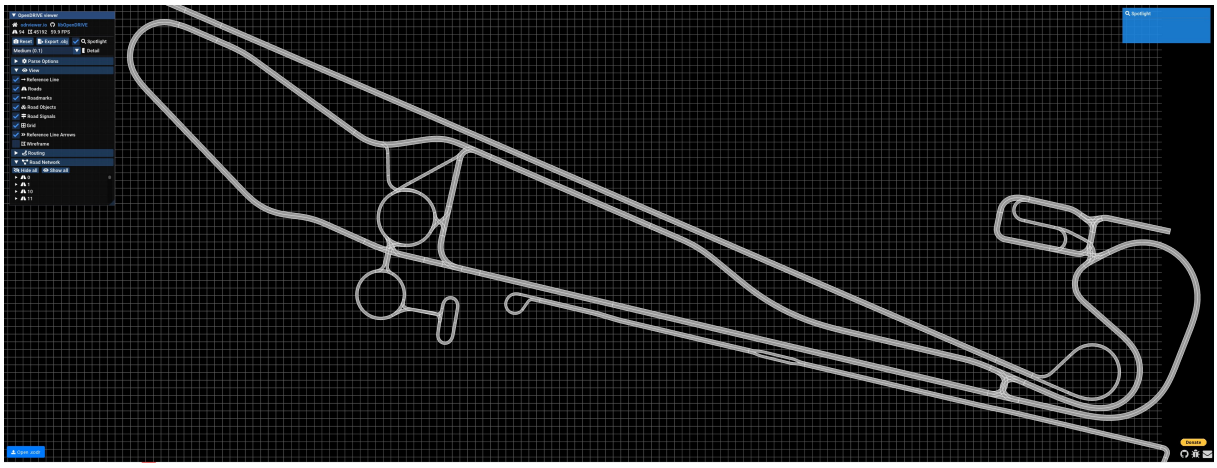


Figure 159: OpenDrive file for the definition and the modelling of the Satory road network (Source: UGE)

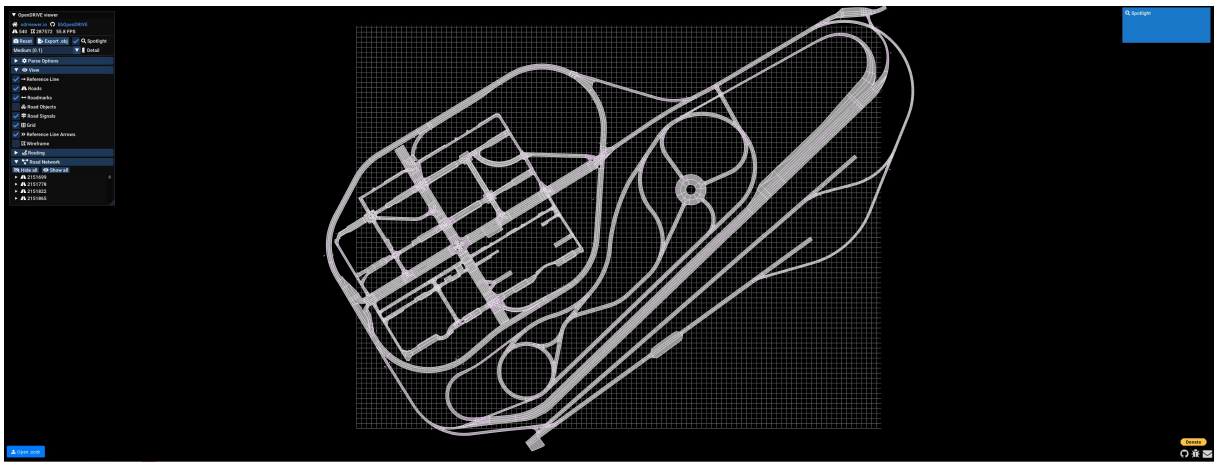


Figure 160: OpenDrive file for the definition and the modelling of the Transpolis road network (Source: UGE)

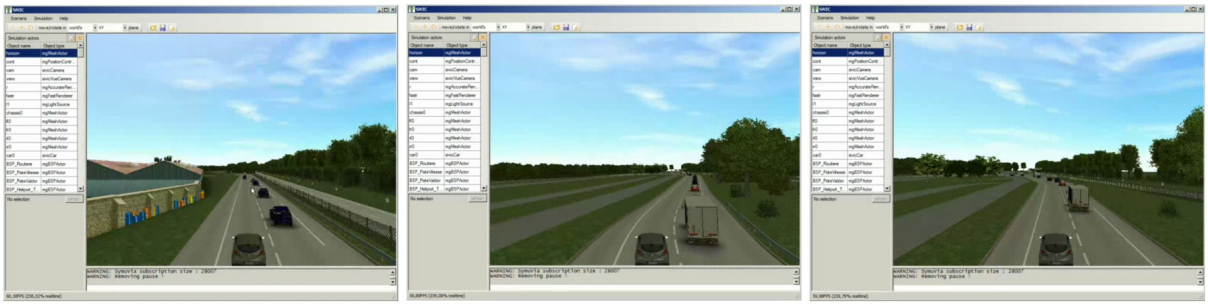


Figure 161: EPICAM: Real time interconnection of Symuvia and Pro-SIVIC (test on the Satory test track) (Source: UGE)

3.9 Degraded and adverse conditions and disturbances

We detail in this section some models of relevant disturbances impacting perceptive sensors (see Figure 162).

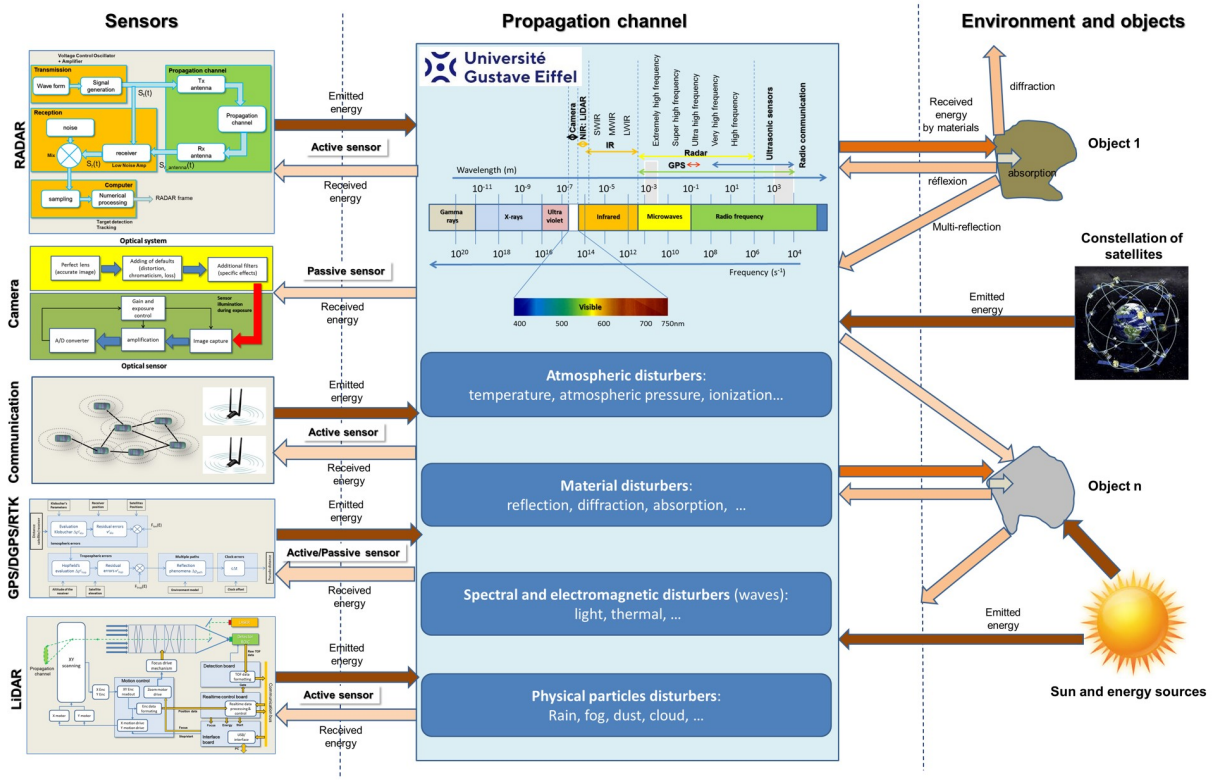


Figure 162: Classes of environment disturbers impacting sensors (source UGE)

We will focus on the modelling of the following factors impacting preceptive sensors performance:

- Weather conditions: fog, rain, rain drops, snow
- Road surface cracks, wears, soiling...
- Light disturbances
- Particles: dust, smoke, spay of water
- Soiling of road and traffic signs
- Material effects: glare, reflection, specularity
- Electromagnetic disturber and interference

The conditions listed below impact the perceptive sensors (camera, LiDAR, radar) due to the light absorption ans scattering by the particles in the atmosphere (fog, rain, snow, dust) or the modification of the light reflection properties of the materials. In order to model the impact of such conditions on perceptive sensors performance, one needs to model the path of a light ray interacting with the particles in the air and the materials of the road scene. The propagation of electromagnetic waves in participating media, such as fog, rain, snow or dust, is governed by the radiative transfer equation (RTE) in which the optical parameters (related to scattering, absorption and extinction) of the medium are considered. The RTE serves to simulate the radiance

$L_\lambda(t,r,u)$ corresponding, for a wavelength λ, to the intensity of the electromagnetic energy flux (in W) of the radiation propagating in the direction u, per unit of area (in m^2) perpendicular to the direction of propagation, per unit of solid angle (in sr) and per unit of wavelength (in microns), and expressed in $W.m^{-2}.\mu m^{-1}.sr^{-1}$. The RTE can be expressed as [78]:

$$\frac{1}{c}\frac{\partial L_\lambda}{\partial t}(t,r,u) + u \cdot \nabla_r L_\lambda(t,r,u) = \\ -\sigma_\lambda(r)L_\lambda(t,r,u) - \kappa_\lambda(r)L_\lambda(t,r,u) + \frac{\sigma_\lambda(r)}{4\pi}\int_{\mathbb{S}^2} L_\lambda(t,r,v)\Phi_\lambda(r,v,u)dv + q(t,r,u), \quad (1)$$

where c is the speed of light, t, r, u, σ_λ, κ_λ, Φ_λ and $q(t,x,u)$ denote, respectively, the time, the position in space, the wave propagation direction, the scattering coefficient, the absorption coefficient, the phase function for the wavelength λ and light sources (including thermal sources if thermal imaging is of interest). The three-dimensional unit sphere is denoted by $\mathbb{S}^2$. Optical passive objects and local light sources are taken into account thanks to the boundary conditions of Equation (1). For each light source occupying the space region S and emitting light from its surface ∂S, we have:

$$\forall\, r \in \partial S,\ \forall\, u \in \mathbb{S}^2,\ u \cdot n_r^S > 0,\ L_\lambda(t,r,u) = E_\lambda^S(t,r,u), \quad (2)$$

where n_r^S denotes the outward normal vector of S at point $r \in \partial S$, and E_λ is given. For each passive object occupying the space region O with a surface denoted by ∂O, we have:

$$\forall\, r \in \partial O,\ \forall\, u \in \mathbb{S}^2,\ u \cdot n_r^O > 0,\ L_\lambda(t,r,u) = \int_{v\in\mathbb{S}^2,\ v\cdot n_r^O<0} L_\lambda(t,r,v)B_\lambda^O(r,v,u)dv, \quad (3)$$

where n_r^O denotes the outward normal vector of O at point r, and $B_\lambda^O(r,\cdot,\cdot)$ is the *Bidirectional Reflectance Distribution Function* (BRDF) of object O at point $r \in \partial O$. When the time can be removed from the physics and under the assumption of a phase function depending only on $v \cdot u$ (scalar product), the following stationary case of Equation (1) can be considered:

$$u \cdot \nabla_r L_\lambda(r,u) = -\beta_\lambda L_\lambda(r,u) + \frac{\sigma_\lambda}{4\pi}\int_{\mathbb{S}^2} L_\lambda(r,v)\Phi_\lambda(v\cdot u)dv + q(r,u), \quad (4)$$

where we note $\beta_\lambda = \sigma_\lambda + \kappa_\lambda$ the extinction coefficient at the wavelength λ. Boundary conditions related to this stationary case are given by Equations (2) and (3) in which the time t is removed.

We detail in the sequel how to model the parameters involved in (1).

3.9.1 Definition and modelling of the weather conditions at a macroscopic scale

The World Meteorological Organisation [79] defined *precipitations* as *the liquid or solid products of the condensation of water vapour falling from clouds or deposited from air onto the ground. It includes rain, hail, snow, dew, rime, hoar frost and fog precipitation. The total amount of precipitation which reaches the ground in a stated period is expressed in terms of the vertical depth of water (or water equivalent in the case of solid forms) to which it would cover a horizontal projection of the Earth's surface. Snowfall is also expressed by the depth of fresh, newly fallen snow covering an even horizontal surface*. For automotive applications, we consider the precipitations: fog, rain and snow. The WMO defines fog as a suspension of very small, usually microscopic water droplets in the air, reducing visibility at the Earth's surface at a

level less than 1 km. The rain is compound of drops with higher sizes than those of fog and with a much higher falling speed. The rain is characterised by its rainfall rate (mm/h) defined as the water volume fallen during one hour on a 1 m^2 horizontal surface. In the METAR terminology (METeorological Aerodrome Report) which is recognised by the WMO, the snow is defined as a *precipitation of snow crystals, mostly branched in the form of six-pointed stars*.

For all these types of phenomena, the extinction of the radiation is a key parameter characterising the impact of weather conditions of optical sensors. The meteorological visibility, or meteorological optical range (MOR), is the distance for which the luminous flux of a collimated light beam is reduced to 5% of its original value [79, 80]. According to this definition, the visibility MOR is related to the extinction coefficient β as follows:

$$\text{MOR} = \frac{-\ln(0.05)}{\beta} \approx \frac{3}{\beta}, \tag{5}$$

It is important to mention that in Equation (5) β is considered in the visible band (at 550 nm wavelength) and is assumed constant by the WMO [79]. This is not relevant for infrared wavelengths [81] or microwave.

Automated vehicles sensors can be affected by other air particles which are not water: smoke or dust. The extinction of radiation due to the presence of these particles is governed by the same physical phenomena as for water particles.

3.9.2 Definition of the particles models and their impacts on the optical properties

The particle models are important to determine the optical properties of the medium (fog, rain, snow, dust, smoke): σ_λ, κ_λ, $\beta_\lambda = \sigma_\lambda + \kappa_\lambda$ and Φ_λ. Particle models are based on the particle size distribution (PSD) and the complex refractive index of the particles. A PSD is a function N (cm^{-3} μm^{-1}) such that $N(r)\,dr$ represents the number of particles contained in a volume of 1 cm^3 whose radii belong to $(r, r+dr)$. In the case of spherical particles, the Lorenz-Mie scattering model is largely used. The Lorenz-Mie theory [82] solves the electromagnetic equations of Maxwell for a spherical particle of radius r with a given complex refractive index $m_p = n_p + ik_p$ embedded in a host medium with refractive index $m_h = n_h + ik_h$. Under the assumption of non-dependent scattering between particles, the extinction and scattering coefficients are then expressed in terms of the PSD N as follows:

$$\sigma^\lambda_{ext}(N) = \int_0^{+\infty} Q^\lambda_{ext}(r)\,\pi\,r^2\,N(r)\,dr \quad ; \quad \sigma^\lambda_{sca}(N) = \int_0^{+\infty} Q^\lambda_{sca}(r)\,\pi\,r^2\,N(r)\,dr. \tag{6}$$

We also note that the absorption coefficient is defined as:

$$\sigma^\lambda_{abs}(N) := \sigma^\lambda_{ext}(N) - \sigma^\lambda_{sca}(N) = \int_0^{+\infty} Q^\lambda_{abs}(r)\,\pi\,r^2\,N(r)\,dr, \tag{7}$$

where

$$Q^\lambda_{abs}(r) := Q^\lambda_{ext}(r) - Q^\lambda_{sca}(r).$$

Similarly, the phase function can be expressed by the following form:

$$\sigma^\lambda_{sca}(N)\,\Phi_\lambda(\mu, N) = \int_0^{+\infty} Q^\lambda_{sca}(r)\psi_\lambda(r,\mu)\,\pi\,r^2\,N(r)\,dr, \tag{8}$$

where the scattering efficiencies and extinction efficiencies are given by:

$$Q_{sca}^{\lambda}(r) = \frac{\lambda^2}{2\,\pi^2\,r^2}\sum_{n=1}^{+\infty}(2n+1)\left(|a_n(r,\lambda)|^2 + |b_n(r,\lambda)|^2\right), \tag{9}$$

$$Q_{ext}^{\lambda}(r) = \frac{\lambda^2}{2\,\pi^2\,r^2}\sum_{n=1}^{+\infty}(2n+1)Re\left(a_n(r,\lambda) + b_n(r,\lambda)\right), \tag{10}$$

and ψ_λ given by

$$\psi_\lambda(r,\mu) = \frac{\lambda^2}{2\,\pi^2\,r^2\,Q_{sca}^{\lambda}(r)}\left(|S_1(\mu)|^2 + |S_2(\mu)|^2\right). \tag{11}$$

S_1 and S_2 are the scattering amplitude functions given by:

$$S_1(\mu) = \sum_{n=1}^{+\infty}\frac{2n+1}{n(n+1)}\left(a_n(r,\lambda)\pi_n(\mu) + b_n(r,\lambda)\tau_n(\mu)\right), \tag{12}$$

$$S_2(\mu) = \sum_{n=1}^{+\infty}\frac{2n+1}{n(n+1)}\left(b_n(r,\lambda)\pi_n(\mu) + a_n(r,\lambda)\tau_n(\mu)\right), \tag{13}$$

where the sequence of polynomials $(\pi_n)_{n\geq 0}$ and $(\tau_n)_{n\geq 0}$ are defined by the recurrences:

$$\begin{cases}\pi_0(z) = 0,\ \pi_1(z) = 1,\\ \forall\, n \geq 2\,, \pi_n(z) = z\dfrac{2n-1}{n-1}\,\pi_{n-1}(z) - \dfrac{n}{n-1}\,\pi_{n-2}(z),\end{cases}$$

$$\begin{cases}\tau_0(z) = 0,\ \tau_1(z) = z,\\ \forall\, n \geq 2\,, \tau_n(z) = z(\tau_n(z) - \tau_{n-2}(z)) - (2n-1)(1-z^2)\,\tau_{n-1}(z) + \tau_{n-2}(z).\end{cases}$$

The coefficients a_n and b_n in equations (9) and (10) are complex numbers called the Lorenz-Mie coefficients, which are defined thanks to spherical Bessel functions. For more details on a_n and b_n, we refer to [82].
For the case of water drops, the extinction efficiencies Q_{ext} and the absorbing efficiencies Q_{abs} are represented w.r.t. the particle radius r at the top of Figure 163 for different wavelengths (one in the visible $0.55\,\mu m$ and three in infrared $8, 10, 12\,\mu m$). At the bottom of Figure 163, we represent these functions for different particle radii w.r.t. wavelengths ranging in $[350\,nm, 2500\,nm]$. The coefficient Q_{ext} is dimensionless and depends on the droplet size and the wavelength varies between 0 and 4 and stabilizes around 2 for drops with a radius of a few microns.

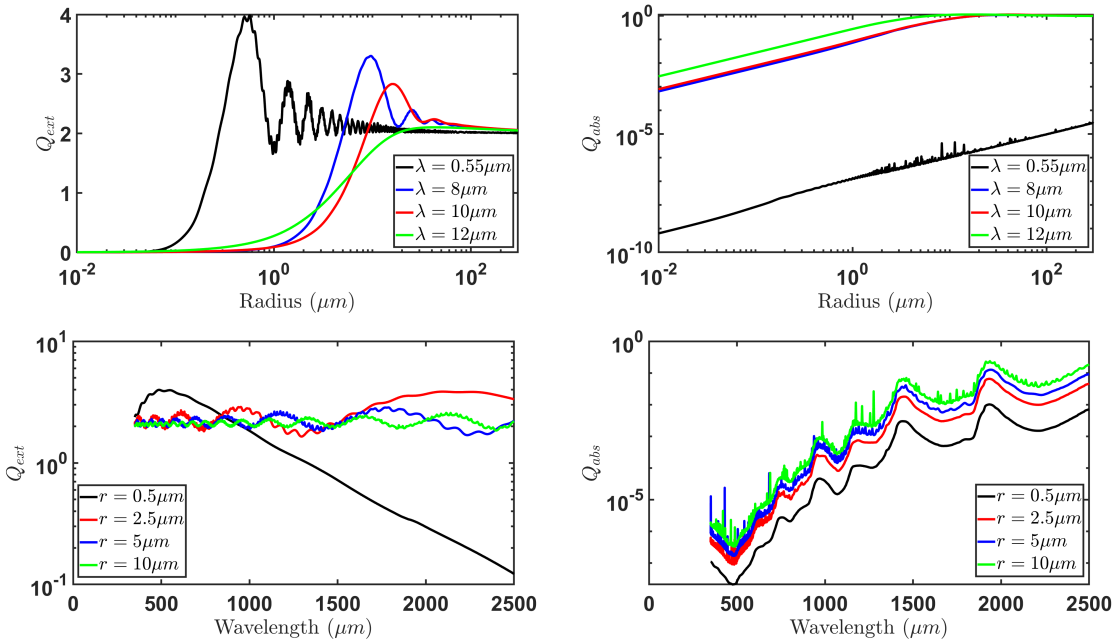


Figure 163: Water drop case. Extinction efficiencies (top left) and absorption efficiencies (top right) for four wavelengths (one in the visible and three in the thermal infrared), as a function of the radius of the sphere. Extinction efficiencies (bottom left) and absorption efficiencies (bottom right) for four particle radii as a function of the wavelength in the band 350-2500 μm.

The numerical computations of the series introduced above require a truncation. The most commonly used truncation, taking into account the numerical difficulties encountered with Bessel functions, is that of Wiscombe [83]:

$$E(v) = \begin{cases} v + 4v^{1/3} + 1 & \text{if} \quad 0.02 \leq v \leq 8, \\ v + 4.05v^{1/3} + 2 & \text{if} \quad 8 < v \leq 4200, \\ v + 4v^{1/3} + 2 & \text{if} \quad 4200 < v \leq 20000, \end{cases} \tag{14}$$

where $E(v)$ is the truncation function of the size parameter $v = 2\pi r/\lambda$.

The physical significance of the phase function Φ_λ is important. Indeed, for a photon moving at the speed of light in a medium, the phase function gives the probability of the resulting direction of the photon when it interacts with a scattering particle (water drop, snowflake, dust aerosol or molecule of the air). The phase function Φ_λ for six radii of spherical water droplets and different wavelengths from the visible to the thermal infrared range is shown in polar coordinates in Figure 164. One angle $\theta = u \cdot v$ is sufficient to represent this phase function due to the spherical symmetry. We can notice a very weak influence of the wavelength on the phase function for small spheres (r = 0.05μm and r = 0.2 μm), which is in accordance with Rayleigh's theory under unpolarized incident light [78, 84]. On the other hand, for sphere radii beyond 0.5 μm, the influence of the wavelength is noticeable, and backscattering gradually disappears. Finally, it should be noted that all of the curves presented in Figure 164 consider the variation in the complex refractive index of water according to the wavelength.

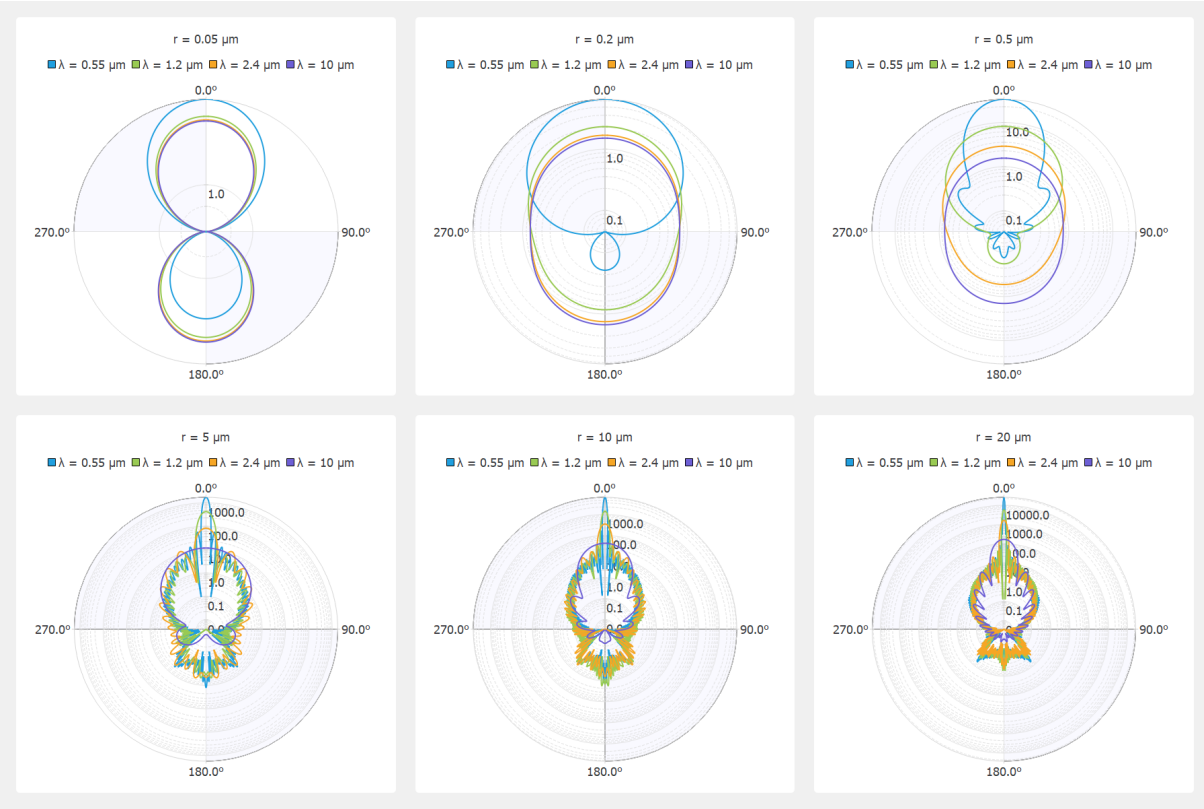


Figure 164: Polar representation of the phase function for six radii (r) of spherical particles and different wavelengths (λ).

An important remark concerns the dependence of the Mie coefficients a_n and b_n on the size parameter $v = 2\pi r/\lambda$. When $v \ll 1$, Mie scattering is well approximated by the Rayleigh scattering. For $v \ll 1$, the laws of optical geometry (Snell-Descartes laws) can be used and in the case of $v \sim 1$, the Mie theory has to be employed. Note that the Mie regime for a rain drop whose size ranges in (0.5 mm,7 mm) corresponds to electromagnetic frequencies between 40 Ghz and 600 GHz. Automotive radars operating around 77 GHz, Mie scattering theory is then relevant for the radar performance assessment in rainy conditions.

The RTE (1) can be simulated thanks to a Monte-Carlo based algorithm (ray tracing) which is reputed to require a lot of computing time. A particular case of Equation (4) is often used in a way to achieve an analytical solution. It consists of eliminating the collision (integral) term in (4) and assuming a constant source q. In this case, assuming there is no object and no local source between points r_0 and $r = r_0 + xu$ for x a real and $u \in \mathbb{S}^2$, we have:

$$L_\lambda(r_0 + xu, u) = L_\lambda(r_0, u)e^{-\beta_\lambda x} + \frac{q}{\beta_\lambda}\left(1 - e^{-\beta_\lambda x}\right), \tag{15}$$

leading to the Beer–Lambert solution if $q = 0$:

$$L_\lambda(r_0 + xu, u) = L_\lambda(r_0, u)e^{-\beta_\lambda x}. \tag{16}$$

The simple case presented above corresponds to the framework of the Koschmieder theory [85, 86] allowing the contrast between a black object and a sky background to be evaluated based on visibility attenuation due to the extinction of the medium between the object and the observer.

This theory is used in image processing to artificially add fog to an image: the intensity $I(x, y)$ of a pixel (x, y) is linked to the intensity $I_0(x, y)$ without fog and an air–light intensity I_s:

$$I(x, y) = I_0(x, y)\, e^{-\beta d(x,y)} + I_s \left(1 - e^{-\beta d(x,y)}\right), \tag{17}$$

where $d(x, y)$ is the real-world distance between the observer (camera) and the real point associated with the pixel (x, y), and β is the extinction coefficient of the medium for the visible range ($\lambda \simeq 550$ nm). The use of this simplified modelling of (4) needs only the knowledge of the extinction coefficient (or equivalently the meteorological visibility), without having to know the PSD. We illustrate how the use of PSD (and then of the phase function Φ_λ) can be necessary for the simulation of radiation propagation in fog. The choice of the RTE modelling is important for the simulated fog added to a clear image. The images in Figure 165 are obtained with the Cerema Monte-Carlo based SWEET simulator [87] without/with fog (a and b) and with the Koschmieder model for the same visibility and the same airlight radiance (c). A blur effect can be noticed in the SWEET image, which is not the case for the Koschmieder image. As we can notice in Figure 165c, Koschmieder's model brightens the foggy image much more than SWEET (Figure 165b). This can be critical because some objects in the scene are not even visible with the SWEET simulation and are partially visible with the Koschmieder model (e.g., the two pedestrians on the right).

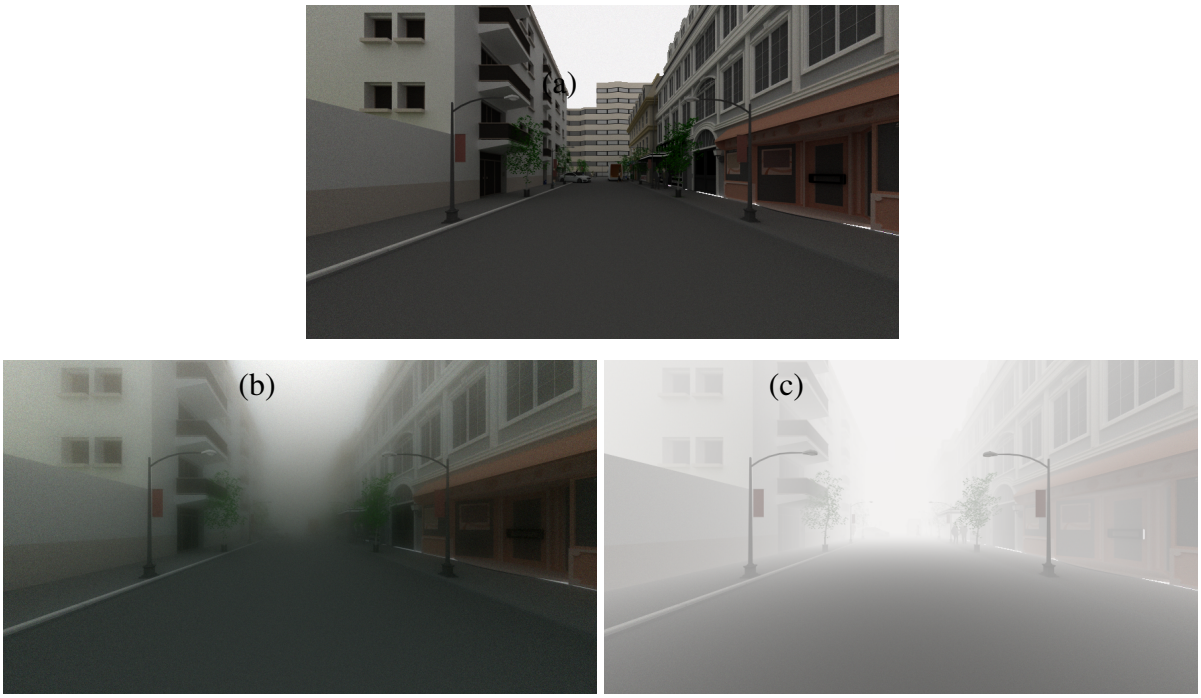


Figure 165: Simulated images for the intra-urban scene with the SWEET simulator without fog (**a**) and with fog (MOR = 20 m, (**b**)) and with the Koschmieder model (**c**) in day conditions.(Source: CEREMA)

In [88], results on experiments made in the Cerema Fog and Rain PAVIN platform show the sensitivity of extinction to the PSD of fog droplets. The simulated radiance (or intensity) thanks to the RTE (4) with two different PSD are showing in Figure 166). It can be observed different behaviour of extinction for two fog PSD corresponding to the same meteorological visibility.

We can remark that the sensitivity depends on the wavelength (visible range at 0,55 μm and thermal infrared range at 12 μm).

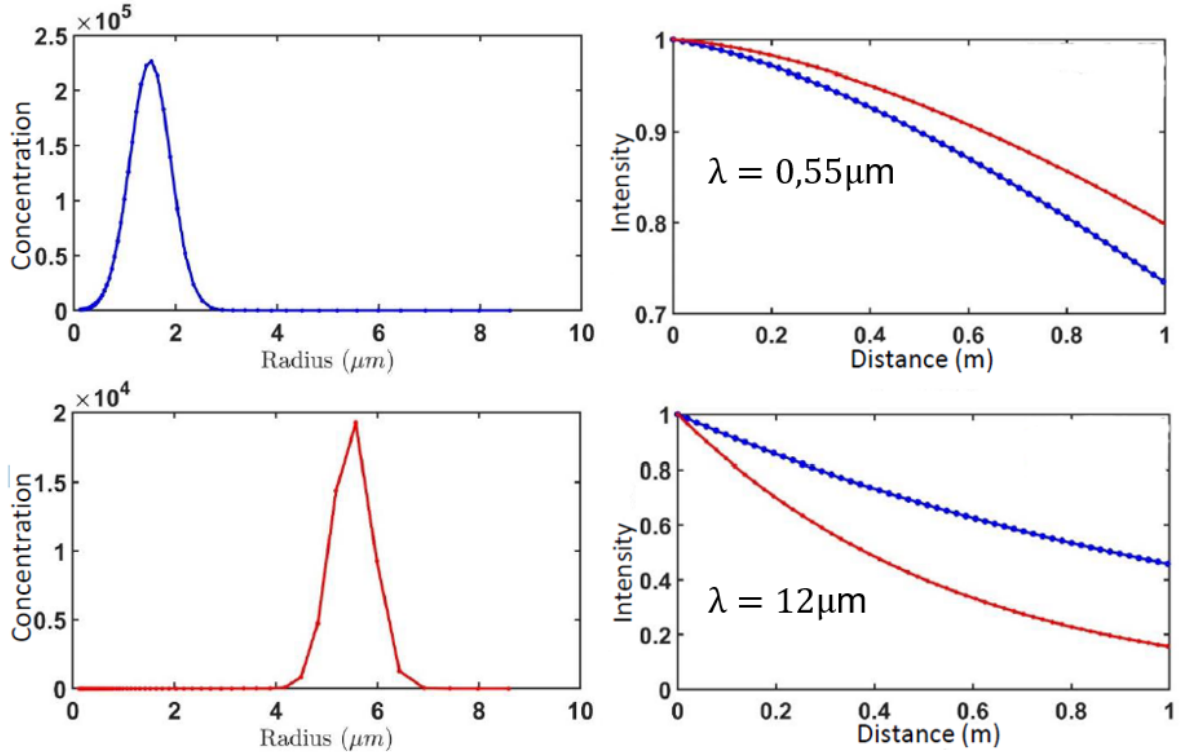


Figure 166: Simulated intensity w.r.t. the distance of a lambertian source for a fog with normalised visibility 0,75 m with small droplets (blue PSD on the left) and bigger droplets (red PSD on the left) at wavelength 0,55μm (top right) and 12μm (bottom right).

3.9.3 Particle size distribution modelling

3.9.3.1 Fog case

Many researches are done around the world on fog drop size distributions or other characteristics like liquid water content, total concentration of drops, mean diameter [89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 52]. Experimental measurements lead to a fog droplet size ranging from a few tenths of a micron to a few tens of microns [99, 100, 51, 101, 102, 103, 104, 105]. Beyond measurements, numerous works were developed to model the fog PSD: shifted gamma laws [106, 107, 108, 109] and log normal laws [107, 110, 91].

Log normal laws are expressed as follows:

$$N(r) = \frac{N_{\text{tot}}}{r\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln r - \ln\tilde{r})^2}{2\sigma^2}\right), \tag{18}$$

where N_{tot} is the number of particles in a referenced volume, σ and $\tilde{r}$ 2 parameters. The modified Gamma law based models for radiation and convection fogs are given e.g. by Shettle and Fen [35]):

$$N(r) = c\, r^{\beta} e^{-dr^{\gamma}},\ r \geq 0, \tag{19}$$

with the following coefficients:

Model	c	β	d	γ	$r_m(\mu m)$
1	4.651	3	0.3	1	10
2	13.399	3	0.375	1	8
3	428.15	6	1.5	1	4
4	211317	6	3.0	1	2

Table 1: Coefficients given in [35] for modified Gamma laws (19).

where r_m represents the peak position for each model.

We represent in Figure 167(b) the advection model of Shettle and Fen PSDs (with big droplets) and in Figure 167(a) the PSD of artificial fog produced in the Cerema PAVIN platform [52].

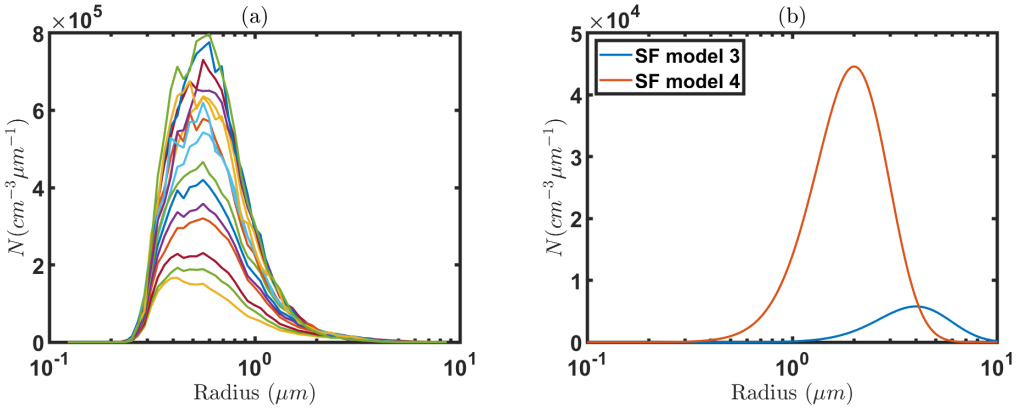


Figure 167: Droplet size distributions N (a) measured at Cerema PAVIN platform and (b) coming from Shettle and Fenn models.

For illustration, the PSDs for actual fog, depicted in Figure 168(a-b), were gathered during the night of March 13 to 14, 2007, at the Palaiseau site in France as part of the Paris-Fog campaign [111]. This type of distribution exhibits a minor peak for particles with a radius of 2 μm, as illustrated in Figure 168b.

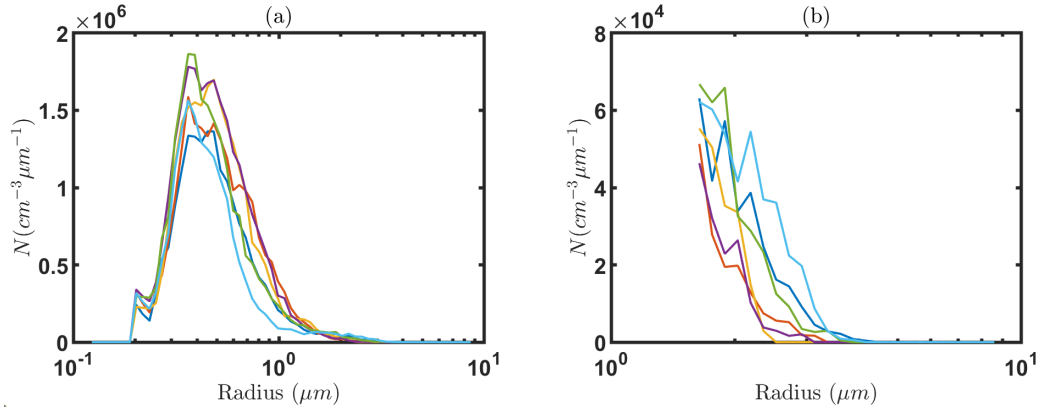


Figure 168: Droplet size distributions N measured during the Paris-Fog campaign over the range 0.1-10 μm (a) and over the range reduced 1-10 μm (b).

To complete the particle models, the refractive index of water has to be given. It is classical to use the Segelstein indices [24]. Figure 169 illustrates the refractive indices of pure water as provided by the Segelstein [24], showing their dependence on different wavelengths.

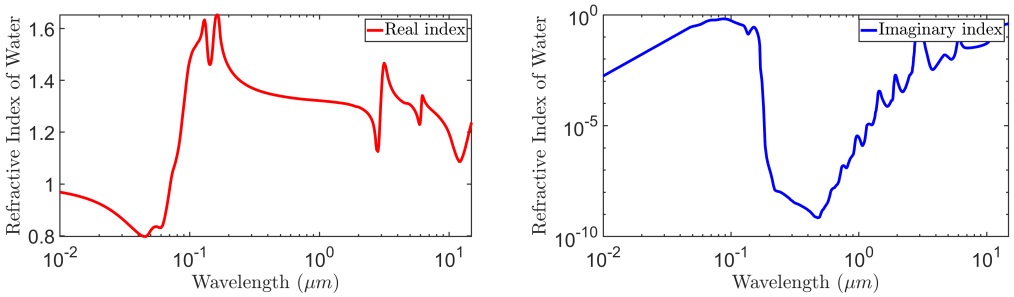


Figure 169: Representation of the complex refractive index of pure water (real part on left and imaginary part on right) [24].

3.9.3.2 Rain case

Raindrops have diameters in the millimetre range, from 0.5 to 5 mm. The usual law to model the size distribution of raindrops is a power law known as Marshall & Palmer [112]:

$$N(r) = N_0 \, (2r)^{\eta} \, e^{-2\Lambda r} \tag{20}$$

where $N(r)$ is the number of droplets per radius interval dr and per unit volume (in cm^{-4}), N_0 is the intercept and Λ is the slope of the curve (inverse of the average diameter of the drops). According to Marshall and Palmer [112]:

$$\begin{cases} N_0 = 0.08 \text{ cm}^{-4} \\ \Lambda = 41 \, R_r^{-0,21} \, cm^{-1} \\ \eta = 1 \text{ for uniform rain} \end{cases} \tag{21}$$

where R_r is the rainfall rate (precipitation rate) expressed in mm/h. We represent in Figure 170 (left) the Marshall & Palmer law for the 3 rainfall rates R_r: 10 mm.h^{-1}, 1 mm.h^{-1} and 0,1 mm.h^{-1}.

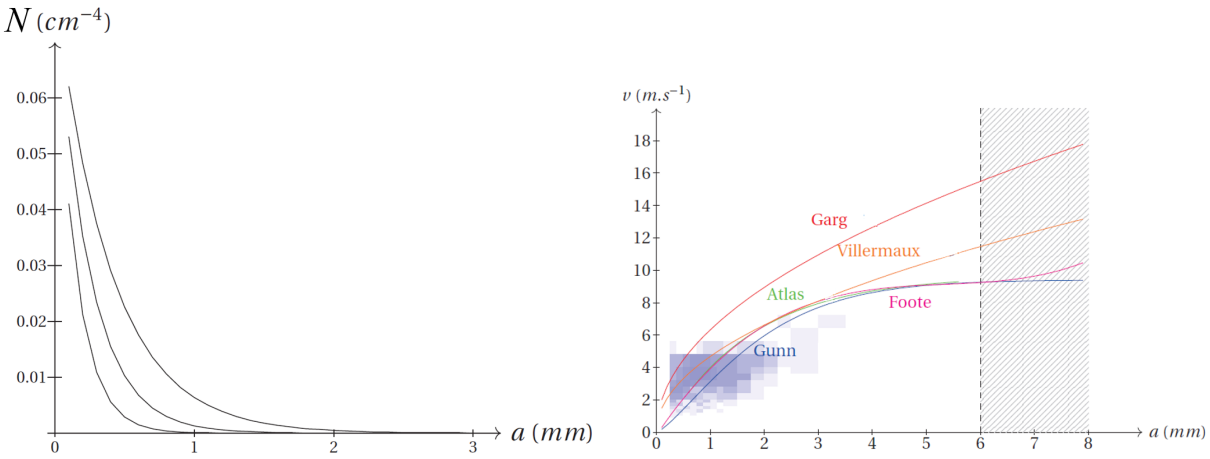


Figure 170: The raindrop size distribution N w.r.t. the diameter a according to the Marshall & Palmer law for the 3 rainfall rates R_r: 10 mm.h^{-1}, 1 mm.h^{-1} and 0,1 mm.h^{-1} (left). The terminal fall velocity of a raindrop w.r.t. the diameter for various models [25] (Garg), [26] (Gunn), [27] (Atlas), [28] (Foote) and [29] (Villermaux), and some measurements with a disdrometer (right).

There is a wide literature focusing on the Marshall & Palmer law validation and we can notice that some gaps can be found [113]. Other laws were developed with Lognormal laws [114] or Gamma laws [115]. These models are done under the assumption of spherical raindrops. Other models consider specific shapes for the droplets and examine the associated light scattering [116, 117, 27].

For the rain, the falling velocity of the drops have to be taken into account. Various laws are proposed to model the velocity of droplets. Free-falling droplets fall in a stable condition after falling 9m and we consider then what we call the terminal fall velocity. The relationship between this velocity and the raindrop size is obtained by the fundamental law of Newton applied with three forces: weight, Archimede Thrust and Friction forces due to air drag whose expressions are respectively:

$$\begin{cases} P = \rho_w \, V \, g \\ \pi = \rho_a \, V \, g \\ f = \dfrac{1}{2} C_x \, \rho_a \, v^2 \, S \end{cases} \tag{22}$$

where g is the gravitational acceleration (m.s^{-2}), ρ_w and ρ_a are respectively the density of water and the air, $C_x \sim 0.5$ is the aerodynamic drag coefficient, v is the raindrop velocity, $V = (4/3)\pi r^3$ and $S = 4\pi r^2$ are respectively the volume and the surface of the spherical raindrop of radius r (mm). The equilibrium of the 3 forces leads to a terminal fall velocity v of the form:

$$v = k\sqrt{a}, \tag{23}$$

where a is the diameter (mm) of the raindrop and $k = 3.5$ a coefficient experimentally estimated in [118]. This framework assumes that the terminal fall velocity is vertical since the wind is not taken into account (we refer e.g. to [119, 120] for the impact of wind on terminal fall velocity). Other models close to the model (23) are detailed in [25] (Garg), [26] (Gunn), [27] (Atlas), [28] (Foote) and [29] (Villermaux). These models are shown in Figure 170 (right). We can observe that the size of the droplets are assumed to be less than 6 mm since a size exceeding

this threshold is extremely rare [121]. We remark that the Gunn model exhibits a velocity limit by the use of an exponential:

$$v(a) = 9,40\left(1 - e^{-3,45.10^3 a^{1,31}}\right) \tag{24}$$

where a is the diameter of the drop (mm) and v is the raindrop velocity (m.s^{-1}).

Like fog, heavy rain can reduce transmission through the atmosphere by scattering light and the Lorenz-Mie theory associated to the particle models (20)-(24) and the RTE (1) allow to simulate the rain effect on electromagnetic propagation. As mentioned previously, the RADAR-based devices are mainly impacted in case of heavy rain fall and the attenuation is modeled thanks to the well-known RADAR equation:

$$P_r = \frac{P_t G^2 \lambda^2 \sigma_T}{(4\pi)^3 r^4} V^4 exp(-0.2\gamma r) \tag{25}$$

with the backscatter effect:

$$(\frac{S_t}{S_B}) = \frac{8\sigma_t}{\tau c \theta_{BW}^2 \pi R_t^2 \sigma_i}. \tag{26}$$

involving the rain attenuation coefficient γ and the rain backscatter coefficient σ_i computed thanks to the Mie theory from the rain PSD. The parameters and variables involved in (25)-(26) are gathered in table 2.

Variables	Names
P_r	Signal Power
P_t	Transmission Power
G	Antenna gain
f	Radar frequency
T	Pulse duration
θ_{BW}	Antenna Beam-width
P_r	Signal Power
σ_t	Radar Cross section of target
F_N	Receiver Noise Figure
B	Receiver Filter Bandwidth
T_0	Thermal Temperature
S_t	Power intensity of target signal
S_b	Power intensity of backscatter signal
c	Speed of light
V	Multipath coefficient
γ	Rain attenuation coefficient
σ_i	Rain backscatter coefficient
λ	Radar Wavelength
r	Distance between radar and target

Table 2: Parameters and variables of radar equation

For moderate rains, dedicated models are developed e.g. for camera. Due to the fall speed of the drops and the exposure time of cameras, rain streaks appear in images. For example,

based on the well-known works of Garg and Nayar [25], a rain simulator was developed in [30] in order to add rain on "clear" images. It does not exactly solve the RTE but is more phenomenological. The method is sketched in Figure 171.

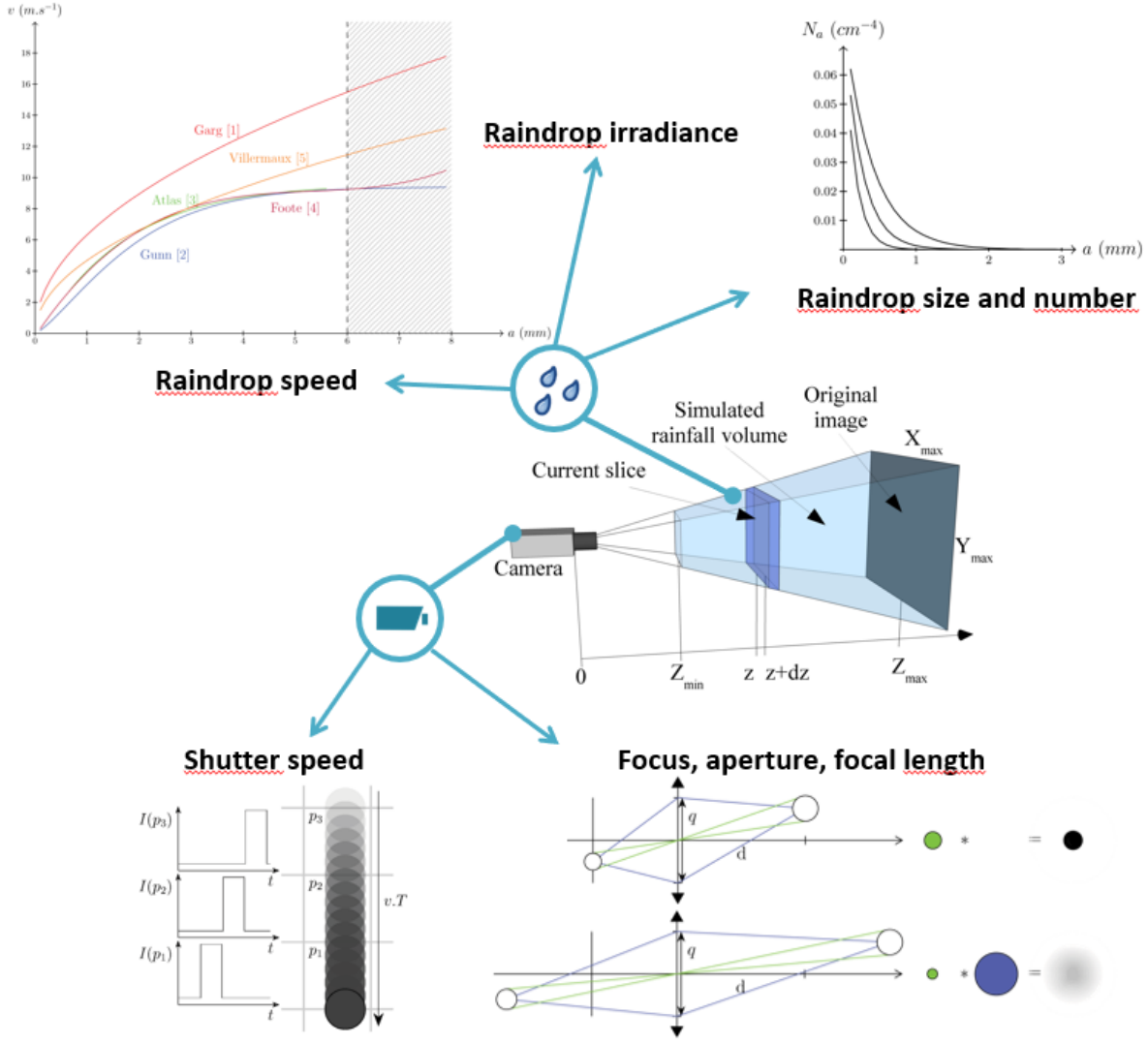


Figure 171: Scheme of the rain simulator developed in [30].

The simulator uses the most well-known laws of rain physics: PSD thanks to the Marshall and Palmer law (20), fall velocity depending on the size of the water drops parametrized by (24), uniform distribution of raindrops in the air volume [25]. The modelled camera is a lens and aperture camera, essential for proper understanding of the phenomena generated by rainfall on images. Contrarily to a pinhole camera, the advantage of this type of model is that it does not have an infinite depth of field as do pinhole cameras. This would give images with consistently sharp raindrops. But [25] has shown that the camera settings (which are used to adjust the field depth) are very important as far as the visibility of rain is concerned. Regarding the luminance emitted by a drop, [25] was able to show that it is on average constant. It actually corresponds to the diffusion of what is behind the drop in an angular field of 160°. In practice, this means

that it can be considered that the luminance of the drops corresponds to that of the sky, as the latter is on average largely predominant (in luminance) in usual outdoor scenes. The luminance of the drops is therefore be set to a value L in the simulator.

3.9.3.3 Spray case

The spray is the production of hydrometeors due to the tire rolling on wet road surfaces. From several simulation-based or experimental research works, the size of the water drops ranges between 0.05 mm and 0.5 mm [122] from CFD simulation-based experiments or between 0 mm and 4 mm [31, 123] from laboratory experiments. The size of the droplets depends on the tire velocity and the Weber number W_e definde by [122] :

$$W_e = \frac{\rho V^2 a}{\sigma} \tag{27}$$

is a key parameter to control the PSD. In the latter formula, ρ is the air density, V is the slip velocity between the droplet and the air (correlated with the tire velocity), a is the drop diameter and σ is the surface tension of the droplet. The higher the rolling velocity is, the thinner the droplet size is as we can see in Figure 172.

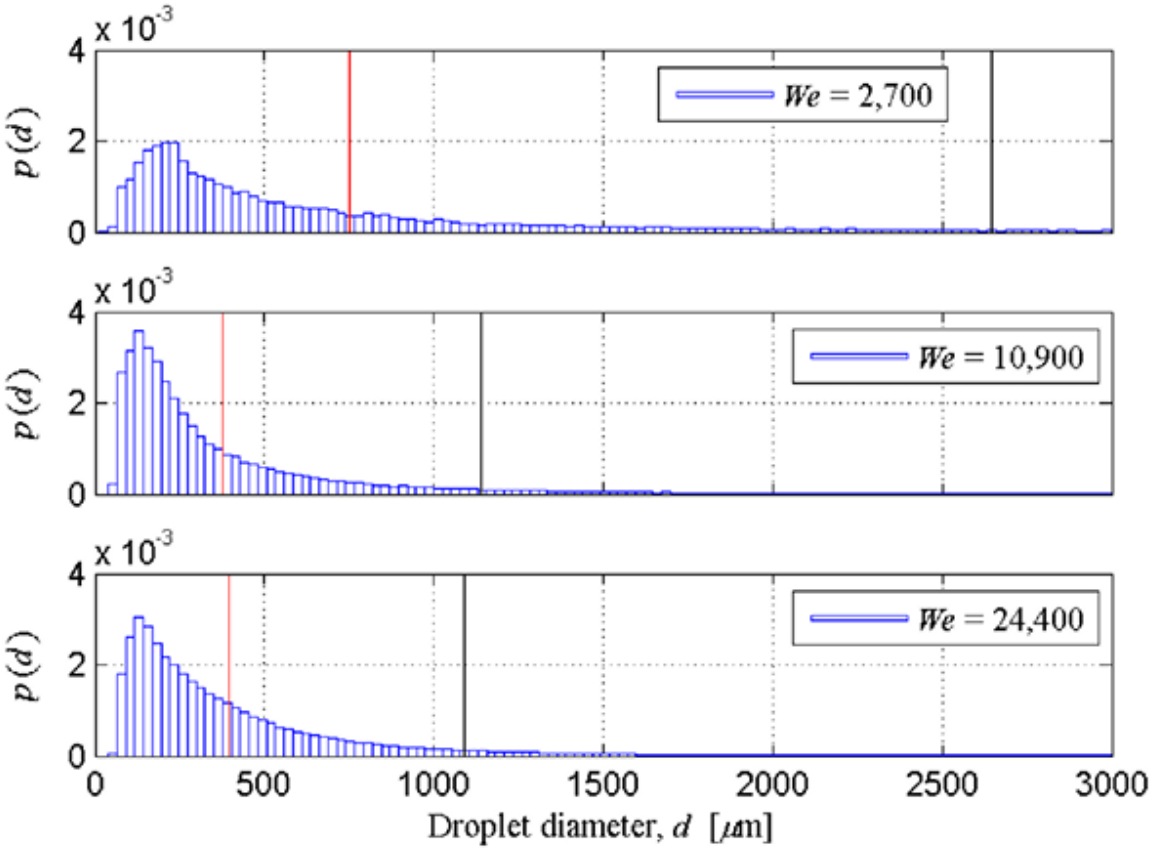


Figure 172: Droplet size distributions of a spray for several values of the Weber number W_e. From [31].

Spray is then a phenomenon with hydrometer sizes ranging between those of fog and rain. The light scattering theory exposed previously can be applied. It is important to note that spray

has an sensitive impact on the visibility reduction: to 90% at a distance of 20 m behind a truck [122] and up to 80% around a moving truck [123].

3.9.3.4 Snow case

The microphysics of snow is a topic largely studied for radar-based teledetection used to predict precipitations [124, 125, 126, 127]. This teledetection is based on inversion methods from light scattering observations. It is then necessary to model the light scattering by ice crystals. Extended Mie scattering theories are developed to model the optical properties of ice crystals whose shapes are very complex [124]. Moreover, some observations of snow particles show that they have densities depending on particle dimensions (the smaller the particle, the denser it is). It is then needed to consider snowflakes not homogeneous and to model them as particles with density that decreases from the center to the edges. Depending on the level of precision and realism required, there are different models of light scattering for snowflakes [124].

In [125, 126, 127], the PSD of the snowflakes are modeled thanks to a similar law as the Marshal & Palmer law (20:

$$N(D) = N_0\, e^{-\Lambda\, D}, \tag{28}$$

where D is the maximal length of the snowflake (1 mm $\leq D \leq$ 15 mm), N_0 and Λ are parameters. Following [127, 126], the values of N_0 and Λ are:

$$\begin{cases} 2.2 \leq \Lambda \leq 8.8 \text{ mm}^{-1} & \text{for dry snowflake} \\ 2380 \leq N_0 \leq 42000 \text{ mm}^{-1}\text{m}^{-3} & \text{for dry snowflake} \\ 1.8 \leq \Lambda \leq 3.1 \text{ mm}^{-1} & \text{for wet snowflake} \\ 1515 \leq N_0 \leq 4800 \text{ mm}^{-1}\text{m}^{-3} & \text{for wet snowflake} \end{cases} \tag{29}$$

The terminal fall velocity v (m.s^{-1}) of a snowflake with maximal size D (m) can be parametrized by different expressions [125, 128]:

$$\begin{cases} v(D) = 2.076\, D^{0.141} \\ v(D) = 2.958\, D^{0.157} \\ v(D) = 4.836\, D^{0.25} \end{cases} \tag{30}$$

3.9.3.5 Smoke and dust cases

Smoke and dust can affect the perceptive sensors of an automated vehicle if there is a fire in the vicinity of the road or if the vehicle is travelling e.g. close to a public works site. The particles of smoke or dust are in suspension in the air like water drops for the fog case. We can use for these particles the light scattering framework developed for fog. The differences between fog case and smoke and dust cases concern the optical properties of the particle and the PSD. Following [129, 130, 32], the smoke particle size is less than 1 μm with a mean diameter around 0.1 μm and the dust particle size ranges between 1 μm and 10 μm. Lognormal laws can be fitted to model the PSD of smoke and dust [32]. For the complex refractive index of smoke and dust, we refer e.g. to [131, 32]. We represent in Figure 173 the complex refractive index of dust w.r.t. wavelength in the range 0.4μm - 40 μ m [32].

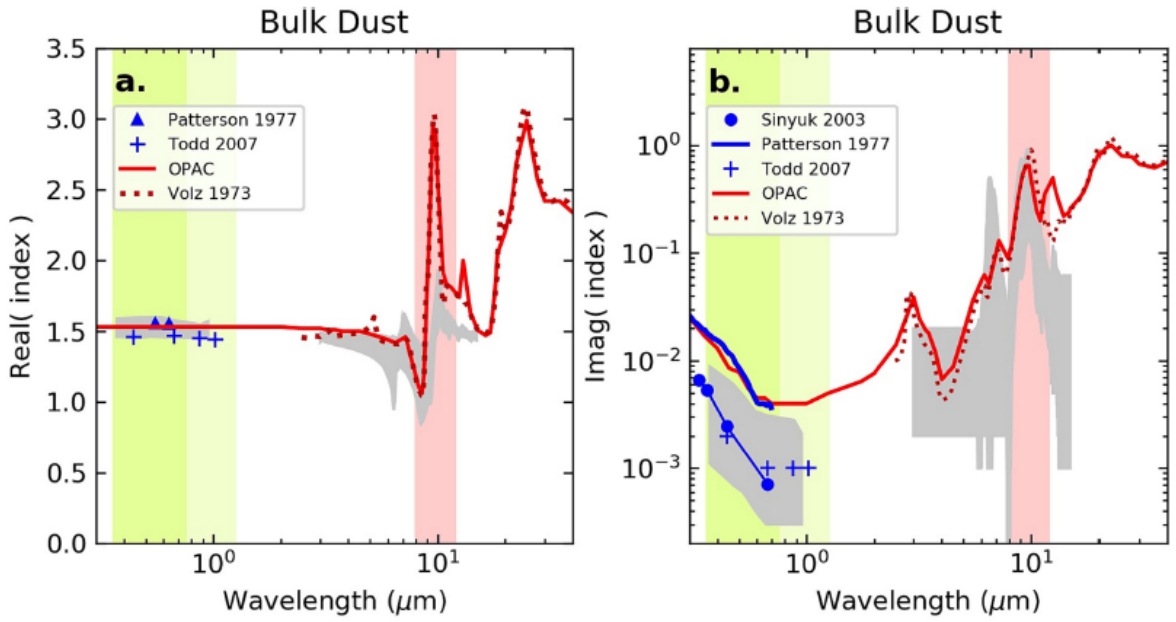


Figure 173: Complex refractive index for dust w.r.t. wavelength in the range 0.4μm - 40 μ m [32].

3.9.4 Light sources modelling

The general model of electromagnetic wave propagation described by the RTE (1) needs to define the light sources present in the road scene: urban lighting, vehicle lighting, IR emission for hot materials, etc. A light source is defined by its geometry (emission surface) and the emitted radiance (or specific intensity) by each infinitesimal surface element dS. The specific intensity I_λ, typically expressed in watts per square meter per steradian $(Wm^{-2}sr^{-1})$, is by definition:

$$I_\lambda(r,u) = \frac{d\phi(r,u)}{\vec{u}\cdot\vec{n}\,dS\,d\Omega dv}, \tag{31}$$

where $d\phi$ is the radiant energy passing through an infinitesimal area dS around the point r in an infinitesimal solid angle $d\Omega$ centered around a direction u and in the wavelength interval $[\lambda, \lambda + d\lambda]$. A light source is totally defined if its emitted specific intensity$I_\lambda(r,u)$ is known for each point r of the surface source, each direction u and each wavelength λ belonging to its emission spectrum. We can remark that this definition can be time-dependant if we have to consider e.g. active sensors like pulsed LIDAR.

The International Commission of Illumination (Commission Internationale de l'Eclairage in French, CIE) defined light source IES profiles for photometric applications (visible range for human vision). Each light source can be characterized by its IES profile tabulated in a specific file whose format is standardized (see Figure 174 (left) for a graphical representation of an IES profile). The emitted flux of the source is given by:

$$\Phi_v = \int_{\theta=0}^{\pi}\int_{\varphi=0}^{2\pi} IES(\theta,\varphi)\,\sin(\theta)\,d\varphi\,d\theta. \tag{32}$$

and is expressed in lumen (lm). The lumen is defined as the luminous flux of light produced by a light source that emits one candela (cd) of luminous intensity over a solid angle of one

steradian (sr). This photometric flux is linked to the radiative flux by the following relation:

$$\Phi_v = K_{cd} \int_{380nm}^{780nm} \Phi_E(\lambda)\, V(\lambda)\, d\lambda, \tag{33}$$

where Φ_E is the spectral power of the source (W), $K_{cd} := 683$ lm/W and the function V is the photopic spectral luminous efficiency, which gives the spectral response of the human eye to various wavelengths of light. This function was adopted by the CIE as the standard in 1924 and is still used today even though modifications have been suggested. A non-linear regression fit to the experimental data yields the approximation (see Figure 174 (right)):

$$V(\lambda) = 1.019\, e^{-285.4(\lambda - 0.559)^2} \tag{34}$$

where λ is the wavelength in micrometer.

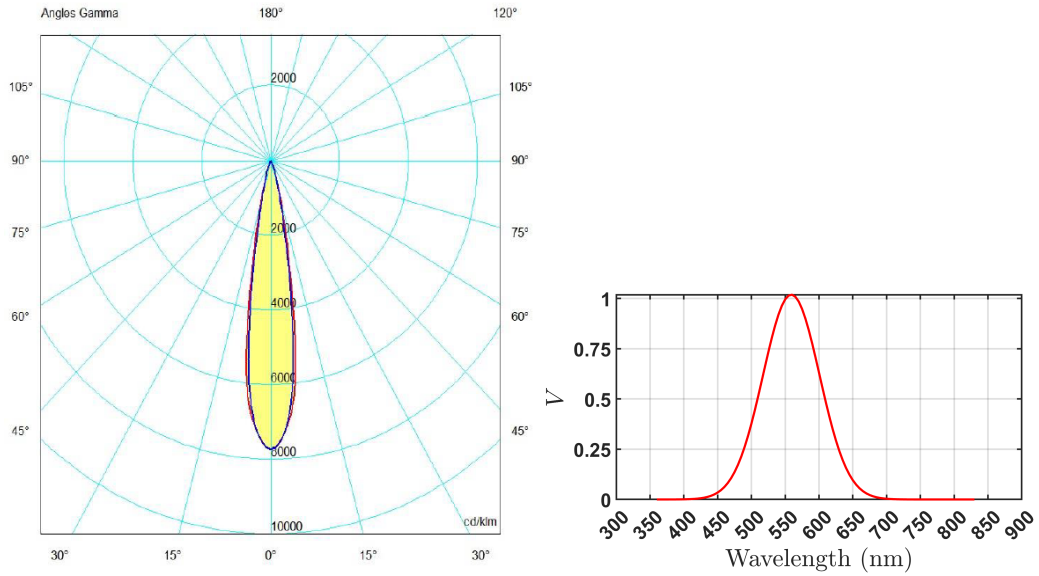


Figure 174: An example of an IES profile (left) and tThe luminous efficiency function V function of the wavelengths (right).

The IES characterization is largely used for urban lighting design for example but it is necessary to extend this modelling for applications operating beyond the visible range like LIDAR, SWIR cameras or LWIR cameras. The most complete characterization to be used is the spectral specific intensity defined in (31).

The modelling of RTE (1) for thermal imaging needs to consider sources in the LWIR range 8 μm - 14 μm. In this range, all surfaces emit radiations due to the black body radiation law of Planck:

$$I_\lambda(r,u) = \epsilon_\lambda \frac{2hc^2}{\lambda^5} \frac{1}{\exp\left(\frac{hc}{\lambda k_B T(r)}\right) - 1}, \tag{35}$$

where $h = 6.62607015 \times 10^{-34}$ J.Hz^{-1} is the Planck constant, $k_B = 1.380649 \times 10^{-23}$ J.K^{-1} is the Boltzmann constant, $c = 2.99792458 \times 10^8$ m.s^{-1} is the light speed, $T(r)$ is the temperature (K) at the point r and ϵ_λ (belonging in [0,1]) is the emissivity of the surface at the wavelength λ. We represent in Figure 175 this law for the temperatures 5600 K (sun), -20°C, 10°C and 100°C.

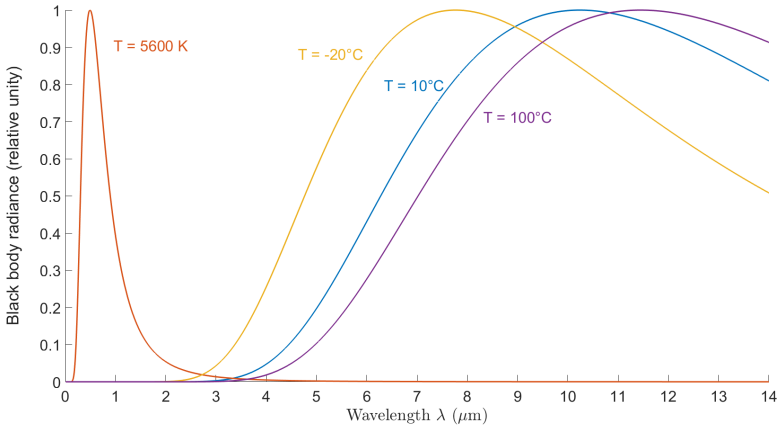


Figure 175: The Planck law for the temperatures 5600 K (sun), -20°C, 10°C and 100°.

We then see that all body with surface temperature between -20°C and 100°C emits in the thermal range. This law has to be into account for the thermal radiation of the atmosphere (including particle like water drop or dust or smoke) by introducing in equation (1)a source term q_λ acting in the whole space and for which, under the assumption of a thermodynamic local equilibrium [**?**], we use the following emissivity (Kirshoff law):

$$\epsilon_\lambda = \beta_\lambda - \sigma_\lambda. \tag{36}$$

The emitting thermal radiation of objects is modeled thanks to the Planck equation (35) with their surface temperature and an emissivity value depending on the surface characteristics. Typically emissivity is around 0.9 for a large variety of road objects. A wet surface has higher emissivity than a dry surface. Concerning the temperatures, vertical road signs can be assumed to have the same temperature than the air due to the thiny tickness of the material but it is not the case for the road surface. For this latter case, it is possible to introduce a specific road thermodynamic model to predict the road surface temperature from air temperature, wind velocity, rainfall rate, snowfall rate and road surface optical parameter (emissivity, albedo, convection exchange coefficient). Considering a road described by its vertical space variable x ($x = 0$ for the surface), the transient heat equation is:

$$C_i \frac{\partial \theta}{\partial t}(x, y, t) - \lambda_i \Delta\theta(x, y, t) = 0, \quad i \in \{1, 2, 3, 4\}, \tag{37}$$

where C_i and λ_i denote specific heat and thermal conductivity of layer i. The road surface boundary condition expresses the energy balance between road and atmosphere:

$$\lambda_1 \frac{\partial \theta}{\partial y}(x, 0, t) = \sigma\varepsilon(t)\theta^4(x, 0, t) + H_v(t)(\theta(x, 0, t) - \theta_a(t)) - R_{atm}(t) - (1 - A(t))R_g(t) + L_f I(t) \tag{38}$$

where the following notations are used:

ε, A : emissivity and albedo of the road surface,
σ : Stefan-Boltzmann constant (5.67×10^{-8} W/m^{2}K^4),
R_{atm}, R_g : atmospheric and global radiation (W/m^2),
θ_a : air temperature (K),
H_v : convection heat transfer coefficient (W/m^{2}K),
I : snow rate (mm.s^{-1}),
L_f : latent heat of fusion of the ice per kg (J.kg^{-1}).

We refer to [132, 133, 134, 135, 136] for more details.

3.9.5 Material effects models

The optical reflection characteristics of road environment materials (including wet materials) have to be considered in the simulators. For the road surfaces, the CIE has defined some optical parameters for the photometric characterisation of a road surface [137]. The surface of a pavement is classified according to its reflection properties. The most characteristic parameter is the luminance coefficient q, which is the ratio between the luminance L (cd/m^2) received by an observer (typically the driver) and the illuminance E (lux) which is incident on the surface: $q = L/E$. Nowadays a derived parameter is used: the r parameter (cd/m^2/lux) leading to r-tables and defined by:

$$r = q \cos^3 \varepsilon, \tag{39}$$

where ε is the angle between the incident light and the normal of the road surface. From this coefficient r, two coefficients Q_0 and S_1 are defined to quantify respectively the lightness and the specularity of the surface. These coefficients are widely used to characterize surface pavement in the visible range and for human vision. Some studies have shown the impact of a water film on the coefficient S_1. We represent in Figure 176 the effect of water on the coefficient S_1 for different pavements [33]. We can observe a peak of specularity during the drying of the pavement surface. The rain on road surface has then an impact and this kind of experimental studies are important to derive models. We mention the ongoing French national project REFLECTIVITY whose objective is to derive such models.

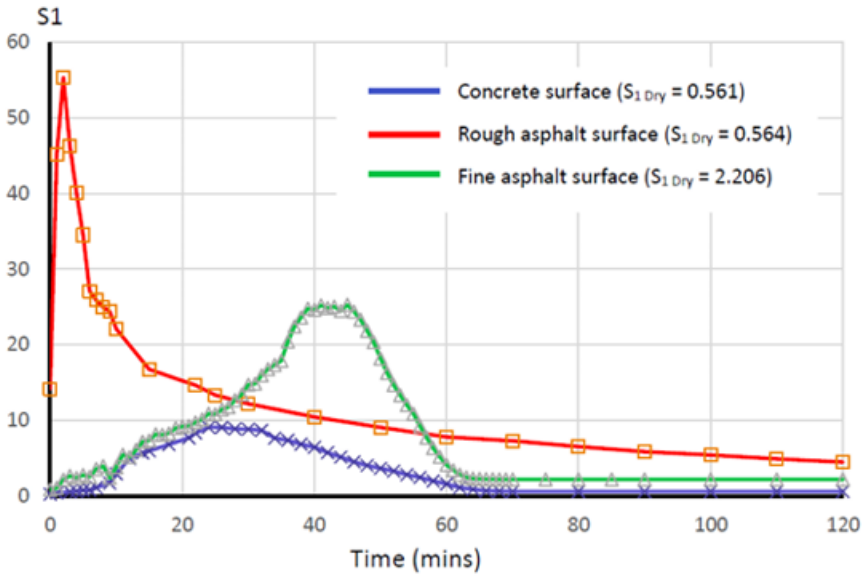


Figure 176: The coefficient S_1 for different pavements after an inundated wet condition [33].

To take into account wavelength beyond the visible range, it is necessary to characterise the reflection of objects in the SWIR range. We mention here the works done in [34] for a study on the spectral reflectance characterisation of the road environment. Thanks to spectroscopy-based measurements, different objects were characterised as shown in Figure 180.

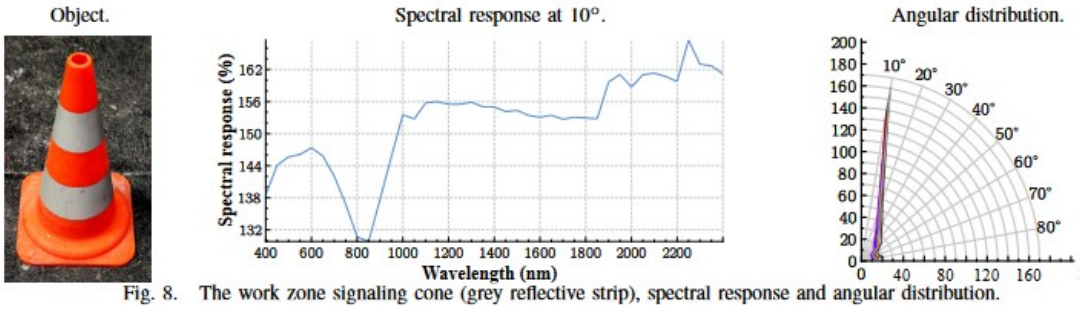


Fig. 8. The work zone signaling cone (grey reflective strip), spectral response and angular distribution.

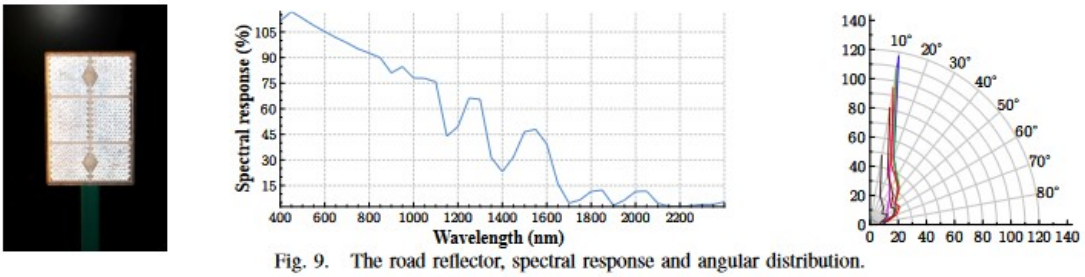


Fig. 9. The road reflector, spectral response and angular distribution.

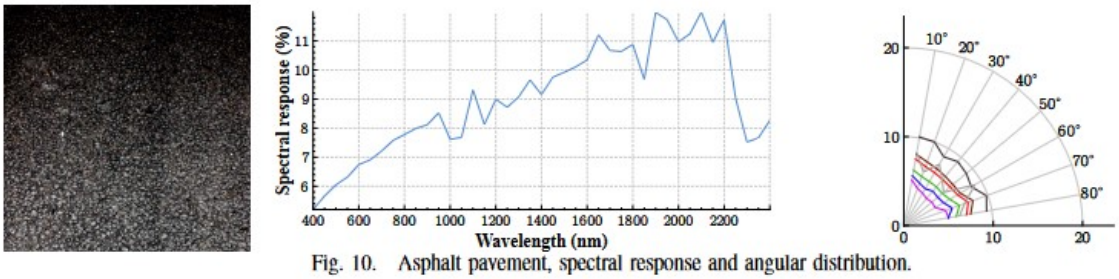


Fig. 10. Asphalt pavement, spectral response and angular distribution.

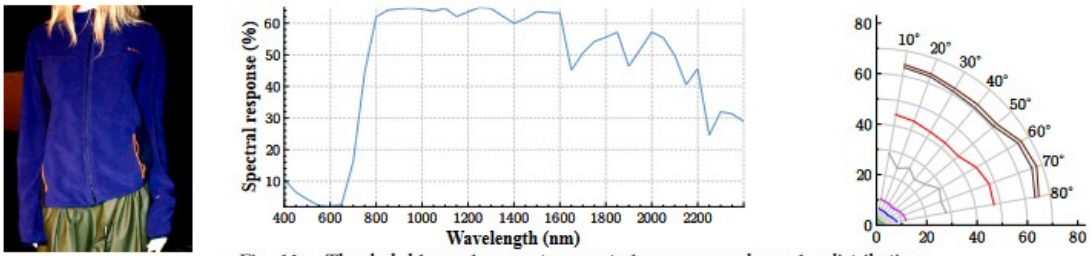


Fig. 11. The dark blue polar sweater, spectral response and angular distribution.

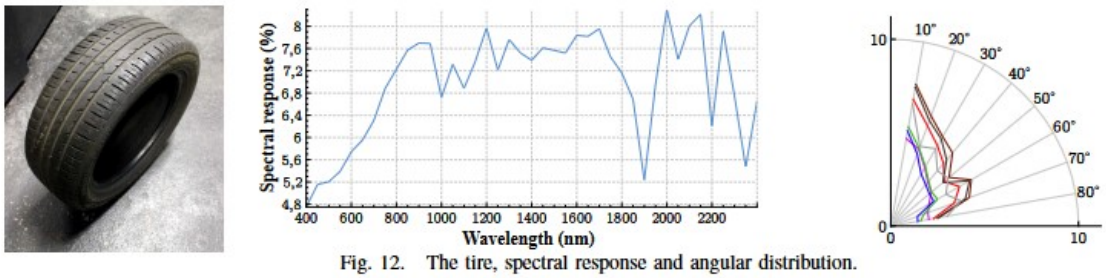


Fig. 12. The tire, spectral response and angular distribution.

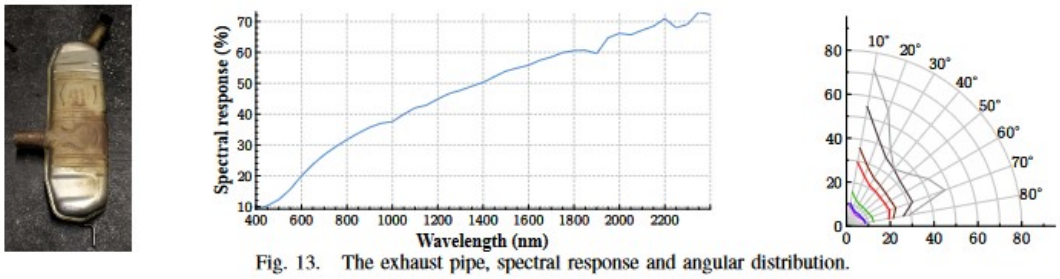


Fig. 13. The exhaust pipe, spectral response and angular distribution.

Figure 177: Spectral response and angular distribution reflection for different road objects (Source: CEREMA [34]).

An example of images generated by the simulator Pro-Sivic are done in Figure 179. We can observe the effect of rain on the windscreen and on the road surface (specularity). Moreover, in the figure 194, it is possible to evaluate the capacities of the existing AI-based systems involving GAN, auto-encoder, Transformers to significantly improve the quality of synthetic images even in degraded conditions with rain and fog.

Figure 178: Rain and snow effect in simulated images of Pro-Sivic (specular reflection on wet materials, drops on windscreens and snow on roads) - (source UGE)

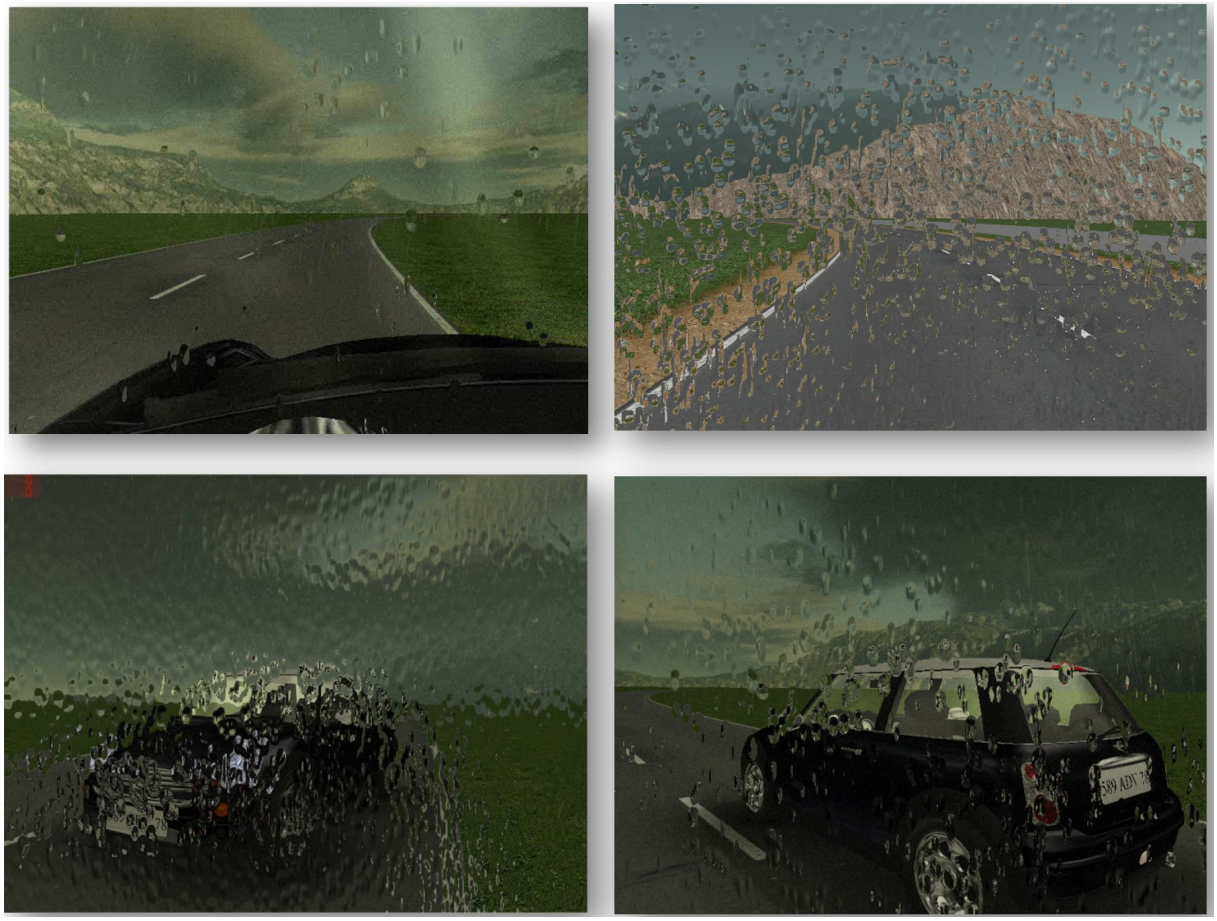


Figure 179: Rain drops effect in Pro-SIVIC on the camera optical part (source UGE).

3.10 Evaluation and assessment

In the context of autonomous vehicle (AV) development, it's crucial to differentiate between simulation and simulator. Simulation refers to the process of modeling real-world scenarios and interactions within a virtual environment, allowing for the testing and validation of AV systems without physical prototypes. On the other hand, a simulator is the tool or platform used to execute simulations, providing a virtual environment where AV algorithms can be tested and evaluated. Simulator evaluation is covered in Deliverable 2.7, here we look at how simulation can be used in an evaluation and certification process.

As the reliance on simulation grows in AV development, there arises a pressing need to establish rules and guidelines governing its usage, particularly concerning the homologation of embedded AI in AVs. Homologation, the process of certifying AVs for public road use, requires rigorous testing and validation of AI algorithms that govern vehicle behavior. While simulation offers benefits such as cost-effectiveness and scalability, its efficacy in ensuring the safety and reliability of AI remains contingent upon the rules and standards imposed around its usage.

Therefore, it becomes imperative to establish clear rules and regulations outlining the criteria for simulation-based homologation of AI in AVs. These rules should encompass factors such as simulation fidelity, scenario diversity, validation methodologies, and transparency in simulation results. By defining such rules, regulatory bodies and industry stakeholders can ensure that simulation serves as a robust tool for homologating AI in AVs, contributing to the advancement and safe deployment of autonomous technology on our roads.

3.10.1 Evaluation and Validation

The evaluation and validation of autonomous vehicles involve a comprehensive process that encompasses various stakeholders (see 2.1.1.1), each playing a crucial role in ensuring the safety, reliability, and compliance of these vehicles. This section provides the significance of simulation in the overall evaluation and validation process, highlighting its importance in ensuring the safety and reliability of autonomous vehicle systems.

There are some issues about the safety barrier validation standards for simulation. This highlights the importance of adopting rigorous methodologies akin to those used in validating safety barriers for real-world applications. Just as safety barriers undergo rigorous risk assessment to ensure their effectiveness in mitigating potential hazards, simulation tools must also undergo thorough risk analysis. This involves identifying potential failure modes, assessing their likelihood and consequences, and implementing measures to mitigate risks. By aligning simulation validation with safety barrier standards, manufacturers can ensure that simulation tools effectively address safety concerns in AV development. Safety barriers are for example subjected to various performance tests to evaluate their ability to withstand impact forces and protect against vehicle collisions. Similarly, simulation tools must undergo performance testing to assess their accuracy, reliability, and robustness in replicating real-world scenarios. This includes validating simulation results against empirical data and conducting sensitivity analyses to identify potential weaknesses. By adhering to performance testing standards, manufacturers can enhance confidence in the predictive capabilities of simulation tools. Safety barrier validation standards often involve verifying compliance with regulatory requirements and industry standards. Similarly, simulation tools must comply with relevant regulations and guidelines governing AV development and testing. This includes ensuring that simulation methodologies align with established best practices, safety standards, and ethical considerations. By aligning

with compliance verification standards, manufacturers can demonstrate the integrity and reliability of simulation tools for AV type-approval. Safety barrier validation standards emphasise the importance of continuous improvement and adaptation to changing conditions. Similarly, simulation tools must evolve to address emerging challenges and advancements in AV technology. This requires ongoing research and development to enhance simulation accuracy, incorporate new features, and address emerging safety concerns. By embracing a culture of continuous improvement, manufacturers can ensure that simulation tools remain effective and relevant in an evolving AV landscape.

Another safety issue is the access to expected behaviour and undesirable behaviour. This is vital for ensuring the safety and security of autonomous vehicle (AV) systems and, unfortunately, simulation is not always the best way to solve this problem. AV systems are designed to operate in a predictable and reliable manner, adhering to predefined rules, regulations, and operational design domains (ODDs). Access to expected behaviour involves defining and validating the behaviours and responses that AVs are expected to exhibit under normal operating conditions. This includes adherence to traffic laws, safe driving practices, and appropriate responses to environmental stimuli. By ensuring that AVs consistently demonstrate expected behaviour, it is possible in theory to enhance safety and build trust among passengers, pedestrians, and other road users. But despite rigorous design and testing, AVs may encounter unforeseen scenarios or edge cases that challenge their ability to operate safely. Access to undesirable behaviour involves identifying and mitigating instances where AVs deviate from expected behaviour or exhibit unsafe actions. This includes detecting and addressing potential failure modes, edge cases, and system vulnerabilities that could compromise safety or security. By proactively addressing undesirable behaviour, engineers can reduce the risk of accidents, collisions, or malicious exploitation of AV systems. So access to expected and undesirable behaviour is essential for safety and security assurance in AV systems. It enables to validate the robustness and reliability of AV algorithms, perception systems, and decision-making processes under diverse scenarios and conditions. By systematically testing and evaluating expected and undesirable behaviours, It can be possible to identify potential risks, weaknesses, and vulnerabilities in AV systems, allowing for timely remediation and enhancement of safety and security measures. Access to expected and undesirable behaviour requires continuous monitoring and improvement throughout the development life-cycle of AV systems. This involves leveraging data analytics, machine learning, and real-world feedback to refine AV algorithms, update operational policies, and enhance system resilience against emerging threats or challenges. By embracing a culture of continuous improvement, engineers can ensure that AVs evolve to meet evolving safety and security standards, maintaining public trust and confidence in autonomous technology.

3.10.1.1 Use of Simulation in the Design Process

Simulation plays a crucial role in the design process of autonomous vehicles, offering a virtual environment to test and validate various aspects of the system before physical prototypes are built. By utilising simulation, manufacturers can accelerate the development cycle, reduce costs, and ensure the safety and reliability of autonomous systems. Here are some key areas where simulation is utilised in the design process:

1. **Algorithm Development**: Simulation allows engineers to develop and refine algorithms for perception, decision-making, and control in a controlled environment. By simulating

different scenarios and edge cases, developers can iteratively improve algorithm performance and robustness.

2. **Sensor Fusion and Perception**: Autonomous vehicles rely on sensor data from cameras, LIDARs, radars, and other sensors to perceive their surroundings. Simulation enables testing of sensor fusion algorithms and perception systems under various conditions, including different lighting, weather, and traffic scenarios.

3. **Vehicle Dynamics and Control**: Simulation allows engineers to model and simulate vehicle dynamics, including acceleration, braking, and steering behaviour. By accurately representing the vehicle's physics in simulation, developers can fine-tune control algorithms and optimize vehicle performance.

4. **Human-Machine Interaction**: Simulation is used to evaluate human-machine interaction (HMI) systems within autonomous vehicles, including user interfaces, displays, and communication systems. Designers can assess usability, accessibility, and safety aspects of HMI systems through simulated user interactions.

5. **Scenario Testing and Validation**: Simulation enables the creation and testing of diverse driving scenarios, including normal driving conditions, emergency manoeuvres, and edge cases. By simulating thousands of scenarios, developers can validate the system's behaviour and identify potential failure modes or safety risks.

6. **Robustness and Fault Tolerance**: Simulation is used to assess the robustness and fault tolerance of autonomous systems against failures or unexpected events. By introducing faults or perturbations in simulation, developers can evaluate the system's ability to detect and recover from errors autonomously.

7. **Regulatory Compliance**: Simulation is instrumental in demonstrating regulatory compliance and safety certification of autonomous vehicles. Manufacturers can use simulation results to provide evidence of system performance and safety to regulatory authorities.

3.10.1.2 Perception AI

Evaluation of perception AI algorithms is critical in ensuring the safe and reliable operation of autonomous vehicles. This assessment primarily focuses on the algorithm's ability to accurately interpret sensor data and detect objects in the vehicle's environment. Several key factors are considered in this evaluation process.

The first is to isolate perceptual AI and test it directly on the basis of data (data which is supposed to be representative of what is captured before processing by the sensors) in relation to given tasks. This is a classic approach to AI evaluation that is not specific to the automotive industry. The second is to simulate the perception organs in a simulator, and to simulate the vehicle in a complementary way.

For the first type of evaluation, before the process begins, it's essential to have a comprehensive dataset representing a wide range of scenarios. This dataset should include various environmental conditions, such as different times of day, weather conditions, and complex traffic situations. Each data point in the dataset should be meticulously annotated (see PRISSMA

deliverable 1.6) to provide ground truth information about the objects and their characteristics. The quality and diversity of the dataset significantly impact the algorithm's performance. In the second case, it will be important to master the scenario generation process to cover the same thing. In fact the perception AI algorithm's evaluation must consider a variety of environmental factors, such as varying light conditions (day, night, and different weather conditions), occlusions (e.g., by other vehicles or objects), and challenging scenarios (e.g., construction sites or crowded urban areas). Robustness in adverse conditions is essential for the safe operation of autonomous vehicles. To calibrate the simulator, and in particular the models, to this variety of environmental factors, it will be important to rely on the ground truth derived from tests on real environments. While simulation provides a controlled environment for testing, real-world testing is crucial for validating the algorithm's performance. Real-world testing allows for the validation of the algorithm's behavior in unpredictable and dynamic environments. It provides insights into how the algorithm performs under real-world conditions and its ability to adapt to new and unforeseen situations. Theses insights are perfect for the calibration of a simulation framework.

Another point of interest is the sensor fusion. Autonomous vehicles rely on a combination of sensors, including LiDAR, radar, cameras, and ultrasonic sensors. The evaluation process must consider how well perception AI algorithms integrate and interpret data from these different sensors. Sensor fusion techniques play a vital role in combining information to generate a comprehensive understanding of the vehicle's surroundings.

It's important to distinguish the tasks that AI must perform for perception, the test protocol must be adapted to each task to be evaluated. Some common tasks:

- Object detection is a fundamental task for perception AI algorithms. The evaluation process should assess the algorithm's ability to accurately detect various objects, such as vehicles, pedestrians, cyclists, and static obstacles.
- Semantic segmentation involves classifying each pixel in an image to a specific object class. This capability is crucial for understanding the surrounding environment accurately. During the evaluation, the algorithm's performance in semantic segmentation should be assessed, ensuring that it can differentiate between different types of objects and the background.
- Instance segmentation takes semantic segmentation one step further by distinguishing between individual objects of the same class. This is particularly important in complex scenarios where multiple objects of the same class are present. The evaluation process should determine the algorithm's ability to perform accurate instance segmentation.
- Reliable metrics must be associated with each of these tasks. Several classical performance metrics are used to evaluate perception AI algorithms, including:
  - Accuracy: The overall correctness of object detection and classification.
  - Precision: The ratio of true positive detections to the sum of true positives and false positives.
  - Recall: The ratio of true positive detections to the sum of true positives and false negatives.

- F1 Score: The harmonic mean of precision and recall, providing a balance between the two.
- Intersection over Union (IoU): Measures the overlap between the predicted bounding box and the ground truth.
- etc (see PRISSMA deliverable 2.3)

When it comes to choosing a simulation framework for this kind of evaluation, we recommend systems with very good sensor models (such as Pro-SiVIC, SCANeR, ANSYS Tool Chain, ...), or at least a model for each on-board sensor used for perception. But also with a simulator able of generating realistic weather conditions like CARLA, Pro-SIVIC, SCANeR, AirSIM, AutonoVi-SIM... (the best being to be able to simulate the interaction between weather degradation and sensor performance). All this while keeping rendering performance as realistic as possible (and if possible without exploding efficiency in terms of cost) by avoiding rather poor engines like Ogre3D for example. In a great part of the simulation environment, the used graphical engine is either Unreal Engine v5, or Unity. Pro-SiVIC has it own extended graphical engine called mgEngine (build from OpenGL library). The section 3.4.1 provides more explanation and interesting references about the current state of the simulation environments.

3.10.1.3 Decision AI

One crucial aspect of evaluating and certifying AI embedded in autonomous vehicles is the validation of decision-making AI algorithms. This involves assessing their ability to generate safe and efficient driving maneuvers across diverse scenarios. Simulation plays a pivotal role in this process. Through simulation, various scenarios, including both routine and edge cases, can be recreated and tested. The validation process typically includes the following steps:

1. **Scenario Generation**: Diverse driving scenarios are created, including normal driving conditions, complex urban environments, adverse weather, and unforeseen situations.
2. **Testing AI Performance**: The AI algorithms are then subjected to these scenarios to evaluate their decision-making processes. Performance metrics such as response time, adherence to traffic laws, and the ability to avoid collisions are measured.
3. **Iterative Improvement**: Based on the results obtained, the algorithms are refined and optimised. This iterative process helps enhance the AI's capability to make decisions in various situations.
4. **Regulatory Compliance**: The final step involves ensuring that the AI algorithms comply with the regulatory standards set for autonomous vehicles. Certification is obtained only when the algorithms consistently demonstrate safe and efficient decision-making abilities across diverse scenarios.

To accomplish this, we can utilise various advanced simulation frameworks, such as CARLA (Car Learning to Act), Gazebo, LGSVL, Pre-Scan, Pro-sivic, IPG Carmaker, SCANeR. These simulation platforms offer different features and advantages.

The special feature of AI evaluation of decision-making is its centrality within systems of systems. In fact, they are not directly linked to the initial information gathering (perception task), but only to the analysis that may follow (often after an information fusion level). Nor are

they the final output level, which falls to the actuator control part. This subsystem is therefore highly dependent on the inputs coming from perception and the applications of their outputs in the actuator control level, making the isolation of this block all the more complex. Classically, there are two approaches to successfully evaluating this module despite this difficulty. The first is to make the perception part ”perfect” but not realistic (bypassing realistic sensor modules, for example, and sending exact scenario information from the simulator) and the control part (reduced to very simple models such as the bicycle model, and not AI-based). This is particularly useful in the AI design phase. The second, more realistic approach is to look at the vehicle as a whole (with each module representing the most realistic subsystems possible) and evaluate on the vehicle's global metrics, taking the risk of not knowing whether the potential wrong (or right) decision comes from the decision module. Of course, we can also monitor the inputs and outputs of the decision module to try and provide more information. This requires information systems that provide easy access to the vehicle's internal information. You should therefore favour simulators that provide the best possible access to these internal data (CARLA, Gazebo, PreScan, SCANeR, IPG, Pro-SiVIC...) and avoid those with more partial access (RoadView, Vissim, Sumo, etc.) and more focus on micro simulation of traffic.

The advantage, however, is that in absolute terms, all you need is input information that is representative of what a perception module could provide. So for the decision module evaluation, aspects such as sensor module viability, rendering performance, physics engine, weather simulation... will not be priorities if you can decouple it from the perception and control aspect. On the other hand, you'll need to achieve maximum scene diversity and avoid simulators that are rather poor in this respect (AutonoVi-Sim, AirSim, Udacity...). On the whole, you'll need to focus on the Scenario Generation aspect.

3.10.1.4 Control AI

AI-assisted actuator control is undoubtedly in the minority when compared with the use of AI for perception. There are, however, current examples, such as the adaptation of night lighting to road conditions and a number of new applications, such as dynamic suspension adaptation, are beginning to appear.

In this particular context, the main simulator features to be favoured are the physical models linked to vehicle mechanics and the physical engine. The actuator control algorithms work directly on the mechanical parts of the vehicle responsible for acceleration, braking, suspension, vehicle stability, trajectory adherence etc. These are the aspects that will be important to model correctly in the simulation system.

When choosing a simulation platform, it is therefore important to ensure that the mechanical functionality controlled electronically by the AI is modelled in a representative way, and this will be the main aspect of platform choice.

3.10.2 Certification - Technical Service & OQA

3.10.2.1 Audit

Audit constitutes a fundamental aspect of the evaluation and certification process for autonomous vehicles. The guidelines for ADS validations (NATM [138]) defined at the UNECE specifies to apply a multi-pillar approach, and audit is one of those pillars with simulation, track testing, real world testing and in-service monitoring. Moreover, simulation is not yet a test

method adopted for the technical service to test the system for the type approval. But simulation and all virtual testing methods are validation elements that a technical service can audit during the type-approval process. The regulation UE 2022-1426 [139] for ADS type-approval defines the principles for credibility assessment for using virtual tool-chain in ADS validation in part 4 of annex III. In fact, since AI-based systems are generally black-boxed at the time of certification, it would be impossible to certify them without auditing the manufacturer's process. It involves a comprehensive examination of various facets of the system's creation.

Simulation and Modelling Audit focuses on assessing the capability, accuracy, correctness, usability, the fit for purpose [139], fidelity, and reliability of simulation tools used in the development and testing of autonomous vehicles. Engineers scrutinise the simulation environment, verifying its ability to replicate real-world scenarios accurately. Demonstration of simulator qualification through audits and real-world validation is a critical aspect of ensuring the reliability and accuracy of simulation tools used in the development and testing of autonomous vehicles (AVs). Audits play a crucial role in assessing the quality and reliability of simulation tools. Through comprehensive audits, engineers evaluate various aspects of the simulator, including its fidelity, accuracy, and adherence to industry standards. By conducting audits, manufacturers can ensure that simulators meet the necessary criteria for accurately representing real-world scenarios, thereby enhancing confidence in the results generated by the simulation. While simulation provides a controlled environment for testing, real-world validation is indispensable for confirming the accuracy and reliability of simulation results. Real-world validation involves comparing the performance of AVs in simulated scenarios with their performance in actual driving conditions. This validation process helps identify any discrepancies between simulation and reality, allowing engineers to refine simulation models and improve their predictive capabilities. The demonstration of simulator qualification involves showcasing the results of audits and real-world validation to stakeholders, including regulatory authorities, industry partners, and the public. By providing evidence of simulator reliability and accuracy through audits and real-world validation, manufacturers can instil confidence in the simulation tools used for AV development and testing. The process of demonstrating simulator qualification is not a one-time effort but rather an ongoing endeavour. Manufacturers must continuously monitor and improve simulation tools based on feedback from audits and real-world validation. This iterative process ensures that simulators remain up-to-date and capable of accurately representing evolving real-world scenarios encountered by AVs. Additionally, the audit evaluates the modelling techniques employed, ensuring they capture the nuances of vehicle behaviour, sensor interactions, and environmental dynamics faithfully. It can apply a virtual tools assessment if the autonomous vehicle development relies heavily on virtual tools, including software frameworks, development environments, and simulation platforms. During the audit, these virtual tools undergo rigorous scrutiny to assess their functionality, usability, and compatibility with the overall system architecture. Engineers evaluate the capabilities of these tools in facilitating design, testing, and validation tasks, ensuring they meet the requirements of the development process effectively. Moreover, the regulation EU 2022 1426 [139] also requires that the engineer team experience shall be audited. In the future, specific training programs should be created to certify the ability of engineers to use simulation tools with efficiency and rigour.

Auditing databases for machine learning algorithms constitutes a critical step in the evaluation and certification of autonomous systems. This process aims to ensure the integrity, quality, and relevance of the data used to train and feed the machine learning models that underpin decisions made by autonomous vehicles. The audit of databases begins with an assessment of the

quality of available data. This involves verifying the accuracy, reliability, and consistency of the data, as well as identifying any potential biases that may affect the model's performance. Raw data may require preprocessing before being used to train a machine learning model. The audit examines the preprocessing methods used, such as imputation of missing values, normalisation of data, and handling of outliers, to ensure they preserve the integrity and representativeness of the data. The audit also evaluates the feature selection process, which involves identifying relevant variables for model training. This includes analysing the importance of features, dimensionality reduction, and validating the adequacy of selected features relative to the problem to be solved. To assess the performance of the machine learning model, the audit typically includes cross-validation of the data. This involves dividing the dataset into training and test sets, then comparing the model's predictions with actual values to evaluate its accuracy and generalisation. The audit also examines the interpretability of the machine learning model, i.e., its ability to explain its predictions in an understandable manner. This may include analysing the weights assigned to features, visualising the model's decisions, and identifying factors that influence its predictions. Finally, auditing databases for machine learning algorithms focuses on detecting and managing potential biases in the data and models. This includes identifying sources of bias, implementing bias mitigation techniques, and validating the fairness and inclusivity of the model across different demographic groups.

The audit process also entails verifying compliance with regulatory standards, industry best practices, and internal guidelines. Engineers review documentation, procedures, and processes to ensure alignment with applicable regulations and safety standards. Additionally, the audit assesses the adequacy of risk management practices, contingency plans, and mitigation strategies to address potential safety and security concerns.

A critical aspect of the audit is the evaluation of system performance against predefined criteria and specifications. Engineers conduct comprehensive tests and analyses to assess the system's capabilities, reliability, and robustness under various operating conditions. Performance metrics such as accuracy, response time, and failure rates are scrutinised to ensure they meet the required standards for safety and functionality.

Throughout the audit process, meticulous documentation review is conducted to ensure completeness, accuracy, and traceability of information. Engineers examine technical documentation, test reports, validation results, and compliance records to verify compliance with documentation requirements. Additionally, documentation related to system architecture, design decisions, and change management processes is assessed to ensure transparency and accountability in the development process.

All aspects of auditing will be developed in WP6 deliverables, but it was important to recall the main principles here

3.10.2.2 Testing for Certification and type-approval

In the pursuit of type-approval of autonomous vehicles, the process of evaluation and validation relies heavily on testing. Simulation testing is an essential pillar of this work for several fundamental reasons:

- **Scenario Variety**: Autonomous vehicles must safely navigate through a multitude of situations and environments. Simulation allows the generation and testing of a wide variety

of scenarios, including rare or extreme cases that may be difficult or dangerous to replicate in real-world tests. This ensures that AVs are prepared to handle a diverse range of driving conditions, which is crucial for obtaining certification or type-approval.

- **Cost and Safety**: Real-world road tests or tests on tracks are expensive, time-consuming, and can pose safety risks for operators and other road users. Simulation reduces these costs and risks by providing a safe and controlled virtual environment to assess AV performance. This allows manufacturers to conduct a much higher number of tests and iterations, thereby speeding up the validation process.

- **Flexibility and Repeatability**: Simulation offers unmatched flexibility and repeatability compared to real-world and track tests. Engineers can easily modify test parameters and conditions, repeat scenarios precisely, and compare AV performances (and especially AI behaviour) under identical conditions. This enables quick identification and resolution of performance and safety issues, helping to ensure that AVs meet the strictest Requirements.

- **Safety System Validation**: AVs must be equipped with robust safety systems to detect and respond to hazardous situations on the road. Simulation allows for the validation of these safety systems in representative virtual environments, testing their ability to detect obstacles, avoid collisions, and make safe driving decisions. This ensures that AVs meet the required safety standards for type-approval.

But to implement these simulation tests, the following aspects need to be mastered:

- **Appropriate sampling** : Testing sampling involves selecting representative scenarios and conditions to evaluate the performance of autonomous vehicles within its ODD and at the ODD boundaries. It is essential to ensure that the selected samples cover a diverse range of situations encountered in real-world driving scenarios. Simulation provides a valuable tool for generating and refining testing samples, allowing engineers to simulate rare or extreme events that may be challenging to encounter during real-world testing. Mastering sampling of virtual testing is also an efficient way to prove coverage of all ODD and OEDR conditions.

- **type-approval test cases** Systematic tests are designed to evaluate specific aspects of autonomous vehicle performance in a controlled and repeatable manner. These tests aim to assess the vehicle's capabilities in various scenarios, such as lane-keeping, obstacle avoidance, and emergency braking. By systematically varying parameters and conditions, engineers can thoroughly evaluate the robustness and reliability of autonomous systems.

A rerunning simulation protocol is also needed. Rerunning simulations is often necessary during the homologation process to validate the performance of autonomous vehicles under different conditions. As testing samples evolve and new scenarios emerge, engineers may need to rerun simulations to ensure that vehicles meet regulatory standards and safety requirements. This iterative process of simulation and validation plays a crucial role in achieving type-approval for autonomous vehicles.

And the most important thing is to keep a link with real tests (whether on real roads or on the track). It's important to remember that, despite the strengths of simulation, the representativeness of tests can never be guaranteed (only real-life tests can), and you'll always need to keep some ground truth to calibrate the simulator modules. Simulation (especially HIL and

VIL) must serve as a bridge between virtual and real-world testing environments, facilitating the transition from simulated scenarios to real-world tests. While simulation provides a controlled environment for testing and validation, real-world tests offer insights into how vehicles perform in dynamic and unpredictable conditions. The correlation between simulation and real-world tests ensures that findings from simulated scenarios can be validated and verified in real-world driving scenarios. Talking about the different levels and types of simulation, there are some concerns regarding the absence of reference frameworks for Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), Hardware-in-the-Loop (HIL), and Vehicle-in-the-Loop (VIL) testing methodologies are paramount in the context of autonomous vehicle (AV) development. The lack of established reference frameworks poses several challenges for AV developers:

- **Standardisation**: Without reference frameworks, there is a lack of standardised methodologies and practices for conducting MIL/SIL/HIL/VIL testing. This inconsistency hampers interoperability and comparability across different testing environments, making it difficult to assess the reliability and effectiveness of AV systems.
- **Quality Assurance**: The absence of reference frameworks raises concerns about the quality and reliability of testing outcomes. Without standardised procedures and metrics, there is a risk of inconsistency and bias in test results, undermining confidence in the validity of testing outcomes and hindering the identification of potential issues or shortcomings in AV systems.
- **Efficiency and Effectiveness**: The lack of reference frameworks can lead to inefficiencies in testing processes, as developers may need to devise ad-hoc methodologies and criteria for conducting MIL/SIL/HIL/VIL tests. This can result in increased time and resources spent on testing activities, potentially delaying the development and deployment of AV systems.
- **Risk Management**: Inadequate reference frameworks may exacerbate safety and security risks associated with AV development. Without established best practices and guidelines, there is a heightened risk of overlooking critical safety considerations or vulnerabilities in AV systems, posing potential hazards to road users and infrastructure.

Addressing concerns regarding the lack of reference frameworks for MIL/SIL/HIL/VIL testing requires concerted efforts from industry stakeholders, regulatory bodies, and research institutions. Establishing standardized methodologies, metrics, and guidelines for conducting these tests is essential to ensure consistency, reliability, and effectiveness in AV development and validation. By promoting the adoption of reference frameworks like the PRISSMA one, the AV industry can mitigate risks, improve testing efficiency, and accelerate the safe deployment of autonomous technology on public roads.

A further point of interest in the use of simulation testing in the validation principle is the integration of sensors into Hardware-in-the-Loop (HIL) testing. HIL testing involves simulating the vehicle's electronic control unit (ECU) and surrounding environment in a controlled laboratory setting, allowing engineers to test the functionality and performance of AV components without the need for physical prototypes or real-world testing. Integrating sensors into HIL testing enables engineers to assess how AV systems interact with sensor inputs and make decisions based on real-world data(see deliverable 2.7). In fact HIL testing involves emulating the behavior of sensors such as LIDAR, radar, cameras, and ultrasonic sensors within the

simulated environment. This emulation replicates the sensor outputs that the AV's perception system would receive in real-world driving scenarios, allowing engineers to test how the AV responds to different sensor inputs. And also, AVs typically rely on sensor fusion algorithms to combine data from multiple sensors and generate a comprehensive understanding of the vehicle's surroundings. Integrating sensors into HIL testing enables engineers to validate sensor fusion algorithms by feeding simulated sensor data into the AV's perception system and evaluating its ability to accurately perceive the environment. HIL testing allows manufacturers to simulate a wide range of driving scenarios and environmental conditions, including challenging situations that may be difficult to replicate in real-world testing. By integrating sensors into HIL simulations, engineers can assess how AV systems respond to various scenarios, such as adverse weather conditions, complex traffic environments, and sensor occlusions. HIL testing also enables to inject sensor defaults and anomalies into the simulation to evaluate the AV's ability to detect and respond to sensor failures. Introducing sensor defaults enables to assess the robustness and reliability of the AV's perception system under adverse conditions. Integrating sensors into HIL testing facilitates the validation and verification of AV systems in a controlled and repeatable environment. Engineers can systematically test the performance of sensor-based algorithms, verify compliance with functional safety standards, and validate the accuracy of perception outputs against ground truth data.

In conclusion, testing sampling, systematic tests, the necessity of rerunning simulations for the validation to be audited for the type-approval process, and the link between simulation and real-world tests are essential aspects of evaluating and validating autonomous vehicles. By leveraging simulation tools and methodologies, engineers can generate representative testing samples, conduct systematic tests, and iteratively refine vehicle performance to meet type-approval requirements. This iterative process of simulation and validation ensures the safety, reliability, and regulatory compliance of autonomous vehicles.

3.10.2.3 Focus on Level 4 Shuttles

The validation and the safety demonstration of Level 4 autonomous shuttles poses unique challenges distinct from traditional vehicle type-approval processes. Level 4 autonomy signifies a significant advancement in autonomous vehicle technology, granting vehicles the capability to operate without human intervention within predefined operational domains. However, the complexities of homologating these shuttles extend beyond their technical capabilities and delve into various regulatory, safety, and operational considerations.

Type-approval of Level 4 shuttles necessitates a meticulous definition and validation of their Operational Design Domains (ODDs). These ODDs delineate the specific conditions and environments within which shuttles can safely operate autonomously. Ensuring the adequacy and accuracy of ODD definitions is crucial to guarantee safe and effective shuttle operation.

With Level 4 shuttles operating without human intervention, safety measures and redundancy systems become paramount. The type-approval process must rigorously evaluate the effectiveness of safety mechanisms and redundant systems to mitigate potential risks and ensure passenger safety in diverse scenarios.

Level 4 shuttles interact with various entities, including pedestrians, cyclists, and other road

users, particularly in shared spaces. The validation process must assess the vehicle's ability to detect and respond to the presence and behaviour of these entities, ensuring seamless and safe interactions in dynamic urban environments.

Type-approval of Level 4 shuttles necessitates compliance with stringent regulatory requirements governing autonomous vehicle operation. Demonstrating adherence to applicable standards and regulations is essential to ensure that shuttles meet legal and safety prerequisites for deployment on public roads or designated routes. Two regulations can be applied here:

- The European Regulation for Automated Driving systems type-approval, also called UE [139] ADS (2022-1426)
- The French project of regulation for the Automated urban shuttle type-approval (NAVURBAUT) [140]

Beyond technical capabilities, the success of Level 4 shuttles hinges on user experience and public acceptance. Factors such as passenger comfort, ease of use, and public perception play pivotal roles in determining the widespread adoption and integration of shuttle services into existing transportation systems. Addressing these aspects in the homologation process is vital to fostering user trust and confidence in autonomous shuttle technology.

In essence, homologating Level 4 shuttles requires a comprehensive approach that encompasses regulatory compliance, safety validation, operational testing, and user acceptance considerations. By addressing these challenges effectively, the type-approval process plays a pivotal role in facilitating the safe and seamless integration of Level 4 autonomous shuttles into urban mobility ecosystems.

4 Definition of a generic and adaptive PRISSMA simulation architecture (all partners)

4.1 Generic simulation framework proposed by AVS (POC 3)

The following is an example of a proposed generic and adaptive PRISSMA simulation architecture in the case of an integration with SCANeR Studios. The proposed architecture contains a generic AI System Bridge that can be based of C++, Python, Simulink or RTMAPS in order to host the AI system. This generic bridge would need to be modified case by case to cater to the required output/input data as needed by the AI System. If needed, the AI System Bridge can utilize Python to send and receive the requested data over the network via UDP/TCP. The Bridge also has the ability to access any data over the SCANeR Network and SHM data protocol, as well as send input data towards the same stream.

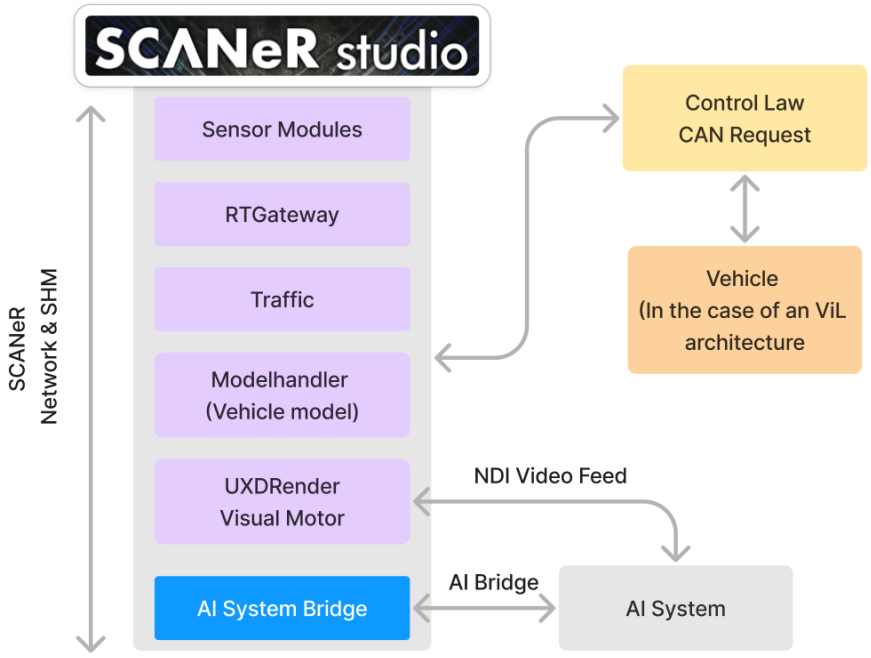


Figure 180: Proposed simulation architecture for SCANeR Studio (Source AVS)

It is also possible to recover the output video of UXDRender via an NDI video feed. The NDI video feed is capable of being broadcasted to any machine connected on the local network.

It is also possible to integrate a ViL simulation, by integrating the Control Law to SCANeR Studio. Evidently, this will also need to be catered and modified on a case by case situation, recovering data either via UDP, C++, Simulink or even RTMaps. With SCANeR Studio, the possibilities are endless, but planning would need to be done beforehand.

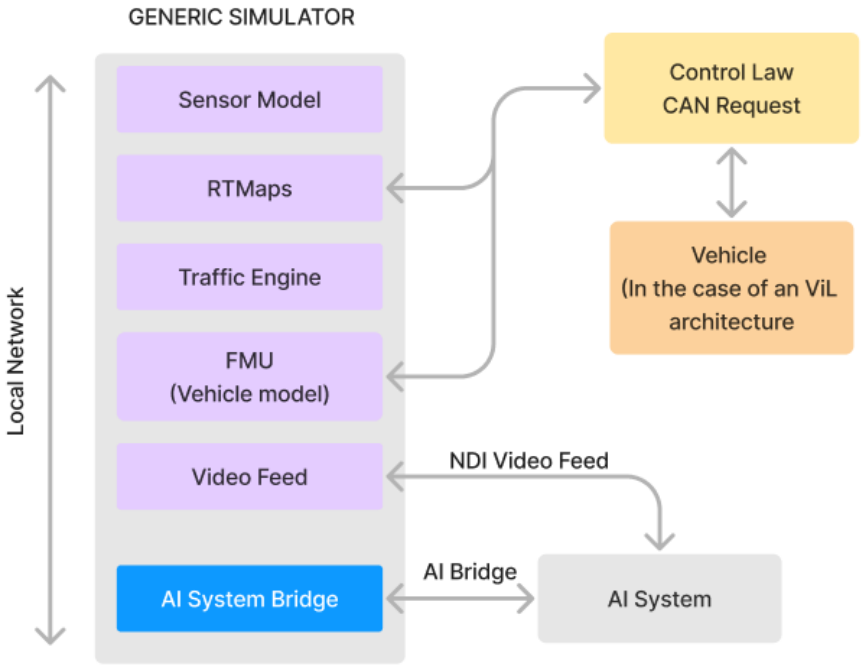


Figure 181: Proposed generic simulation architecture (Source AVS)

Figure 181 proposes a generic PRISSMA Architecture for the POC 3 with AVS and UTAC. The video feed would be extracted and fed into the AI system in order for it to make decisions. RTMaps as well as the vehicle model would be implemented into the the control law in the case of a ViL structure. RTMaps would ensure the real time working of the system and that there is no latency between the two architectures. All information from the Sensors would be conveyed via the local networks and recovered within the AI System bridge.

The analysis of the scenarios would be done externally, either by saving and exporting the simulation data or via checkpoints/time markers. This analysis would thus then be processed externally to see if the scenario is a success or not, or simply to just observe the behaviour of the AI System.

4.2 Generic simulation framework proposed by UGE (POC 1)

In PRISSMA, from the implementation made for the POC 1, it appeared a clear generic framework involving the different tools, models, and platforms defined in the previous sections of this deliverable. It appeared too that propose a generic simulation framework is very complex and needs to share this problem in sub-problems. From the global framework (non exhaustive) presented in the figure 182 and proposed by UGE, we have an overview of this complexity. in order to be more understandable, a simpler generic framework has been proposed in the figure 183. A part of this framework has been used and implemented in the POC 1 (figure 184). It is interesting to emphasise that similar functions and tools with different complexities appear in this framework. It is the case for the traffic generation and management. With low density traffic situation, it is recommended to use the models (vehicles, 2 wheels, and pedestrians) proposed in the simulation platform (in our case Pro-SIVIC). In this condition, each vehicle (generally limited to 15 till 20 vehicles) can be controlled by a specific decision-making and path planning algorithm implemented as a package or a DLL in the application environment (RTMaps). In

the case of a very dense traffic with a couple of hundred vehicles, it is recommended to use a dedicated simulator (Traffic generation tool in the generic framework). Of course, both can be used in same time. We favor the complex and dynamic model in Pro-SiVIC for the ego-vehicle, and the traffic simulator for populating the environment with simpler evolution models (i.e. IDM).

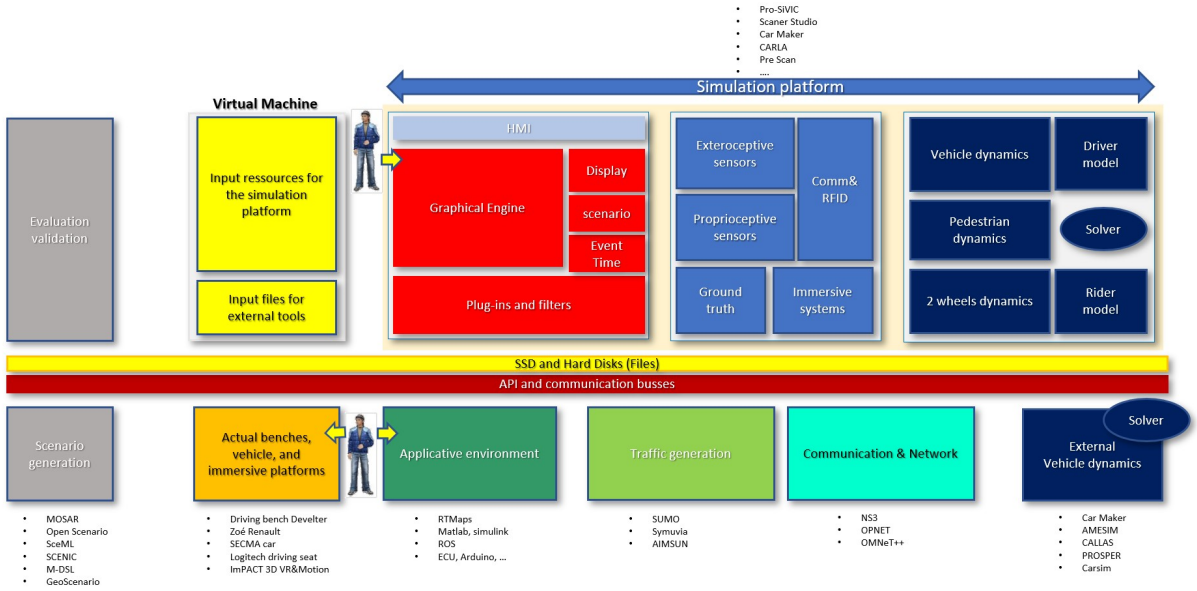


Figure 183: Simplified generic simulation framework proposed and developed by UGE (Source: UGE)

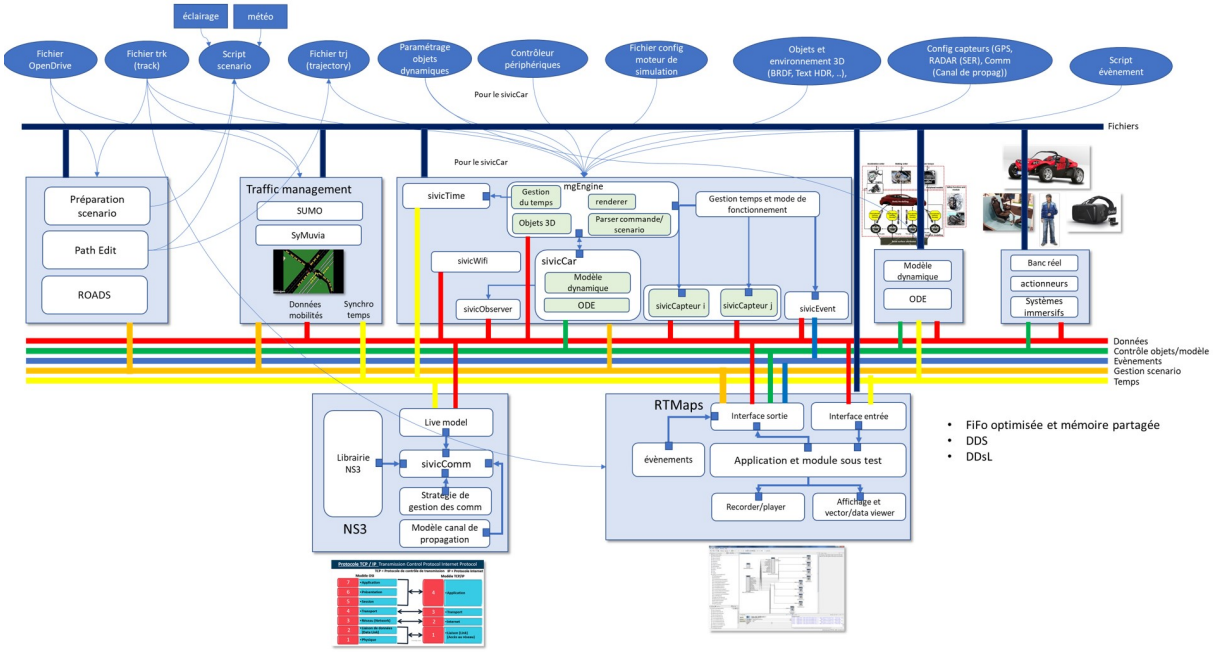


Figure 184: Implementation of the POC 1 simulation platform (Source UGE)

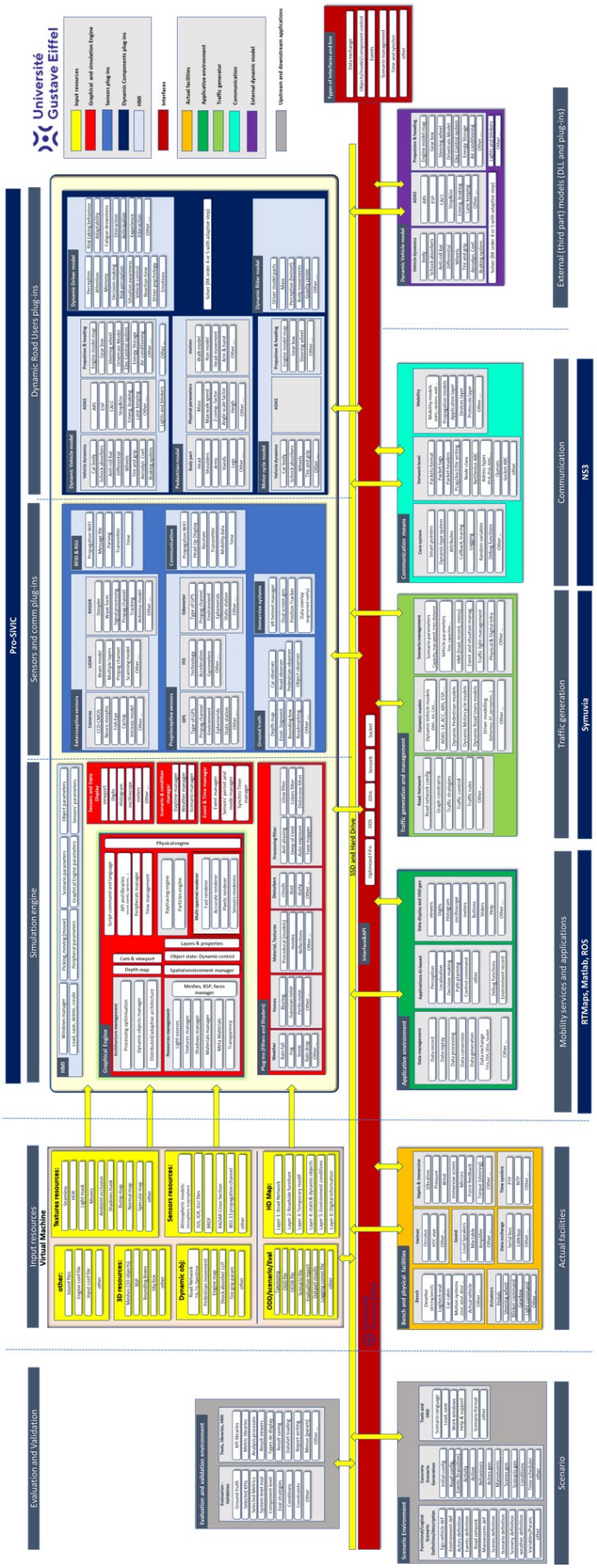


Figure 182: Final overview of the generic simulation framework proposed by UGE for the evaluation and validation of AV (Source: UGE)

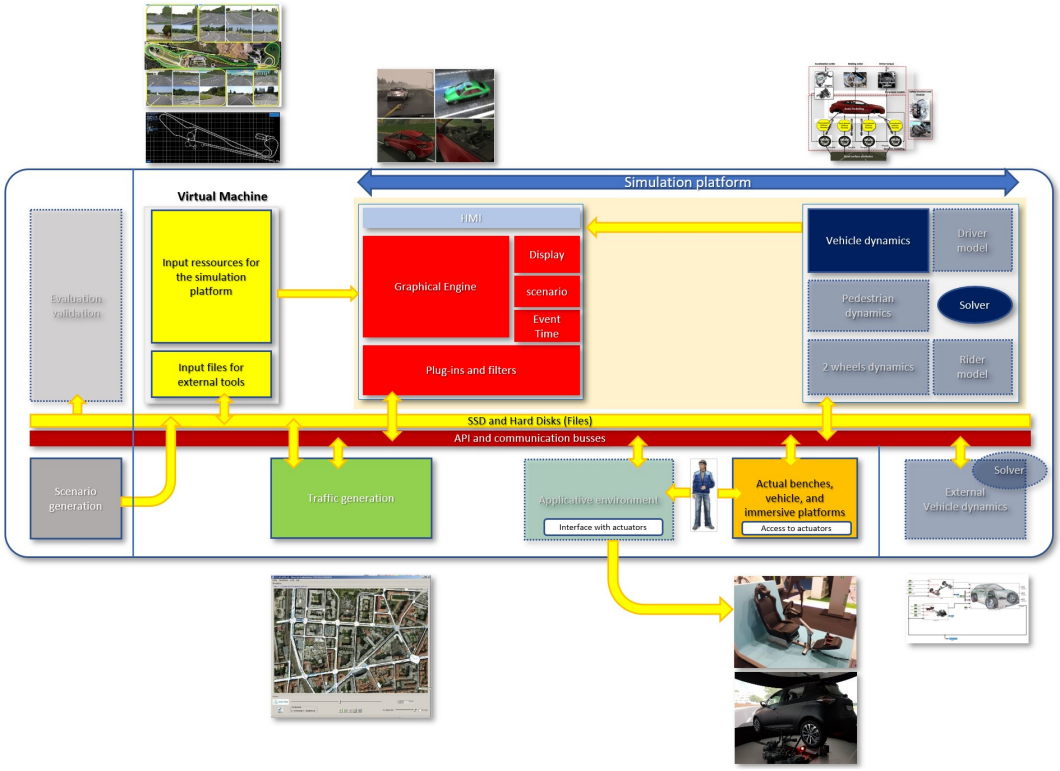


Figure 185: Proposal of a Driving Simulation architecture from the generic framework proposed by UGE (Source UGE)

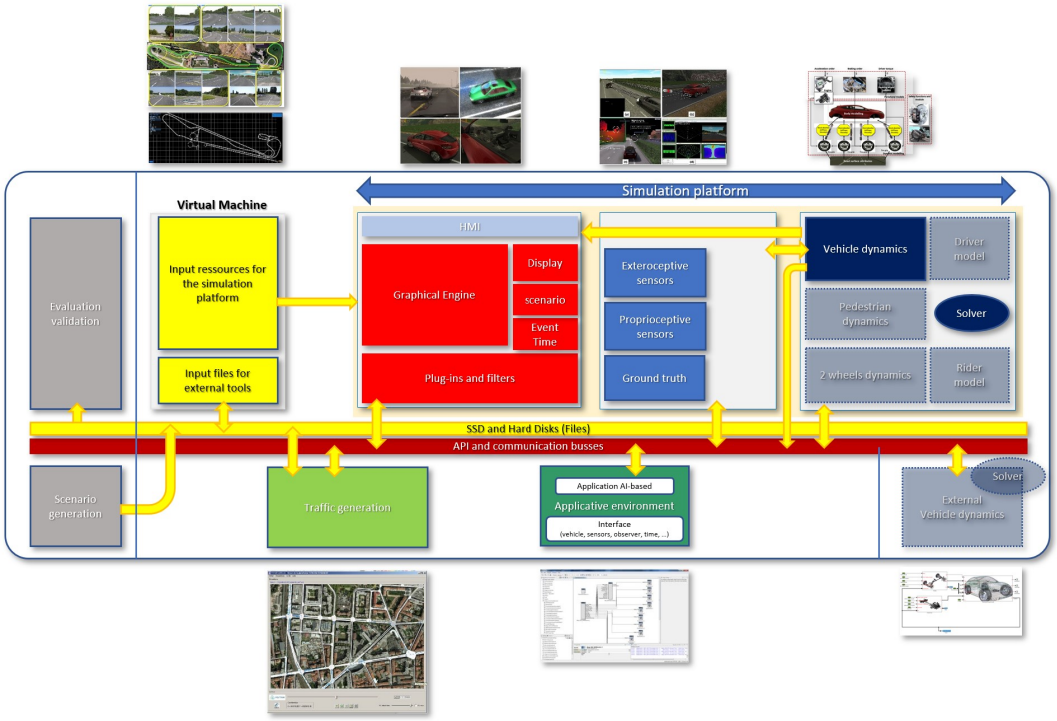


Figure 186: Proposal of an Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

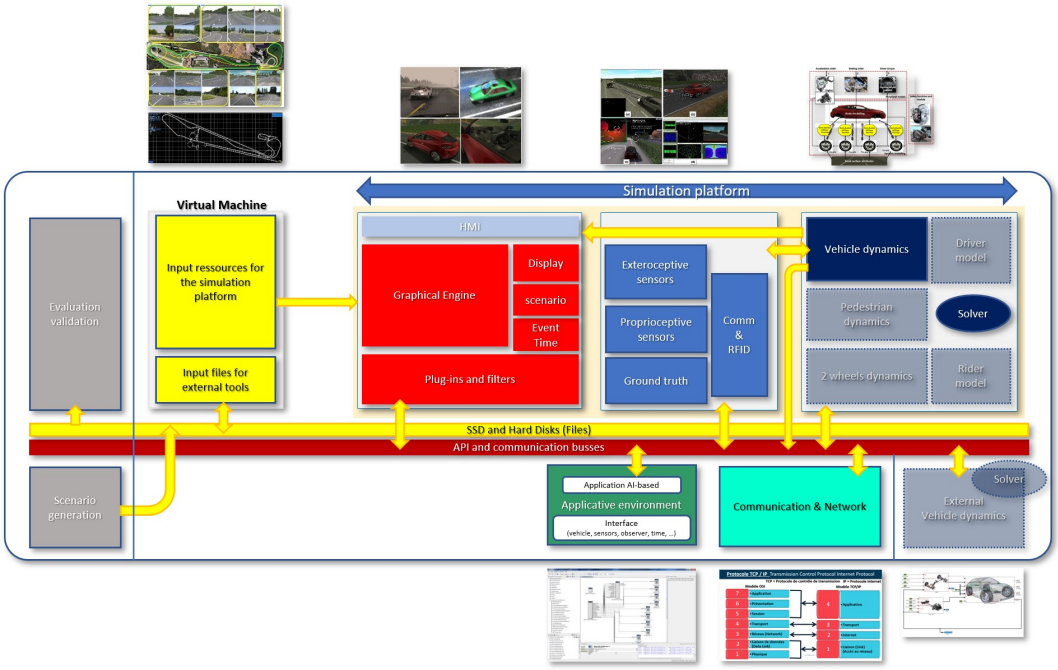


Figure 187: Proposal of a Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

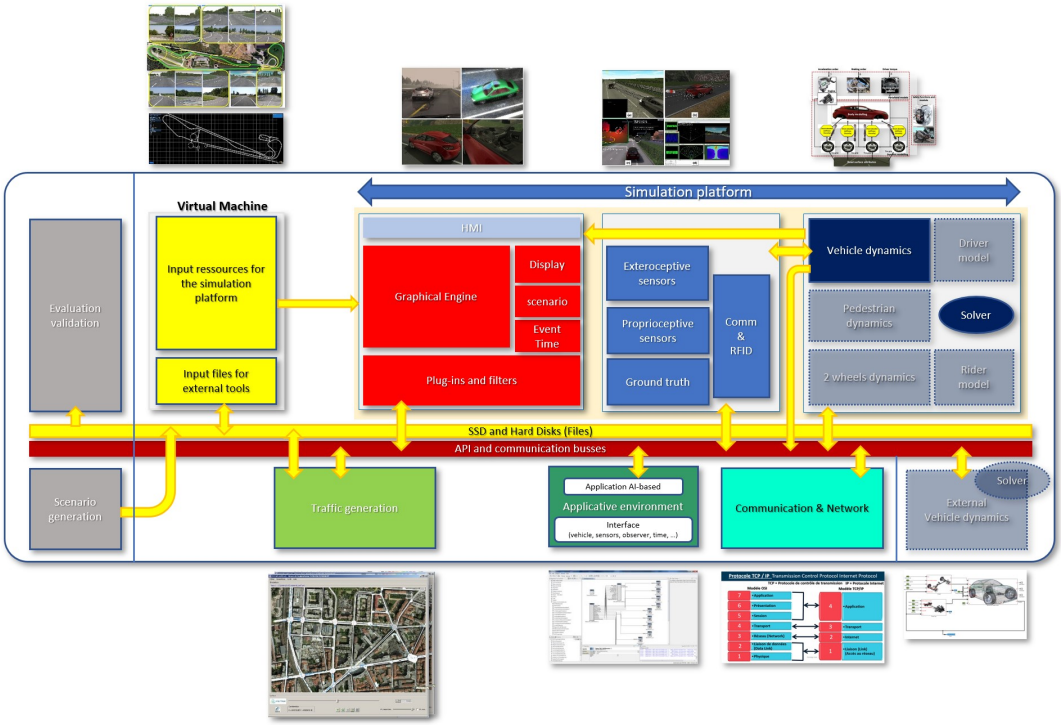


Figure 188: Proposal of a Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

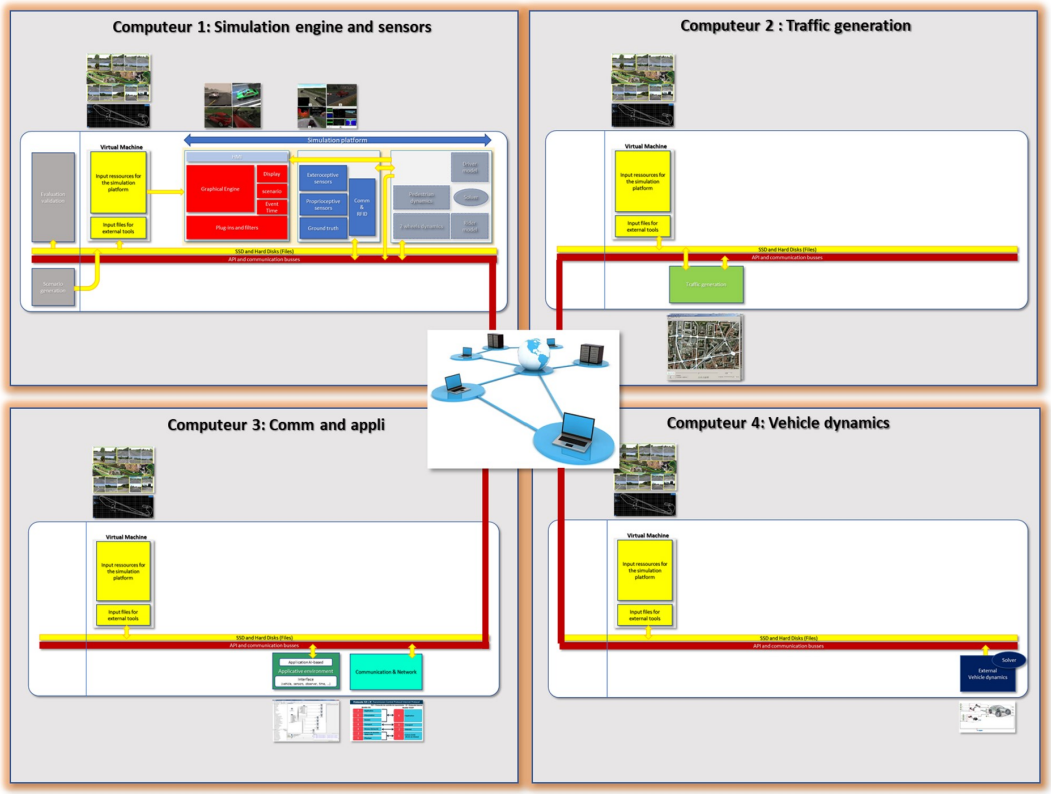


Figure 189: Proposal of a Distributed Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

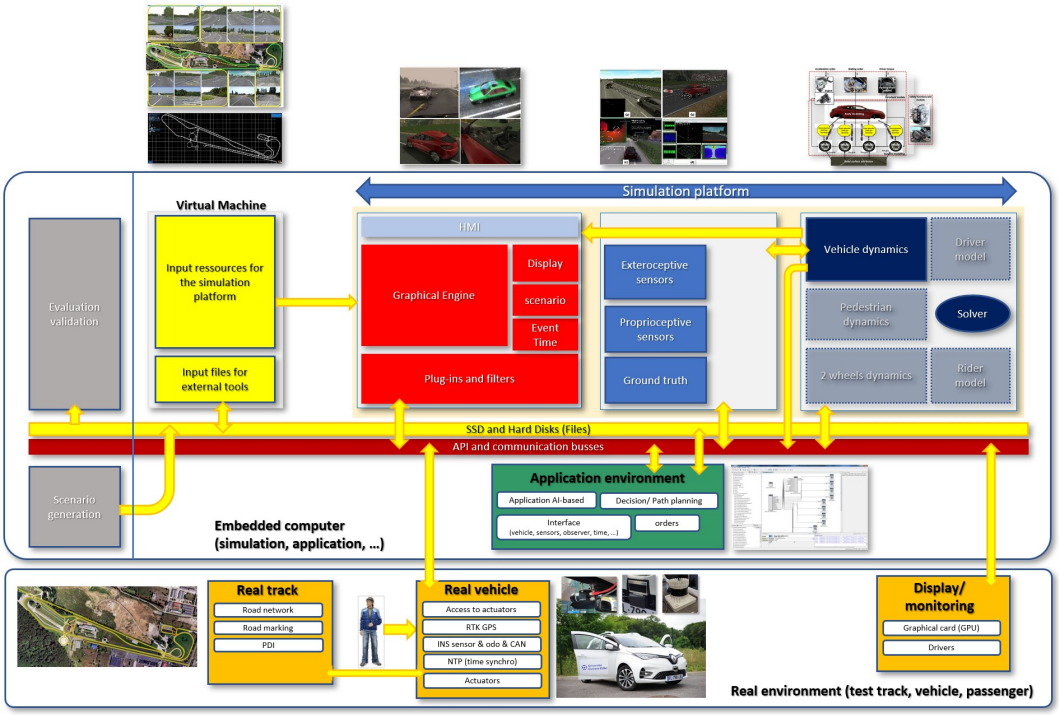


Figure 190: Proposal of a Vehicle in the Loop architecture involving AV simulator and application environment from the generic framework proposed by UGE (Source UGE)

The figure 191 presents a complex architecture merging a simulation framework, a dynamic and immersive platform (CKAS, Zoé cabin, dashboard, sound, steering wheel and force feedback, ...), and the real automated Zoé Renault. This platform is called ImPACT 3D and is developed by UGE under the responsibility of Dominique Gruyer. In this architecture, the first ego vehicle is controlled by using API and control law using the external vehicle dynamics implemented in an external tool or library (DLL). The control laws generate orders used by the CKAS motion system. The simulation engine provides the rendering of the road scene using the Digital Model of the Satory test tracks. The second ego-vehicle corresponds to the real automated Zoé Renault moving on the real Satory test tracks. The embedded RTK GPS and INS/Odo sensors provides the reference about the second ego-vehicle state vector in order to control the virtual second ego-vehicle (avatar) in the simulation environment. Concerning the first ego-vehicle, the state vector is sent to the real ego-vehicle in order to provide an augmented reality usable by the embedded application environment. In this real environment, the LiDAR data and the camera images are augmented and enriched with the projection of the virtual ego-vehicle. In both real vehicle and simulation platform, it is possible to put in the loop 2 real drivers. The synchronisation of the different part of this distributed platform is done by using of PTP and NTP. NTP used for synchronisation at the application level. NTP has a coarse-level granularity, and a lack of synchronisation guarantee requirement. But it is enough to synchronise the computers and applications in the real ego-vehicle. NTP is mainly a software synchronisation. PTP is a hardware synchronisation system used for accurate synchronisation. PTP is used for critical applications and deletes the network and equipment delay and jitter in the time accuracy.

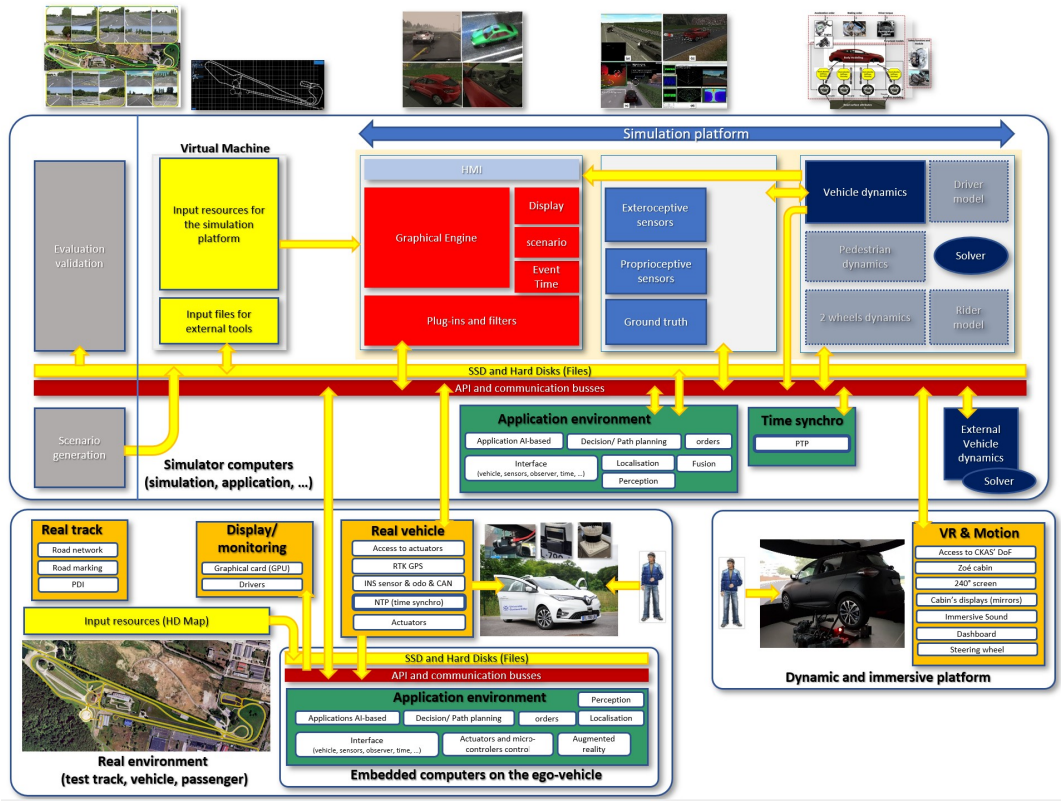


Figure 191: Proposal of an Interconnected platform as the concept of ImPACT 3D. ImPACT 3D is developed by UGE and will work in real time with a real proprtype on the test track and a dynamic and immersive simulation platform. This distributed, interconnected, and dynamic platform relies on the generic framework proposed by UGE (Source UGE)

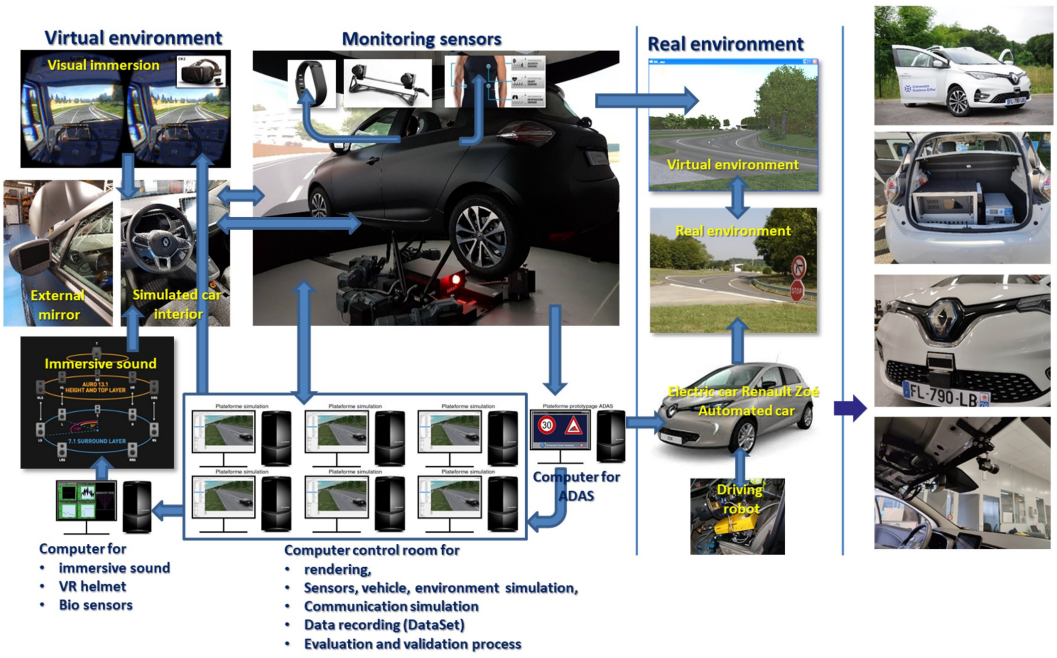


Figure 192: Impact 3D: an interconnected platform with dynamic and immersive platform and real automated vehicle (Source UGE)

4.3 Generic simulation framework proposed by Inria (POC 4)

The figure 193 presents the ROS-based framework from POC 4. ROS, short for Robot Operating System, is a flexible, open-source framework designed for robot software development. It provides a modular structure, enabling complex systems to be divided into simpler, independent components, each responsible for a specific task. ROS offers a wide range of libraries and tools that simplify tasks such as message passing, service calls, and hardware abstraction. Communications through ROS are based on messages with standardised types, for example LiDAR point clouds are exchanged using built-in *PointCloud2* type and frame transforms and coordinates are exchanged using built-in *TransformStamped* type on the standard */tf* topic. If required, custom types can be defined and used in the same way. Latest versions of ROS (ROS2) is built on top of DDS (Data Distribution Service), an industry standard for real-time communication.

This framework comes with the constrain of using ROS librairies and ROS message types. However, it is possible to use bridges to interface an existing simulator or ADS with ROS. POC4 used Gazebo, which natively supports ROS, but other simulators like Carla offer bridges and tools to interface with it. Additionally, in the case of the simulator, only required data like LiDAR point clouds, camera images, and vehicle state data are exchanged with other components, reducing the complexity of the interface. More generally, every component of the framework (e.g. simulator, perception system, scenario manager, ground truth generator) is compartmentalised and can be replaced by a different implementation as long as it respects the same interface. This allows for easy integration of new components and tools, and for the framework to be adapted to different use cases and requirements.

An other key aspect of the framework is the use of ROS bags for data recording and replay. The recording tool automatically subscribes to any desired topics and records all exchanged messages to a bag file. Reception of messages is timestamped and included with the message data in the bag file. Timestamps are used when replaying the bag file to simulate the scenario in the same chronological order. Additionally, message data is automatically serialised and compressed for efficient storage. ROS bags are a versatile tool for data recording and replay, during the following evaluation of the scenarios, bags using the common interface but from heterogeneous implementations of the framework can be used together. Configurations of the framework can be saved in the bags to adjust the evaluation while the data across the bags is in the same type and format.

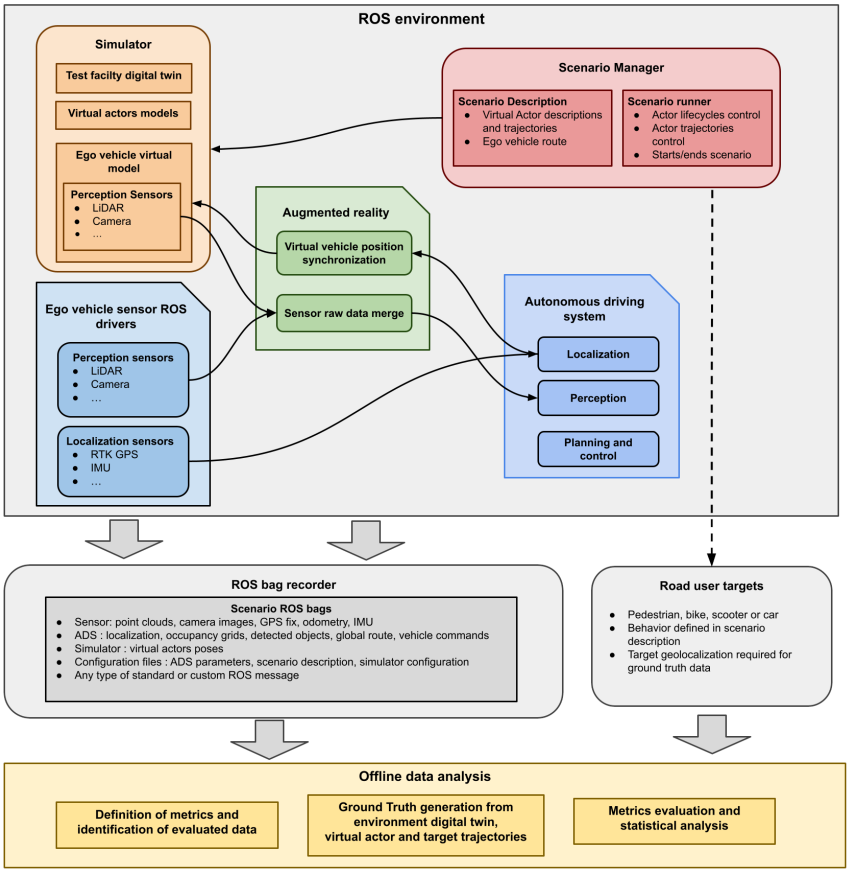


Figure 193: ROS-based framework from POC 4, rounded boxes represent key components of the framework, they are grouped under categories in dashed boxes. Arrows represent key messages exchanged between components.

5 Advice, recommendations, issues, challenges, future developments

This section tries to provide, from the different studies, developments, and implementations made in PRISSMA, a set of advice, recommendations, issues, challenges, and future developments regarding simulation environments, platforms, tools, and models for the evaluation and validation of automated mobility systems:

5.1 Advice and Recommendations

- High-fidelity simulation environment: it seems to be relevant to invest in real-time high-fidelity simulation environments that accurately replicate real-world conditions to ensure realistic testing of automated mobility systems. The realism and fidelity aspects in simulation environments are essential and simulation environments should accurately replicate real-world conditions to ensure that the behaviour of automated mobility systems can be realistically evaluated. This includes factors such as road conditions, weather, traffic, pedestrian behaviour, and sensor data.

- The collaboration between industry partners, research institutions (and pool of end users, beta testers, citizens), and regulatory bodies is essential in order to establish common standards and best practices for simulation-based testing will be mandatory.

- In order to develop a large and extended community of developers around these topics of virtual evaluation and validation, it seems relevant to develop and to share open-source simulation platforms and tools to foster collaboration, innovation, and knowledge sharing within the research community.

- A lot of simulation platforms claiming the generation of high fidelity and sensor realistic data exist. Nevertheless it seems mandatory to guarantee the quality and reliability of these processes of simulation-based testing. It is essential to ensure that simulation results are validated against real-world data and scenarios. It is mandatory to provide some procedures allowing to apply formal proof.

- Propose a global strategy for the data , ground truth, and DataSets generation for the training, evaluation, and validation of applications and components involved in the design and the development of automated driving systems and services.

- As presented in the Deliverable D2.8 of PRISSMA, it is important to invest in the scenario definition, generation, and validation in order to guarantee to cover the sets of scenarios defined in regulations (EURONCAP and EU ADS), the nominal scenarios and also critical scenarios involving rare events and situations. This aspect includes the simulation of cyber-attacks and of events called ”black swans” (event with a probability of occurrence close to 0).

- A simulation environment needs to guarantee capabilities of Customisation and Flexibility. Indeed, different use cases and scenarios could require different simulation setups. A good simulation platform should allow for easy customisation and configuration of scenarios, traffic patterns, road layouts, and environmental conditions.

- Validation and Verification Tools are mandatory for effective validation and verification process. These tools are necessary for assessing the performance and safety of automated

mobility systems. This includes tools for analysing simulation results, identifying edge cases, and assessing the robustness of AI algorithms.

- Open Standards and Interoperability are essential to develop and to design generic simulation frameworks. To foster collaboration and innovation, simulation platforms should adhere to open standards and support interoperability with other tools and platforms. This enables researchers and developers to leverage existing models, datasets, and simulation environments.

- Data Generation and Analysis are the core of the evaluation and validation processes (in simulation, in controlled environment, and in open road environment). Simulation platforms should support the generation and analysis of large quantities of data, including sensor data, vehicle trajectories and state vector, application outputs, and system performance results (from the using of KPIs and metrics). The evaluation and validation processes are essential for training AI models, refining algorithms, and improving system performance. Real-Time Simulation and Hardware-in-the-Loop Testing can offer Real-time simulation capabilities to test real-time performance of automated mobility systems with human in the loop (for level 3 and 4 of automation). Hardware-in-the-loop testing, which involves integrating physical components such as sensors and actuators into the simulation, can further enhance realism and accuracy (real physical movement of the ego-vehicle and improved feeling of the human in the loop (i.e driver)).

- Simulation platforms should support regulatory compliance and safety assurance processes by providing tools for documenting simulation results, conducting safety assessments, and demonstrating compliance with industry standards and regulations.

- The field of automated mobility is rapidly evolving, and simulation platforms need to evolve accordingly. Continuous improvement and updates, including the integration of new technologies and methodologies, are essential for keeping simulation environments relevant and effective.

5.2 Issues and Challenges

- **Data Quality and Evaluation** : Ensuring the quality and reliability of simulation data, including sensor data and environmental conditions, can be challenging and may impact the accuracy of simulation results. It is obvious that the next generation of simulation platform will use AI-based methods in order to generate sensor-realistic data. Indeed, in the majority of simulation engines currently on the market, sensor rendering uses physical models that do not integrate AI. In addition, simulation engines or graphics engines do not have AI, except for populating scenes and for managing the behaviour of dynamic actors. This issue brings some challenges and a new issue about the performance and the real quality evaluation of the data generated by AI-based simulation environment. Some interesting studies already exist about this topic with the use of GAN, encoder, Transformer approaches. Nevertheless, it will be mandatory to provide in the same time a set of scores and metrics allowing to evaluate and to label the simulated data in order, for these data, to be usable in a training, evaluation, and validation process.

- **Computational Complexity**: Simulating large-scale scenarios with multiple automated vehicles, a large set of complex sensors (embedded and on the infrastructure), a large

set of disturbances (adverse and degraded weather conditions, light and shadows effects, reflection, ...), and complex traffic patterns requires significant computational resources and may introduce scalability challenges. Depending on the targeted use case and system under test, it seems to be mandatory to develop distributed and remote computational resources and computers. The use of server centers (cloud and computer clusters with high capacities of parallel and graphical computation (GPU farms)) will be mandatory if we seek to manage large scale and very complex (in terms of components and actors) environment (concept of metaverse with the Digital Shadows and Digital Models covering not only a local and short range area but a large area like a city, a long distance highway, or a region or a country). In these conditions, several issues will need to be addressed and managed like the synchronisation of the components, elements, and models distributed in a cluster of computers allowing to manage a distributed "world" twin (currently called a metaverse). The main challenge consists to share the simulation and simulation elements/components on several computers or clusters of computers and to manage accurately and efficiently the data flow propagation with a high level of performance for the time management and synchronisation constraints. Indeed, the world has it own high frequency operating and each components (sensors, PDI, ...) have their own frequencies based on the world reference (world frequency).

- **The data management**: In this context, the huge quantity of data to generate, to use, to process, to propagate, to record, and to analyse will be a huge challenge. This aspect needs to take into account the interfaces between softwares and computers, and the proposal of data format. The goal consists to propose an harmonised framework and a set of interface standard (i.e FMU and FMI) allowing to guarantee an efficient interoperability and evolution of the simulation environment. Data management must also take into account the best practices and legislations derived from Community law in terms of the management of aggregated data and the use (possibly for commercial purposes) of this data acquired as part of the development of simulation models (RGPD, data governance act, ...).

- **Validation and Verification**: Propose some methodologies and processes allowing to evaluate and validate the capability of a simulation platform to be efficiently and accurately used for an evaluation and validation process. In this context the challenge consists to provide the adapted and relevant metrics allowing to guarantee the fidelity and the representativeness of the data (sensors, dynamic objects behaviours, ground truth, ...) generated by the simulation environment and components.

- **Regulatory Compliance**: Meeting regulatory requirements for simulation-based testing and demonstrating compliance with industry standards and regulations poses challenges, particularly in highly regulated industries such as automotive. In the case of AI-based systems, it will be mandatory to apply and to respect the AI-act accepted in February 2024.

- **Ethical Considerations**: Addressing ethical considerations related to simulation-based testing, such as ensuring the privacy and security of simulation data and minimising the risk of unintended consequences, requires careful consideration.

- **Cyber security and cyber attacks simulation**: How to propose a generic methodology allowing to generate a large set of possible attacks as well as from perception level (distur-

bances on the environment furnitures and road signs), or communication means (impact on the message content, or message access, or message propagation (delay, repetition, flooding, ...), or sensors (electromagnetic, light, temperature interferences)

5.3 Future Developments

- **Enhanced Realism**: Future simulation environments will likely incorporate advancements in virtual reality (VR), augmented reality (AR), and artificial intelligence (AI) to enhance realism and immersion. As mentioned in the deliverable D2.1, D2.2, D2.3 of the WP2 of PRISSMA, a lot of recent works already propose solutions in order to improve the current rendering of simulation environment. Enhanced Realism and augmented reality technologies will enable simulation environments to provide enhanced realism by overlaying virtual objects and information onto real-world scenes. Also this can improve the immersion and fidelity of simulation experiences for real users in the loop. Some AI-based generative methods have been applied on synthetic images coming from Pro-SiVIC in several environment like Satory test tracks and Transpolis test tracks. The results are presented in figures 194, 195, and 196. Some work need to be done in order to manage and to guarantee the spatial and temporal consistency of the improved images. Moreover, improvements of the methods need to be done in order to reduce significantly the processing time.

- **Edge Computing**: Edge computing technologies will enable distributed simulation environments that can offload computational tasks to edge devices, reducing latency and improving scalability.

- **Increase of Multi-Modal, Multi-technology, Multi-conditions Simulation**: Simulation environments will increasingly support multi-modal simulations (including interactions between automated vehicles, pedestrians, cyclists, and other road users, to replicate complex urban environments), multi-technology simulations (including sensors, PDi, IOT, ...), and multi-condition simulation (generation of weather, light, and environment conditions).

- **Digital X and metaverse**: In order to manage large scale environments, it will be important to develop methodologies allowing to generate Digital Twin, Digital Shadows, and Digital Models of virtual environments. Digital-X and metaverse are strongly link. Indeed, The metaverse encompasses vast, immersive virtual environments where users can interact with each other and digital objects in real-time. The metaverse is persistent, meaning it continues to exist and evolve even when human users are not logged in. This continuity allows for a consistent experience where changes and developments remain over time, similar to the real world. This mean that a fleet of CAV could continu to evolve without human in the loop in order to evaluate their capacities overtime. In the metaverse, the users can also interact with each other through the use of avatars, which are digital representations of themselves. Moreover, a significant aspect of the metaverse is its ability to take into account new contents developed by users (i.e specific virtual structures, new furnitures, ...). The metaverse can blend elements of the physical world with the virtual one. For instance, augmented reality can overlay digital information onto the real world, while virtual reality can offer immersive simulations that feel almost real. It was the case in the POC 4 with the generation of an augmented reality with LiDAR impacts generated from virtual objects and merged with the real LiDAR impact frame.

- **Automated Validation**: AI-based validation techniques, such as generative adversarial networks (GANs) and reinforcement learning (RL), will be used to partially automate the validation and verification of simulation results and improve the efficiency of testing. In this context, AI-based system could also generate the references and ground truth needed in the evaluation and validation process. Of course, even if AI-based procedures will clearly facilitate the task of human evaluators, we must keep in mind that there will always be a need for one or more human operators providing an outside perspective and, above all, expertise. on the results produced.

- **XiL architectures**: Integration of simulation environment with Physical Components seems to be an efficient way to link the existing validation process in controlled environment with the simulation capabilities. VIL architecture involves integrating physical vehicles or components into simulation environments, allowing for real-time interaction between virtual and physical elements. This enables more realistic testing of automated mobility systems by considering the physical constraints and limitations of real vehicles. Hardware-in-the-Loop (HIL) Testing consists to incorporate and connect physical components such as sensors, actuators, and controllers are connected to simulation environments. This enables testing of real-world hardware and software interactions under controlled conditions. Closed-Loop Testing is directly linked with Digital Twins and allows to take into account feedback from physical components in order to feed back the simulation in real-time. This enables dynamic testing of automated mobility systems in realistic scenarios, including interactions with other vehicles, pedestrians, and environmental conditions. Of course XiL architectures will be used in complement of some existing open loop evaluation and validation procedures.

- **Verification and Validation (VV) of Models, Tools, and Simulation Platforms**: In PRISSMA, a first study has been done about the process and methodology usable in order to provide a label on the models, tools, and platforms developed for the automated mobility components and applications. This work will be addressed in the future for the following objectives: The model verification and validation will involve validating simulation models, algorithms, and sensor models against real-world data and scenarios to ensure accuracy and reliability. This may include benchmarking against industry standards and best practices. The simulation tools and platforms will undergo validation to ensure that they accurately replicate real-world conditions and produce reliable results. This may involve comparing simulation results with empirical data and conducting sensitivity analyses. The simulation platforms will be validated to ensure that they meet regulatory requirements and industry standards for safety, reliability, and performance. This may involve conducting rigorous testing and certification processes. The evaluation of the level of realism and fidelity of simulated data involves assessing how well simulation environments replicate real-world conditions and behaviours. This may include analysing factors such as sensor accuracy, environmental variability, and scenario complexity.

- **Standards and Interoperability**: Efforts to establish common standards and interoperability between simulation platforms, tools, and models will continue to facilitate collaboration and knowledge sharing across the stakeholders. Moreover, these aspects will improve the efficiency and the quality of the evaluation and validation process in simulation.

Figure 194: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Satory's test tracks with foggy and rainy conditions. The first image on the top left is the initial generated image from Pro-SIVIC. The height other images are generated from AI-based methods with different parameters allowing to fit with the initial image. (Source: UGE).

Figure 195: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks. The top left and bottom left images are the initial images generated from Pro-SiVIC. The other ones are generated from 2 AI-based methods (Source: UGE).

Figure 196: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks. The top left image is the initial image generated from Pro-SIVIC. The other images are generated from AI-based methods with a variation of some parameters (Source: UGE).

The future developments to continue or to start will address these different domains:

- **Technology and hardware**: with the development of new sensors, it will be necessary to obtain the information and data from OEM in order to develop realistic model. In order to use distributed, complex, large scale simulation environment, it will be necessary to propose and to develop powerful and parallel computers architecture (distributed computer architecture, cluster of GPU and computers). ViL and HiL architectures. (Digital Twins)
- **Software**: Develop new generation of algorithms for accurate and high quality renderings of data and simulation of dynamic and interactive objects. (Digital Shadows and Digital Twins)
- **Data management**: How to efficiently generate, use, process, propagate, record, and analyse data. This aspect will address the issue of data format.
- **Standards**: for Dataset generation and recording (involving Digital Models), for simulation environment dedicated to evaluation and validation, to AI-based application testing, for scene and scenario management and generation, for evaluation and validation process, ...
- **Ethics**: link to governance, data issues, security and beyond ?

REFERENCES

[1] JAMA. (2020, October) Automated driving safety evaluation framework ver.1.0. Report from Japan Automobile Manufacturers Association, Inc. (JAMA), AD Safety Assurance Expert Group. [Online]. Available: http://www.jama-english.jp/publications/Automated_Driving_Safety_Evaluation_Framework_Ver1.0.pdf

[2] C. Linnhoff, P. Rosenberger, S. Schmidt, L. Elster, R. Stark, and H. Winner, "Towards serious perception sensor simulation for safety validation of automated driving - a collaborative method to specify sensor models," in 24th International Conference on Intelligent Transportation Systems (ITSC), Indianapolis, IN, USA, September 19-22 2021, September 2021. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9564661

[3] N. Druml, I. Maksymova, T. Thurner, and D. van Lierop, "1d mems micro-scanning lidar," in SENSORDEVICES 2018 : The Ninth International Conference on Sensor Device Technologies and Applications, 2018. [Online]. Available: https://personales.upv.es/thinkmind/dl/conferences/sensordevices/sensordevices_2018/sensordevices_2018_3_30_20065.pdf

[4] K. Montalban, C. Reymann, D. Atchuthan, P.-E. Dupouy, N. Riviere, and S. Lacroix, "A quantitative analysis of point clouds from automotive lidars exposed to artificial rain and fog," MDPI Atmosphere, vol. 12, no. 6, p. 738, June 2021. [Online]. Available: https://www.mdpi.com/2073-4433/12/6/738

[5] D. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review." MDPI Sensors, vol. 21, no. 2140, pp. 1–37, March 2021. [Online]. Available: https://www.researchgate.net/publication/350156437_Sensor_and_Sensor_Fusion_Technology_in_Autonomous_Vehicles_A_Review

[6] S. Patole, M. Torlak, D. Wang, and M. Ali, "Automotive radars. a review of signal processing techniques," IEEE Signal Processing Magazine, Signal ProceSSing for Smart Vehicle technologieS: Part 2, vol. 34, no. 2, pp. 22–35, March 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7870764

[7] N. Li, W. Huai, S. Wang, and L. Ren, "A real-time infrared imaging simulation method with physical effects modeling of infrared sensors," Infrared Physics Technology, vol. 78, no. 37, pp. 45–57, September 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1350449516302456?via%3Dihub

[8] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. D. Carlo, and C. E. Campanella, "Gyroscope technology and applications: A review in the industrial perspective," Sensors, vol. 17, no. 10, October 2017.

[9] M. T. A. H. Nakayasu H, Nakagawa M, "Human cognitive reliability analysis on driver by driving simulator," in IEEE 40th international conference on computers and industrial engineering (CIE). 25-28 July 2010, Awaji, Japan, 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5668250

[10] S. Hamdar, Driver Behavior Modeling. London: Springer London, 2012, pp. 537–558. [Online]. Available: https://doi.org/10.1007/978-0-85729-085-4_20

[11] N. M. Negash and J. Yang, "Driver behavior modeling toward autonomous vehicles: Comprehensive review," IEEE ACCESS, vol. 11, pp. 22 788–22 821, February 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10054035

[12] Z. Wang, Y. Shi, W. Tong, Z. Gu, and Q. Cheng, "Car-following models for human-driven vehicles and autonomous vehicles: A systematic review," Journal of Transportation Engineering, Part A: Systems, vol. 149, no. 8, p. 04023075, 2023.

[13] Y. QAMSANE, J. MOYNE, M. TOOTHMAN, I. KOVALENKO, and al, "A methodology to develop and implement digital twin solutions for manufacturing systems," IEEE ACCESS, pp. 1–22, March 2021. [Online]. Available: https://www.researchgate.net/publication/350081269_A_Methodology_to_Develop_and_Implement_Digital_Twin_Solutions_for_Manufacturing_Systems

[14] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, and M. Yang, "Choose your simulator wisely: A review on open-source simulators for autonomous driving," IEEE Transactions on Intelligent Vehicles, vol. 13, no. 4, pp. 1 – 19, March 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10461065

[15] S. Malik, M. A. Khan, A. anf Hesham El-Sayed, F. Iqbal, J. Khan, and O. Ullah, "Carla+: An evolution of the carla simulator for complex environment using a probabilistic graphical model." Drones, vol. 7, pp. 1–28, February 2023. [Online]. Available: https://www.mdpi.com/2504-446X/7/2/111

[16] S. TANG, Z. ZHANG, Y. ZHANG, J. ZHOU, Y. GUO, and al, "A survey on automated driving system testing: Landscapes and trends," ACM Transactions on Software Engineering and Methodology, vol. 32, pp. 1292–1321, July 2023. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3579642

[17] L. Zhang, A. Zhu, and Y. Zhou, "Simulation of atmospheric visibility impairment," vol. 30, pp. 8713–8726, 2021.

[18] S. Yun, J. Lee, W. Jang, D. Kim, M. Choi, and J. Chung, "Dynamic modeling and analysis of a driving passenger vehicle," Applied Sciences, vol. 13, pp. 1 – 25, May 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/10/5903

[19] A.-T. Nguyen, T. Q. Dinh, T.-M. Guerra, and J. Pan, "Takagi-Sugeno Fuzzy Unknown Input Observers to Estimate Nonlinear Dynamics of Autonomous Ground Vehicles: Theory and Real-Time Verification," IEEE/ASME TRANSACTIONS ON MECHATRONICS, vol. 26, pp. 1328–1338, june 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9314225

[20] H. Li, V. P. Makkapati, L. Wan, E. Tomasch, H. Hoschopf, and A. Eichberger, "Validation of Automated Driving Function Based on the Apollo Platform: A Milestone for Simulation with Vehicle-in-the-Loop Testbed." Vehicles, vol. 5, p. 718–731, june 2023. [Online]. Available: https://www.mdpi.com/2624-8921/5/2/39

[21] S. Sedran, "A truck dynamics model for driving simulators," CHALMERS UNIVERSITY OF TECHNOLOGY, Tech. Rep., March 2016, master's thesis

in Automotive Engineering. [Online]. Available: https://publications.lib.chalmers.se/records/fulltext/233463/233463.pdf

[22] A. G.-A. Blanco, D. García-Vallejo, and E. Freire, "An electric kickscooter multibody model: equations of motion and linear stability analysis," Multibody System Dynamics, no. 2, February 2024. [Online]. Available: https://www.researchgate.net/publication/378967189_An_electric_kickscooter_multibody_model_equations_of_motion_and_linear_stability_analysis

[23] H. Masuda, O. E. Marai, M. Tsukada, T. Taleb, and H. Esaki, "Feature-based vehicle identification framework for optimization of collective perception messages in vehicular networks," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, vol. 72, no. 2, pp. 2120–2129, February 2023. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9910433

[24] D. J. Segelstein, "The complex refractive index of water," Theses, M.S. Thesis, University of Missouri–Kansas City, 1981.

[25] K. Garg and S. K. Nayar, "Vision and Rain," International Journal of Computer Vision, vol. 75, no. 1, pp. 3–27, feb 2007. [Online]. Available: http://link.springer.com/10.1007/s11263-006-0028-6

[26] R. Gunn and G. Kinzer, "The terminal velocity of fall for water droplets in stagnant air," Journal of Meteorology, 1949. [Online]. Available: http://journals.ametsoc.org/doi/abs/10.1175/1520-0469(1949)006{%}3C0243:TTVOFF{%}3E2.0.CO;2

[27] M. N. Chowdhury, F. Y. Testik, M. C. Hornack, and A. A. Khan, "Free fall of water drops in laboratory rainfall simulations," Atmospheric Research, vol. 168, pp. 158–168, Feb. 2016.

[28] G. B. Foote and P. S. Du Toit, "Terminal Velocity of Raindrops Aloft." Journal of Applied Meteorology, vol. 8, no. 2, pp. 249–253, Apr. 1969.

[29] E. Villermaux and F. Eloi, "The distribution of raindrops speeds," Geophysical Research Letters, vol. 38, no. 19, 2011. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011GL048863

[30] P. Duthon, F. Bernardin, F. Chausse, and M. Colomb, "Methodology used to evaluate computer vision algorithms in adverse weather conditions," Transportation Research Procedia, vol. 14, pp. 2178–2187, 2016, transport Research Arena TRA2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146516302368

[31] C. Radovich and D. Plocher, Experiments on Spray from a Rolling Tire. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 403–417. [Online]. Available: https://doi.org/10.1007/978-3-540-85070-0_37

[32] P. Castellanos, P. Colarco, W. R. Espinosa, S. D. Guzewich, R. C. Levy, R. L. Miller, M. Chin, R. A. Kahn, O. Kemppinen, H. Moosmüller, E. P. Nowottnick, A. Rocha-Lima, M. D. Smith, J. E. Yorks, and H. Yu, "Mineral dust optical properties for remote sensing and global modeling: A review,"

Remote Sensing of Environment, vol. 303, p. 113982, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425723005345

[33] K. Pattanapakdee and S. Chotigo, "Experimental investigation of pavement light reflection characteristics in wet conditions," 06 2019, pp. 1790–1795.

[34] M. Colomb, P. Duthon, and F. Bernardin, "Spectral reflectance characterization of the road environment to optimize the choice of autonomous vehicle sensors," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 1085–1090.

[35] E. P. Shettle and R. W. Fenn, "Models for the aerosols of the lower atmosphere and the effects of humidity variations on their optical properties," 1979.

[36] J. Stevens, "The how and why of open architecture," Undersea Warfare, no. 37, pp. 6–9, 2008.

[37] S. Yoo and A. Jerraya, "Introduction to hardware abstraction layers for soc," in 2003 Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 336–337.

[38] A. Gazis and E. Katsiri, "Middleware 101: What to know now and for the future," Queue, vol. 20, no. 1, pp. 10–23, 2022.

[39] D. H. Martinen, M. Lagalaye, J. Pfefferkorn, and J. Casteres, "Modular and open test bench architecture for distributed testing," SAE Technical Paper, Tech. Rep., 2017.

[40] D.-S. Cho, S. Yun, H. Kim, J. Kwon, and W.-T. Kim, "Autonomous driving system verification framework with fmi co-simulation based on omg dds," in 2020 IEEE International Conference on Consumer Electronics (ICCE), 2020, pp. 1–6.

[41] N. Mohan and M. Törngren, "Ad-eye: A co-simulation platform for early verification of functional safety concepts," SAE Technical Paper 2019-01-0126, 2019.

[42] J. Khan, "Autonomous driving validation using game engine technology," Tata Elxsi, Tech. Rep., 2019. [Online]. Available: https://tataelxsi.com/storage/insights/March2021/BlySkLFzuJCrQolOvk1V.pdf

[43] D. Gruyer, S. Laverdure, J.-S. Berthy, P. Desouza, and M. Hadj-Bachir, "From virtual to real, how to prototype, test, evaluate and validate adas for the automated and connected vehicle?" in From AI to Autonomous and Connected Vehicles, Advanced Driver-Assistance Systems (ADAS), ser. Digital Science, T. Bapin and A. Bensrhair, Eds. New York: ISTE Ltd., London, and John Wiley and Sons, 2021, ch. 4. [Online]. Available: http://iste.co.uk/book.php?id=1800

[44] J. Yoon, B. Son, and D. Lee, "Comparative study of physics engines for robot simulation with mechanical interaction," MDPI Applied Sciences, vol. 13, no. 680, January 2023.

[45] R. M. Templet, "Game physics: An analysis of physics engines for first-time physics developers," CALIFORNIA STATE UNIVERSITY, NORTHRIDGE, thesis of Master of Science in Software Engineering, Tech. Rep., December 2020. [Online]. Available: https://scholarworks.calstate.edu/downloads/z890s004h

[46] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in 2015 IEEE International Conference on Robotics and Automation (ICRA), 26-30 May 2015. IEEE, 2015, pp. 1–19. [Online]. Available: https://homes.cs.washington.edu/~todorov/papers/ErezICRA15.pdf

[47] M. Müller, M. Durner, R. Siegwart, W. Stürzl, and R. Triebel, "A photorealistic terrain simulation pipeline for unstructured outdoor environments," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September 2021, pp. 9765–9772.

[48] Y. Li, P. Duthon, M. Colomb, and J. Ibanez-Guzman, "What happens for a tof lidar in fog?" IEEE Transactions on Intelligent Transportation Systems, 2021.

[49] B. Chretien. (2012, January) Simulation of a new automotive concept based on a centralized approach for driver assistance system activation decision. THÈSE DE DOCTORAT de l'Université d'Evry Val d'Essonne, spécialité Automatique, Systèmes Productiques et Robotique.

[50] S. Glaser. (2004, Janvier) Modélisation et analyse d'un véhicule en trajectoires limites application au développement de systèmes d'aide à la conduite. THÈSE DE DOCTORAT de l'Université d'Evry Val d'Essonne, spécialité Automatique, Systèmes Productiques et Robotique.

[51] I. Gultepe, R. Tardif, S. C. Michaelides, J. Cermak, A. Bott, J. Bendix, M. D. Müller, M. Pagowski, B. Hansen, G. Ellrod, W. Jacobs, G. Toth, and S. G. Cober, "Fog research: A review of past achievements and future perspectives," Pure and Applied Geophysics, 2007.

[52] P. Duthon, M. Colomb, and F. Bernardin, "Fog classification by their droplet size distributions: Application to the characterization of cerema's platform," Atmosphere, vol. 11, no. 6, 2020. [Online]. Available: https://www.mdpi.com/2073-4433/11/6/596

[53] L. Li, P. Kooi, M. Leong, and T. Yeo, "On the simplified expression of realistic raindrop shapes," Microwave and optical technology letters, vol. 7, no. 4, pp. 201–205, Mar. 1994.

[54] P. Räisänen, A. Kokhanovsky, G. Guyot, O. Jourdan, and T. Nousiainen, "Parameterization of single-scattering properties of snow," The Cryosphere, 2015.

[55] G. Haessig, D. Joubert, J. Haque, M. B. Milde, T. Delbruck, and V. Gruev, "Pdavis: Bio-inspired polarization event camera," in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 17-24 June 2023, Vancouver, BC, Canada, June 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10209002

[56] W. HUANG, X. YAN, S. ZHANG, Z. LI, and al, "Mems and moems gyroscopes: A review," Photonic Sensors, vol. 13, no. 4, pp. 1–26, June 2023. [Online]. Available: https://link.springer.com/article/10.1007/s13320-023-0693-x

[57] C. Campolo, A. Vinel, A. Molinaro, and Y. Koucheryavy, "Modeling broadcasting in ieee 802.11p/wave vehicular networks," IEEE COMMUNICATIONS LETTERS,

vol. 15, pp. 199–201, February 2011. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:585205/FULLTEXT01.pdf

[58] D. Bankov, E. Khorov, A. Krasilov, and A. Otmakhov, "Analytical model of 5g v2x mode 2 for sporadic traffic," arXiv:2309.12901v1 [cs.NI], 2023. [Online]. Available: https://arxiv.org/pdf/2309.12901.pdf

[59] M. Sepulcre, M. Gonzalez-Martín, J. Gozalvez, R. Molina-Masegosa, and B. Coll-Perales, "Analytical models of the performance of ieee 802.11p vehicle to vehicle communications," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, vol. 71, no. 1, January 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9599363

[60] R. Mardiati, N. Ismail, and A. Faroqi, "Review of microscopic model for traffic flow," ARPN Journal of Engineering and Applied sciences, vol. 9, no. 10, pp. 1794–1800, 2014.

[61] T. Bellet, J.-C. Bornard, P. Mayenobe, and G. Saint Pierre, "A computational model for car drivers situation awareness simulation: Cosmodrive," in DHM 2011 First International Symposium on Digital Human Modeling, 2011, p. 8p.

[62] F. Vrbanić, D. Čakija, K. Kušić, and E. Ivanjko, "Traffic flow simulators with connected and autonomous vehicles: A short review," Transformation of Transportation, pp. 15–30, 2021.

[63] G. Mahapatra, A. K. Maurya, and P. Chakroborty, "Parametric study of microscopic two-dimensional traffic flow models: a literature review," Canadian Journal of Civil Engineering, vol. 45, no. 11, pp. 909–921, 2018.

[64] R. Chandler, R. Herman, and E. Montroll, "Traffic dynamics: studies in car following, operation research," 1958.

[65] P. G. Gipps, "A behavioural car-following model for computer simulation," Transportation Research Part B: Methodological, vol. 15, no. 2, pp. 105–111, 1981.

[66] R. Wiedemann, "Simulation des straßenverkehrsflusses. schriftenreihe heft 8," Institute for Transportation Science, University of Karlsruhe, Germany, 1974.

[67] S. Krauß, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," Physical Review E, vol. 55, no. 5, p. 5597, 1997.

[68] E. Brockfeld, R. D. Kühne, and P. Wagner, "Calibration and validation of microscopic models of traffic flow," Transportation Research Record, vol. 1934, no. 1, pp. 179–187, 2005.

[69] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," Physical review E, vol. 62, no. 2, p. 1805, 2000.

[70] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," Physical review E, vol. 51, no. 2, p. 1035, 1995.

[71] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," Transportation Research Record, pp. 82–82, 1992.

[72] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," Journal de physique I, vol. 2, no. 12, pp. 2221–2229, 1992.

[73] G. F. Newell, "A simplified car-following theory: a lower order model," Transportation Research Part B: Methodological, vol. 36, no. 3, pp. 195–205, 2002.

[74] A. Duminil, S. song Ieng, and D. Gruyer, "Exploration of fidelity quantification in computer-generated images," MDPI Sensors, vol. 24, pp. 1–22, April 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/8/2463

[75] F. Peyret, J. Monsifrot, D. Bétaille, and C. Moriana Varo, "How gnss performance standardization can support the deployment of critical its applications)," in The 22nd ITS World Congress, Bordeaux, France, 2015.

[76] SaPPART. (2017) Cost Action TU1302. Assessment of positioning performance in ITS applications. [Online]. Available: https://reflexscience.univ-gustave-eiffel.fr/lire/ouvrages/sappart-manuel

[77] M. Diachuk, O. Lykhodii, D. Leontiev, L. Ryzhykh, and Y. Aleksandrov, "Dynamic modeling of semitrailer trucks equipped by steered wheels," JOURNAL OF MECHANICAL ENGINEERING AND SCIENCES, vol. 16, p. 8691 – 8705, November 2021.

[78] M. Modest, "Radiative heat transfer," Elsevier Science, 01 2003.

[79] World Meteorological Organization, Guide to Meteorological Instruments and Methods of Observation World Meteorological Organization, 2014. [Online]. Available: https://www.weather.gov/media/epz/mesonet/CWOP-WMO8.pdf{%}7D{%}7Bhttps://www.weather.gov/media/epz/mesonet/CWOP-WMO8.pdf

[80] J. I. Gordon, "Daytime visibility, a conceptual review," no. 11, p. 22, Nov. 1979, sIO Ref. 80-1. [Online]. Available: http://misclab.umeoce.maine.edu/education/VisibilityLab/reports/SIO_80-1.pdf

[81] A. Arnulf, J. Bricard, E. Curé, and C. Véret, "Transmission by Haze and Fog in the Spectral Region 035 to 10 Microns," Journal of the Optical Society of America, vol. 47, no. 6, p. 491, 1957.

[82] D. Deirmendjian, Electromagnetic Scattering on Spherical Polydispersions. Santa Monica, CA: RAND Corporation, 1969.

[83] W. J. Wiscombe, Improved mie scattering algorithms. Appl. Opt., 19, 1980.

[84] H. Van de Hulst, "Light scattering by small particles," 1957.

[85] H. Koschmieder, "Theorie der horizontalen sichtweite," Beiträge zurPhysik der freien Atmosphäre, vol. 12, pp. 33–55, 1924.

[86] Z. Lee and S. Shang, "Visibility: How applicable is the century-old koschmieder model?" Journal of the Atmospheric Sciences, vol. 73, no. 11, pp. 4573 – 4581, 2016. [Online]. Available: https://journals.ametsoc.org/view/journals/atsc/73/11/jas-d-16-0102.1.xml

[87] A. Ben-Daoued, P. Duthon, and F. Bernardin, "SWEET: A Realistic Multiwavelength 3D Simulator for Automotive Perceptive Sensors in Foggy Conditions," vol. 9, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2313-433X/9/2/54

[88] P. Duthon, M. Colomb, and F. Bernardin, "Light transmission in fog: The influence of wavelength on the extinction coefficient," Applied Sciences, vol. 9, no. 14, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/14/2843

[89] D. A. Stewart and O. M. Essenwanger, "A survey of fog and related optical propagation characteristics," pp. 481–495, 1982.

[90] M. Colomb, K. Hirech, P. André, J. Boreux, P. Lacôte, and J. Dufour, "An innovative artificial fog production device improved in the European project "FOG"," Atmospheric Research, vol. 87, no. 3-4, pp. 242–251, mar 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0169809507002037

[91] M. Colomb, F. Bernardin, and P. Morange, "Paramétrisation de la visibilité et modelisation de la distribution granulométrique à partir de données microphysiques," in Séminaire AMA 2008 Météo France, 2008, pp. 1–10.

[92] T. Elias, J. C. Dupont, E. Hammer, C. R. Hoyle, M. Haeffelin, F. Burnet, and D. Jolivet, "Enhanced extinction of visible radiation due to hydrated aerosols in mist and fog," Atmospheric Chemistry and Physics, vol. 15, no. 12, pp. 6605–6623, 2015.

[93] J. A. Garland, "Some fog droplet size distributions obtained by an impaction method," Quarterly Journal of the Royal Meteorological Society, vol. 97, no. 414, pp. 483–494, 1971.

[94] M. Kumai, "Arctic Fog Droplet Size Distribution and Its Effect on Light Attenuation," Journal of the Atmospheric Sciences, vol. 30, no. 4, pp. 635–643, 1973.

[95] J. Goodman, "The Microstructure of California Coastal Fog and Stratus," Journal of Applied Meteorology, vol. 16, no. 10, pp. 1056–1067, 1977.

[96] F. García-García, U. Virafuentes, and G. Montero-Martínez, "Fine-scale measurements of fog-droplet concentrations: A preliminary assessment," Atmospheric Research, vol. 64, no. 1-4, pp. 179–189, 2002.

[97] M. B. Meyer, J. E. Jiusto, and G. G. Lala, "Measurements of visual range and radiation-fog (haze) microphysics." Journal of the Atmospheric Sciences, vol. 37, no. 3, pp. 622–629, 1980.

[98] Q. Liu, B. Wu, Z. Wang, and T. Hao, "Fog droplet size distribution and the interaction between fog droplets and fine particles during dense fog in tianjin, china," Atmosphere, vol. 11, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2073-4433/11/3/258

[99] R. G. Pinnick, D. L. Hoihjelle, G. Fernandez, E. B. Stenmark, J. D. Lindberg, G. B. Hoidale, and S. G. Jennings, "Vertical Structure in Atmospheric Fog and Haze and Its Effects on Visible and Infrared Extinction," Journal of the Atmospheric Sciences, vol. 35, no. 10, pp. 2020–2032, 1978.

[100] H. E. Gerber, "Microstructure of a radiation fog." Journal of the Atmospheric Sciences, vol. 38, no. 2, pp. 454–458, 1981.

[101] J. He, X. Ren, H. Wang, Z. Shi, F. Zhang, L. Hu, Q. Zeng, and X. Jin, "Analysis of the microphysical structure and evolution characteristics of a typical sea fog weather event in the eastern sea of china," Remote Sensing, vol. 14, no. 21, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/21/5604

[102] I. Gultepe, J. A. Milbrandt, and B. Zhou, Marine Fog: A Review on Microphysics and Visibility Prediction. Cham: Springer International Publishing, 2017, pp. 345–394. [Online]. Available: https://doi.org/10.1007/978-3-319-45229-6_7

[103] M. Mazoyer, F. Burnet, and C. Denjean, "Experimental study on the evolution of droplet size distribution during the fog life cycle," Atmospheric Chemistry and Physics, vol. 22, no. 17, pp. 11 305–11 321, 2022. [Online]. Available: https://acp.copernicus.org/articles/22/11305/2022/

[104] F. S. Boudala, D. Wu, G. A. Isaac, and I. Gultepe, "Seasonal and microphysical characteristics of fog at a northern airport in alberta, canada," Remote Sensing, vol. 14, no. 19, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/19/4865

[105] J. Price, "Radiation Fog. Part I: Observations of Stability and Drop Size Distributions," Boundary-Layer Meteorology, vol. 139, no. 2, pp. 167–191, May 2011.

[106] D. Deirmendjian, "Electromagnetic Scattering on Spherical Polydispersions," 1969.

[107] E. P. Shettle and R. W. Fenn, "Models for the aerosols of the lower atmosphere and the effects of humidity variations on their optical properties," Environmental Research Paperr, vol. 676, no. 6, p. 89, 1979.

[108] J. V. Mallow, "Empirical fog droplet size distribution functions with finite limits," Journal of the atmospheric sciences, vol. 32, no. 2 (February, 1975), pp. 440–443, 1975.

[109] F. Tampieri and C. Tomasi, "Size distribution models of fog and cloud droplets in terms of the modified gamma function," Tellus, vol. 28, no. 4, pp. 333–347, 1976.

[110] F. Bernardin, M. Colomb, F. Egal, P. Morange, and J. Boreux, "Droplet distribution models for visibility calculation," in 5th International Conference on Fog, Fog Collection and Dew, no. July, Münster, 2010, pp. 2–5.

[111] T. Bergot, M. Haeffelin, L. Musson-Genon, R. Tardiff, M. Colomb, C. Boitel, G. Bouhours, T. Bourriane, D. Carrer, J. Challet, P. Chazette, P. Drobinski, E. Dupont, J.-C. Dupont, T. Elias, C. Fesquet, O. Garrouste, L. Gomes, A. Guérin, F. Lapouge, Y. Lefranc, D. Legain, P. Morange, C. Pietras, A. Plana-Fattori, A. Protat, J. Rangognio, J.-C. Raut, S. Remy, D. Richard, B. Romand, and X. Zhang, "Paris-Fog : des chercheurs dans le brouillard," La Météorologie, vol. 8, no. 62, pp. 48–58, 2008. [Online]. Available: https://meteofrance.hal.science/meteo-00350944

[112] J. Marshall and W. Palmer, "The distribution of raindrops with size," Journal of meteorology, 1948. [Online]. Available: http://journals.ametsoc.org/doi/abs/10.1175/1520-0469(1948)005{%}3C0165:TDORWS{%}3E2.0.CO;2

[113] N. Desaulniers-Soucy, S. Lovejoy, and D. Schertzer, "The hydrop experiment: an empirical method for the determination of the continuum limit in rain," Atmospheric Research, vol. 59-60, pp. 163–197, 2001, 13th International Conference on Clouds and Precipitation. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169809501001156

[114] G. Feingold and Z. Levin, "The lognormal fit to raindrop spectra from frontal convective clouds in israel," Journal of Applied Meteorology and Climatology, vol. 25, no. 10, pp. 1346 – 1363, 1986. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/25/10/1520-0450_1986_025_1346_tlftrs_2_0_co_2.xml

[115] C. W. Ulbrich, "Natural variations in the analytical form of the raindrop size distribution," Journal of Applied Meteorology and Climatology, vol. 22, no. 10, pp. 1764 – 1775, 1983. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/22/10/1520-0450_1983_022_1764_nvitaf_2_0_co_2.xml

[116] K. Andsager, K. V. Beard, and N. F. Laird, "Laboratory Measurements of Axis Ratios for Large Raindrops." Journal of the Atmospheric Sciences, vol. 56, no. 15, pp. 2673–2683, Aug. 1999.

[117] K. V. Beard and C. Chuang, "A new model for the equilibrium shape of raindrops," Journal of Atmospheric Sciences, vol. 44, no. 11, pp. 1509 – 1524, 1987. [Online]. Available: https://journals.ametsoc.org/view/journals/atsc/44/11/1520-0469_1987_044_1509_anmfte_2_0_co_2.xml

[118] G. Rodgers, G. Poots, J. Page, and W. Pickering, "Theoretical predictions of raindrop impaction on a slab type building," Building Science, vol. 9, no. 3, pp. 181–190, 1974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0007362874900164

[119] A. Tokay and K. V. Beard, "A field study of raindrop oscillations. part i: Observation of size spectra and evaluation of oscillation causes," Journal of Applied Meteorology and Climatology, vol. 35, no. 10, pp. 1671 – 1687, 1996. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/35/10/1520-0450_1996_035_1671_afsoro_2_0_co_2.xml

[120] F. Mook, van, M. Wit, de, and J. Wisse, "Computer simulation of driving rain on building envelopes," in Proceedings of the 2nd European & African Conference on Wind Engineering, Genova, Italy, 22-26 Ju 1997, pp. 1059–1066.

[121] P. Allamano, A. Croci, and F. Laio, "Toward the camera rain gauge," Water Resources Research, vol. 51, no. 3, pp. 1744–1757, 2015. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014WR016298

[122] J. S. Paschkewitz, "Simulation of spray dispersion in a simplified heavy vehicle wake," Lawrence Livermore National Laboratory (LLNL), Livermore, CA, Tech. Rep., 2006.

[123] G. Dumas and J. Lemay, "Splash and spray measurement and control: Recent progress in quebec," in The Aerodynamics of Heavy Vehicles: Trucks, Buses, and Trains, R. McCallen, F. Browand, and J. Ross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 533–547.

[124] D. Ori, T. Maestri, R. Rizzi, D. Cimini, M. Montopoli, and F. S. Marzano, "Scattering properties of modeled complex snowflakes and mixed-phase particles at microwave and millimeter frequencies," Journal of Geophysical Research: Atmospheres, vol. 119, no. 16, pp. 9931–9947, 2014. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014JD021616

[125] J. M. Straka, D. S. Zrnić, and A. V. Ryzhkov, "Bulk hydrometeor classification and quantification using polarimetric radar data: Synthesis of relations," Journal of Applied Meteorology, vol. 39, no. 8, pp. 1341 – 1372, 2000. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/39/8/1520-0450_2000_039_1341_bhcaqu_2.0.co_2.xml

[126] F. S. Marzano, G. Botta, and M. Montopoli, "Iterative bayesian retrieval of hydrometeor content from x-band polarimetric weather radar," IEEE Transactions on Geoscience and Remote Sensing, vol. 48, no. 8, pp. 3059–3074, 2010.

[127] F. S. Marzano, D. Scaranari, and G. Vulpiani, "Supervised fuzzy-logic classification of hydrometeors using c-band weather radars," IEEE Transactions on Geoscience and Remote Sensing, vol. 45, no. 11, pp. 3784–3799, 2007.

[128] E. A. Brandes, K. Ikeda, G. Zhang, M. Schönhuber, and R. M. Rasmussen, "A statistical and physical description of hydrometeor distributions in colorado snowstorms using a video disdrometer," Journal of Applied Meteorology and Climatology, vol. 46, no. 5, pp. 634 – 650, 2007. [Online]. Available: https://journals.ametsoc.org/view/journals/apme/46/5/jam2489.1.xml

[129] W.-H. Dong, X.-E. Sheng, S. Wang, and T. Deng, "Experimental study on particle size distribution characteristics of aerosol for fire detection," Applied Sciences, vol. 13, no. 9, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/9/5592

[130] M. Xiao, C. Lu, H. Wei, and W. Jiang, "Research on dust aerosol particles size distribution," in Proceedings of the 2017 7th International Conference on Applied Science, Engineering and Technology (ICASET 2017). Atlantis Press, 2017/05, pp. 31–37. [Online]. Available: https://doi.org/10.2991/icaset-17.2017.6

[131] Y. H. Park, I. N. Sokolik, and S. R. Hall, "The impact of smoke on the ultraviolet and visible radiative forcing under different fire regimes," Air, Soil and Water Research, vol. 11, p. 1178622118774803, 2018. [Online]. Available: https://doi.org/10.1177/1178622118774803

[132] L.-P. Crevier and Y. Delage, "Metro: A new model for road-condition forecasting in canada," Journal of Applied Meteorology, vol. 40, no. 11, pp. 2026–2037, 2001.

[133] L. Bouilloud and E. Martin, "A coupled model to simulate snow behavior on roads," Journal of Applied Meteorology and Climatology, vol. 45, no. 3, pp. 500–516, 2006.

[134] S. Asfour, F. Bernardin, E. Toussaint, and J.-M. Piau, "Hydrothermal modeling of porous pavement for its surface de-freezing," Applied Thermal Engineering, vol. 107, pp. 493–500, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1359431116310523

[135] Bernardin, Frederic and Munch, Arnaud, "Modeling and optimizing a road de-icing device by a nonlinear heating," ESAIM: M2AN, vol. 53, no. 3, pp. 775–803, 2019. [Online]. Available: https://doi.org/10.1051/m2an/2018056

[136] F. B. Sarah Asfour and E. Toussaint, "Experimental validation of 2d hydrothermal modelling of porous pavement for heating and solar energy retrieving applications," Road Materials and Pavement Design, vol. 21, no. 3, pp. 666–682, 2020. [Online]. Available: https://doi.org/10.1080/14680629.2018.1525418

[137] V. Muzet, J. Bernasconi, P. Iacomussi, S. Liandrat, F. Greffier, P. Blattner, J. Reber, and M. Lindgren, "Review of road surface photometry methods and devices – proposal for new measurement geometries," Lighting Research & Technology, vol. 53, no. 3, pp. 213–229, 2021. [Online]. Available: https://doi.org/10.1177/1477153520958454

[138] UNECE, "New assessment/test method for automated driving (natm) guidelines for validating automated driving system (ads)," 2023. [Online]. Available: https://unece.org/sites/default/files/2023-04/ECE-TRANS-WP.29-2023-44e.pdf

[139] European Commission, "Regulation (eu) 2022/1426 - laying down rules for the application of regulation (eu) 2019/2144 of the european parliament and the council as regards uniform procedures and technical specifications for the type-approval of the automated driving system (ads) of fully automated vehicles - 5 august 2022," Brussels, 2022.

[140] MINISTÈRE DE LA TRANSITION ÉCOLOGIQUE - TRANSPORTS. Arrêté du xxx définissant les conditions d'homologation, d'exploitation et de circulation des navettes urbaines équipées d'un système de conduite automatisé.