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

[DELIVERABLE 2.7] METHODOLOGY, PROCEDURES AND PROTOCOLS FOR EVALUATION OF APPLICATIONS AND VIRTUAL TEST FACILITIES

[LIVRABLE 2.7] MÉTHODOLOGIE, PROCÉDURES ET PROTOCOLES D'ÉVALUATION DES APPLICATIONS ET DES MOYENS D'ESSAIS VIRTUELS

Main authors : D. Gruyer (UGE), R. Régnier (LNE), G. Mouchard (CEA), G. Durand, C. Chaves, S. Prabakaran (AVSimulation), K. Quintero (IRT SystemX), C. Bohn (IRT SystemX), Keilatt Andriantavison (Valeo), Sio-Song Ieng (UGE), Cedric Gava (SPHEREA), Alexandra Duminil (UGE), K. Mann (INRIA), J.-B. Horel(INRIA), A. Renzaglia (INRIA), C. Laugier (INRIA), H. Chabin (UTAC), S. Kierszbaum (Airbus Protect), H-M Hussain (Airbus Protect), F. Bernardin (Cerema), A. Ben Daoued (Cerema), P. Duthon (Cerema), D. Bétaille (UGE), P. A. Laharotte (UGE)

Keywords: Simulation, evaluation, AI systems, validation plan and requirements, validation methods, verification process, evaluation protocol, metrics, KPI, risk assessment, performance indicators, simulation tools, sensor models, traffic simulation, graphical and physic engines, simulation environments, POC

Abstract. This document describes the verification, evaluation, and validation methodologies, procedures, and protocols allowing to calculate both the tools/models and system of systems/ AI-based components levels of performance/quality/representativeness. This methodology will be applied on the 5 POCs defined in the D2.8. A set of relevant and adapted metrics and KPI is proposed and presented.

Résumé. Ce document décrit les méthodologies, procédures et protocoles de vérification, d'évaluation et de validation permettant de calculer les niveaux de performance, de qualité, et de représentativité des outils et des modèles, et des systèmes et composants utilisant des modules et des algorithmes à base d'IA (notamment de Machine Learning). Cette méthodologie sera appliquée sur les 5 POC définis dans le L2.8. Un ensemble de métriques et KPI pertinents et adaptés est proposé et présenté.

Contents

List of Figures

List of Tables

Acronyms

ADS Automated Driving System.

ARTS Automated & Autonomous Road Transport System.

CAV Connected Autonomous Vehicle.

DDS Data Distribution Service.

OD Operational Domain.

ODD Operational Design Domains.

OEDR Object and Event Detection and Response.

1 Introduction

Virtual testing is introduced to reduce the burden of physical tests and effectively provides evidence on the AI performance across the operational domain of a Automated & Autonomous Road Transport System (ARTS) (Connected Autonomous Vehicle). Virtual testing, evaluation, validation, and certification enter a specific design plan adapted from the V-cycle, which is the reference to present the design life cycle of a product such as an ADAS (Advanced Driver-Assistance System) or an Automated Driving System (ADS) (Advanced Driving System) as shown in Figure 1 ([33]). The validation stream is always related to the specification stream, meaning that validation plans are designed concerning the specifications. However, specifying and validating complex systems of systems such as a Connected Autonomous Vehicle (CAV) is a challenging process. To operate validation plans showing a suitable level of safety and reliability with an acceptable time and budget, virtual method tests from MIL (Model-In-The-Loop) to VIL (Vehicle-In-The-Loop) now supplement physical testing: closed site tests and open road tests.

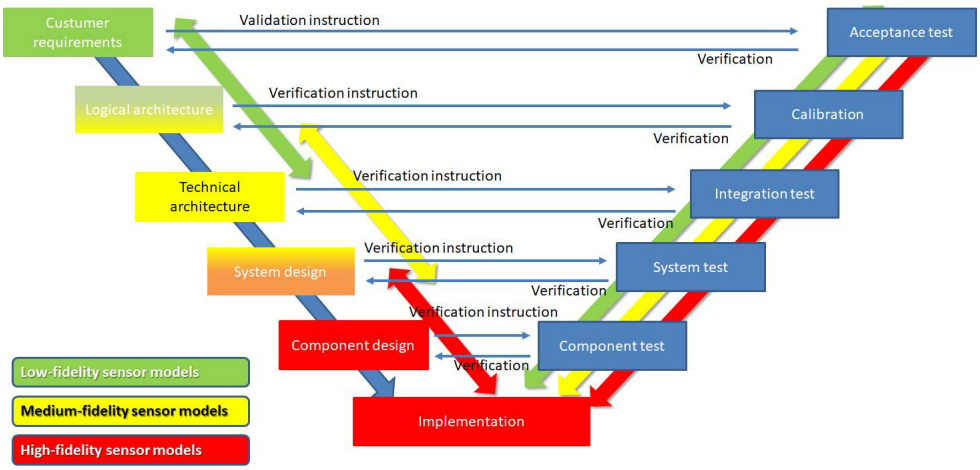


Figure 1: V-cycle for virtual prototyping, test, evaluation, and validation([1])

This deliverable addresses these aspects with 2 points of view:

- The first one is focused on the Verification/Validation of virtual test means for their qualification and performance assessment. This does not in itself address the issue of AI evaluation and certification (which will be addressed in the second aspect), but only the evaluation, verification, and validation of the virtual means used to test AI-based systems or systems of systems. Virtual test means involve tools as well as models. This work has been done in the task T2.5 of the WP2. More accurately, this evaluation will go through the definition of the criteria and metrics allowing both the verification and the evaluation of the simulation platforms as well as the qualification of the level of fidelity to the reality, the relevance and the validity of the ground truths corroborated by the WP3 and WP4. For this, this deliverable proposes to define a set of indicators validating the representativeness and the fidelity of the simulation for the evaluation of ARTS. Concerning the formal approaches, a verification of their properties on the models will be proposed. This will involve the generation of conformance tests, and the validity of the simulation. These actions and relevance of the proposed methodology, procedures, and protocols will be tested with the 5 POCs defined and presented in the deliverable D2.8. For the last POC

(CEREMA and LNE), the results are provided in the deliverable of the WP3 presented the PRISSMA's evaluation and validation methodology for the controlled environment.

- The second one proposes procedures and protocols for the evaluation and validation of AI-based systems under test and using simulation tools (verified, validated, and respected the requirements of WP1). This part corresponding to the task 2.6 aims to propose procedures and protocols for the evaluation and validation of critical AI-based systems and subsystems from the evaluation environments chosen in T2.3 and validated in T2.5 (particularly on the level of realism, the relevance of simulation tools, and the quality of ground truths in terms of evaluation references). The choice of test equipment will of course have to be assessed in the light of the recommendations made in Task 2.5, but we won't deal with this issue here. In addition, this task will address best practice procedures and implementation of experimentation plans using simulation and co-simulation platforms. In order to enable the implementation of the evaluation and validation stages, we will propose definitions, criteria and evaluation metrics for AI-based systems (with the outputs of the other PRISSMA workpackages as well as the collaboration with Pillar 1 of the French program "Grand Défi Sécuriser, fiabiliser et certifier des systèmes dondés sur l'IA"). For the proposed simulation environments, an identification of the optimum scope of use (including dysfunctional through the injection of failures, etc.) will be carried out. These actions will be tested on the 5 POCs defined and presented in the deliverable D2.8 (a fifth POC, straddling WP2 and WP3, was also carried out between CEREMA and LNE on the use of test benches and simulation resources for scenarios in degraded climatic conditions, but will be evaluated as part of WP3). The proposed evaluation and validation process has to take into account metrics and KPI (for example on performance, security...) adapted to the level of complexity linked to the applications (AI-based systems, system of systems, communication and cyber security systems) and the objectives sought.

Partners	POC 1 (BuSAS): UGE' Zoé, XiL approach (real&virtual tracks) (perception/decision/action)	POC 2 : URBAN Driving virtual&real (decision/action)	POC 3 : UTAC VIL virtual&real (perception/path planning/action)	POC 4 : VIL virtual&real (perception)
Types of AI methods	Detection/identification/tracking: yoloV5 + DeepSort++; YoloP, YoloV8 Road marking: Ultra-Fast-Lane-Detection, YoloP		Openpilot	Probabilistic grid method
Sensors	RGB camera, LiDAR	LiDAR, RGB camera, RADAR	RGB camera	LiDAR
Test sites	Satory	Créteil (Paris2Connect)	UTAC	Transpolis
AVS			X	
UGE	X			
LNE				
AIRBUS PROTECT				
ANSYS	X	X		
CEREMA				
ESI	X			
INRIA				X
STRMTG				
SPHEREA	X			
SYSTEMX	X			
TRANSPOLIS				X
UTAC			X	
VALEO		X		
CEA			X	

Figure 2: Table summarising the four POCs implement in the WP2 of PRISSMA and used to test the PRISSMA's verification/evaluation/verification methodology.

2 Method for Verification/Validation of virtual test means and virtual models

2.1 Section objectives

Globally, subsection 2 is dedicated to the definition of the PRISSMA verification and evaluation protocol for the models and tools used in the scope of the simulation platforms, which implies modifications to the classic V protocol presented in the introduction.

The subsection 2.2 will return to some key concepts of systems analysis, which will help us to understand what we actually need to study when validating PRISSMA's virtual test facilities.

The subsection 2.3 is more generic than the PRISSMA framework and illustrates what has been achieved in the aeronautics sector, the only sector outside the automotive industry to have tackled the issue of simulation for certification head on and serve as the state-of-the-art. This example highlights the main guidelines for the validation of virtual test resources in the context of future homologation in another field, and enables us to set out some major principles, such as the adaptation of the V-cycle in particular. This example can serve as a concrete example of how virtual test facilities can be evaluated and this example can evaluate also AI-based virtual testing facilities which makes it even more special and interesting.

Subsection 2.4 will concretely fit in with the PRISSMA project in terms of validation of virtual test facilities. In particular, it deals with the evaluation of each sub-part of an automotive simulator and falls within the scope of the PRISSMA project. This section will go into the technical details of the PRISSMA procedure for validating virtual test equipment.

The final section will will synthesize the PRISSMA procedure presented in the previous sub-section, and will provide a clear vision of the procedure to be followed for the validation of virtual test equipment.

2.2 Virtual testing system analysis

2.2.1 Reminder of system analysis major principles

A system is a group of interacting, interrelated, or interdependent elements forming a complex whole. A System of Interest (SOI) is the system of concern to those who have interest in it.

System engineering is a key discipline in the design and realization of complex systems. At the heart of System engineering lies the system analysis, a process that starts with the crucial definition of the system-of-interest boundaries. This boundary determines what is considered internal to the system, what is external, and what its environment is for the analysis.

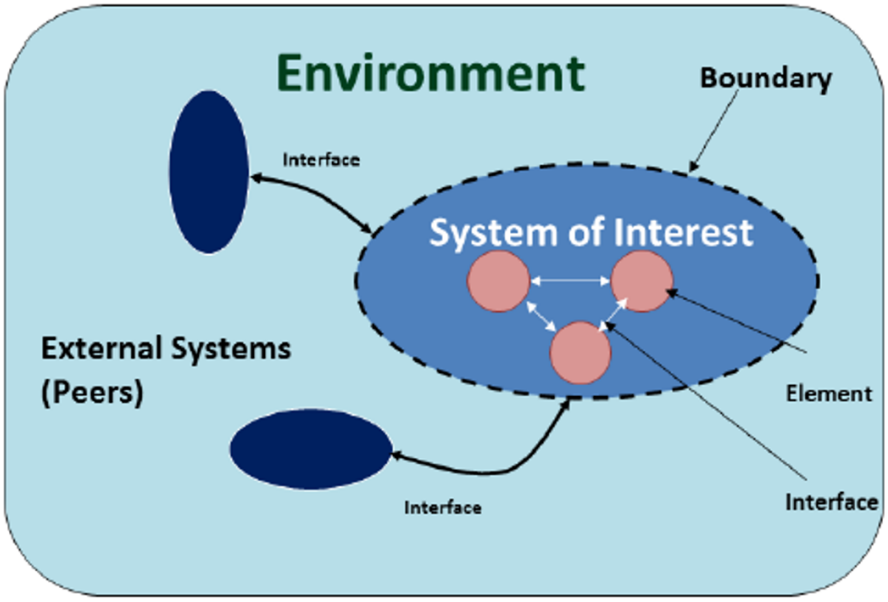


Figure 3: System context

For example, a car can be analyzed as the system of interest. The internal components include the engine, transmission, the braking system the steering system and other internal systems. External to the vehicle are elements such as road infrastructures, other vehicles, and weather conditions, which interact with the automobile but are not part of its intrinsic composition. The environment, meanwhile, encompasses broader aspects such as climate conditions, traffic regulations, and user behaviors, influencing the performance and design of the vehicle without being directly connected to its components.

The definition of these boundaries has major impacts on understanding, designing, and efficiently managing the life cycle of this system of interest, allowing engineers to target key interactions and better anticipate potential challenges.

The concept of system-of-interest is a role attached to a system by stakeholders interested in a particular system. The perception and definition of a particular system, its architecture and its system elements depend on a stakeholder's interests and responsibilities: One stakeholder's system-of-interest can be viewed as a system element in another stakeholder's system-of-interest. Furthermore, a system-of-interest can be viewed as being part of the environment for another stakeholder's system-of-interest Example: For the engineering team designing the ADS of a CAR, the ADS is the system-of-interest, and the Braking System, or the Steering system are parts of the environment of this ADS. For the engineering team designing the braking system, the braking system is the system-of-interest, and the ADS and the Steering system are parts of the environment of the braking system. And for the team designing the whole car, the car is the system of interest, and the ADS, Breaking and Steering system are system's elements

2.2.2 The enabling systems

Some of the systems that belong to the environment of a particular system-of-interest have a particularity: they are not present during the utilisation of the system. Such systems are called Enabling Systems of a given system of interest.

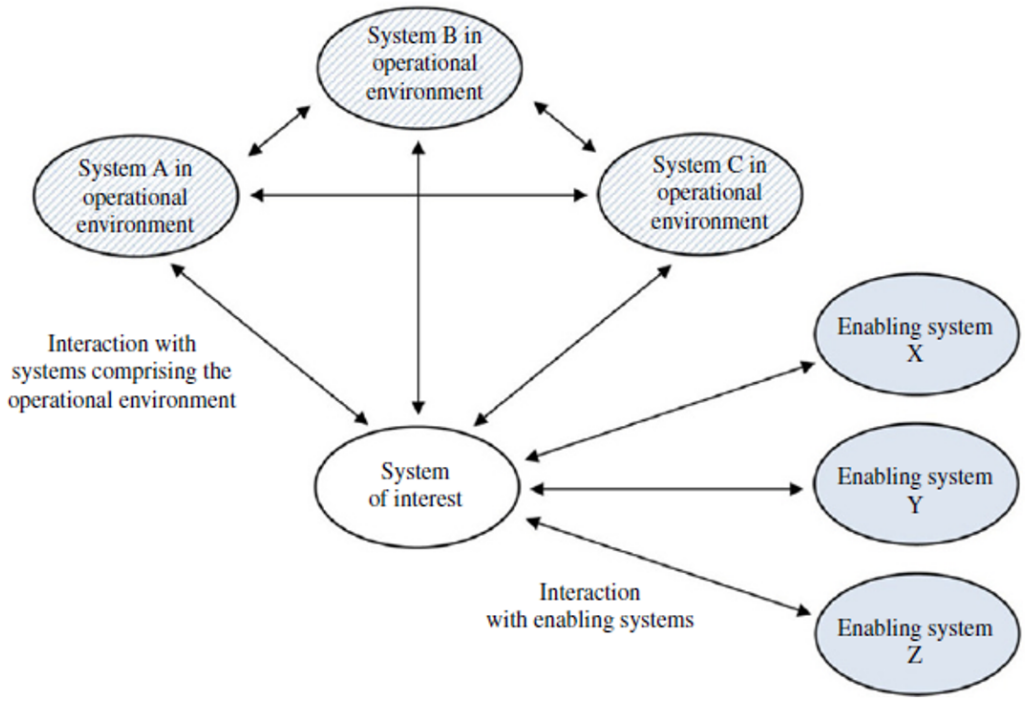


Figure 4: Enabling system [extract from INCOSE SeBOK]

Enabling systems have some specific properties compared to the other systems belonging to the environment of a given system of interest:

1. The end users of a given system of interest have generally no particular interest for the enabling systems, because they don't use it. For example, except for the people loving to repair their cars, no one think to buy a car lift at the same time of the car. The main consequence is that the budget for the enabling systems are usually more underestimated than the budget of the system of interest.

2. In many cases where the system of interest is new, the enabling systems do not exist yet, but they are still required at the early stages of the life cyle of the system of interest. A virtual testing environment or a dedicated software factory are good examples of enabling systems that are required very early in the development stages, and are also usually not available out of the box

One consequence on the systems used for the development (like simulators or test systems), the end users are the developers of the system of interest. This leads to common bias in the engineering activities where the developers of a system of interest also try to develop some of this enabling systems themselves. Very often, the needs statement, specification and architecture activities are rigorously carried out only for the system of interest, but not for the enabling systems, leading to poor designs, technical debt and many more issues. A common example is the creation of complex excel sheets with macro scripts, that, in the end, took days of work for test and development, became critical for the enterprises key activities and cost even more to maintain than to develop.

2.2.3 When an enabling system becomes a system of interest

For the team designing simulators or test systems, the system-of-interest is the simulator or test system itself, and the system needing simulation or tests is part of the environment of this system-of-interest. When specifically addressing test systems, the tested system is called the system Under test.

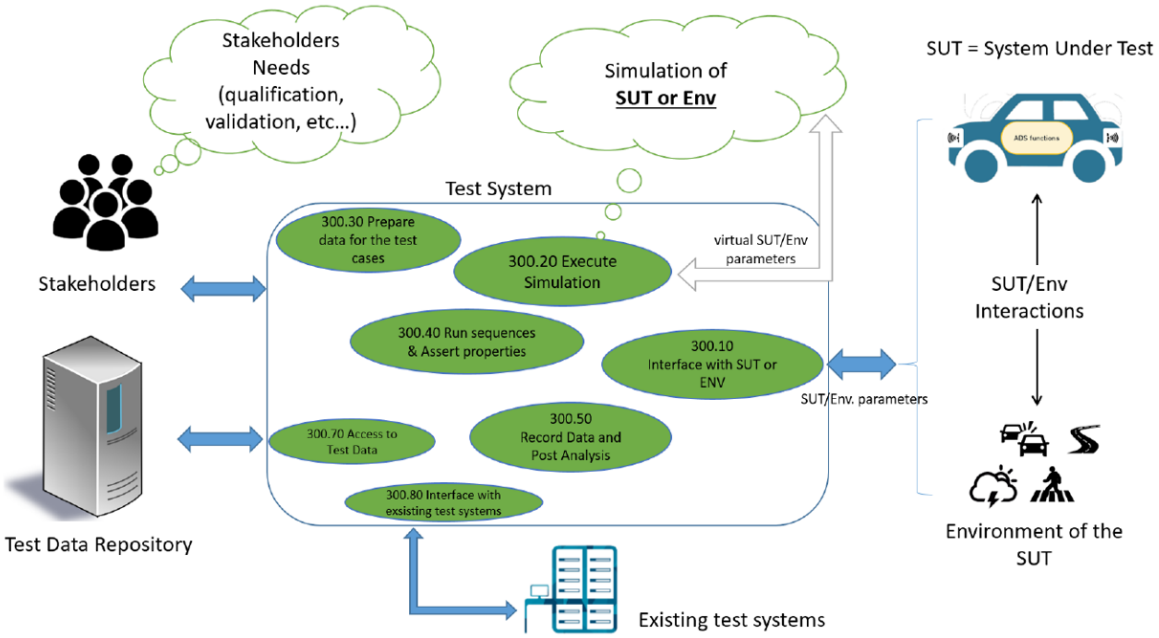


Figure 5: When a test system is the system of interest

The consequences, when describing describing a V&V process using V cycle is that the system of interest targeted by this V&V process must be clearly identified when the system of interest is an enabling system of another system. For example: when dealing with ARTS as a system of interest, the level of decomposition targeted must be identified (the ARTS in its globality, the vehicles, some sub-components like the connected infrastructures or supervision, or a particular vehicle subsystem like perception or decision).

* The insightful reader will have noticed that the system engineering process associated with the development, production, commissioning, operational monitoring, maintenance, and retirement of a particular system of interest are part of the set of enabling systems for the particular system which is the simulator or the test system.

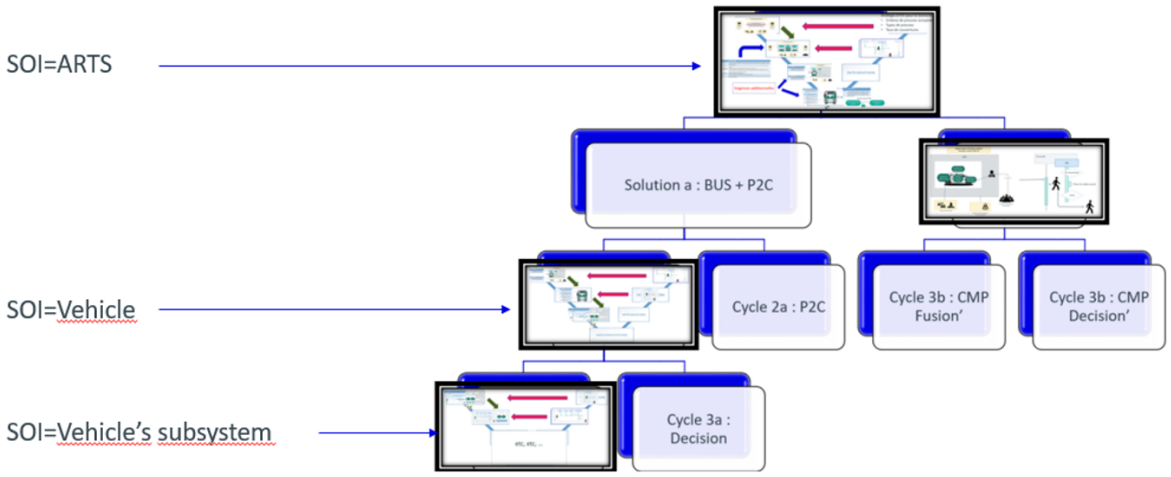


Figure 6: example of nested V processes for the system of interest

Since simulator or virtual testing tools fall in the category of enabling systems of an ARTS or one of its components or subcomponent, it's important, at the first stages of the V&V process of these tools, to verify they have been correctly specified, including the environment diagrams or architecture depicting the systems they are supporting.

And when adressing an enabling system, which can be decomposed into system element when treated as the system of interest, mentioning the original system this enabling system is supporting is mandatory.

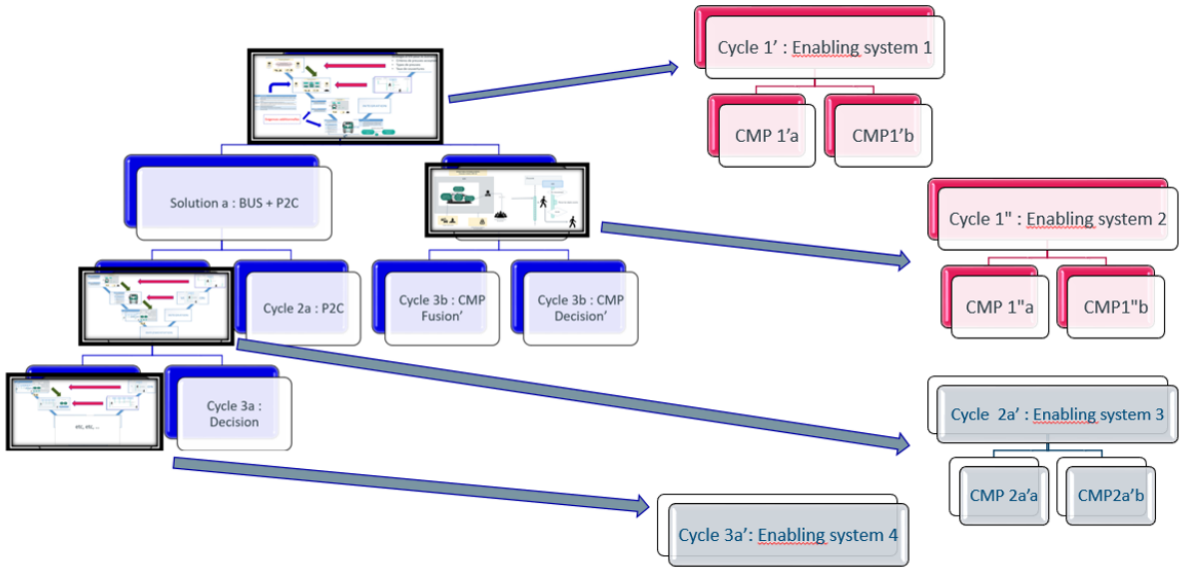


Figure 7: example of associated nested V processes for the enabling systems

Since simulator or virtual testing tools fall in the category of enabling systems of an ARTS or one of its components or sub-component, it's important, at the first stages of the V&V process of these tools, to verify they have been correctly specified, including the environment diagrams or architecture depicting the systems they are supporting.

2.3 Example of the aeronautic methodologies and approaches for verification and validation

2.3.1 Protocol for AI based software

The first question to handle in the certification or qualification of the software tool for AI based application is: are these tools independent of AI? For the purpose of these paragraphs, it has been supposed they are themselves containing AI in order to be closer to the application they are evaluating. Therefore, it seems like we are stuck in a repeated loop. How can an AI-based application be qualified through the use of AI-based software tools?

First, all the best practices used to develop the main AI application remain applicable for the simulation device. Second, the simulation must go through a rigorous safety assessment process that takes into account the severity and the frequency of consequences on the application Operational Design Domains (ODD).

In the following paragraphs, it is assumed that the simulation tools are AI-based. Hence, their development cycle must follow the adapted software engineering cycle. All the components including AI, do follow the "W" approach rather than the classical "V" approach (see. 8).

The following approach is issued from development in aeronautics concerning AI systems qualification and certification of systems including AI-based software modules. The following figure from EASA concept Paper: First usable guidance for Level 1 machine learning applications suggests separating the AI-based subsystem from the classical components. The classical components go through the normal V&V process while the AI based element follows the W shaped cycle (steps in blue). Note: EASA designates European Union Aviation Safety Agency that is a certification organism.

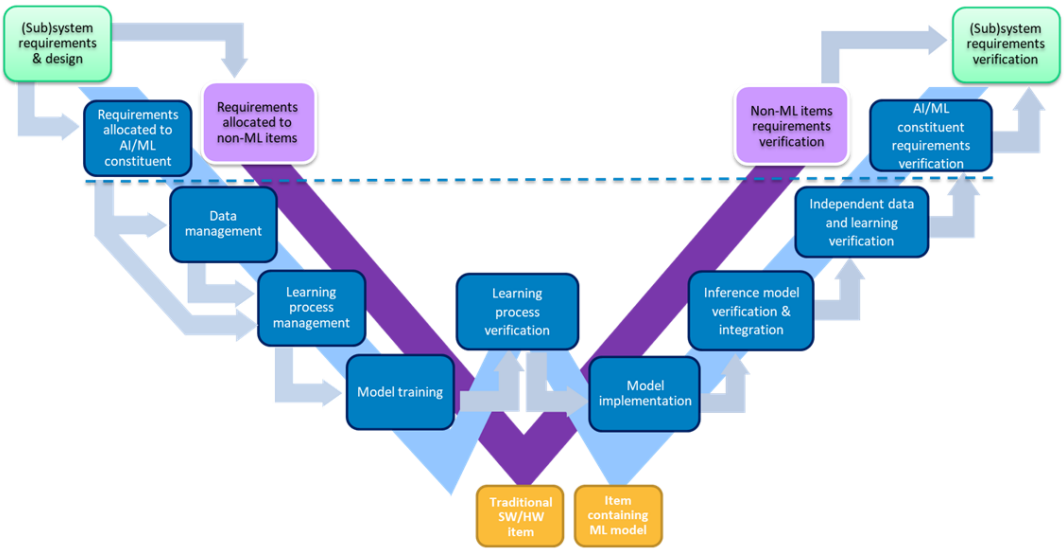


Figure 8: Global view of learning assurance W-shaped process, non-AI/ML content V-cycle process.

This process remains iterative as shown in the figure below. Learning process verification affects requirements, data management, learning process management and model training for the main application that is the automated or autonomous road transport system in the context of PRISSMA (see 9).

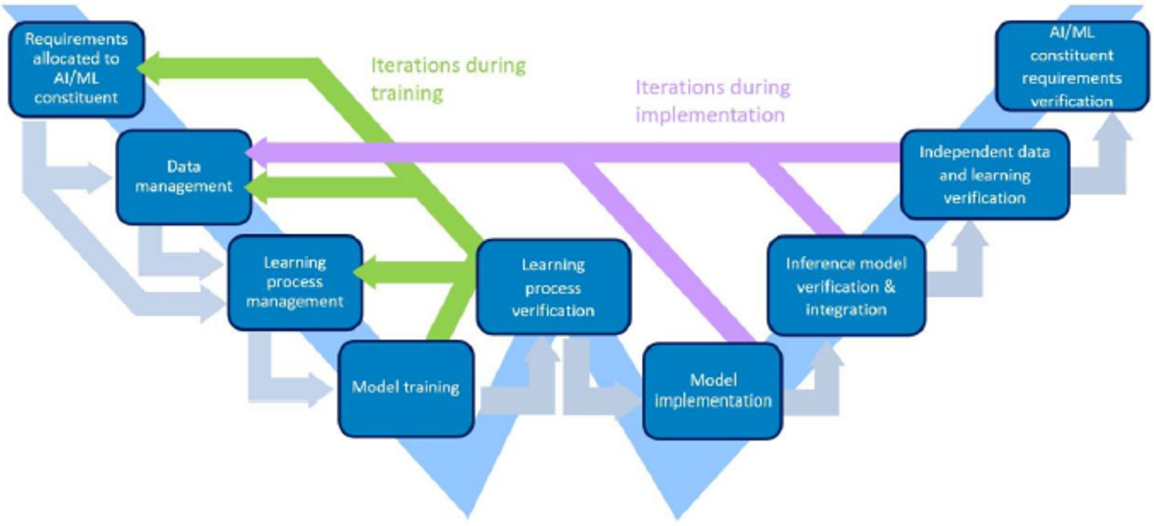


Figure 9: Iterative nature of the learning assurance process.

2.3.2 Example of the application in aeronautic field

Drones including AI have been among the first AI-based aeronautic systems submitted to validation or qualification. When drones entered the airspace, JARUS (Joint Authorities for Rulemaking on Unmanned Systems) devised a safety plan specific for the drones under the name of SORA: Specific Operation Risk Assessment. It allowed drone manufacturers sometimes unacquainted with classical safety methods to have a simple safety method that was still rigorous but user-friendly. The same way EASA concept Paper: First usable guidance for Level 1 machine learning applications (https://www.easa.europa.eu/en/easa-concept-paper-first-usable-guidance-level-1-machine-learning-applications-proposed-issue-01pdf) shares a matrix with requirements referring to steps of W cycle that may be applicable to the ARTS.

The complete matrix is available in the concept paper. This paper does not only shape requirements but also shares examples of Means of Compliance. To build trustworthiness EC (Ethic and Compliance) ethical Guidelines rely on seven criteria: Accountability, technical robustness and safety, oversight, privacy and data governance, non-discrimination and fairness, transparency, societal and environmental well-being. Errors in simulation tool accuracy could lead to accidents with human or material loss. The severity of all errors must be identified and the attention on the simulation must be proportional to the severity of the accidents. Errors in simulation tools, if they are not detected may result in accidents much later. That is why they must be anticipated.

2.3.3 Best practices for AI software accuracy

In the Aeronautics domain, the European Union Aviation Safety Agency (EASA) published the MLEAP deliverable Phase 2 - Interim Public Report on Machine Learning Application Approval (MLEAP). Currently it is the most recent work dealing with validation and qualification of machine learning for multiple transportation domains including aeronautics, automotive, railway, among others. The MLEAP tasks are:

- Task 1: Data completeness and representativity, with handling of the simulator. This task allows to evaluate the level of coverage for the dataset. The goal consists to validate the coverage of the operating domain. In this part, a formal proof is used in order to detect and identify the missing space/data and the bounds of the dataset (Generalisation bounds module). Numalis is the start-up developing this concept.

- Task 2: Model development, through the handling of the generalisation properties (related to the Learning Process Management and Independent data and learning verification steps in the W cycle). This part is the classical training of the model for supervised Ai-based systems.

- Task 3: Model validation, in particular in terms of robustness and stability (related to the Learning Process verification, Interference model verification and integration and Independent data and learning verification steps in the W cycle). In this part, depending of the validation quality of the model, it could be possible to restart the training stage.

The following paragraphs will focus on Task 3 model evaluation. The testing phase will require indicators and metrics. In case of ARTS the simulation tools shall take in account the

nominal behaviour but also degraded behaviour of AI based systems like sensors.

Worst cases, edge's cases, corner cases, rare cases must be "pushed to the limit" to be modelled correctly.

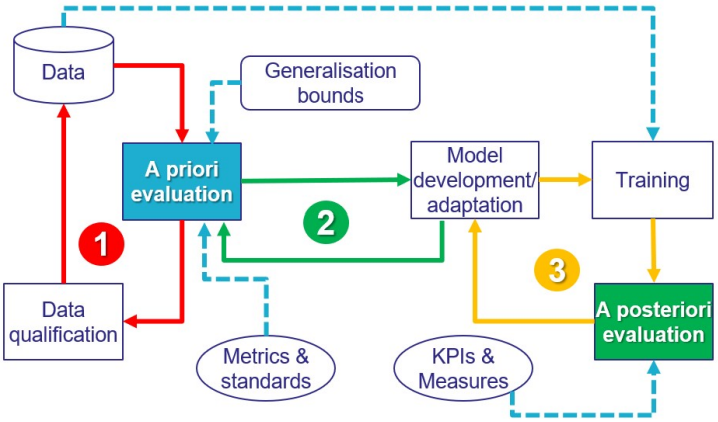


Figure 10: Model evaluation and updating process. 1: A priori evaluation of the coverage of the dataset allowing to validate the operational domain; 2: model development/adaptation with consist to train the model; 3: training; 4: implementation and embedding

The simulator for ARTS AI-based application allow to measure the quality of training of these two applications. The evaluation for ARTS AI based application needs to guarantee that the application AI modules are stable and robust. The reports presents two methods based on performance measures based on empirical data and validation of explicit properties to verify stability.

This report gives access to three methods of evaluating desired generalising ability of ML/DL through the Random labelling, data corruption and finally through evaluation of ML approaches. Random labelling consists of tagging the data with the wrong labels for example labelling a dog picture as an air plane while keeping a set of data that is correctly labelled. Then to run the learning algorithm in parallel to compare the results of a model trained with natural data vs randomised data. "The hypothesis is that if it turns out to be the same in both cases, it cannot even distinguish learning from natural data (where generalisation is possible) from learning on randomised data (where no generalisation is possible)."

Data corruption can also be used to compare the behaviour of a model with natural data vs partially corrupted data, shuffled pixel data, random pixel data and compare learning process and performance evolution. Finally, Machine Learning different characteristics can be evaluated separately. ML correctness, robustness and fairness can be evaluated using tools such as DeepXplore and Themis. The core of ML/DL module can be tested on tools like Tensor Flow and Scikit-learn. Finally the workflow and application scenarios can be evaluated separately.

ML can be tested through adversarial attacks where the aim is to confuse the model to train it is robust against such situations. Data integrity and bias: since data is collected by humans, it may reflect a bias that can remain undetected if the focus of testing is solely on performance.. Behavioural tests can help detect the bias. ML can present failure modes due to performance

bias failures, robustness failures or model input/output failures. These failures should be taken in account and to ensure correct evaluation.

2.4 Description of verification and validation protocol for models and tools: the PRISSMA scope

2.4.1 Platforms, systems and simulation engines

When it comes to simulation-based assessment, the first thing to do is to define the tools required for the simulation platform. In the PRISSMA project, this important task has been addressed by deliverable 2.4 and the future deliverable 2.5 (end-of-program update of 2.4).

To summarise these deliverables, the choice of implementation of this platform is mainly based on the preliminary choices of requirements, ODDs, use cases and scenarios to be addressed. From there, it is possible to choose the different tools, models, simulation/graphical engines and simulation platforms needed to answer the evaluation and validation requirements and constraints. The final stage consists in choosing the evaluation and validation tools, both for the system under evaluation and for the sub-study platform.

To illustrate, Figure 11 shows an example of the simulation environment set up for WP2 POC 1.

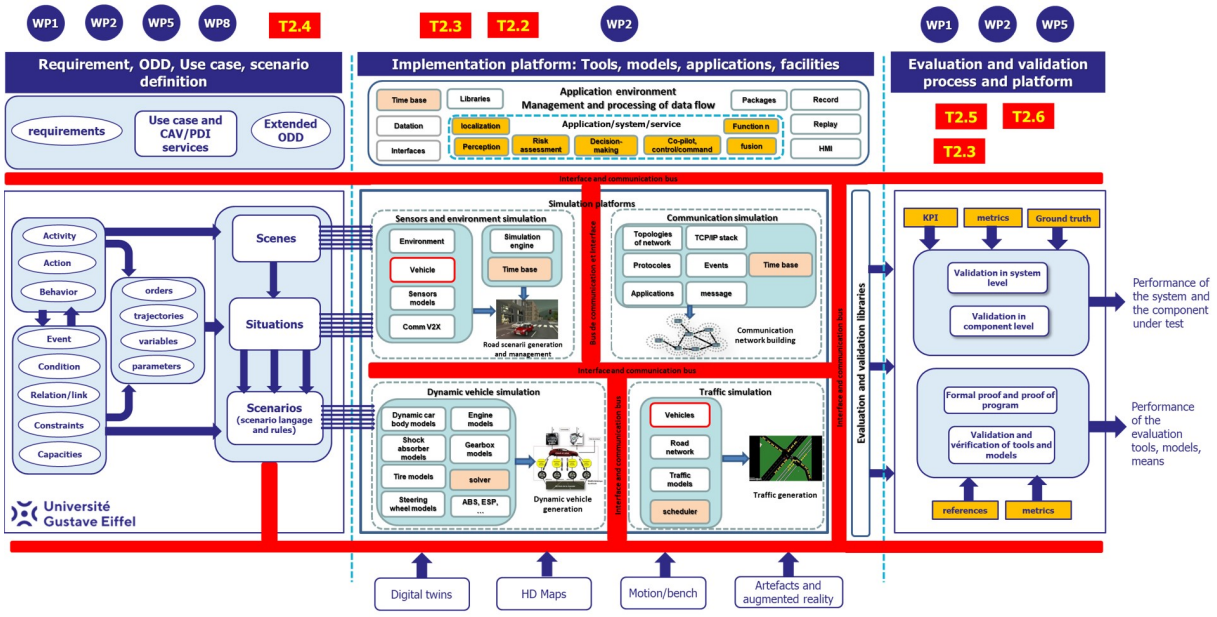


Figure 11: Global view of the simulation environment for evaluation process with its systems, functions, and components

In the rest of this section, we'll look at the various elements of the simulation platform and the means we have to validate and calibrate them.

2.4.1.1 Definition of the types of models and components involved

In this chapter, we will review definitions of different types of models and components commonly used in vehicle simulations:

- **Vehicle dynamics model**:
  - Multi-body dynamics model: represents the mechanical systems of the vehicle, including shock absorbers, chassis, steering, tires, and drivetrain. It simulates the movement, forces, and interactions between these components based on Newtonian physics principles.
  - Powertrain model: simulates the engine, transmission, and other powertrain components, accounting for torque, gear ratios, fuel consumption, and performance characteristics.
- **Environment Model**:
  - Terrain model: represents the road surface and terrain features, including elevation changes, curvature, friction characteristics, and surface irregularities affecting vehicle dynamics.
  - Weather and environmental conditions model: simulates weather conditions (such as rain, snow, fog), lighting, visibility, and other environmental factors that influence vehicle performance and handling.
- **Sensor Models**:
  - 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.
  - Radar model: emulates Radar sensors that use radio waves for object detection and speed measurement.
  - 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...).
  - Navigation models:
    * 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.
    * INS model : inertial Navigation System model. An Inertial Navigation System (INS) is a navigation aid that uses a computer, motion sensors (accelerometers), and rotation sensors (gyroscopes) 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.
    * Odometer model: an odometer model refers to a simulated representation of an odometer, the instrument used to measure the distance travelled by a vehicle.

  - 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.

- **Control Systems Model**: 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).

- **Driver Behaviour Model**: Simulates human drivers' behaviour, including decision-making, reaction times, and driving styles, which influences vehicle operation and response in the simulation.

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

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

- **User Interface and Visualisation Tools**: Interfaces for users to interact with the simulation, visualise data, and analyse results.

- Data Analysis, Validation and Calibrations Tools: Software tools used to analyze simulation results, compare against real-world data, and validate the accuracy and reliability of the simulation models.

2.4.1.2 Verification and validation methods and metrics

This section will focus only on Simulation engines and graphical engines (games engines) with both the verification and validation of 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, ...). More over in a more global view, we address the Verification of the quality of interfaces (data transmission, synchronisation, calculation time).

From [34] 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 specificaly the graphic and simulation engine:

- **R1. Multi-resource constraint**: The platform must support multiple sources of input data to facilitate world creation and scenario generation.

- **R2. 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.

- **R3. 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.

- **R4. Transparent code**: The platform should use open source code as much as possible for control and decision logic.
- **R5. 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.
- **R6. Simulation Fidelity and Quality**: The platform will support physics-based worlds.
- **R7. Sensor Modelling**: The platform must support editable sensor models.
- **R8. 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.
- **R9. References and ground truths**: The platform must provide ground truth data during simulation execution.
- **R10. Ego-Vehicle Control**: The platform must provide the ability to enable a control channel to control the simulated ego-vehicle.
- **R11. 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.
- **R12. Ensure process parallelisation**: The platform must support the ability to run and evaluate multiple control channels simultaneously.
- **R13. Signal scheduler**: The platform must offer the possibility of prioritising control signals between the different channels.
- **R14. Control Signals**: Each control channel must provide control signals conforming to the same specification.
- **R15. Data flow management and scheduler**: The platform must be able to distinguish control signals coming from different channels.
- **R16.Sensors and sources separability**: The platform must be able to distinguish between several sensors of the same type.
- **R17. Unexpected events and rare scenarios**: The platform will support the creation of handcrafted worlds and scenarios, as well as their subsequent adaptation to take into account rare and unexpected scenes and scenarios.
- **R18. Usability of Datasets**: The scenario database must be reusable in the sense that it must be independent of the perception and control logic used.

- **R19. Data channel adaptability**: The connection of the sensors to the control channel must be configurable to adapt to the needs of each channel.
- **R20. 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).
- **R21. Generic scripting language**: The platform must offer the ability to script fault injection in the same interface as test automation.
- **R22. Generic architecture respecting standard**: The platform must support co-simulation standards such as FMI for large and specialised simulations.
- **R23. Reproductibility**: The portability of code generated from the platform should not be limited to a particular hardware configuration.
- **R24. Repeatability**: The platform must provide the same result after n times the same scenario with the same platform configuration.
- **R25. 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.

In [35], a study is proposed on Determinism of Game Engines used for Simulation-based Autonomous Vehicle Verification. This work address a critical issue related to a key requirement for simulation-based development and verification: Is the graphic engine determinism? Indeed, a deterministic process will always produce the same output given the same initial conditions and event history. Thus, in a deterministic simulation environment, tests are rendered repeatable and yield simulation results that are trustworthy and straightforward to debug. However, game engines are seldom deterministic. This work reviews and identifies the potential causes and effects of non-deterministic behaviours in game engines and more specifically in the well known CARLA simulation using Unreal engine from NVIDIA. In this work, a method of a general nature is proposed, that can be used to find the domains of permissible variance in game engine simulations for any given system configuration. The main potential sources of non-determinism into game engines and coming from the literature review ([36]) and found in the investigation made by [35] are the following:

- **The Floating-Point Arithmetic**: The using of floating-point number representation requires rounding due to a fixed bit width. Consequently, floating-point arithmetic lacks associativity, leading to potential variations in results based on the order of execution.
- **Scheduling, Concurrency and Parallelisation**: Runtime scheduling is a strategy for managing resources, distributing computational resources among tasks with varying priorities based on the operating system's scheduler policy. Scheduler policies can be optimised for various objectives, such as maximising task throughput, meeting deadlines, or minimising latency. Maintaining stability in these optimisations across repeated runs

is crucial. However, in scenarios where certain elements of the game loop are multi-threaded or if the scheduler randomly selects from a set of threads with equal priority, it has the potential to disrupt an otherwise deterministic sequence of events.

- **Non-Uniform Memory Access (NUMA)**: In a repetitive test utilising multiple cores and governed by a CPU scheduling policy, memory access time can fluctuate based on the physical proximity of memory to the processor. Generally, a core experiences lower latency when accessing its own memory compared to that of another core, leading to reduced inter-processor data transfer costs. Variations in latency during repeated tests could potentially result in non-deterministic operation of the game engine. This is particularly true if tasks are processed out of sequence with equal priority scheduling or, at the very least, could lead to increased data transfer costs, translating to slower performance.

- **Error Correcting Code (ECC) Memory**: ECC (Error-Correcting Code) Memory is widely employed in commercial simulation facilities and servers to identify and rectify single-bit errors in DRAM memory. These errors can arise from various sources such as malfunctioning hardware, ionizing radiation (including cosmic or environmental sources), or electromagnetic radiation. If left uncorrected, single-bit errors can lead to inaccurate computational results, introducing the potential for non-deterministic behavior due to the probabilistic nature of such errors. Estimating the error rate is a complex task, contingent on factors such as hardware specifications, environmental conditions, and computer cycles.

- **Game Engine Setup**: Consideration should be given to the type and version of the executed engine code, particularly in managing pseudo-random numbers, maintaining fixed physics calculation steps (dt), utilising a fixed actor navigation mesh, and ensuring deterministic ego vehicle controllers. The loading rates of engine textures, especially in perception stack testing, are crucial. In Unreal Editor, performance metrics can be monitored using commands like ”unit” to assess Frame, Game, and Draw times. For effective perception stack testing, control over weather, lighting conditions, and dynamic elements in the game engine is essential. This includes managing reflections from surface water and ensuring that textures are not randomly generated.

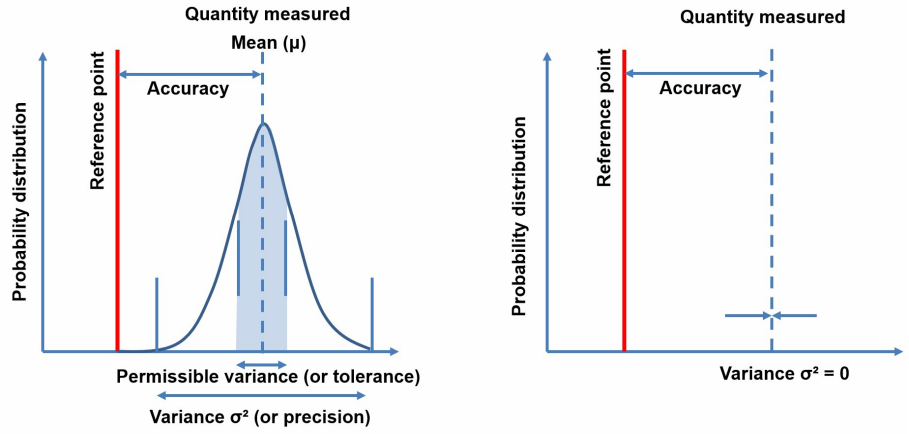


Figure 12: Accuracy, variance, precision and tolerance meaning in the context of non-determinism (right) and determinism (left)

In [36] addressed the issue of detection and identification of the occurrence of intermittently failing in software and simulation platform. While the identification of root causes for such failures has been extensively studied at the unit testing level for open source software, there is a notable gap in research at the system test level, particularly in the context of testing industrial embedded systems. In this study, the authors introduced a novel metric for categorizing test cases as intermittent. Through the analysis of over half a million test verdicts, they discerned between intermittently and consistently failing tests, pinpointing their root causes using various sources. The findings revealed that approximately 1-3% of all test cases exhibited intermittent failures. The study identified 9 factors associated with test case intermittence, including assumptions within: test cases, testing complexity, software or hardware faults, test case dependencies, resource leaks, network issues, random number problems, test system issues, and challenges in code maintenance. These information have been took into account in order to propose a methodology identifying the potential sources of failure and non determinism (issue of repeatability and non respect of tolerance constraints) and are summarise in the figure 14. In order to validate a simulation platform for an evaluation and validation process, it is necessary to first monitor the different potential failure and non determinism sources.

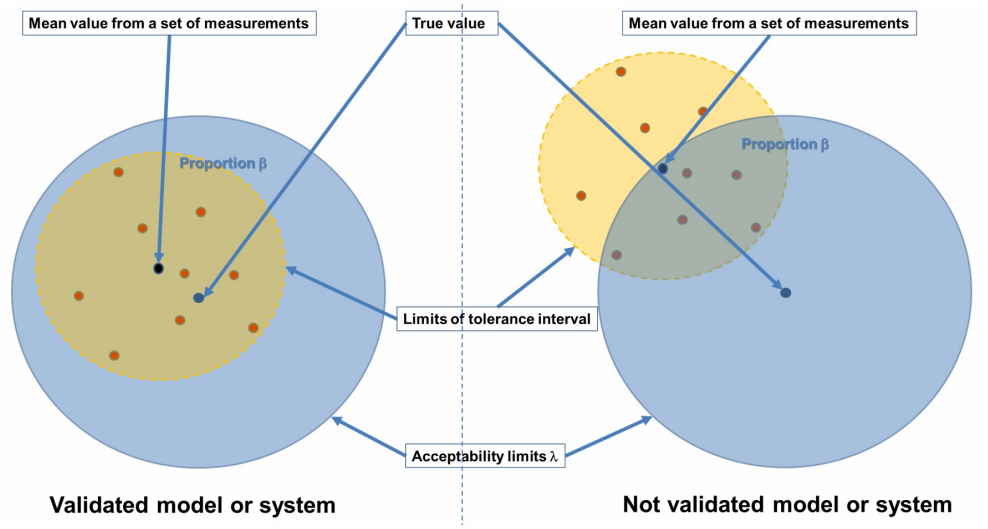


Figure 13: Determinism and non determinism of a simulation system, a question of tolerance and acceptability

Figure 14: Presentation of the different level of non determinism and failures

In [37], the author presents the main requirements and functionalities needed to have a graphic engine usable for evaluation and validation stages. In this report (Figure 1), 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 behaviors, model libraries and hardware dependenciest.

From the different studies and the research work develop in University Gustave Eiffel [38], 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 16)**: 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.
- **Shadows generation and management (see figure 16)**: 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 19), 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 17, noise, blur, fog, Depth of Field, optical deformation, colour, self Exposure, auto focus, ...
- **Spatial management**: The simulation engine need to implement quaternion librarie 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 objets. 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 [39] and are ODE, Bullet, Vortex, and MuJoKo. For the game engine ([40]), the main physic engine are PhysX, Bullet, Havok, MuJoCo, and ODE. In [2], a comparison of these physical engine is made (see figure 15). 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 ressources 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/synchronization 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, climat, ...). 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 [41] which proposes offers a photorealistic 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 (see figure 22) and the crash facilities (foam dummy) (see figure 23).

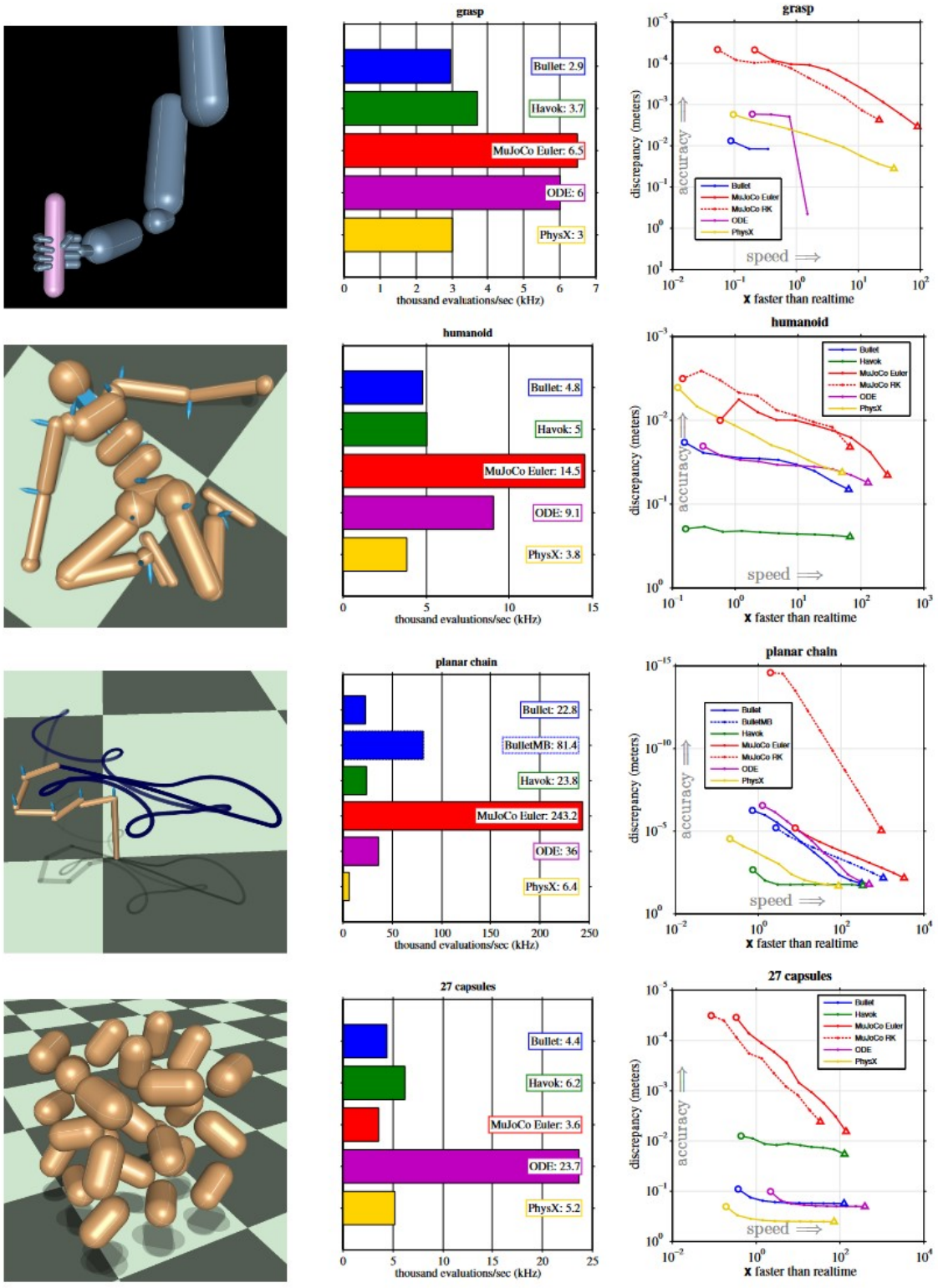


Figure 15: Comparison of the main physic engine used for robotics and game engine. The first column provides the dynamical system. The second column shows raw speed as thousands of evaluations per second for each physic engine. The third column shows the speed-accuracy trade-off in terms of consistency measure. ([2])

Figure 16: Pro-SiVIC, some mechanism allowing to take into account the different types of shadows (catch, cast, self-shading, occluded) and light conditions with light masks allowing to generate realistic headlight with a volumic approach

Figure 17: Pro-SiVIC, some mechanism allowing to take into account the different types of environment reflections (planar, cubic, cylindrical reflections)

Figure 18: LGSVL Simulator with an accurate rendering of light and light mask, shadows, and reflections

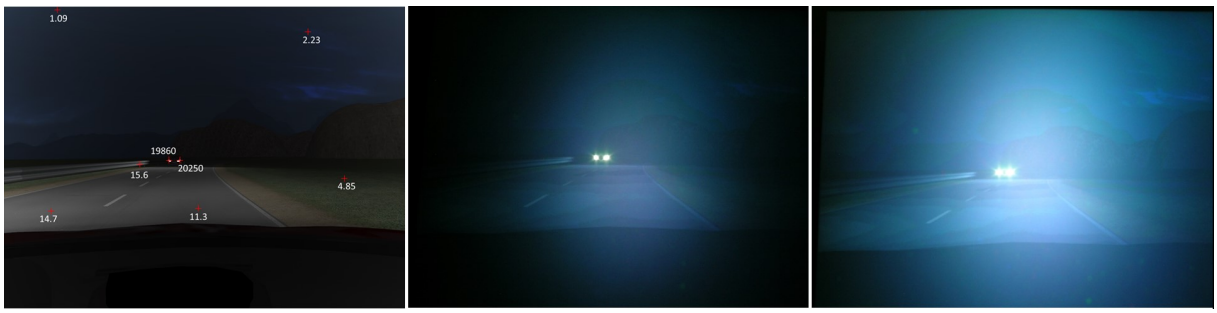


Figure 19: Pro-SiVIC, HDR texture and HDR image generation with a HDR video projector and HDR screen

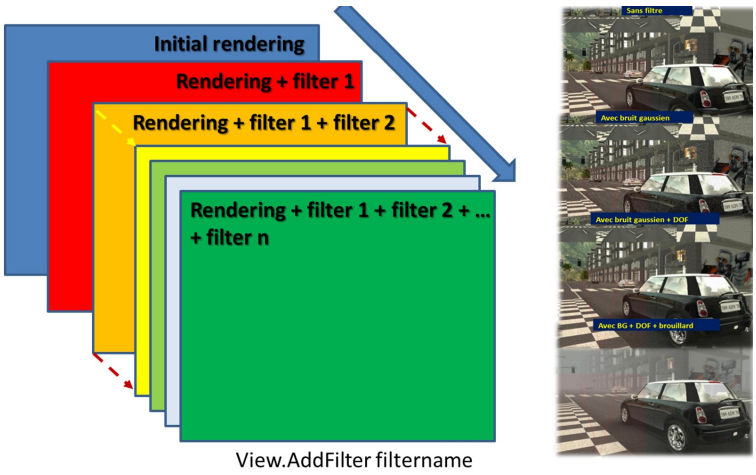


Figure 20: Pro-SiVIC, multiple filter library for the implementation of physical and rendering filters useful for sensor data generation

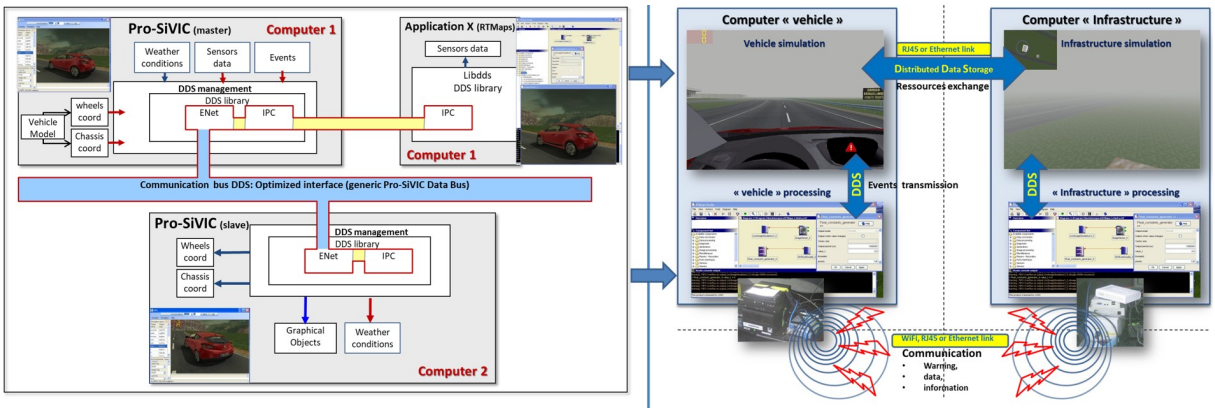


Figure 21: Pro-SiVIC, Dynamic and distributed architecture to share processing on several CPU/GPU or remote computers. Implementation made in DIVA project for fog detection from infrastructure and fog warning service in the ego-vehicle

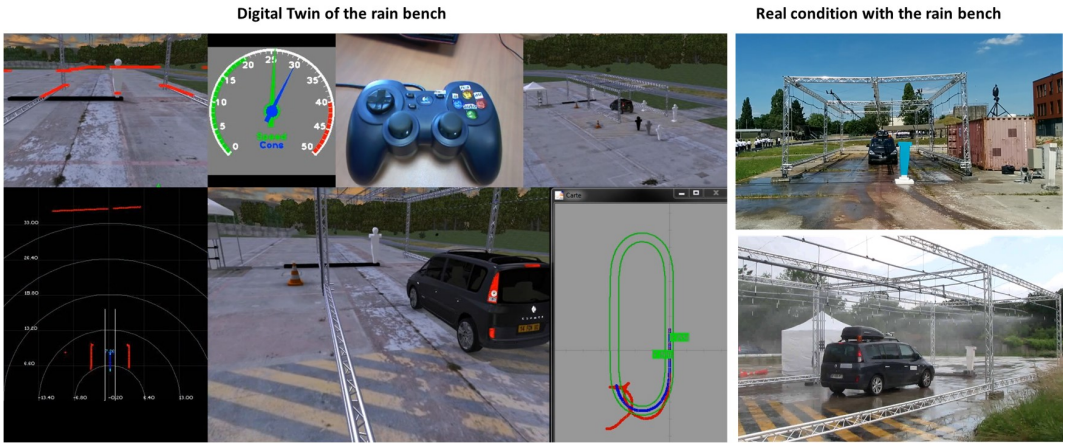


Figure 22: Pro-SiVIC, Digital Twin of the rain bench facility. Righ part illustrate the Digital Twin in Pro-SiVIC, and the righ part provide real picture of this rain bench

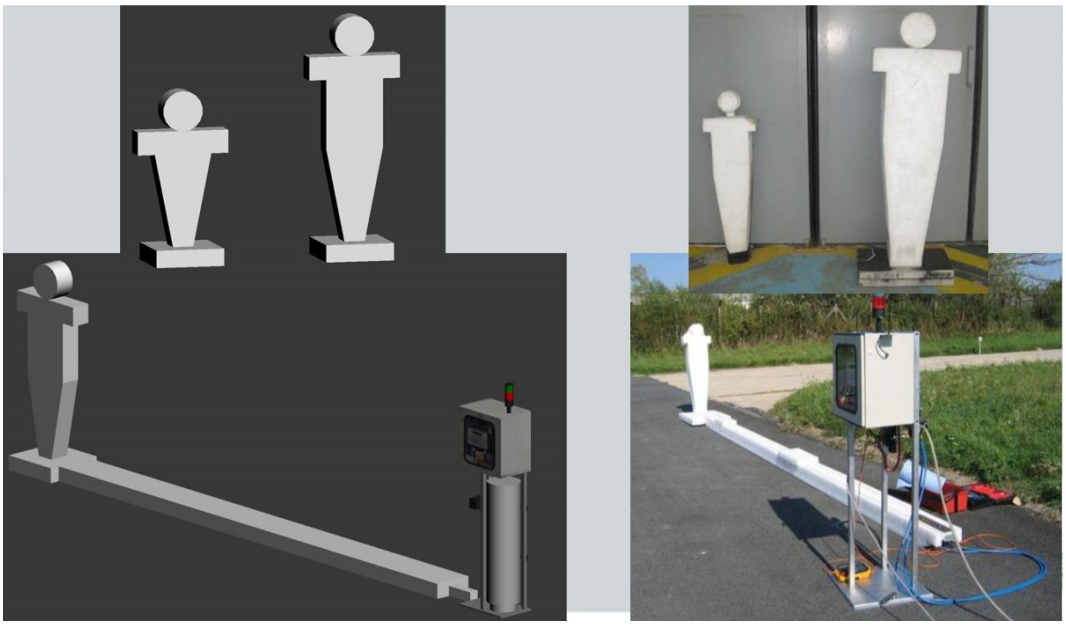


Figure 23: Pro-SiVIC, Digital Twin of the foam dummy used in UGE for crash test and collision mitigation systems. Left part is the Digital Twin of the dummy and the dummy projection system with an IR detector and a hydraulic cylinder. Right part is the real system

The previous section was focused on the graphic engine and simulation engine requirements and constraints to respect in order to be usable for evaluation and validation processes. In the next sections, the different sub part of a global simulation platform will be addressed. It is the case for the sensors, the dynamic modelling of dynamic objects (i.e. vehicle, pedestrian, motocycle, cyclist), the traffic management and generation, the communication means, and the weather conditions.

In summary, in order to apply a simulation engine or a graphic engine in a process of evaluation and validation, it is necessary to verify and validate the following criteria:

- The level of fidelity: its capacity to physically mimic and copy the real conditions involving the key components of the real environment

- The level of quality:
- The level of reproductibility: Ability of the device/architecture/model/platform to provide repeated results, regardless of the operator performing the test (variation between operators). In the PRISSMA framework, it could be the capability to reproduce an evaluation and validation process and platform with different models and tools having the same requirements and capacities/functions to generate equivalent results with an level of acceptability and tolerance. (see figure 13)
- The level of repeatability: This aspect is link to the determinism concept. Schumann et al. describe determinism as the property of causality given a temporal development of events such that any state is completely determined by prior states ([42]). However, in the context of simulation this should be expanded to include not just prior states but also the history of actions taken by all actors. Therefore, a deterministic simulation will always produce the same result given the same history of prior states and actions.
- The level of adaptativity and modularity:
- The level of Time and timestamp management:

Repeatability and reproducibility, commonly called R&R, are link and provide a statistical method used in process control (SPC) to measure the precision and variation present in an evaluation/validation system involving measurers/observers (in our case the platform simulation reproducing a real environment and its conditions in simulation) and the effectiveness of the virtual or real architecture (instrument/system/platform) which is built to use it as an evaluation and validation instrument. As the name suggests, this R&R method has two components, repeatability linked to the ability of the device/architecture/model/platform itself to produce consistent results (accuracy) during repetitive tests and experiments according to the same method, and the reproducibility or ability of the device/architecture/model/platform to provide repeated results, regardless of the operator performing the test (variation between operators). A complete R&R study involves at least 10 reference scenarios (target for a camera) and three different operators conducting the experiments, each carrying out 3 tests on each reference scenario. The total of the 90 tests (for 10 reference scenarios) indicates whether the platform or model meets the tolerance and is acceptable for use as an evaluation and validation system. A short R&R study involves 10 different reference scenarios (or conditions/scenes/situation) and 3 tests on each by a single operator, thus totalling 30 tests.

The tolerance aspect of a model or process is relatively simple. The tolerance limit for a system or model is the deviation permitted from a given dimension, generally specifications and precision ranges given by the manufacturer. Tolerance is equal to the difference between the maximum and minimum limit value expected in the operation of a system or model. For a sensor, the tolerance is directly link to the datasheet and specification of the sensor provider. For a LiDAR, the distance accuracy tolerance is related to the range inacuraccy (5% of the distance for the IBEO LiDAR). A visual explanation is given in the figure 13.

Moreover in [43], the authors present a Comparative Study on the main open source Game Engines (Unity, CryEngine, Unreal Engine). In this context, a great number of initiative are proposed to provide answer to the prototyping, test, evaluation, and validation in specific domain (Robotics, medecine, monitoring, ...). For instance, Isaac Sim 2020 (see figure 24) has been developed for mobile robotics and Omniverse Robotics Experience. This simulator developed by NVIDIA uses UNREAL engine (https://developer.nvidia.com/isaac-sim

). Omniverse Kit is NVIDIA's state of the art platform for simulating the world and natively supports RTX for photo-realism with real-time ray and path tracing, comes with built-in latest PhysX for stable, fast and GPU enabled physics, and supports MDL materials for physically based rendering. Omniverse Kit provides extensible functionality and user-customized UI, python scripting for scene management, and supports workstation, headless and cloud deployment options. Omniverse Connect facilitates robotics scene modelling through plugins for Maya, Revit, SketchUp, Rhino and Unreal Engine 4. Unfortunately, a dedicated testing tool specifically designed for physical simulators is currently lacking. In [44], the authors try to address this issue and propose a falsification framework that can be seamlessly integrated with physical simulators, e.g., Isaac Sim, as well as OpenAI Gym environments.

Figure 24: Isaac Sim based on Unreal engine, RTX for photorealism rendering, MDL materials for physically based rendering, and PhysX for dynamic modelling and objects interactions

2.4.2 Sensor models

Research work on the perception system of an AV generally presents sensors as black boxes. Often the problems that are addressed concern the processing of the data provided by these sensors. However, to evaluate the perception system of an AV or other subsystems using simulation, it is crucial to be able to model the sensors used for the surrounding environment perception with great fidelity in terms of the physical laws that govern their operation. The articles by Francisca Rosique et al. [45] or Jorge Vargas et al. [46] present these sensors from the electromagnetic spectrum in which they operate actively or passively. Indeed, the high fidelity of the simulation is a good mean to verify and validate the representativeness of the simulation compared to the real situation. The first part to evaluate a sensor model is to define the ODD of the sensor and check that the results of using the model are in the ODD. We have to describe the part of the real sensors. Then we have to propose a procedure and metrics allowing to validate the behaviour and the quality of the data provided by the sensor and maybe each subpart of the sensor. In the following sections, the approaches to verify and validate the sensor simulation models most used in an automated vehicle will be presented.

2.4.2.1 High level Verification and validation process and concepts for sensors modelling and simulation

In Figure 25, the term *Reality of Interest* denotes the physical system from which data is acquired. Essentially, it encapsulates the specific problem under investigation, be it a unit prob-

lem, component issue, subsystem intricacy, or the entire system in focus. Take, for instance, the simulation of a GPS sensor – the scrutiny could be centred on a component of the receiver, the receiver's output, the propagation channel, or the holistic system inclusive of the satellite constellation. The Verification and Validation (V&V) process for a comprehensive system necessitates the iterative execution of the steps illustrated in Figure 25. This repetition becomes imperative as the model's development progresses, transitioning from unit-level concerns to addressing the intricacies of the entire system. The *Mathematical Model* component encompasses the conceptual model, mathematical equations, and modelling data essential for characterising the Reality of Interest. Typically, the Mathematical Model adopts the form of partial differential equations (PDEs), constitutive equations, geometry specifications, initial conditions, and boundary conditions. These elements collectively formulate the mathematical framework required to precisely describe the underlying physics of the system under scrutiny. As the cornerstone of the simulation process, the Mathematical Model plays a pivotal role in bridging the conceptualisation of the physical system and the computational framework employed for analysis.

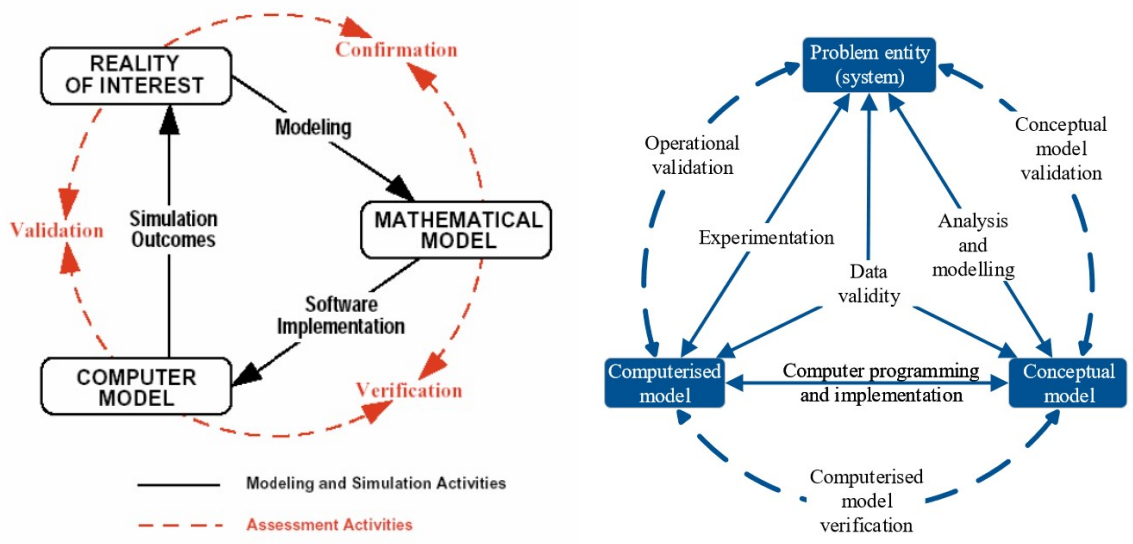


Figure 25: Simplified overview of the verification and validation process of a model.

The Computer Model serves as the embodiment of the Mathematical Model, typically manifesting as numerical discretization, solution algorithms, and various parameters integral to the numerical approximation, along with established convergence criteria. Within the Computer Model, one encounters the computer program (code) itself, underlying conceptual and mathematical assumptions, inputs driving the code, constitutive model and its corresponding inputs, grid size specifications, solution alternatives, and predefined tolerances. Expanding further, the comprehensive representation within the Mathematical and Computer Model may extend to encompass a performance (or failure) model. This inclusion aims to provide insights into the system's behaviour under various conditions, adding a layer of depth to the simulation. Moreover, an uncertainty analysis method, diverse solution options, and specified tolerances may further refine the model's sophistication. The preceding stages of the modelling process involve critical activities. *Modelling* is the meticulous process of selecting vital features and formulating mathematical approximations that aptly represent the intricacies of the Reality of Interest in the Mathematical Model. Following this, *Confirmation* takes precedence, involving

the evaluation of the correctness and fidelity of the modelling choices made. The subsequent Verification activity zooms in on the meticulous identification and rectification of errors within the Software Implementation of the Mathematical Model. This step aims to enhance the reliability and accuracy of the Computer Model by ensuring it aligns precisely with the intended mathematical representation. The meticulous attention to detail during these phases is crucial for the robustness and effectiveness of the overall simulation process.

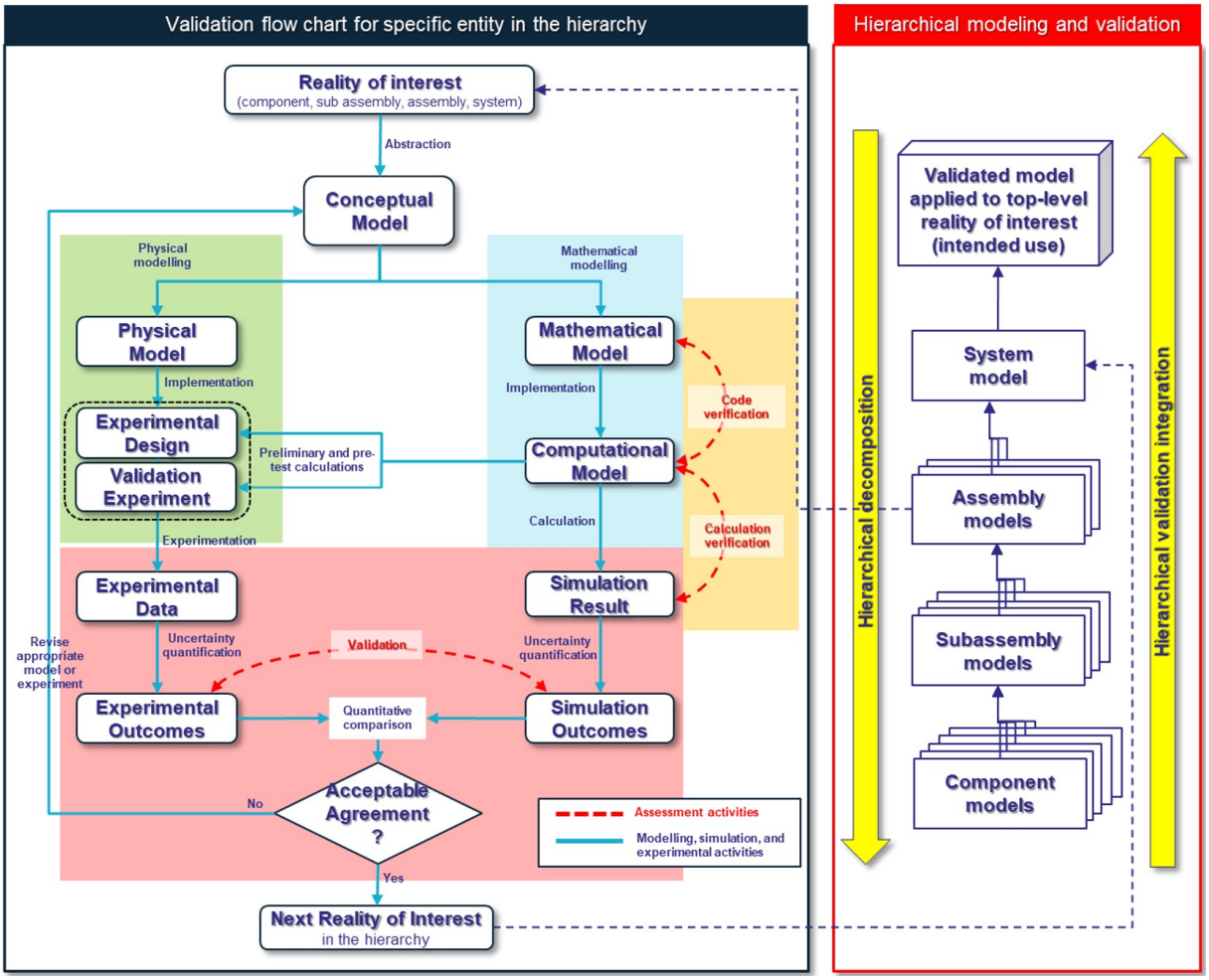


Figure 26: High-level overview of the development, verification, and validation process of a model.

Verification encompasses two distinct yet interconnected activities: Code Verification and Calculation Verification. Code Verification revolves around the meticulous identification and elimination of errors within the software code. On the other hand, Calculation Verification delves into quantifying the errors introduced during the application of the code to a specific simulation. Notably, a critical facet of Calculation Verification involves undertaking a grid or time convergence study. This entails progressively refining the mesh or time step until a satisfactory level of accuracy is achieved. In the culminating phase, the Validation activity is undertaken with the objective of gauging the accuracy of the model. This is achieved through comprehensive comparisons between experimental data and the Simulation Outcomes derived from the Computer Model. It's crucial to recognise that Validation is an ongoing process, evolving as experiments improve or parameter ranges are expanded. Strictly speaking, complete model

validation is challenging; instead, the focus lies in validating specific model calculations or a range of calculations for a particular class of problems, using the code as a reliable tool for simulation. By intertwining Verification and Validation activities, the modelling and simulation process strives for robustness, ensuring that the Computer Model not only operates without coding errors (Code Verification) but also provides accurate representations of the intended physical phenomena in simulated scenarios (Calculation Verification and Validation). This iterative and systematic approach is fundamental to the development of reliable and predictive simulation models.

To establish an effective verification and validation process, it is imperative to articulate and comprehend several key facets:

- **Comprehensive Representation of Activities**: Clearly delineate the diverse activities encompassing the design, execution, and presentation of experimental results. This involves a meticulous overview of the experimental process, ensuring that each step is well-defined and contributes to the overall validation effort.

- **Parallel and Cooperative Roles**: Emphasise the parallel and cooperative roles of both experimentation and simulation. Recognise the symbiotic relationship between these two components, acknowledging how they complement each other in refining and enhancing the overall validation process.

- **Quantification of Uncertainties**: Address the quantification of uncertainties inherent in both experimental and simulation outcomes. This involves a thorough understanding of the potential sources of uncertainties and the development of methodologies to quantify and manage these uncertainties effectively.

- **Objective Mechanism for Improvement**: Establish an objective mechanism for improving the agreement between experiment and simulation. This involves iterative feedback loops that allow for the refinement of simulation models based on insights gained from experimental outcomes and vice versa.

In Figure 26, an intricate depiction of a detailed verification and validation process is presented, encapsulating the interplay of these various aspects. This visual representation offers a holistic overview of the systematic steps involved in ensuring the reliability and accuracy of simulation models through rigorous verification and validation methodologies. The figure serves as a comprehensive guide, illustrating the interconnected nature of the processes and emphasising the importance of a well-structured approach to achieve robust and dependable simulation outcomes.

In Figure 26, the right branch delineates the progression of developing and applying the model, while the left branch outlines the steps involved in acquiring pertinent and high-quality experimental data through physical testing. The closed boxes within the diagram symbolise objects or data, with the connectors represented by black solid lines signifying various modelling or experimental activities. Additionally, the connectors illustrated by dashed red lines denote critical assessment activities integral to the overall verification and validation process. The right branch encapsulates the iterative cycle of model development and utilisation. This involves the formulation of the conceptual model, its translation into mathematical equations, the implementation of a computer model, and the subsequent exercise of this model to simulate the intended physical system. This iterative loop ensures the refinement and improvement of the model over

successive cycles. Conversely, the left branch elucidates the steps associated with obtaining reliable experimental data. This process encompasses the design and execution of physical tests, the collection of relevant data, and the meticulous validation of these experimental outcomes. The continuous interaction between experimentation and simulation, represented by the connectors, underscores the cooperative nature of these two realms in refining and validating the overall model. The connectors in dashed red lines highlight the crucial assessment activities that bridge the two branches. These activities involve a comprehensive evaluation of the agreement between experimental data and simulation outcomes, allowing for the identification of disparities and the subsequent refinement of the model to enhance its accuracy and reliability. The figure provides a visual road-map, emphasising the interconnected nature of these processes and underscoring the significance of a well-integrated and systematic approach in achieving robust verification and validation outcomes. The Mathematical Model depicted in Figure 1 undergoes a detailed breakdown into a Conceptual Model and a Mathematical Model, as illustrated in Figure 26. An ideal scenario involves collaborative development by both the model developer and experimenter. The process of developing the Conceptual Model is intricate, involving a comprehensive consideration of various factors:

- **Computational Objective**: Clearly define the computational objective, outlining the specific goals the model aims to achieve within the simulation framework.
- **Agreement Level Requirements**: Determine the required level of agreement between experimental and simulation outcomes, establishing the benchmarks for success in aligning these two realms.
- **Domain of Interest**: Identify the precise domain of interest, encapsulating the specific spatial and temporal boundaries that the model will encompass.
- **Physical Processes and Assumptions**: Enlist all significant physical processes and assumptions that need to be incorporated into the Conceptual Model to ensure a comprehensive representation of the real-world scenario.
- **Failure Mode Considerations**: Address the failure mode of interest, emphasising potential scenarios where the system might deviate from expected behaviour, allowing for a robust analysis of critical conditions.
- **Failure Mode Considerations**: Address the failure mode of interest, emphasising potential scenarios where the system might deviate from expected behaviour, allowing for a robust analysis of critical conditions.
- **Validation Metrics**: Define validation metrics, specifying the quantities to be measured and establishing the basis for comparison between experimental data and simulation outcomes.

Collaborative efforts between the model developer and experimenter during the development of the Conceptual Model contribute to a more holistic and accurate representation of the underlying system. By addressing these key aspects, the Conceptual Model serves as the foundational framework guiding subsequent steps in the modelling and simulation process.

Following the development of the Conceptual Model, the modelling process proceeds with the construction of the Mathematical Model, while concurrently, the experimenter formulates

the design for the Validation Experiment. The Mathematical Model represents a set of mathematical equations specifically crafted to articulate the intricacies of physical reality. In the realm of mechanics, this model encompasses fundamental components such as:

- **Conservation Equations**: Inclusion of conservation equations governing mass, momentum, and, in certain cases, energy. These equations provide a quantitative framework for understanding the fundamental principles underlying the physical system.
- **Spatial and Temporal Domain Specification**: Definition of the spatial and temporal domain that the model will encapsulate. This delineates the geographical and time-related boundaries within which the mathematical representation operates.
- **Initial and Boundary Conditions**: Specification of initial conditions, representing the state of the system at the onset of the simulation, and boundary conditions, outlining the interactions with the external environment.
- **Constitutive Equations**: Integration of constitutive equations that define the material properties and behaviours, contributing to a more nuanced understanding of how the system responds to various stimuli.
- **Uncertainty Relationships**: Incorporation of relationships describing the uncertainty inherent in the model. This acknowledgement of uncertainty is crucial for a comprehensive analysis, particularly when dealing with real-world complexities.

Simultaneously, the experimenter undertakes the design of the Validation Experiment, a crucial step in aligning the model with empirical observations. This experiment serves as a benchmark, offering empirical data to validate and refine the Mathematical Model. The cohesive development of the Mathematical Model and the execution of the Validation Experiment form a symbiotic relationship, fostering a robust and accurate representation of the physical system. This iterative process allows for continuous refinement and improvement, ensuring that the model aligns closely with the intricacies of the real-world scenario under investigation. The Computer Model serves as the practical realisation of the equations formulated in the Mathematical Model, typically manifested through numerical discretization, solution algorithms, and convergence criteria. This implementation involves translating the mathematical abstraction into a tangible form that can be processed by a computer. The Computer Model is essentially a numerical procedure, often employing finite element, finite difference, or similar techniques, to solve the equations outlined in the Mathematical Model through the execution of a computer code. In the context of complex sensor modelling like RADAR, the codes employed are equipped with methodologies for discretizing equations and modules both in space and time. This involves breaking down the continuous mathematical representation into discrete elements, facilitating computational efficiency. Additionally, these codes incorporate algorithms specifically designed for solving the resultant approximate equations. This intricate process ensures that the Computer Model is not just a static representation but a dynamic tool capable of simulating the behaviour of the physical system over both space and time. Moreover, the Computer Model plays a pivotal role in transforming abstract theoretical concepts into actionable insights. The numerical methods utilised, whether finite element or finite difference, enable the computation of complex interactions, providing a practical framework for analysing mechanical systems. This integration of computational tools with mathematical principles enhances the

model's ability to simulate real-world scenarios accurately, contributing to the overall efficacy of the simulation process.

Within the *Computer Model*, a segment is dedicated to incorporating nondeterministic solution methods, uncertainty characterisations, and the corresponding convergence criteria. This aspect introduces a layer of complexity to the simulation process, acknowledging and managing uncertainties inherent in the modelling. Various nondeterministic theories (theories of uncertainties), such as probabilistic methods, fuzzy and possibilistic sets, interval theory, and evidence theory, find application in this domain, allowing for a more nuanced representation of uncertainty. The uncertainties within the model are characterised by the chosen methodology to represent them. For instance, a probability distribution might be employed to capture the variability in data generated by a sensor, or intervals could be used to delineate bounded inputs. This approach forms the foundation of Uncertainty Quantification, a crucial step in the simulation process aimed at quantifying the impact of all input and model form uncertainties on the computed simulation outcomes. In addition to the primary model response, Simulation Outcomes now encompass more than just a deterministic result. They include quantified error or confidence bounds, providing a comprehensive understanding of the potential variations in the model's predictions. This extension enhances the interpretability/explainability/understanding of simulation results, offering a measure of the uncertainty associated with the computed model response. The inclusion of uncertainty considerations and quantification not only reflects the realism of real-world scenarios but also facilitates more informed decision-making based on a thorough understanding of the simulation's inherent uncertainties.

The Computer Model undergoes rigorous assessments in the form of Code and Calculation Verification to systematically identify and rectify potential errors within the programming. These assessments aim to enhance the reliability and accuracy of the model by addressing various aspects, including insufficient grid resolution, solution tolerances, and considerations related to finite precision arithmetic. The next 4 stages have to be addressed in the Computer Model verification process:

- **Code Verification**: Code Verification focuses on the identification and elimination of errors in the programming itself. This involves a meticulous examination of the code to ensure that it accurately reflects the intended mathematical model. Common issues such as syntax errors, logical inconsistencies, and programming oversights are identified and rectified during this phase.

- **Calculation Verification**: Calculation Verification delves into the numerical aspects of the Computer Model. It addresses issues related to grid resolution, ensuring that the discretization adequately represents the underlying mathematical equations. Solution tolerances are scrutinised to guarantee that the numerical solution aligns with the desired accuracy. Additionally, considerations regarding finite precision arithmetic, where numerical errors may arise due to the limitations of computer arithmetic, are carefully evaluated and mitigated.

- **Enhancing Precision**: The assessments involve strategies to enhance precision and minimise numerical errors. Techniques such as refining grid resolution, adjusting solution tolerances, and employing advanced numerical methods contribute to the overall improvement of the model's accuracy.

- **Iterative Refinement**: Code and Calculation Verification are iterative processes, where identified errors lead to refinements in the model. The goal is to create a robust and

accurate representation of the physical system, minimising discrepancies between the simulated outcomes and the intended mathematical model.

By systematically addressing programming errors, numerical considerations, and precision-related challenges, Code and Calculation Verification ensure that the Computer Model aligns closely with the intended mathematical representation. This iterative refinement process contributes to the creation of a reliable and effective simulation tool for analysing complex systems. This step is essential and critical in the new complex models of sensors sharing the codes on CPU, GPU, and remote computers.

Within the experimental domain depicted on the left side of Figure 26, the inception and design of a physical experiment give rise to what is termed a Validation Experiment. The fundamental objective of a Validation Experiment is to furnish the requisite information essential for validating the model. This necessitates a meticulous approach where all assumptions are not only acknowledged but are comprehensively understood, well-defined, and rigorously controlled throughout the experiment. To facilitate this precision, the undertaking of Pretest Calculations, which may encompass sensitivity analyses, proves invaluable. These calculations serve multiple purposes, such as identifying optimal locations for measurements within the experiment and determining the most effective types of measurements required. The insights gained from these calculations are pivotal in streamlining the experimental design, ensuring that the gathered data will be not only relevant but also instrumental in validating the corresponding model. The dataset acquired from a Validation Experiment extends beyond mere response measurements. It encompasses a comprehensive array of measurements, including those crucial for defining model inputs and elucidating uncertainties associated with loadings, initial conditions, and boundary conditions. For instance, the symmetrical placement of sensors within the experiment allows for the quantification of obstacle/tracking/recognition variabilities for a RADAR sensor. Concurrently, conducting multiple validation experiments aids in quantifying test-to-test variations, contributing to a more robust understanding of the experimental setup. By embedding this meticulous approach into the experimental design phase, researchers can leverage both the Validation Experiment and Pretest Calculations to not only validate the model but also to uncover critical insights into the variability and uncertainties inherent in the physical system under study. This comprehensive understanding is pivotal for ensuring the accuracy and reliability of both the experimental setup and the subsequent model validation process.

The connection denoted by Pretest Calculations in Figure 26 serves as a vital link bridging the experimental and computational branches. This connection underscores the critical interaction between the modeller (sensor model provider) and the experimenter, emphasising the necessity for ongoing collaboration to guarantee that the measured data not only aligns with expectations but is also indispensable, pertinent, and accurate for the modelling process. Throughout the Pretest Calculations phase, it is imperative for the modeller and experimenter to engage in a collaborative dialogue. This ensures a mutual understanding of the experimental requirements and the information essential for validating the computational model. The intricacies of the physical system, the model assumptions, and the expected outcomes need to be clearly communicated, fostering a shared vision that enhances the likelihood of a successful and informative experiment. However, once the Validation Experiment and Pretest Calculations are concluded, a shift occurs, and the modeller and experimenter often proceed independently. This phase allows each party to focus on their specific domain of expertise. The modeller refines and advances the computational model, incorporating insights gained from the experimental data. Simultaneously, the experimenter may delve into post-experiment analyses, ensuring the accu-

racy and reliability of the acquired data. The subsequent independent work serves to optimize both the experimental and computational components. When the time arrives for comparing outcomes from the experiment and the simulation, the independence fostered in the interim phase facilitates a more objective and comprehensive assessment. This iterative and collaborative approach, with distinct phases of interaction and independence, ensures that the final validation process is robust, accurate, and mutually beneficial for both experimental and computational aspects of the study.

The experimental phase (*Experimentation*) encompasses the systematic gathering of raw data from diverse *reference* and *ground truth* sensors deployed in the physical experiment. In addition to the real sensor to test (real version of the simulated sensor), the sensors used as "ground truth" must be accurate enough, different from the sensor under test, and need to be calibrated. The outcome of this data collection process is the generation of Experimental Data, which may comprise a multitude of information such as strain measurements, time histories of responses, as well as visual records. In certain instances, experimental data can be further transformed into distinct "features" to enhance its direct utility for comparison with simulation results. To fortify the robustness of the experimental findings and to facilitate the quantification of uncertainties inherent in the experimental setup, it is often imperative to conduct repeat experiments. These repetitions serve a dual purpose: firstly, they aid in quantifying the lack of repeatability attributed to systematic errors or biases inherent in the experimental apparatus, and secondly, they help in capturing uncontrollable variabilities that may influence the outcomes. In the pursuit of experimental validation, it is crucial to consider not only the primary measurements but also the derived features that can enhance the compatibility with simulation results. Moreover, the inclusion of repeat experiments underscores a commitment to rigorously assessing the reliability and consistency of the experimental outcomes. This comprehensive approach ensures that the experimental data, in its various forms, is not only rich in information but is also well-characterised in terms of uncertainties, contributing to a more robust and insightful comparison with the results obtained from computational simulations.

Following the experimental phase, Uncertainty Quantification becomes a pivotal step in systematically assessing and quantifying the impact of various sources of uncertainty on Experimental Outcomes. This process involves a comprehensive evaluation of factors such as measurement errors, design tolerances, as-built uncertainties, and fabrication errors, among others. The outcome of Uncertainty Quantification typically manifests as Experimental Outcomes presented alongside associated error bounds, elucidating the range of uncertainty as a function of time or load. It is noteworthy that Uncertainty Quantification is emphasised on both the left and right branches of Figure 26, underscoring its crucial role in elucidating uncertainties and instilling confidence in both the experimental and simulation outcomes. This dual consideration ensures a holistic understanding of the uncertainties inherent in both domains. The subsequent phase involves the Quantitative Comparison of Experimental and Simulation Outcomes, a process essential for drawing meaningful insights from the validation exercise. This comparison extends beyond a mere visual assessment and takes the form of a statistical statement regarding selected validation metrics. For instance, if the chosen metric involves the difference between simulation and experimental outcomes (commonly referred to as "error"), the Quantitative Comparison articulates the expected accuracy of the model. An example statement might be, "We are 98% confident that the error is between 1% and 5%." This statistical approach offers a nuanced understanding of the model's performance and establishes a quantifiable measure of confidence in the alignment between experimental and simulation outcomes. This aspect imply the using of the tolerance and acceptability bounds using. In essence, the tandem application

of Uncertainty Quantification and Quantitative Comparison fortifies the validation process, ensuring a comprehensive evaluation of uncertainties and providing a statistically sound basis for assessing the agreement between experimental and simulation results.

The assessment of Model Validation is a critical step in determining the accuracy of a model as a representation of the real world, particularly in the context of its intended applications. This evaluation seeks to answer whether the model achieves an Acceptable Agreement with the experiment, signifying alignment with predefined criteria specified in the Conceptual Model. However, it is crucial to recognise that the question of adequacy for the intended use is more expansive than the decision block depicted in Figure 26. The decision point for Acceptable Agreement concentrates solely on the level of concordance between Experimental and Simulation Outcomes. This alignment is assessed against predefined criteria embedded in the Conceptual Model, providing a quantitative measure of how well the model reflects the real-world behaviour as observed in the experiment. In instances where the agreement falls short of acceptability, there are two potential avenues for improvement: Model Revision and Experiment Revision. Model Revision involves adjusting fundamental aspects such as assumptions, structure, parameter estimates, boundary values, or initial conditions to enhance alignment with experimental outcomes. Conversely, Experiment Revision entails modifying aspects of the experimental design, procedures, or measurements to improve agreement with simulation outcomes. The decision on whether to revise the model, the experiment, or both is a nuanced judgement made by the model developer and experimenter. This process requires a careful consideration of the nature of discrepancies, insights gained from the validation process, and the overall objectives of the study. It is through this iterative and collaborative decision-making that the model evolves and refines, progressively becoming a more accurate and reliable representation of the real-world system for its intended applications.

In [4], the authors provide a succinct examination of existing Verification and Validation (V&V) methods commonly employed in traditional simulations (see figure 27). Building upon this comprehensive review, they introduce a pioneering framework specifically tailored for the V&V of simulations designed to predict the behaviours of autonomous robots. To demonstrate the efficacy of this novel framework, the authors apply it to a practical use-case involving the navigation task of an autonomous Unmanned Ground Vehicle (UGV). In the application of the framework, the authors focus on the crucial aspect of model validation, specifically targeting sensor models associated with the Global Positioning System (GPS), inertial measurement unit (IMU), and RGB camera. This validation process is essential for ensuring the accuracy and reliability of these sensor models in capturing real-world data and translating it into simulation outcomes. Furthermore, the framework extends its application to the validation of these sensor models within the context of a specific camera-based autonomous navigation algorithm—stop sign detection. This additional layer of validation demonstrates the versatility and adaptability of the proposed V&V framework to address diverse functionalities within autonomous robotic systems. By delving into the intricacies of sensor models and their application in a real-world scenario like autonomous navigation, the authors contribute to advancing the validation methodologies specifically tailored for autonomous robots. This research not only underscores the importance of rigorous V&V processes in autonomous systems but also provides a practical and applicable framework for enhancing the reliability of simulations in predicting the behaviours of autonomous robots.

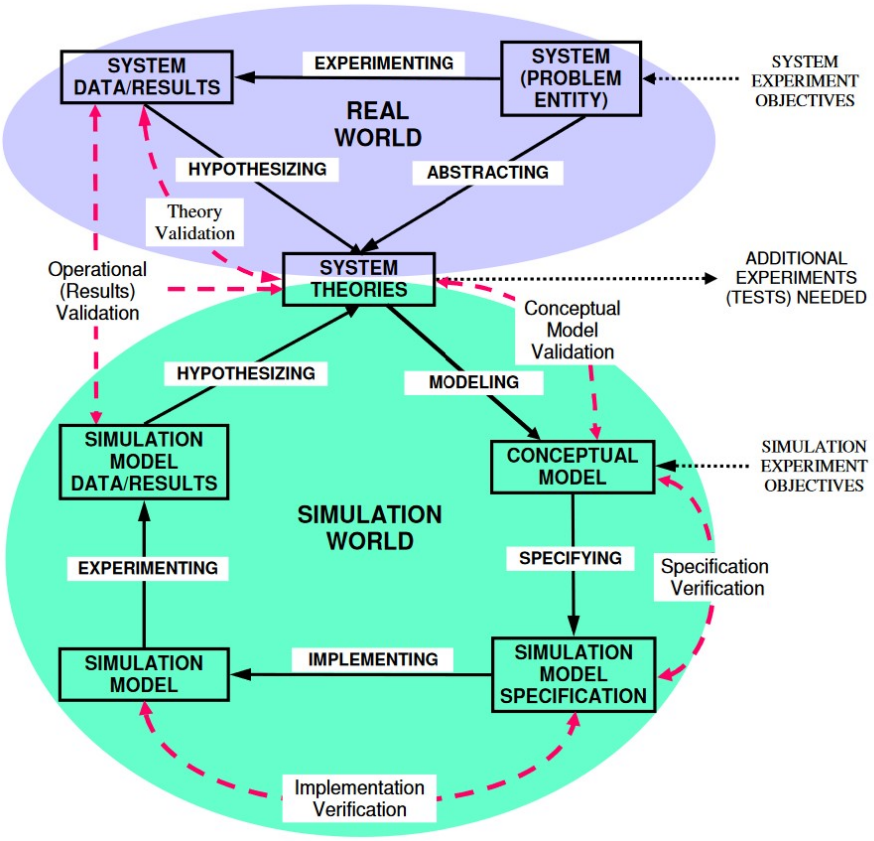


Figure 27: Real World and Simulation World Relationships with Verification and Validation, and development cycle proposed by [3] and used in [4]

In 2023 and in the referenced work [5], the authors introduce a methodology aimed at facilitating the establishment and using of assumptions within the modelling process. The methodology encompasses a structured approach for conducting simulation projects, an associated assumptions taxonomy, and a method designed to streamline the management of these assumptions. Additionally, the authors advocate for an expansion of the conventional Validation, Verification, and Accreditation processes, proposing a comprehensive framework comprising eight distinct phases (as illustrated in Figure 28). These phases collectively span the entire life cycle of a model, offering a more thorough and integrated approach to ensure the reliability and accuracy of simulation outcomes. This methodology is mainly focus on the proposal of a Continuous Process for Validation, Verification, and Accreditation of Simulation Models.

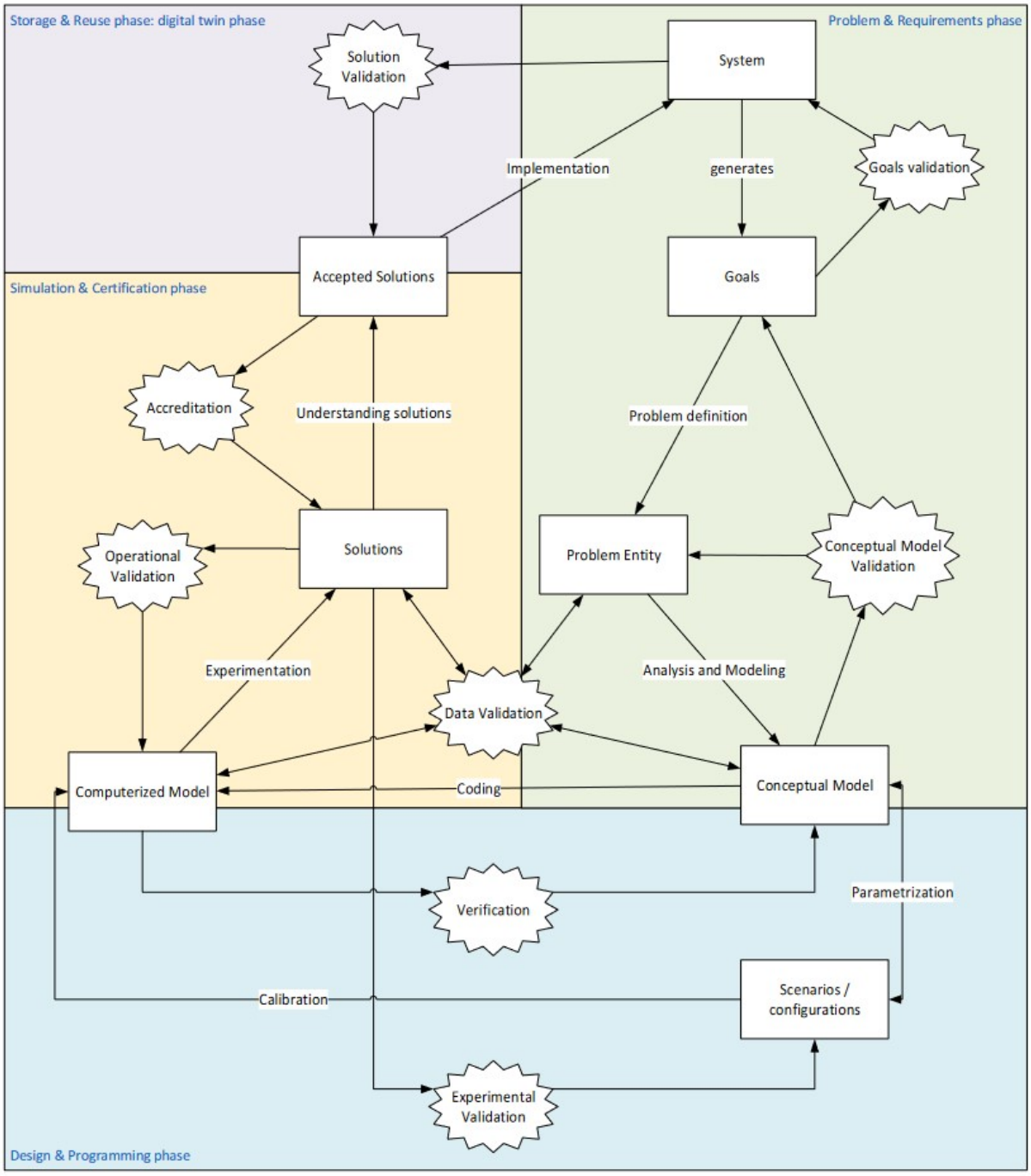


Figure 28: Life cycle for a simulation project based on 8 distinct phases involving Verification, Validation, and Accreditation. Work proposed by [5]

The proposed life cycle for a simulation project is delineated through a visual representation in Figure 28. Within this schematic, squares signify various products, encompassing elements such as the definition of the problem, model coding, and obtained solutions. Validation and verification actions, denoted by stars, are integral to ensuring the accuracy of these products. If at any point the validation or verification process fails, the project iterates back to the preceding product. Navigating from one product to another involves specific actions, exemplified by processes like coding to transition from the *Conceptual Model* to the *Computerised Model*. The

cyclical nature of this process is mapped across different phases as proposed by [19].

In Figure 28, eight key products pertinent to the simulation project are represented within squares:

- **Goals**: The elucidation of relevant elements governing the organisation, constituting a standalone product.
- **Problem Entity**: The definition of the problem slated for analysis, generated by the organisational goals.
- **Conceptual Model**: A comprehensive and unambiguous representation of the model under consideration.
- **Scenarios/Configurations**: The diverse scenarios or configurations earmarked for analysis.
- **Computerised Model**: The simulator responsible for coding and embodying the model.
- **Solutions**: Outcomes derived through simulation.
- **Accepted Solutions**: Solutions validated and accepted through model-based discussions.
- **System Itself**: The system is modified based on the solutions accepted through the model-based discussions.

This structured approach guides the evolution of a simulation project through distinct phases, ensuring a systematic and iterative progression while maintaining a focus on validation, verification, and continual refinement of the modelled system.

In [6], the author discerns that the omission of uncertainties, limited information in validation results, restricted extrapolation capability, and the ensuing narrow application scope are impeding the contemporary validation processes from meeting evolving requirements. Through a thorough analysis scrutinising over twenty frameworks, this work elucidates that statistical methods possess significant potential to address these four critical inadequacies. The study extensively justifies that adopting consistent statistical validation is not only necessary but also important and crucial for the precise quantification of reliability. This precision, in turn, facilitates accurate model selection, knowledge development, and decision-making within the realm of modern automotive vehicle-dynamics simulations. The incorporation of statistical methods is posited as a transformative approach, bridging gaps and enhancing the robustness of the validation process to meet the heightened demands of contemporary simulation requirements.

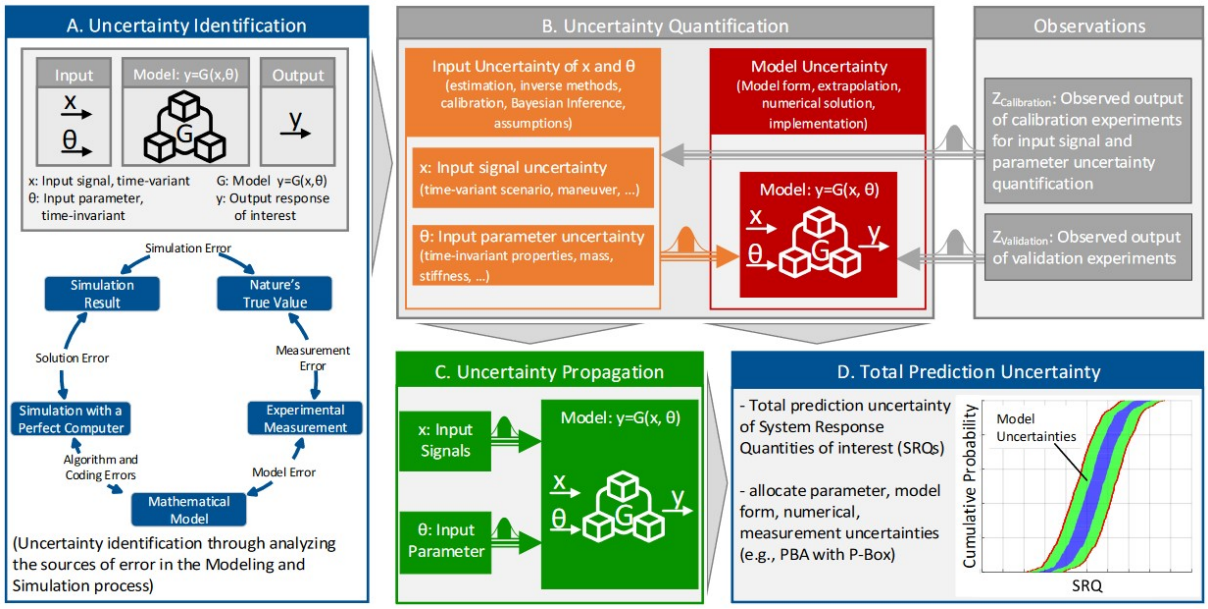


Figure 29: Overview of statistical validation showing the four main steps A. uncertainty identification, B uncertainty quantification, uncertainty propagation and D total prediction uncertainty. Work proposed by [6]

2.4.2.2 Some requirements, properties, and concepts about synthetic data verification and validation

In order to encapsulate the full breadth of applications and approaches to synthetic data, [47] proposes the following definition for synthetic data:

Definition: Synthetic data is data that has been generated using a purpose-built mathematical model or algorithm, with the aim of solving a (set of) data science task(s).

We contrast synthetic data with real data, which is generated not by a model but by real world systems (e.g camera image, radar signal, GPS frame etc.). The model – the synthetic data generator – can take many forms, from a real time graphical engine with rendering process, deep learning architectures such as the popular Generative Adversarial Networks (GANs), or Variational Auto-encoders (VAEs), through agent-based and bio-inspired models, to a set of (stochastic) differential equations modelling a physical or/and dynamical system.

From [47], it appears that a good synthetic data generator (for the simulation of data generated by specific sensor model reproducing actual data provided by actual sensor) should simultaneously satisfy the following properties:

- **Syntactical accuracy**: The generated data should be plausible (e.g. a synthetically generated road environment need to involve a road surface with signage, road side furniture, and dynamic or static elements above the road surface). However, this also requires that certain structural/physical properties of the data are preserved. This means to guarantee a physical, spatial, and temporal coherence. For example, with time-series data, it is mandatory to ensure that data points are not generated using information coming from the extrapolation process or knowledge from the future.

- **Privacy**: It should be possible to accurately quantify the amount of information about the original data revealed by the diffusion of the synthetic sample of data. Exactly how

one measures privacy will depend on the specific task at hand and the type of synthetic data that will be generated. Although differential privacy is a popular way to assess the amount of information disclosed through synthetic data generators, a different notion may be necessary when data is sparse or one wishes to move away from the limits of worst case. For example, the use of datasets that have not been properly anonymity and are used to build synthetic data generation models could, when using the model, generate data containing confidential and potentially critical information. Additionally, information produced by the generator must not inadvertently regenerate confidential information.

- **Statistical accuracy**: It should be possible to accurately quantify the statistical similarity (or lack thereof) between the synthetic and the original data. When measuring statistical accuracy, one might be interested in capturing certain marginal distributions and certain relationships between variables, but not others. A good synthetic data generator should allow for control over this. In order to quantify the level of fidelity and quality of a synthetic data relatively to the actual one, it is necessary to apply a set of metrics defined in the next sections.

- **Efficiency**: The algorithm should scale well with the dimension of the data space (i.e. feature space). It is well known that, in general, approximation of distributions can suffer from the curse of dimensional, and consequently sampling from unstructured distributions is an NP-hard problem.

While it is relatively straightforward to design algorithms that satisfy a subset of the desired properties for Synthetic Data Generation (SDG), there is currently no systematic framework to achieve all four properties simultaneously. Challenges arise, particularly in the conflict between generating statistically accurate and private data. In the realm of deferentially private synthetic data, it is demonstrated the non-existence of a computationally efficient algorithm that simultaneously ensures differential privacy and preserves correlations between pairs of features. This holds as a general result, implying that there is no universal algorithm that works for all datasets. However, it is possible that:

- for a specific application (e.g., dataset), it is possible to efficiently generate DP synthetic data;

- one may not be interested in the correlations being preserved (i.e. the application may demand a different fidelity notion).

Despite the impossibility of a universal deferentially private synthetic data generation method, it emphasises the necessity to evaluate privacy and fidelity on a case-by-case basis. The absence of a "one-size-fits-all" solution is underscored. In a related study (reference [48]), the authors demonstrate that a computationally efficient algorithm exists by relaxing the requirement to match most correlations instead of all. While promising, this raises the challenge of quantifying which aspects of the data structure are not being matched. These findings emphasise the importance of tailoring synthetic data generation to specific use cases. It suggests that for a given application, relevant statistical properties can be preserved, while others may be disregarded in favor of privacy. Concrete examples of such use cases are provided, accompanied by a caution against misguided endeavours in synthetic data applications.

In order to provide a label on a synthetic data in a order to be 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.
  - Metrics: Evaluation involves comparing statistical measures (mean, variance, correlations, etc.) between the original and synthetic datasets.
- **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.
  - Metrics: Assessment involves various criteria, such as the ability of synthetic data to maintain privacy guarantees, support intended analyses, and align with the goals of the data generation task.
- **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.
  - Metrics: Entropy: Measures the unpredictability or disorder in the dataset, reflecting the diversity of information; Coverage: Evaluates the extent to which the synthetic dataset covers the entire spectrum of possible values and scenarios present in the real data; Distribution Matching: Compares the statistical distributions of features in synthetic and real datasets to ensure similar patterns; Representativeness: Assesses how well the synthetic data represents the diversity of instances, ensuring a balanced portrayal; Variability: Measures the extent of variation in features, emphasising diversity in different dimensions; Cluster Analysis: Examines the formation

of distinct groups or clusters within the synthetic data, indicating diverse subgroups; Augmentation of Rare Events: Ensures that rare or uncommon events in the real data are appropriately represented in the synthetic dataset.

in [7], the authors propose a comprehensive 3-dimensional metric, (α-Precision, β-Recall, Authenticity), designed to evaluate the fidelity, diversity, and generalisation capabilities of generative models across diverse application domains. This metric combines statistical divergence measures with precision-recall analysis, allowing for both sample- and distribution-level assessments of model fidelity and diversity. Additionally, it introduces generalisation as a crucial dimension for evaluating how well a model reproduces training data, especially relevant in scenarios involving sensitive information. The three components of the metric are interpretable probabilistic quantities, and their estimation can be achieved through sample-level binary classification. This sample-level nature inspires a novel application called model auditing, where individual sample quality from a (black-box) model is assessed, leading to the identification and removal of low-quality samples, ultimately enhancing the overall model performance in a post-hoc manner.

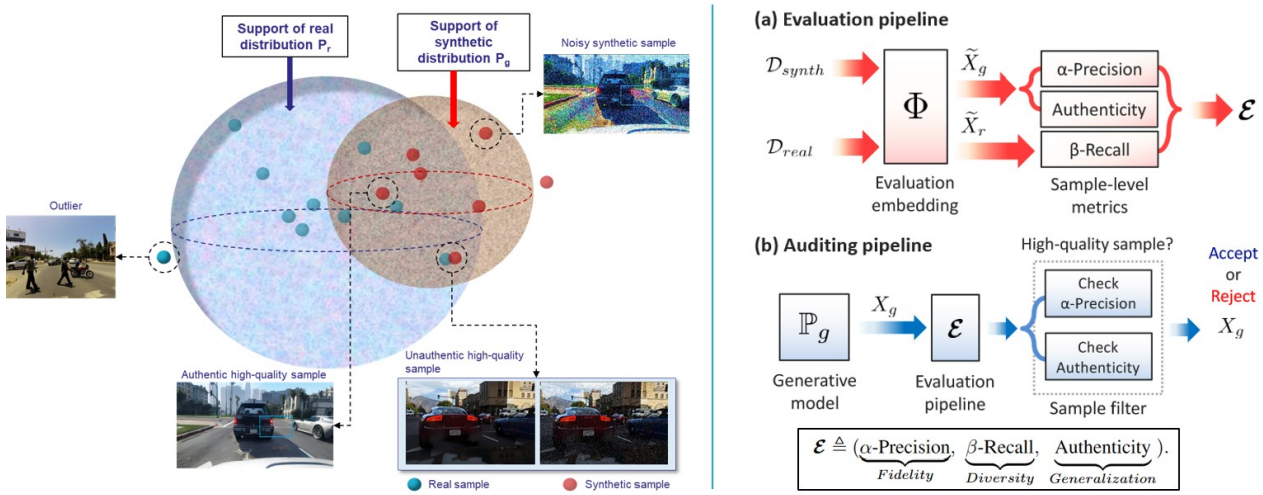


Figure 30: Evaluation and auditing pipelines for Generative Models ([7])

In summary, fidelity primarily addresses the statistical resemblance between synthetic and real data, while quality extends the evaluation to include considerations of usability, relevance, and the fulfilment of specific requirements. The last aspect (diversity) aims to produce high-quality datasets that mirror the intricacies of the real-world data distribution. These 3 aspects are crucial in determining the suitability of synthetic data for various applications, with fidelity focusing on accuracy, quality encompassing a broader set of attributes, and diversity providing a large coverage of encounterable situations/conditions/configurations.

2.4.2.3 Verification and validation methods and metrics for cameras

The principle of operation of the sensor by accumulation of charges is a source of different biases which may depend on the environment observed by the camera lens. Reading noise is inherent to the sensor whatever the situation. On dynamic scenes, motion blur is visible

depending on the sensors and the integration time chosen. In high light conditions, blooming and smearing phenomena appear due to the saturation of the CCD sensors. In the Table 1, we present the result of work done in eMOTIVE regarding the faults of a camera and the associated measurement protocols to highlight them. Faults highlighted in bold are those that are typically used to calibrate camera models. In difficult environmental situations, certain defects modify the acquired images significantly, that is to say sufficiently to disrupt the behaviour of the sensor and the perception systems using the images generated by the sensor. Noise and vignetting become significant compared to other defects in low light conditions; the blooming occurs when the brightness is too strong; distortion can cause bias when attempting pattern recognition; and motion blur occurs in the event of sudden dynamic movements.

In order to validate the quality of the data generated and provided by camera, it is necessary to validate 2 sub parts of the camera model in close relationship with the camera's defaults mentioned in Table 1: the intrinsic operating/behaviour of the camera and the extrinsic aspect involving the image generation depending on the current environment with its disturbers.

Intrinsic parameters: The first sub part is relative to the intrinsic parameters. In order to address this validation, the current way will be to use the virtual sensor model with a set of digital twin for the calibration chart and a digital twin of the DXO bench shown in the figure 31. The process consists, as proposed in [49] [50], firstly to generate both virtual and real data coming from cameras (virtual and real) by using the real and virtual calibration chart in the DXO bench. Secondly, once these data collected, we use the DXO software to assess the intrinsic parameters of both cameras (real and virtual). Thirdly, a comparison is apply on the two dataset generated and a gap is generated. Depending of the gap, the virtual camera will be accepted or rejected.

DxO bench: The DxO Calibration Bench is a tool used to evaluate the image quality of cameras in various industries, such as aerospace, automotive, photography, medical, smartphones, computer vision, action cameras, defence, surveillance and many others. It is designed to measure, compare and optimize the image quality of cameras using image quality evaluation software, equipment and test protocols. The Analyser system offers a complete laboratory suite for measuring and optimising camera image quality. The software includes a comprehensive library of image quality assessment metrics and can process both still images and videos. The Analyser system is modular and highly configurable, working with both raw and processed data. It is a valuable tool for image signal processing (ISP) optimisation and provides both mandatory features and optional premium services. DxO Analyser uses a comprehensive library of image quality evaluation metrics to measure, compare, and optimize camera image quality. Here are some of the metrics used in the Analyser system:

- Effective Focal Length, distortion,
- Effective Focal LengthColor Fidelity and Sensitivity and consistency,
- Noise (SNR, Dynamic Range, Tonal Range)
- White Balance Accuracy etc.

For example, the bench used in UGE includes:

Main camera defaults				
Characteristics	**Variables**	**Measurement tools**	**Technics**	**Measurement**
Gain (linearity)	light	uniform calibration target, luxmeter	white calibration target with uniform lightning	linear function
Offset	Temperature, exposure time, gain	temperature sensor	darkness measurement, constant exposure time	offset for each pixel
Uniformity	light	calibration target unsaturated, light sensor	uniform light range	response distribution, standard deviation
Thermal noise	temperature, exposure time			mean, standard deviation
Reading noise	Temperature, exposure time			mean, standard
photonics noise	light quantity			
Distance focal/pitch		graduated calibration target, range sensor	variation of the distance plan focal-target : resolution of the pinhole model equations	focal distance/ pitch
Vignetting		uniform calibration target, homogeneous ambient lighting	variation of attenuation over average values of groups of ROI pixels	attenuation coefficient
geometric aberration		checker-board calibration pattern	distortion computation	distortion coefficient
optical definition	focusing distance, diaphragm	frequencial calibration target, luxmeter	image analysing	cutoff frequency
out of focus	focusing distance, diaphragm	sinusoidal calibration target, lux-meter	contrast and FTM analysing, measurement of the spatial frequency	depth of field, cutoff frequency evolution
Flare	Position of the light source	calibration target with black square and white background, position sensor	image analysing	position and dimension of the flares
Blooming	light intensity, integration time	lux-meter	image analysing	number of saturated pixels depending on light intensity and integration time
Smearing	light	lux-meter	image analysing	position and dimension
Shutter	light	black and white rotating calibration target, speed sensor	response time analysing	integration time
Dynamic range	contrast	contrast calibration target	image analysing	dynamics
Response time	light variation, gain level	lux-meter, chronometer	image analysing	response time
blur of movement	integration time, object/camera speed	lux-meter, chronometer, black and white rotating calibration target	image analysing	black/white transition depending on the speed of the calibration target and the integration time

Table 1: Summary of the main camera defaults

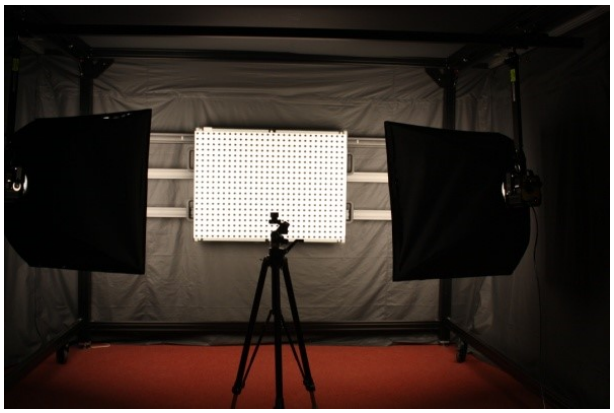
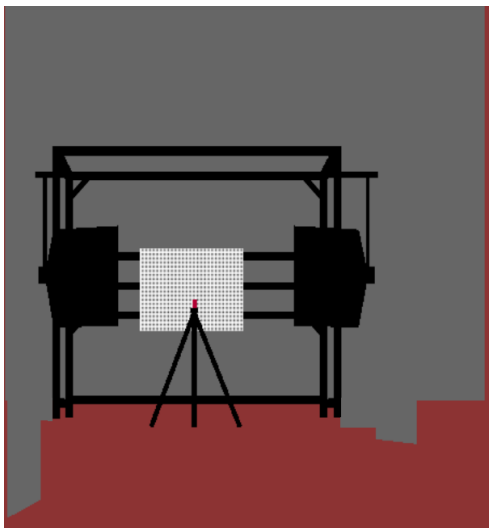
(a)  (b) 

Figure 31: (a) DxO Bench. (b) DxO bench's digital twin

- a structure for the calibration as it is shown in the figure 31,
- Two light projector of 2000W each
- Different calibration targets
- A lux meter
- An accurate Laser telemeter

But other additional components can be added.

In order to compare real cameras and simulated ones, we follow a well-defined characterisation protocol. First of all, we perform a camera calibration on the DxO optical bench. To analyse an optic, we capture a calibrated target image. The focus must be extremely accurate, as well as the parallelism between the target and the plane of the sensor. These levels are checked by auto collimation.

For the calibration, and the characteristic comparison between digital and real images, some calibration targets can be used. For instance in [50], three calibration targets are used and are shown in figure 32:

- A $1.2m \times 0.9m$ dot chart that is used for the lens distortion, the focal length and vignetting measurement. Each dot has a diameter of $2cm$. The tolerance for the test targets is in the range of $0.2mm$ on the diameter and the position of the dots.
- The retro-lighting chart or noise targets is a transmission target placed on top of a uniform light box. The transmission target is a DxO's design made of a thick black plastic plate with precision-drilled holes. These holes (or "patches") are equipped with range of neutral density filters designed to absorb light in the same way for all wavelengths. The filters are made of pure optical glass with no structures that could be measured as noise. The light box behind the target features two fluorescent daylight spectrum tubes with a diffuser sheet on top to achieve perfect uniformity on each filter. The luminance is approximately $1500cd/m^2$. The light absorption levels of the filters range from 0% to 99.99% in order to test across a dynamic range of 4 density steps.When shooting such

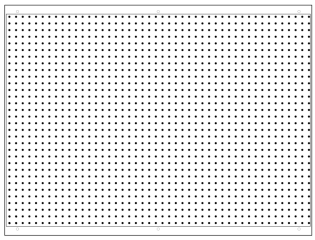
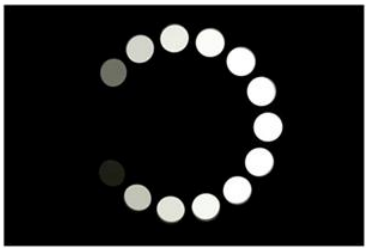
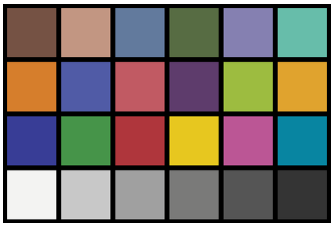
(a)  (b)  (c) 

Figure 32: The three calibration targets used with DxO Bench: (a) DxO's Dot chart (b) The noise target is a transmission target placed on top of a uniform light box for noise and tonal range calibration. (c) Macbeth Chart for the colour sensitivity measurement.

a chart, the sensor of the test camera is exposed to a wide range of light levels, with a 1/10,000 ratio from minimum to maximum.

- The Macbeth chart is a colour calibration target consisting of a cardboard-framed arrangement of 24 squares of painted samples. The chart's colour patches have spectral reflectances intended to mimic those of natural objects such as human skin, foliage, and flowers, to have consistent colour appearance under a variety of lighting conditions, especially as detected by typical colour photographic film, and to be stable over time. Colour sensitivity measurement requires knowing the RAW response of the sensor to a selection of known colours.

In parallel with this work, the optical bench as well as the different test patterns are digitally reproduced in order to be integrated into the Simulation software. From the different parameters characterising each camera, it is possible to reproduce a simulated camera module with characteristics similar to the real sensor. Subsequently, we seek to qualify the fidelity of the optical sensor model by comparing characteristics of digital images to those of the real images. Following the application of different filters and parameters to the simulated camera module (same characteristic as the real camera), the synthetic images of three targets are acquired in the same way as for the real sensor. The results obtained using the simulator camera module are compared to those of the real system. For a first metric we can use the relative error:

$$E = \frac{measure_{sim} - measure_{real}}{measure_{real}} \tag{1}$$

As example, hereafter, the results of the study by [50] are presented: three cameras and their digital twins are calibrated utilising the procedure: Philips SPC1030NC webcam, JAI's CM040MCL with a $4.0mm$ focal length objective and JAI'sGE040CB. The calibration will be carried out for focal length, distortion, vignetting and the sensor linearity.
First, the figure 33 shows the images of the targets by the three real cameras and by the simulated optical sensors.

Finally, the quantitative difference between the digital twins' images for focal length, distortion and vignetting. For the focal length calibration provide the following results presented in the Table 2

For the distortion, the mathematical model that fits the three tested camera is the radial model defined by the equation (2) . Radial distortion causes straight lines to appear curved. Radial distortion becomes larger the farther points are from the centre of the image.

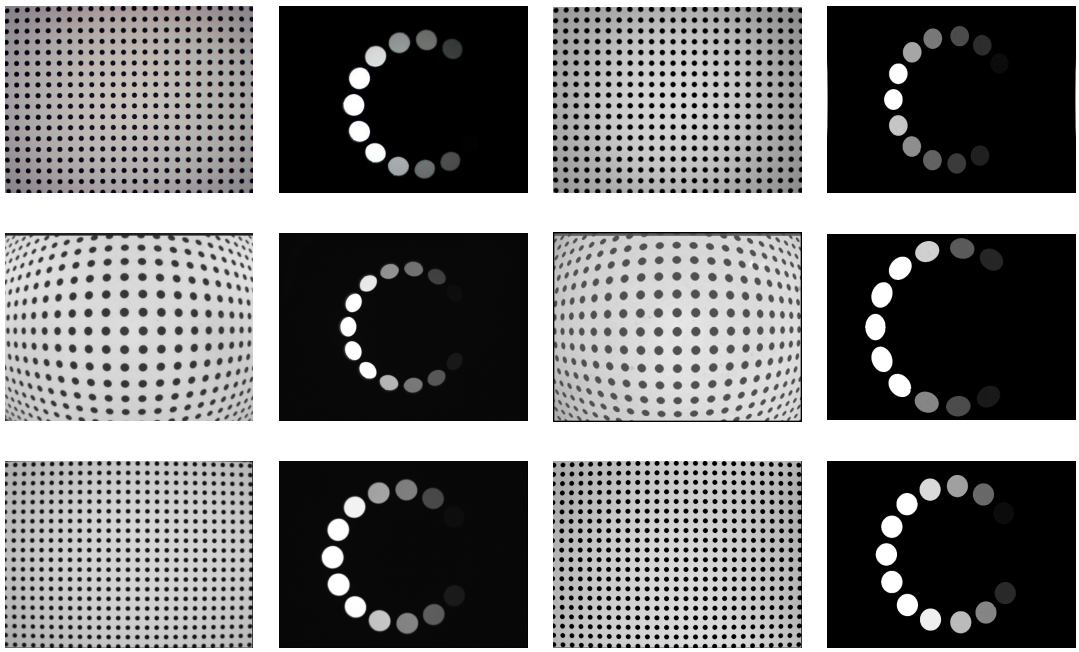


Figure 33: Images of the dot and retro-lighting charts for real cameras in the two right-hand side column and simulated ones in the two left-hand side column. The first line is the images for the Philips Webcam. The second line is the images for JAI CM040CL camera and the last line is the images for JAI GE040CB camera.

Table 2: Focal Length measurement with the uncertainty in milimeter(mm)

optical systems:	Philips SPC1030NC	JAI CM040MCL	JAI GE040CB
Real device	2.89 ± 0.01	4.01 ± 0.01	11.99 ± 0.01
Simulated device	2.88 ± 0.01	4.06 ± 0.01	11.96 ± 0.01

$$\begin{cases} x_{distorted} = x(1 + k_1 r^2 + k_2 r^4) \\ y_{distorted} = y(1 + k_1 r^2 + k_2 r^4) \\ r^2 = x^2 + y^2 \end{cases} . \tag{2}$$

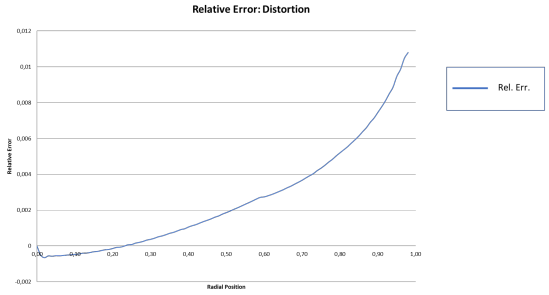
The two parameters k_1 and k_2 of the equation (2) are determined from the measurements a the dot charts in the real camera calibration step. These parameters are used as parameters in the simulated camera. The table 3 shows the distortion parameters of the three cameras:

Table 3: Distortion parameters k_1 and k_2 for the tested cameras

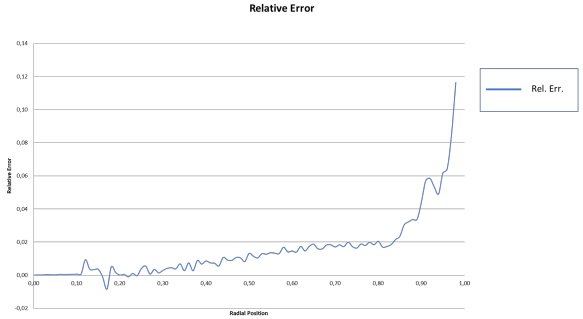
optical systems:	Philips SPC1030NC	JAI CM040MCL	JAI GE040CB
k_1	0.9752	-0.2894	0.3102
k_2	-0.1464	0.0641	0.2654

As it is shown in the figure 34, the relative error is increasing with the radial position of the pixel.

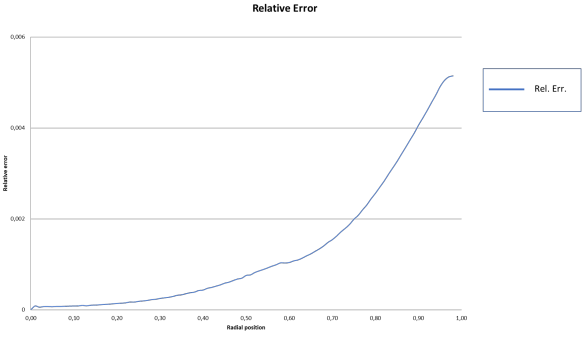
Optical vignetting is the attenuation of the illumination received by the system with increasing field. We take into account of this in the form of a map. The user can easily obtain such a



(a) Philips Webcam.



(b) JAI CM040MCL



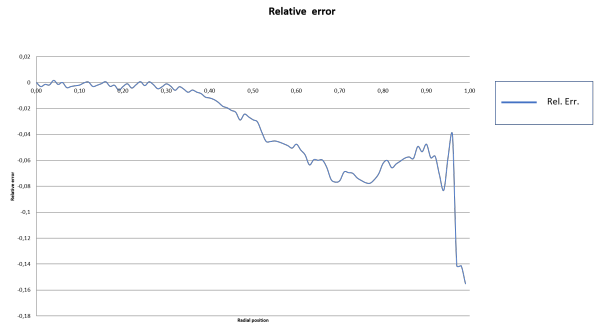
(c) JAI GE040CB

Figure 34: Relative error of distortion for the three tested cameras

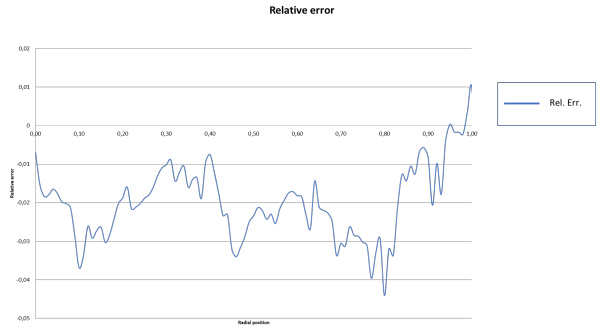
map by acquiring an image of a target illuminated homogeneously and that does not saturate the camera. The figure 35 shows the relative error of the vignetting effect of the simulated cameras. It is obvious that the relative error of the tested camera depends on the camera model and for the Philips webcam, the error is quite important with value more than 15% while for the two other camera, the error is less than 6%.

In this first part, from the different intrinsic parameters characterising each camera, we reproduce a simulated camera module with characteristics similar to the real sensor. Subsequently, we seek to qualify the fidelity of the optical sensor model compared to reality using measurements. To achieve this part, a calibration bench, such as DxO bench, is very useful.

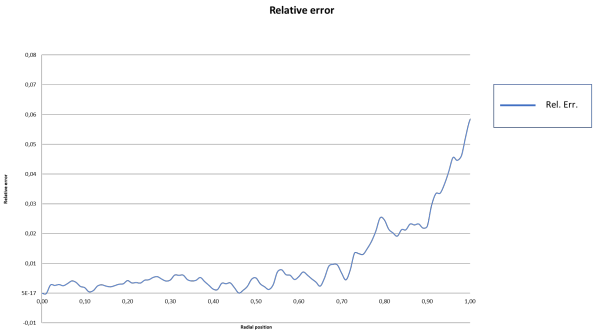
Image Generation: The second sub part is relative to the image generation using the road infrastructure and the environment conditions (weather, light). This second sub part validation is a critical stage for not only the verification and validation of the camera model and its capability to generate realistic data, but for the simulation engine and its capabilities to generate the effects allowing to generate a realistic rendering. For a couple of years, a significant number of synthetic datasets have emerged to overcome the lack of real data with the objective of creating increasingly large and realistic synthetic datasets, which is crucial for AI-based algo-



(a) Philips Webcam.



(b) JAI CM040MCL



(c) JAI GE040CB

Figure 35: Relative error of vignetting effect for the three tested cameras

rithms. But to what extent can we say that the computer-generated images are faithful to reality? Then it is essential to propose some metrics to quantity the level of fidelity of these kinds of images. Plenty of metrics already exist to quantify image quality but none to quantify the fidelity of synthetic images. The term fidelity is preferred to realism due to its subjective nature, making quantification a complex task. Additionally, a comprehensive conceptual framework of fidelity has already been proposed. In the PRISSMA methodology for simulation verification and validation, the fidelity can refers to the extent of similarity between some selected features in the virtual environment and their corresponding reference features in the real environment. Therefore, high-fidelity simulations can correspond to a faithful representation of the real environment's features, while low-fidelity simulations may correspond to a simpler representation of these features in comparison to the real world. The figure below illustrates the comprehensive diagram of the proposed verification method, which consists of introducing a set of metrics to quantify the fidelity of synthetic images. Learning and statistic-based approaches are used to exploit the image information such as textures and others that are relevant to real images.

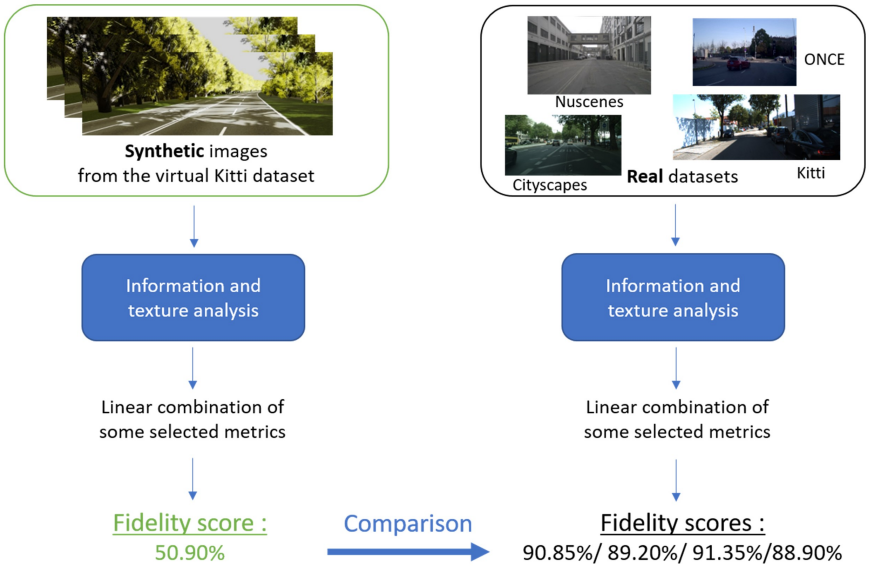


Figure 36: Diagram of the proposed verification method to assess the fidelity level of a virtual image coming from a virtual camera

A linear combination of these metrics allow to computes the fidelity scores for the synthetic datasets. The same process is applied to the real datasets to provide reference scores, giving an indication of the fidelity level of synthetic images. This method has shown promising results, but it needs to be extended to various types of scenes. The experiments were conducted using data from urban areas under clear daytime conditions. The next step is to apply this method to different scenes and under adverse weather conditions. The image information will vary significantly depending on scenarios, like under foggy conditions.

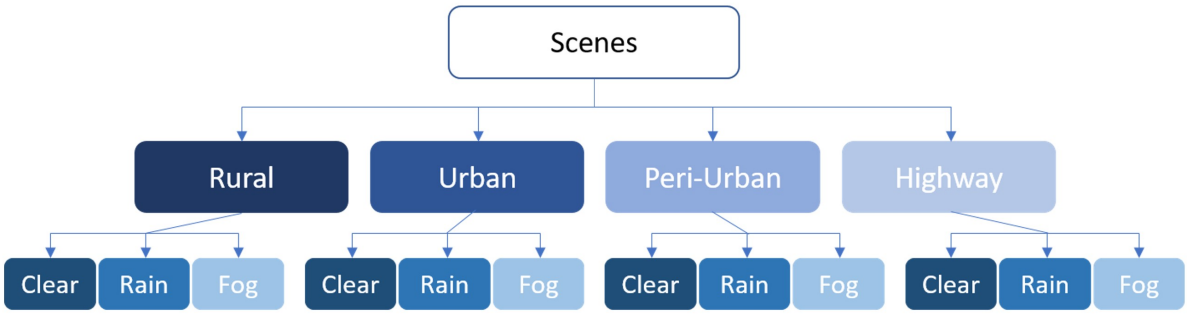


Figure 37: Taxonomy of the scenario encounter by a virtual camera

Generating a score is essential to determine whether a virtual data set will be sufficiently representative and true to reality to be used in learning, assessment, and validation procedures. At this moment, researches focused on this topic address mainly the generation of realistic images from the human point of view and not from the camera point of view. Indeed, concerning human vision and the level of realism of a synthetic image, certain metrics or criteria commonly used to characterise it are sometimes questionable and weakly correlated with human vision. In this part, we only deal with images produced by a camera, which is very different from the

constraints and conditions posed by the human eye as a sensor. Compared to a camera, the human eye is a "poor quality" sensor. As we can observe in Figure 38, the part producing precise information as a camera would do is very small.

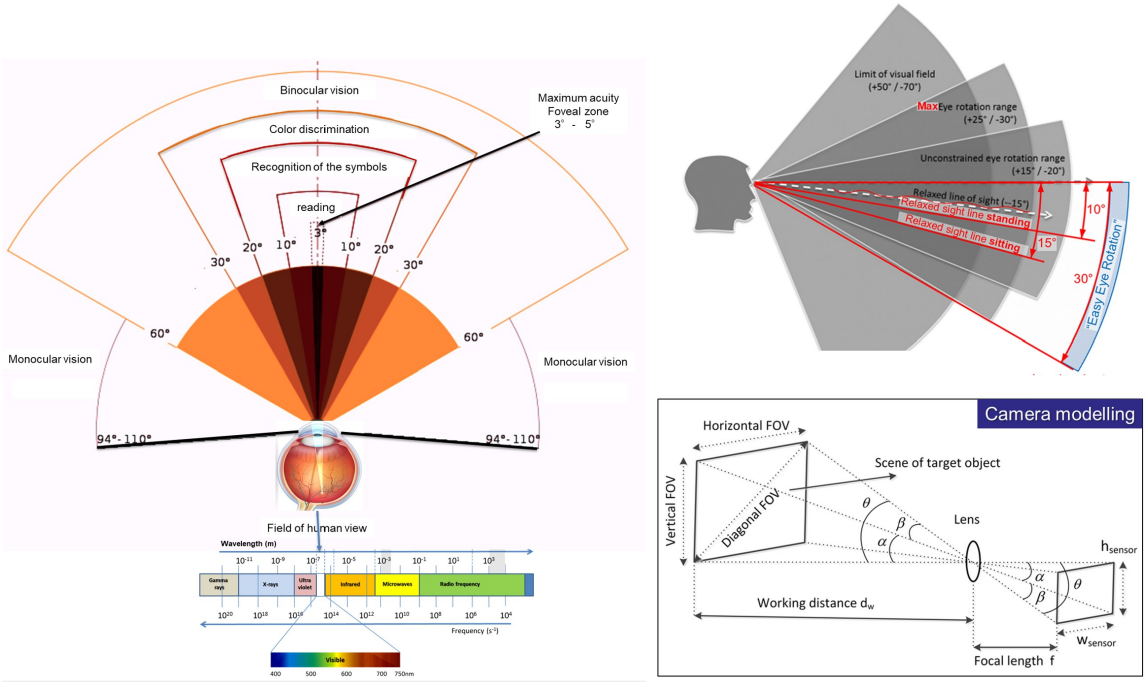


Figure 38: Modelling of the human eye with the different fields of view with their angles and functions. Bottom right provide the simple modelling of a RGB camera with the main parameters.

For instance, in the recent work presented in [51], the authors propose an exploration of Metrics and Datasets to Assess the Fidelity of Images Generated by Generative Adversarial Networks. Ensuring the quality of synthetic images involves evaluating them. The classical metrics used in this objective are metrics like:

- **FID (Frechet Inception Distance)**[52], which is more consistent than the Inception Score. The Fréchet distance also known as Wasserstein-2 distance measures the difference of two Gaussians (i.e synthetic and real-world images). The Fréchet distance d(., .) between the Gaussian with mean (m, C) obtained from p(.) and the Gaussian with mean (mw, Cw) obtained from pw(.) is called the "Fréchet Inception Distance" (FID). The FID is consistent with increasing disturbances and human judgement. The FID seems to capture efficiently the disturbance level. Even if this distance is strongly used, because it does not differentiate the fidelity and diversity aspects of the generated images, recent papers have introduced variants of precision and recall metrics to diagnose those properties separately. FID is calculated with $d^2 = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2\sqrt{C_1 \cdot C_2})$
- **IS (Inception Score)**: The inception score is a metric designed to measure the image quality and diversity of generated images. It was initially created as an objective metric for generative adversarial networks (GAN). In order to measure the image quality, generated images are fed into a pre-trained image classifier network in order to obtain the probability score for each class. If the probability scores are widely distributed then the generated image is of low quality. This metric was shown to correlate well with human

scoring of the realism of generated images from the CIFAR-10 dataset. The IS uses an Inception v3 Network pre-trained on ImageNet and calculates a statistic of the network's outputs when applied to generated images. A wide probability distribution is commonly referred to as "high entropy". $entropy = -\sum(p_i \cdot log(p_i))$

- **KID (Kernel Inception Distance)** [53]: KID uses polynomial kernel methods to measure the disparity between empirical distributions of features in real and generated images. The compared features are extracted from the $pool3$ layer of an inception network. The KID metric is useful for assessing the quality and diversity of generated images in the context of generative models. Moreover, it has been demonstrated to be superior to IS and FID [53]. The polynomial kernel used is : $k(x, y) = (\frac{1}{d}x^T y + 1)^3$ where d is the dimension representation.

- **LPIPS(Learned Perceptual Image Patch Similarity)**, which align with human perception of image quality, but not with camera perception. LPIPS essentially computes the similarity between the activation of two image patches for some pre-defined network. This measure has been shown to match human perception well. A low LPIPS score means that image patches are perceptual similar. In [54], a new version of this similarity distance has been proposed (R-LPIPS (Robust Learned Perceptual Image Patch Similarity)).

It is important to emphasise that these methods are open to criticism because they all depend on the Inception network. The rating is good if the type of image has already been seen by the network, otherwise the rating that will be given will be low and potentially of no real value.

Moreover, for GAN, proper dataset segmentation and pre-processing are crucial for accurate model training. GANs face yet challenges in training due to the constant competition between generator and discriminator networks, leading to instability. Many researchers overlook the importance of correctly filtering datasets for GANs, impacting image generation quality. Careful selection and preparation of datasets are essential and mandatory for optimal results in GAN-based image generation. Improving realism in generating images of individuals requires addressing road actors variability through advanced segmentation techniques, promising better representation and more authentic images with diverse datasets. This work which analyse metrics and datasets generated by GANs, provides some advice and a good practice guideline for researchers in order to understand the strengths, weaknesses, and areas for further research on generative adversarial networks. Also this work ultimately enhances image generation precision and control by detailing dataset preprocessing and some quality metrics for synthetic images.

To evaluate synthetic data sets, in addition to the metrics proposed to [51], some other works use:

- **Moving Average (MA)** which computes the time-average of parameters. In time series analysis, the moving-average model (MA model), also known as moving-average process, is a common approach for modelling uni variate time series. The moving-average model specifies that the output variable is cross-correlated with a non-identical to itself random-variable.

- **Exponential Moving Average (EMA)** computes an exponentially discounted sum. Exponential smoothing, also known as exponential moving average (EMA), is a technique for smoothing time series data by applying an exponential window function. Unlike simple moving averages that assign equal weights to past observations, exponential smoothing utilises exponentially decreasing weights over time. This method is simple to under-

stand and apply, making it a popular choice for making determinations based on user-defined assumptions like seasonality. Exponential smoothing finds frequent use in the analysis of time-series data. This simple form of exponential smoothing is also known as an exponentially weighted moving average (EWMA). Technically it can also be classified as an autoregressive integrated moving average (ARIMA) (0,1,1) model with no constant term.

- **Precision/Recall (P1/R1)** ([55]) and **Improved Precision** & **Recall (P&R)** ([56]): In the second case, the quality of a generator is decoupled into two separate values to aid in the detection of mode collapse and mode dropping. Mode dropping refers to the case wherein modes of Pr are underrepresented by Pg, while mode collapse describes a lack of diversity within the modes of Pg.

- **Density/Coverage (D/C)** ([57]). The precision and recall metrics are not reliable enough to assess fidelity and diversity (coverage) aspects. For instance, they fail to detect the match between two identical distributions, they are not robust against outliers, and the evaluation hyper parameters are selected arbitrarily. The main motivation behind this two-value metrics is to provide a capability to have a diagnosis involving the fidelity and diversity of generated images.

- **Parzen window likelihood (P W)** [58]) Given a set of d -dimensional samples, the Parzen window estimate the underlying probability distribution. When log-likelihoods are unavailable, a common alternative is to use Parzen window estimates. Here, samples are generated from the model and used to construct a tractable model, typically a kernel density estimator with Gaussian kernel. A test log-likelihood is then evaluated under this model and used as a proxy for the true model's log-likelihood. It is suggested to fit the Parzen windows on both samples and training data, and to use at least as many samples as there are images in the training set.

- **Peak Signal-to-Noise Ratio (PSNR)**: Signal-to-noise ratio is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. SNR is defined as the ratio of signal power to noise power, often expressed in decibels. A ratio higher than 1:1 indicates more signal than noise. PSNR is commonly used to quantify reconstruction quality for images and videos subject to lossy compression. PSNR is calculated using the mean squared error (MSE) or L2 distance. $PSNR = 20 \cdot log_{10}(MAX_I) - 10 \cdot log_{10}(MSE)$

- **Structural Similarity Index (SSIM)** or Perceptual Quality (PQ): The Structural Similarity Index (SSIM) is mainly used as a perceptual metric that quantifies image quality degradation caused by processing such as data compression or by losses in data transmission. This L2 Distance uses pixel differences to measure the structural differences between two images (sample and reference). In [59], the class of SSIM metrics are studied with a mathematics approach. The main conclusions are that SSIM can be partitioned into a set of components (mainly 2 or 3). For instance, the SSIM metric can identifies these 3 metrics from an image: Luminance $l(x, y)$, Contrast $c(x, y)$, and Structure $s(x, y)$. Each of these components can be transformed into a valid distance metric. The metrics can be either combined to a single scalar-valued distance metric applying a norm with the increasing property or be used to form a vector-valued generalised (cone) metric.

Like the MSE, the SSIM index and SSIM-based metrics are preserved under orthogonal or unitary transformations. Convexity, quasi-convexity, and generalised convexity have been locally hold for the metrics derived from SSIM. The properties may find broad applications in many optimisation problems in image processing where objective functions correlated with perceptual image quality are desirable. Instead of taking global measurements of the whole image some implementations of SSIM take measurements of regions of the image and then average out the scores. This method is known as Mean Structural Similarity Index (MSSIM) and has proven to be more robust. $SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma}$ and $MSSIM(X, Y) = \frac{1}{M} \sum_{j=1}^{M} SSIM(x_j, y_j)$

- **Wasserstein distance (W)** is used for many tasks in statistical machine learning including: Two-sample testing without smoothness, goodness-of-fit, analysis of mixture models, image processing, dimension reduction, generative adversarial networks, domain adaptation, signal processing. But this distance does have problems. First, it is hard to compute. Second, a way to do inference for the distance is not available. This reflects the fact that the distance is not a smooth functional which is, itself not a good thing.

- **Kernel Inception distance (KID)** measures the maximum mean discrepancy between real and generated images after transforming Inception features using a kernel function.

A part of these metrics are more dedicated to the fidelity assessment. For the diversity aspect in the generated images, it is relevant to use measures such as entropy, diversity score, or coverage. In a nutshell, a part of the metrics mentioned above serve distinct and complementary purposes in evaluating two sets of images:

- Inception Score: Measures the realism of an image concerning a pretrained model.

- FID Score: Assesses the similarity between one group of images and another group.

- LPIPS Score: Evaluates changes in the structure of a patch and quantifies the extent of those changes.

- SSIM, MSSIM, PSNR: Gauges the level of noise in a generated image in comparison to a ground truth image.

The effectiveness of these metrics depends on the specific characteristics and requirements of the task at hand.

In [8], the authors propose the **feature likelihood score (FLS)**: a novel sample-based score from December 2023 that captures sample fidelity, diversity, and novelty (see figure 40). FLS enjoys the same scalability as popular sample-based metrics such as FID and IS but crucially also assesses sample novelty, over-fitting, and memorisation. From the authors, evaluation using FLS has many consequential benefits:

- **Explainability**: Samples that contribute the most (and the least) to the performance are identified.

- **Diagnosing Overfitting**: As over-fitting begins (i.e., copying of the training set) FLS identifies the copies and reports an inferior score despite any drop in sample fidelity and diversity.

- **Holistic Evaluation**: FLS simultaneously is the only score proposed in the literature that simultaneously evaluates the fidelity, diversity, and novelty of the samples
- **Universal Applicability**: FLS applies to all generative models, including VAEs, Normalising Flows, GANs, and Diffusion models with minimal overhead as it is computed only using samples.
- **Flexibility**: Because of its connection with likelihood, FLS can be naturally extended to conditional and multi-modal generative modelling.

Architecture	Datasets	Metrics
GAN	ImageNet, CIFAR 10, STL 10	FID, IS, MA, EMA
WGAN-GP	CIFAR 10, CIFAR 100, ImageNet, MNIST	FID, IS, GAN-Train, GAN-Test
DCGAN	CIFAR 10, CelebA-HQ, LSUN Bedroom	FID, IS, KID, MS-SSIM
StoryGAN	CIFAR 10, CelebA-HQ, LSUN Bedroom	MS-SSIM
WGAN, WPGAN, MSGAN	BraTS	SSIM, PSNR
GAN	MNIST, HAR, EST	FID, IS
WGAN, cGAN	AffectNet, RAF-DB	W-DISTANCE
WGAN-GP, BigGAN	CIFAR 10	FID, IS, W-DISTANCE
DCGAN, WGAN, ProGAN, StyleGAN2, BigGAN, CoCoGAN	FFHQ, CelebA, LSUN	RGB, HSV, YCbCr
StyleGAN2	CelebA, CASIA WebFace	FID, IS
StyleGAN	CIFAR 10, STL 10, ImageNet, CASIA HWDB1.0	FID, IS
VanillaGAN, SNDCGAN, BigGAN	CIFAR 10	FID
Enlighten-GAN	Sentinel-2	GSM, PI, PSNR, LPIPS
SPGAN	VGGFace2, CelebA, Helen, LFW, CFP-FP, AgeDB-30	FID, PSNR, SSIM
PCGAN, StyleGAN2, CoCoGAN	Inria Aerial	FID, KID
PF-GAN	Celeb-HD	Similarity Score
GAN	VoxCeleb	PSNR, SSIM
StyleGAN	SHHQ	FID
FCG-GAN	CelebA	PSNR, SSIM
StyleGAN-XL	FFHQ, Pokemon	FID, IS
DDM-GAN	MNIST	FID, KL-divergence
StyleGAN3	CelebA-HQ	FID, LPIPS, SSIM, PSNR, MSE
StyleGAN2	CelebA, LFW, FFHQ, Caltech	LPIPS, MSE
Triple-BigGAN, BIG	SEA Faces	LPIPS, MSE
StyleGAN	FFHQ	FID, IS, LPIPS, MS-SSIM
InterFaceGAN	CelebA	KL-divergence
PCGAN, InterFaceGAN	CelebA-HD	ACRD, ACSD, SBC
AgeTransGAN	FFHQ-Aging	FID

Figure 39: Succinct outline of the metrics, datasets, and GAN architectures used for the generation of high fidelity synthetic images

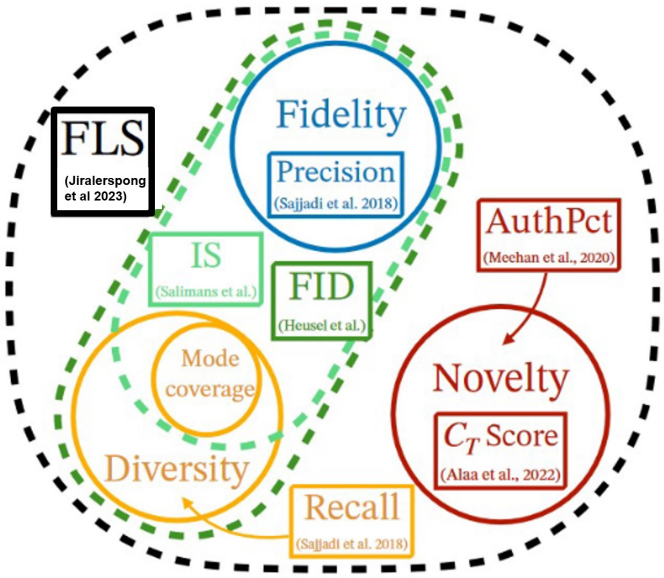


Figure 40: Sub-spaces of generative model evaluation with the "trichotomy": fidelity, diversity and novelty [8]

In [60], the authors address the same issue (generation of high fidelity images and quantification of the level of fidelity) but not in the road eco-system, but in the complicated underwater environment where lighting conditions lead to severe influence on the quality of underwater imaging. o effectively evaluate the quality of underwater images, an underwater image quality assessment dataset is constructed from synthetic to real-world, and then a new objective underwater image assessment method based on the characteristics of the underwater imaging is proposed (UICQA). Considering that the transmission map can effectively reflect the characteristics of the underwater imaging, statistical features are extracted from the transmission map for distinguishing underwater images of different quality. Further, considering that the transmission map negatively correlates with scene depth, a local-to-global transmission map weighted contrast feature is constructed. Additionally, the color features of human perception and texture features based on fractal dimensions are proposed. Finally, the experimental results of this work show that the proposed UICQA method exhibits the highest correlation with ground truth scores compared to state-of-the-art UIQA (Underwater Image Quality Assessment) methods. In this method, statistical methods were used to analyze the USRD dataset (DataSet generated with 420 underwater synthetic images and 320 real underwater images). A statistical method is used to test whether mean opinion score (MOS) value (subjective evaluation made by humans) matches the image quality of different waters. 3 classes of features are estimated in order to build a fidelity score:

- **The statistical features**: The mean subtracted contrast normalised (MSCN) coefficients of the images on both scales, transmission map and 0.5× transmission map, are fitted to the generalized Gaussian distribution (GGD) and asymmetric generalised Gaussian distribution (AGGD) to obtain the statistical feature characteristics.
- **The Perceptual features**: In this part, local-to-global Transmission Map Weighted Contrast Features are extracted. Moreover the colourfulness is also extracted. Combining colorfulness features in the spatial domain with those in the frequency domain (DCT map) provides a more comprehensive representation of colourfulness feature (CF).

- **The texture features**: The fractal dimension is extracted to represent the texture features of the underwater image.

The three extracted feature sets represent all properties of the underwater images, including statistical features on the transmission map, local-to-global weighted contrast features, colorfulness feature, and texture feature, as shown in Table 3. According to MOS value and 73 dimensional features of underwater image, the SVR (for feature regression) is used for UICQA. Then, the matching image quality scores is computed.

This last method could be the basis of a verification and validation of the fidelity level of synthetic images generated from a real image with the addition of specific disturbances like the fog, the rain, the snow fall, the smoke, and the dust cloud.

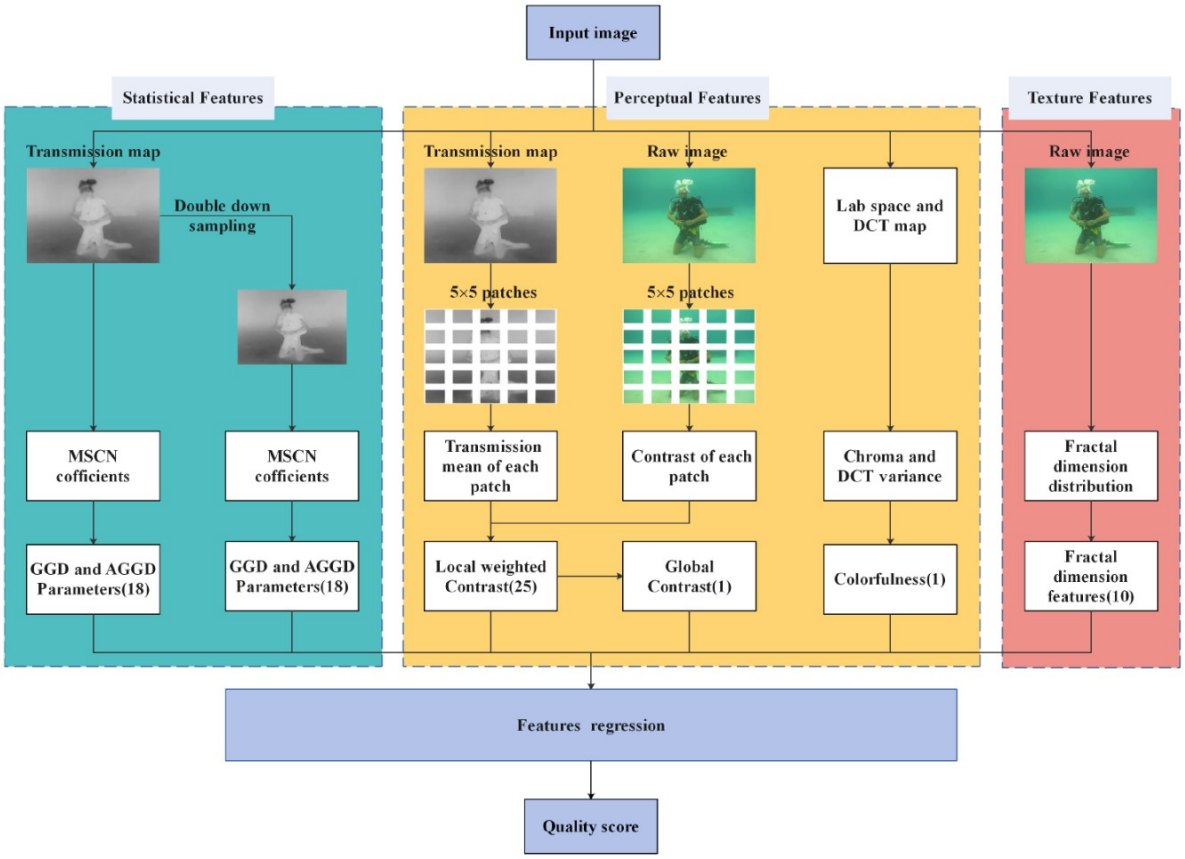


Figure 41: Flow chart of the UICQA method shared in 3 dedicated classes of features: Statistical, Perceptual, and Texture ([9]

In [10], the same ”underwater” image issue is addressed but with another interesting approach. They propose an Underwater IQA (UIQA) method based on color space multi-feature fusion. In this context, the proposed method converts underwater images from RGB color space to CIELab color space, which has a higher correlation to human subjective perception of underwater visual quality. In this context, we do not speak about the level of fidelity of a synthetic image but the generation of a quality score for an image. The interesting aspects of this work for PRISSMA methodology and the evaluation of the level of fidelity of a synthetic image concern the type of features extracted from an image and which space seems to be relevant. The proposed method extract histogram features, morphological features, and moment statistics from

luminance and color components and merge these features to obtain fusion features to better quantify the degradation in underwater image quality. After features extraction, support vector regression(SVR) is employed (like in [60]) to learn the relationship between fusion features and image quality scores, and gain the quality prediction model. In this method, the different features extracted are the following:

- f1: LBP histogram feature
- f2: Luminance component morphological feature
- f3: Luminance component moment statistic feature
- f4: Luminance histogram feature
- f5: Chromatic component moment statistic feature
- f6: Chromatic component morphological feature

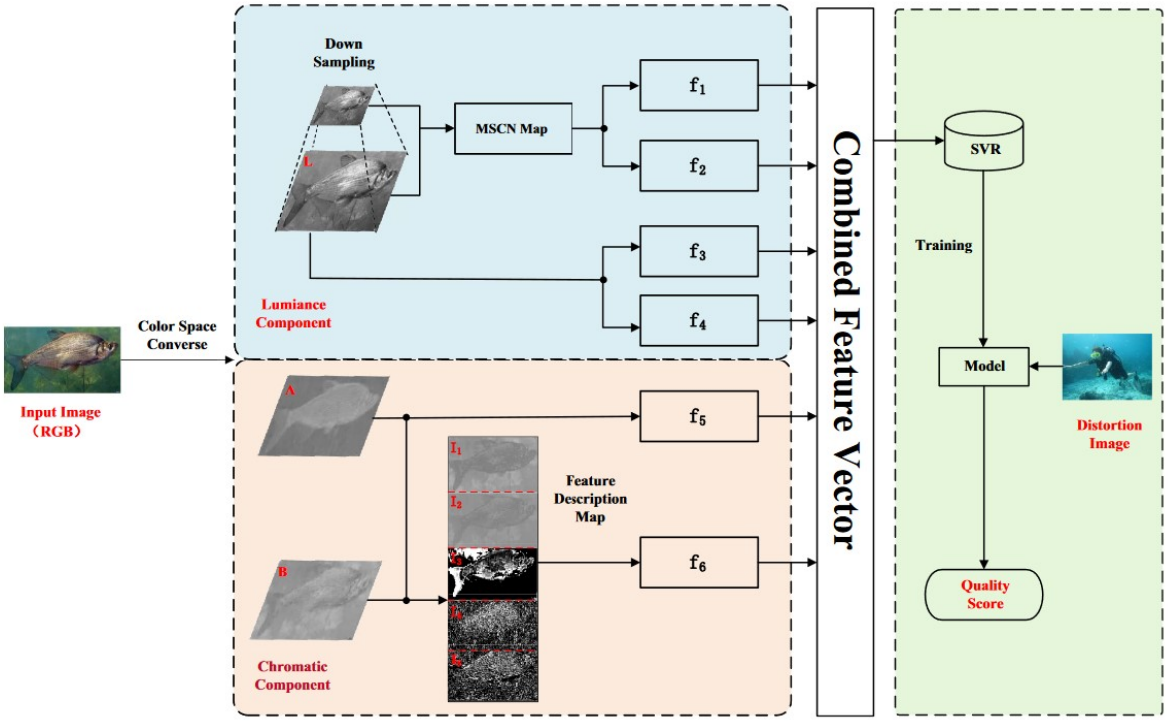


Figure 42: Evaluation of image quality using the fusion of luminance and chromatic features ([10]

But what is a good fidelity score to be able to validate a simulated image or a dataset to construct a module in an AI-based system? This clearly depends on the task to be carried out, in AI there is no ”good” fidelity score in absolute terms, no standard or regulation currently sets a score to validate the quality of the simulated data used. Consequently, the threshold to be taken into account for this score will depend solely on the interpretation of the expert in relation to the complexity of the task and the existing state of the art. 50% may be a “good” score for a business expert on a certain task and disastrous for another task.

2.4.2.4 Verification and validation methods and metrics for Neuromorphic/event-based camera

The event based camera is new sensor that is developed in the early 2000's [61]. Although, the research interest is growing in this field, it is still very seldom to use an event based camera in autonomous driving. To the best of the author's knowledge, the only known event-based camera models are published in [62, 63, 64, 65]. We can here classified these models into simplified model [62] and medium-fidelity models [63, 64, 65]. Thus, the V&V of the models are not yet as well-developed as for the other sensors. However, the V&V principle remind the same but we need to take into account of the sensor's specificities. As for the RGB camera and other sensors, our method is to set up a real calibration bench (For the Event-based camera the DxO bench can not be used without modification). Here are two levels of V&V of the sensor model that it is interesting to discuss: the calibration level and the event point cloud level.

calibration level : A digital twin of the calibration bench must also be set up. The calibration targets are different compared to the RGB camera's calibration target. Due to the specificities of the event-based camera, the calibration target must blink or move during the calibration step. Thus, it will be to set up a calibration that allow movements with known speed or a screen that displays a blinking target. The usually used target is the chessboard. The camera manufacturers like Prophesee or IniVation provide the chessboard pattern image file in the development kit.

event point cloud level : The event-based camera doesn't 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 $p \in 0, 1$ 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 raytracing technique allows for high simulation fidelity, the simulation of the event simulation needs a high frequency in the event simulation. Following [65], to quantify the accuracy of the event simulation, it is not a right solution to compare the generated event point-cloud with respect to the ground truth event point clouds because there is still no clear metric to compute the similarity between two point clouds. The proposed solution is the evaluation of a sampling strategy by comparing the estimated brightness signal with respect to a ground truth brightness signal obtained by oversampling the visual signal at 10Khz. The metric used is the root mean square error. Furthermore, to produce an event point e, the observed object must move or blink. In order to compare the sensor model and the real sensor, we should define some important parameters such as the speed of the object, the trajectory of the object and the frequency of the blinking. As example of validation, in [65], the movement of the camera is simulate at various speeds and rotations up to 1000 degrees per second. Thus, the V&V scenarios should emphasise the movement of the sensor or the movement of the observed obstacle. It is also need to take into account of the texture and patterns of the environment.

2.4.2.5 Verification and validation methods and metrics for IR cameras

There are two types of infrared-based sensors: active (which emit IR) and passive. The validation of active IR camera models emphasises the source and emission of infrared, while for passive IR cameras, the focus is on accurately modelling thermal characteristics and temperature variations. Both types of cameras require thorough validation, but the specific criteria and validation steps differ based on their distinct operation.

Compared to the validation and calibration stages of conventional cameras such as RGB, calibration for an infrared camera involves adjustments to factors like temperature sensitivity, sensor response to IR wavelengths, and sometimes correction for IR lens distortion. Calibration tools for IR cameras might include black-body calibration devices that emit known levels of infrared radiation or specialised targets designed for IR wavelengths. Calibration is centred around parameters linked to infrared radiation sensitivity, temperature mapping, and other IR-specific features like reflective properties of materials in the IR spectrum.

One example of a black-body calibration device available commercially is the ”FLIR Systems Black-body Calibration Source.” FLIR Systems is a prominent manufacturer of thermal imaging cameras and related equipment, including black-body calibration sources. They offer various models designed for different calibration needs in the field of thermal imaging and infrared cameras. Another well-known manufacturer of black-body calibration devices is Santa Barbara Infrared (SBIR). They produce black-bodies under the name ”Black-body Calibration Sources” tailored for calibrating infrared cameras, sensors, and systems used in various industries such as automotive, aerospace, and defence. These devices are engineered to emit precise and stable infrared radiation at known temperatures, allowing accurate calibration and validation of the temperature measurement capabilities of infrared cameras and sensors. The black-body sources from these manufacturers often come in different sizes, temperature ranges, and features to suit specific calibration requirements across industries and applications.

Validating an infrared (IR) camera model within an automotive simulation platform involves a comprehensive process to ensure the accuracy and reliability of the simulated camera’s behaviour. Establishing ground truth data is essential for validation. This involves having precise information about the actual temperatures, material properties, and any other relevant factors present in the scenes captured by the real IR camera. Ground truth data serves as a reference for validating the simulation’s output. Utilising the collected dataset and ground truth data, the simulation model’s output is compared against the real-world IR images. This involves adjusting and calibrating the simulation parameters to minimise the differences between the simulated IR images and the real ones. Parameters adjusted during calibration might include temperature sensitivity, emissivity settings, noise levels, and other factors specific to IR imaging.

Metrics are established to quantitatively evaluate the performance of the simulated IR camera model. These metrics can include temperature accuracy, spatial resolution, signal-to-noise ratio, and other image quality parameters.

2.4.2.6 Verification and validation methods and metrics for RADAR

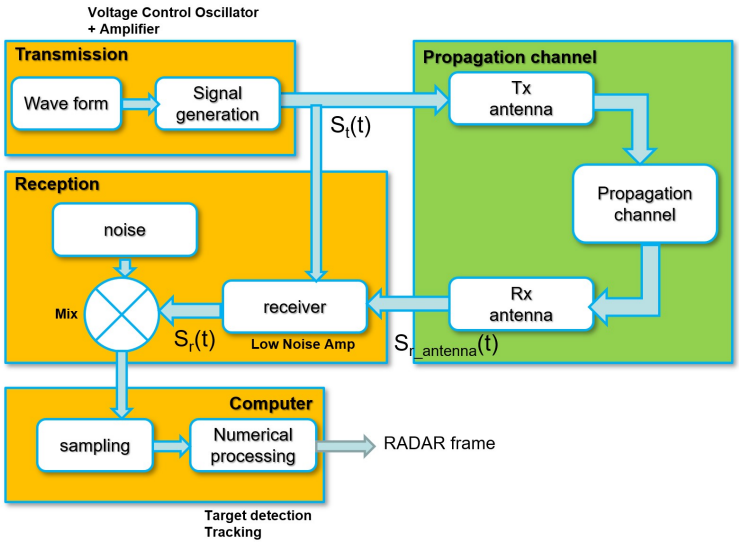


Figure 43: RADAR functions in a Digital Twin of the sensor

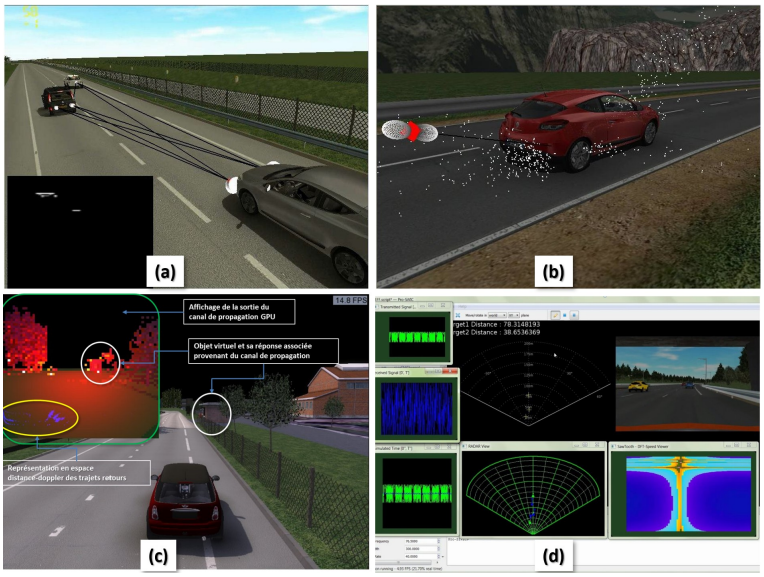


Figure 44: Different levels of RADAR modelling implemented in ProSIVIC®. (a) presents 2 simplified models of RADAR. First one with Depth map, voting method with objects impacted by RADAR signal, multi-object tracking generating a Continental RADAR message, second one with ray-tracing (transmitted power, 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 43) an illumination of the environment allowing multi-reflections (waveguide effect), energy concentrations, scattering, absorptions. 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 43)

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.

The verification and validation (V&V) method must take into account the complexity of the model. The method must take into account the sensor, the target and the environment. First, to V&V a given RADAR model, it is compulsory to have the real RADAR as reference. The first step will be the calibration of the real sensor and the digital twin as well. Secondly, for an accurate V&V, it is useful to use some test bench and facilities. For the RADAR, a serious measurement must be carried out with a anechoic chamber or some smaller test bench that includes an anechoic chamber.

Calibration targets : RADAR calibration targets are used to ensure that the data obtained from the RADAR is accurate. There are several types of calibration targets for RADARs, including point targets, distributed targets, and corner reflectors. Point targets are small objects that are used to calibrate the range and velocity measurements of a RADAR system. Distributed targets are larger objects that are used to calibrate the spatial resolution of a RADAR system. Corner reflectors are objects that are designed to reflect the RADAR signal back to the RADAR system in a specific direction. These targets are designed to reflect the RADAR signal back to the RADAR system with a known RCS (Radar Cross Section) value. The RCS value of the target is used to calibrate the RADAR system so that it can accurately measure the RCS of other objects.

The calibration targets are used to refine the inaccurate transmitter and receiver positions. A nearby corner reflector or target of known location could be used to calibrate range and antenna position but with drawbacks. When a corner reflector is used, the target signal can be obscured by ground clutter (A unique radar calibration target https://ieeexplore.ieee.org/document/6203713).

Anechoic Chamber: This is a room designed to stop reflection or echoes of electromagnetic waves. The chambers range from small compartments with the size of a microwave oven to ones as larges as an air plane hangar. The size of the chamber will depends on the size of the objects and frequency being tested. The internal appearance of the radio frequency (RF)

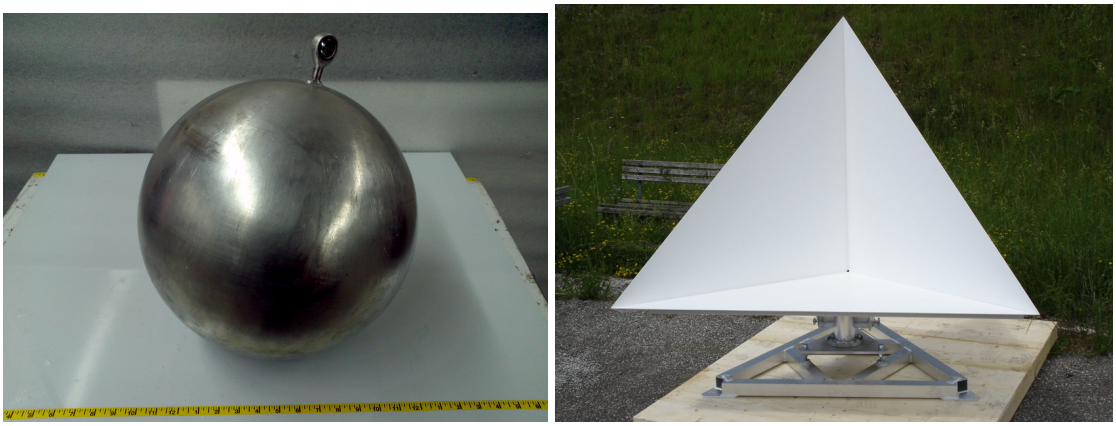


Figure 45: Passive calibration target for RADAR.

anechoic chamber is similar to that of an acoustic anechoic chamber, however, the interior surfaces of the RF anechoic chamber are covered with radiation absorbent material (RAM) instead of acoustically absorbent material. The RF anechoic chambers are used to test antennas and radars, and they are typically used to house the antennas for performing measurements of antenna radiation patterns and electromagnetic interference.

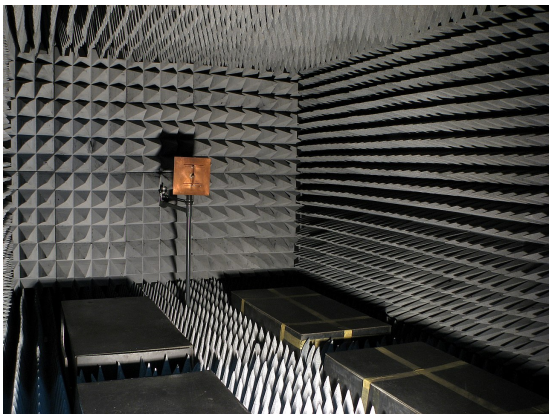


Figure 46: A RF anechoic chamber.

V&V of a simplified RADAR model For the simplified model, the V&V can be carried out by comparing outputs of the simulation that are high level data such as FoV in azimuth and elevation, positioning and speed of an ”ideal target” and the real measurements from a physical simulation using the anechoic chamber or tests benches from dSPACE or Rohde & Schwartz companies.

In [66], a more advanced way to V&V the model is presented. The approach is to include the Hardware in the simulation so that the RADAR sensor model can be compared to the real sensors output with the same scenarios. The so-called ASGARD1®is a RADAR Target Simulator (RTS) that enable Over-the-Air target stimulation for the Radar Under Test. Typically, the device is positioned in front of the radar under examination and can stimulate the radar to identify one or more simulated targets with different ranges and velocities. By intercepting the radar signals, the RTS can perform over-the-air validation tests by modifying the signals based on target information such as speed and distance before returning them to the radar. The target

Figure 47: The dSPACE test bench and the Rohde & Schwartz test bench

information is usually simulated using either models of the target's response or data derived from real recordings. This solution with the test bench shown in the figure 47 is also provided by dSPACE as it is illustrated in the figure 48.

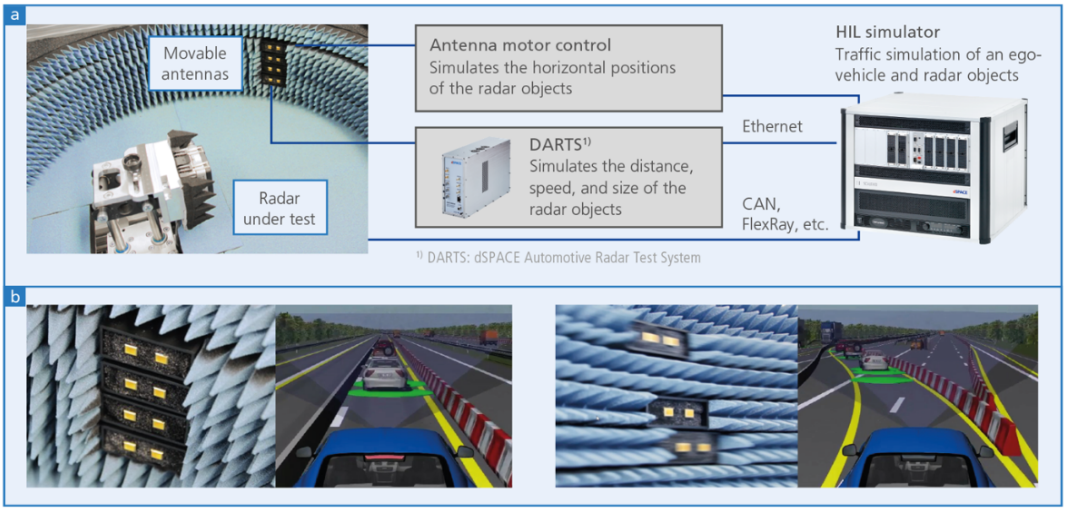


Figure 48: dSPACE's solution for a V&V based on HIL.

V&V of a (high fidelity) Physics based RADAR model With more complex physics based RADAR model, the V&V of the model is more involved. With such kind of models, it is also necessary to consider the position of the antennas (SISO, SIMO and MIMO), the antenna diagram or the RCS. These characteristics from the sensors and the targets are important for the fidelity of the model and are critical for the simulated outputs (speed, range etc.). Thus, it is necessary that theses simulated features are as closed as possible to the real signals.

To validate the simulated RCS, the usual solution is to measure the RCS of the real road users targets and compare the measurements to the simulated output as it is proposed in [67]. It is also possible to verify the simulated antenna diagram by comparing it to the real diagram of the real RADAR by measuring the antenna diagram using the anechoic chamber. For instance, in [66], an example of simulated antenna diagram by ProSIVIC®platform is shown with and without the bumper effect.

2.4.2.7 Verification and validation methods and metrics for LiDAR (UGE)

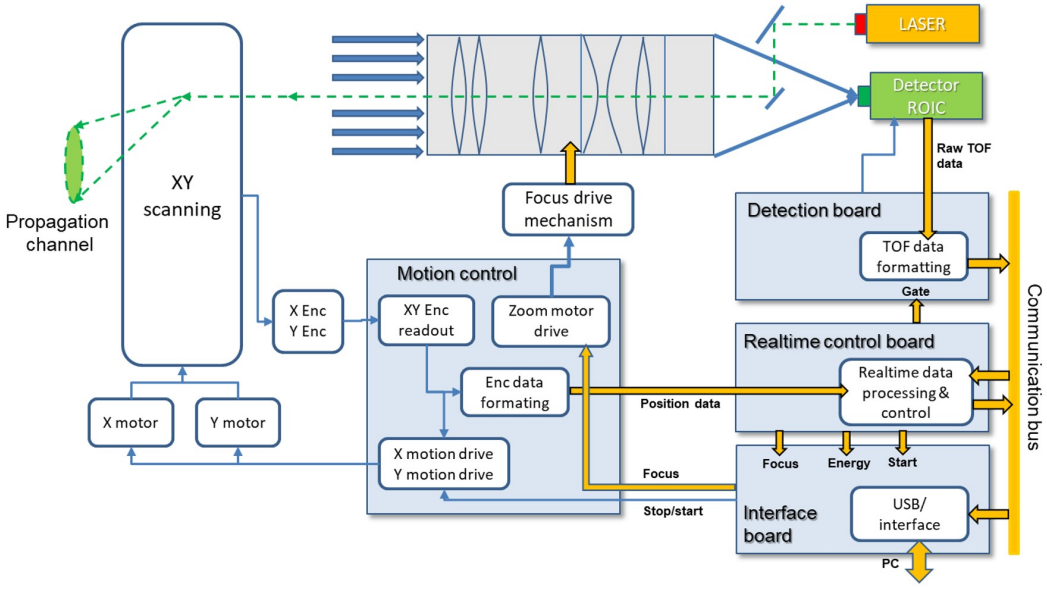


Figure 49: LiDAR functions in a Digital Twin of the sensor

	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 50: LiDAR Parameters for the main type of automotive LiDAR [11]

The general specifications and intrinsic and extrinsic parameters of automotive LiDARs are presented in the figure 50 and involved at least:

- frames per second (FPS)
- accuracy (Acc.)
- detection range (RNG)
- vertical FoV (VFOV)
- horizontal FoV (HFOV)
- horizontal resolution (HR)
- vertical resolution (VR)
- wavelength (λ)
- diameter (Ø)

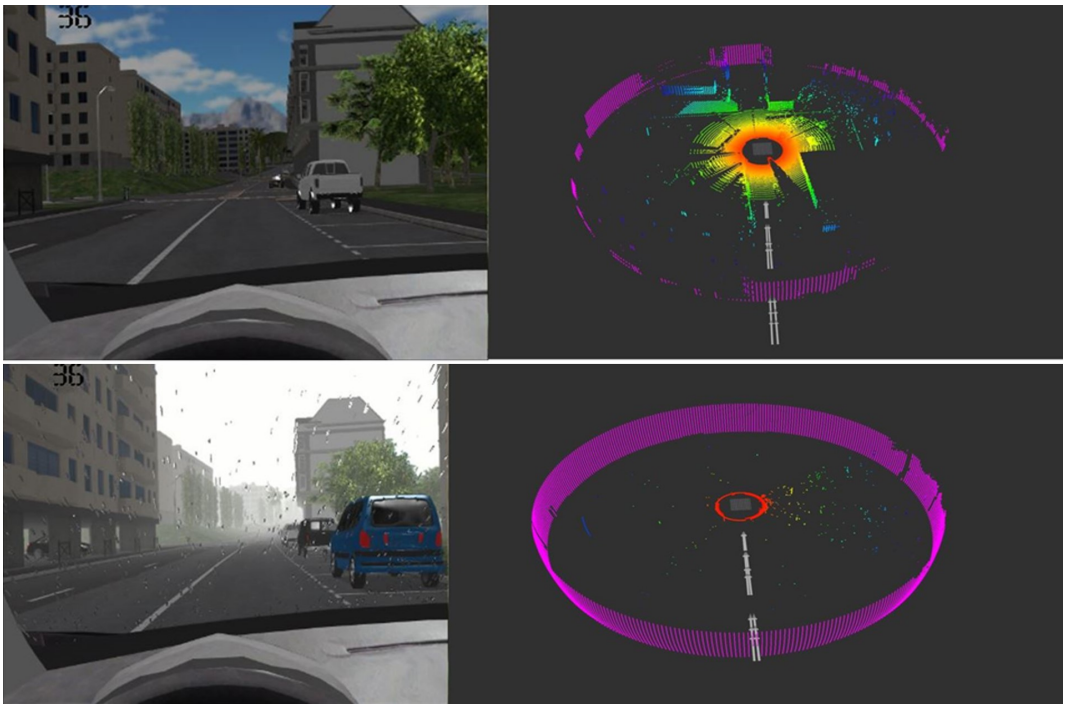


Figure 51: LiDAR modelling in ProSIVIC®with multiple layers and impact of weather conditions on the received laser beams

As for the radar, the simulation of the sensor can be more or less complex and detailed model. The classical approach to construct LiDAR models are:

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 LiDAR signal processing block chain. The outputs of this sensor are ideal targets with appropriate distance and speed.

2. **Physics based LiDAR**: Utilise physics-based modelling to simulate the behaviour of light and disturbers (rain, fog, snow, dust, smoke) in the environment. This approach involves either ray-tracing techniques to emulate laser propagation, reflection, and refraction accurately, or Z-buffer techniques with some physical models simulating the propagation channel. Sometime, a hybride version merge both ray-tracing and model based approach.

3. **Sensor Fusion and Machine Learning**: Combine LiDAR simulation with other sensor models (such as cameras or radars) and employ machine learning techniques to enhance the accuracy of the generated point cloud or to improve object detection and classification.

4. **Hybrid approaches**: Combine different methods, such as using pre-existing LiDAR sensor models and customising them to match specific requirements, or integrating simplified physics-based models with machine learning algorithms to enhance realism.

The construction of a physics based LIDAR model could follow these steps:

- Understanding LiDAR Principles: Before starting, grasp the fundamental principles of LiDAR, including laser emission, time-of-flight measurement, and point cloud generation.

- Choose Simulation Software: Select a simulation platform or software that supports the creation of custom sensor models and has capabilities for physics-based simulations.

- Sensor Characteristics and Parameters: Define the specifications of the virtual LiDAR sensor, such as its field of view, range, resolution, scanning pattern, wavelength, and accuracy. Real-world LiDAR specifications can serve as a reference.

- Simulating Laser Emission and Propagation: Implement algorithms to simulate the emission of laser pulses from the LiDAR sensor and model their propagation through the virtual environment. Consider factors like beam divergence, attenuation, and reflection properties.

- Detecting and Recording Returns: Simulate how laser pulses interact with objects in the environment, calculate the time-of-flight for each reflected pulse, and generate a virtual point cloud by recording the position and intensity of returns.

- Error Modelling and Calibration: Incorporate error models to account for sensor inaccuracies, noise, and other real-world limitations. Calibration procedures may be simulated to adjust for imperfections.

- Integration with Simulation Environment: Integrate the LiDAR model into the simulation software, allowing it to interact with other simulated components like vehicles, pedestrians, or dynamic environments.

When constructing a LiDAR model for simulation software, it's crucial to validate the model against real-world data or known scenarios to ensure its accuracy and reliability in various conditions. Additionally, iterate and refine the model based on feedback and new developments in LiDAR technology.

We can use these steps to evaluate and calibrate a LiDAR model:

- Evaluation of LiDAR model:

  - Data collection: gather real-world LiDAR data from sensors in various environments, capturing different scenarios, such as urban, suburban, rural, or indoor settings. This data will serve as a reference for evaluating the simulated LiDAR output.
  - Comparative analysis: compare the virtual LiDAR-generated point clouds with the real-world data. Use metrics like point cloud density, accuracy in object detection, range measurements, and the ability to capture detailed features.
  - Error analysis: assess discrepancies between the simulated LiDAR output and the real-world data. Identify sources of error, such as inaccuracies in sensor parameters, simulation environment modelling, or algorithms used for simulation.
  - Validation scenarios: design specific validation scenarios to stress-test the LiDAR model in challenging conditions, such as low-visibility situations (fog, rain, snow) or scenarios with complex reflective surfaces (glass, mirrors).

- Calibration of LiDAR model:

  - Adjusting sensor parameters: modify the parameters of the virtual LiDAR sensor based on the observed differences between simulated and real-world data. This includes fine-tuning values like field of view, range, resolution, beam divergence, and sensor orientation.
  - Error correction algorithms: implement correction algorithms to account for systematic errors identified during the evaluation phase. These algorithms can compensate for sensor noise, calibration errors, or environmental factors affecting LiDAR performance.
  - Model refinement: update the LiDAR model based on insights gained from the evaluation process. Refine the simulation algorithms, sensor behaviour, or environmental interactions to better mimic real-world LiDAR performance.
  - Iterative process: calibration is often an iterative process. Make incremental adjustments to the model parameters, validate the changes against real-world data, and continue refining until the simulated LiDAR output closely matches observed real-world behaviour.
  - Validation and verification: validate the calibrated LiDAR model in various scenarios and environments to ensure its improved performance and accuracy across a wide range of conditions.

Tools and Techniques:

- Use of accurate, calibrated tools and targets.
- Statistical analysis to quantify discrepancies between simulated and real data.
- Optimisation algorithms for parameter tuning.
- Machine learning techniques to fine-tune models based on observed errors.
- Sensor calibration procedures to mimic real-world calibration processes.

Calibration and evaluation are ongoing processes in LiDAR model development, ensuring that the virtual sensor's behaviour aligns closely with real-world performance and remains accurate across different scenarios and environments. But for the calibration part, we can recommend doing it with both a real sensor with a calibration bench and its digital model and calculating the error $\epsilon = |v_{model} - v_{real}|$ made by the model. As an example, in [68], a very sophisticate calibration and evaluation bench is proposed using motorised rotation stage and motorised goniometer by Newport company to have an accurate measurement of horizontal and vertical Field of View and the angular resolution. The devices are shown in the figure 52. Moreover a rail system is also used for an accurate positioning of calibration targets or heavy obstacle.

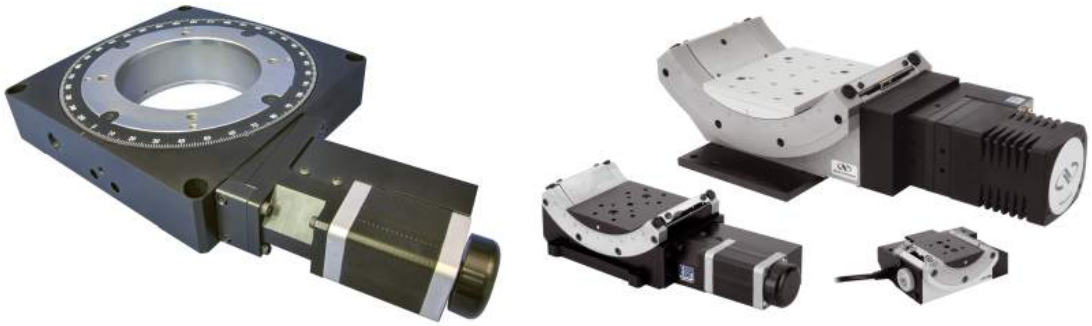


Figure 52: Motorised rotation stage and goniometer by Newport for the sensor's field of view measurement.

For the evaluation part, it will also interesting to use the hardware-in-the-loop (HIL) solution as it this presented in [66] with the RADAR but the solution can be also applied to LiDAR using the HIL simulation tools by dSPACE as shown in figure 48.

2.4.2.8 Verification and validation methods and metrics for GPS

Developing a GPS simulation for autonomous driving involves addressing various high-level requirements to ensure accurate and realistic positioning information. Here are a set of the main requirements to consider:

- **Realistic Geographical Data**: Utilise high-quality and up-to-date geographical data, including maps, road networks, landmarks, and relevant infrastructure. This requirement allows to take into account realistic environments for the simulation to accurately represent real-world driving scenarios having 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 requirement implies to take into account the computation of the pseudo-distances and to use these pseudo-distances in order to generate NMEA frame. The goal consist to mimic the continuous and instantaneous updates required for accurate autonomous navigation.
- **Satellite Constellation Simulation**: Simulate a realistic satellite constellation, including the positions and movements of GPS satellites, to generate authentic GPS signals. This

requirement imply the generation of clock correction. The use of the ephemera file with positioning of satellites ensure that the GPS simulation accurately replicates the satellite signals received by an actual GPS receiver.

- **Atmospheric layers modelling**: Simulate a high quality propagation a the GPS signal (CA frame, L1 and L2 signals ...) though the atmospheric layers with the impact and interaction on the signal transmission and the pseudo-distance accuracy. The main layer to model are the ionospheric and the troposphere layers.
- **Accuracy and Precision Control**: Enable control over the accuracy and precision of the GPS simulation to mimic different real-world scenarios and GPS receiver capabilities. Allow 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 how well the system can navigate 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 (Global Navigation Satellite System) receivers. Reflect the real-world scenario 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 the system's 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 station allowing to provide positioning error for the GPS correction (DGPS and GPS RTK). Mimic the complementary use of a reference using the same satellites in order to send by using of communication means the positioning correction.

The objective of the verification and validation protocol for GPS simulation consists to respect the requirements identify for GPS and ensure the accuracy, reliability, and performance of the GPS model integrated into the vehicle simulation software. The final objective consist to validate that the GPS model provide 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)

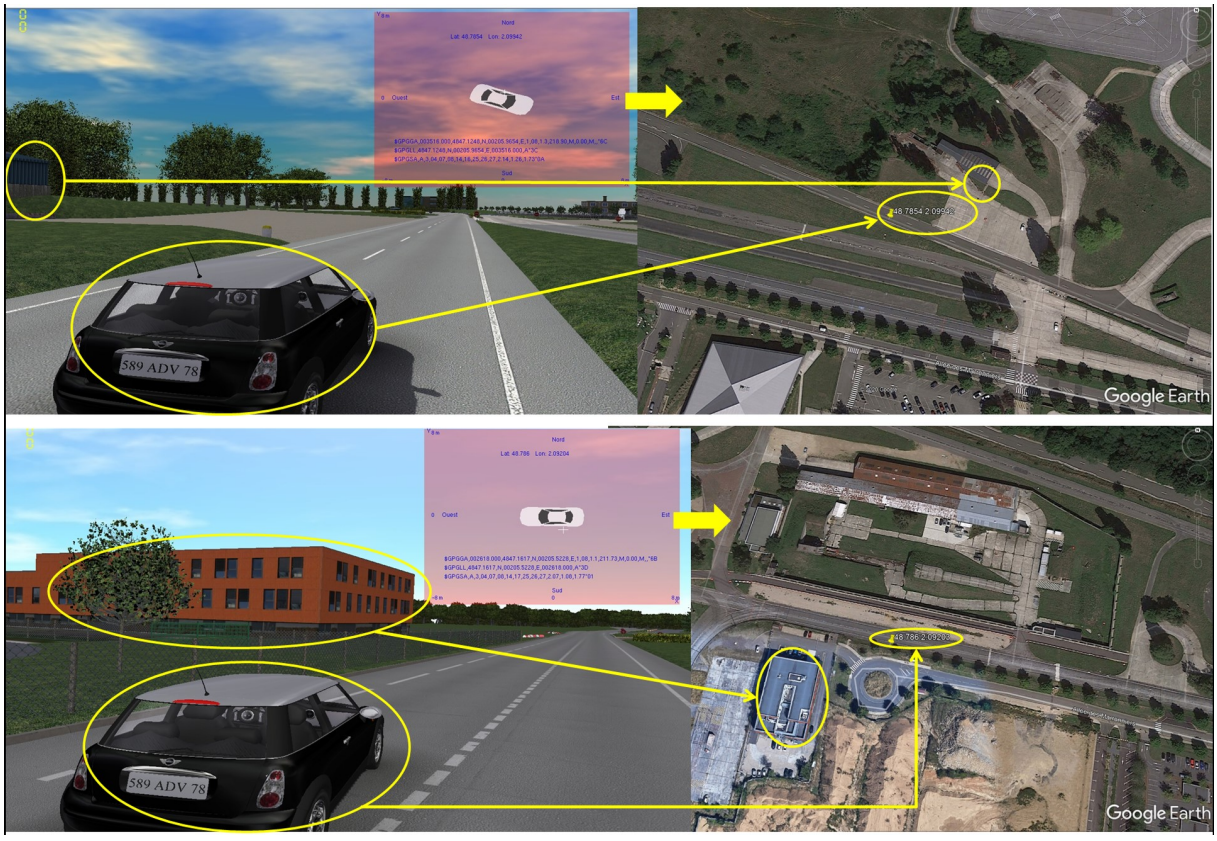


Figure 53: Result of NMEA frame generation and projection in Google Earth

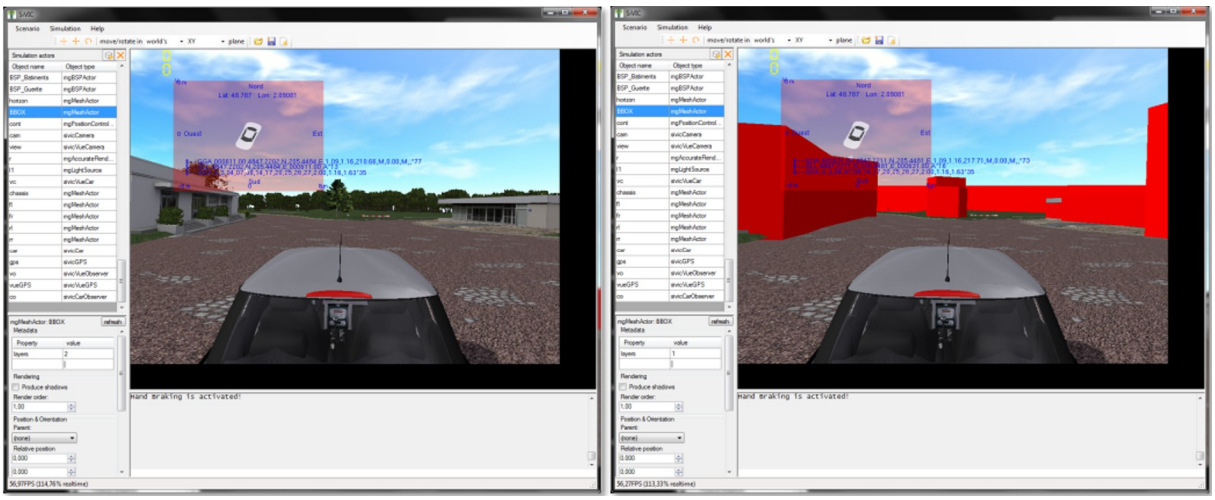


Figure 54: Modelling of the environment in order to take into account ground disturbances (multiple reflection and occlusion)

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 55), and the different function of the receiver (see figure 56). All these functions represent a complex process for the modelling of a high level GPS simulation.

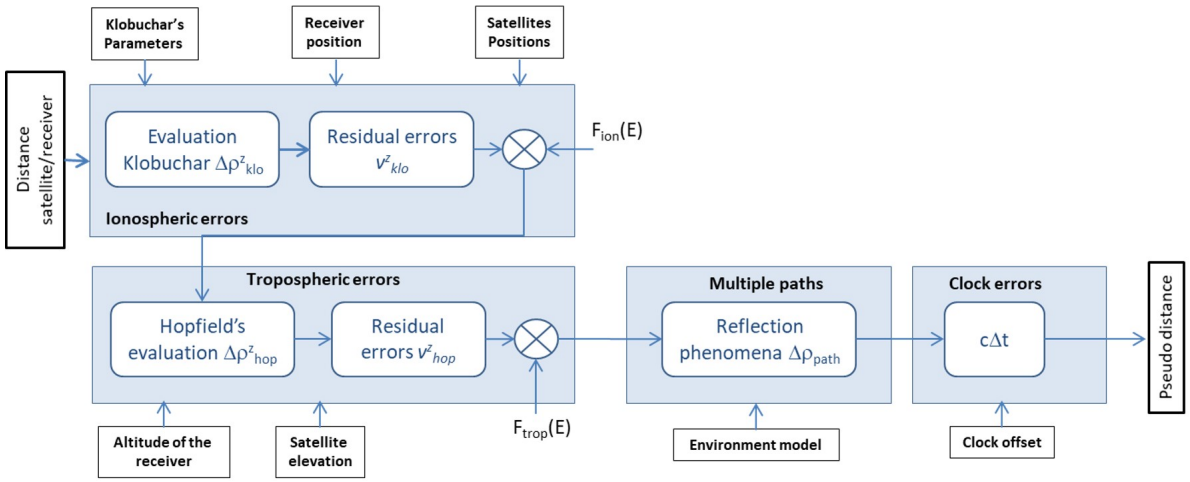


Figure 55: Diagram of the different disturbances on the GPS signal (functions involved in the Pro-SiVIC's GPS model)

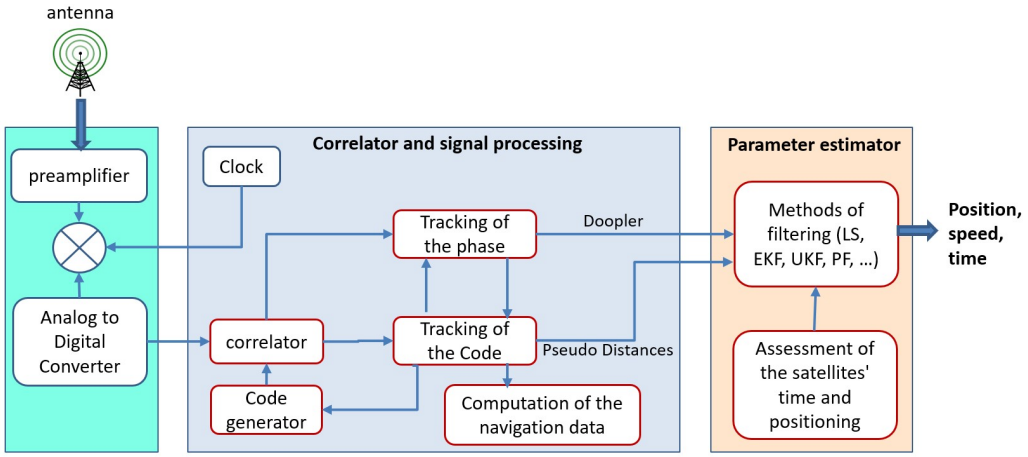


Figure 56: Diagram of the different functions involved in the GPS reveiver)

In [12], a GNSS model is proposed for urban area with the modelling of the main issues encountered by real GNSS. From the figure 57 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.

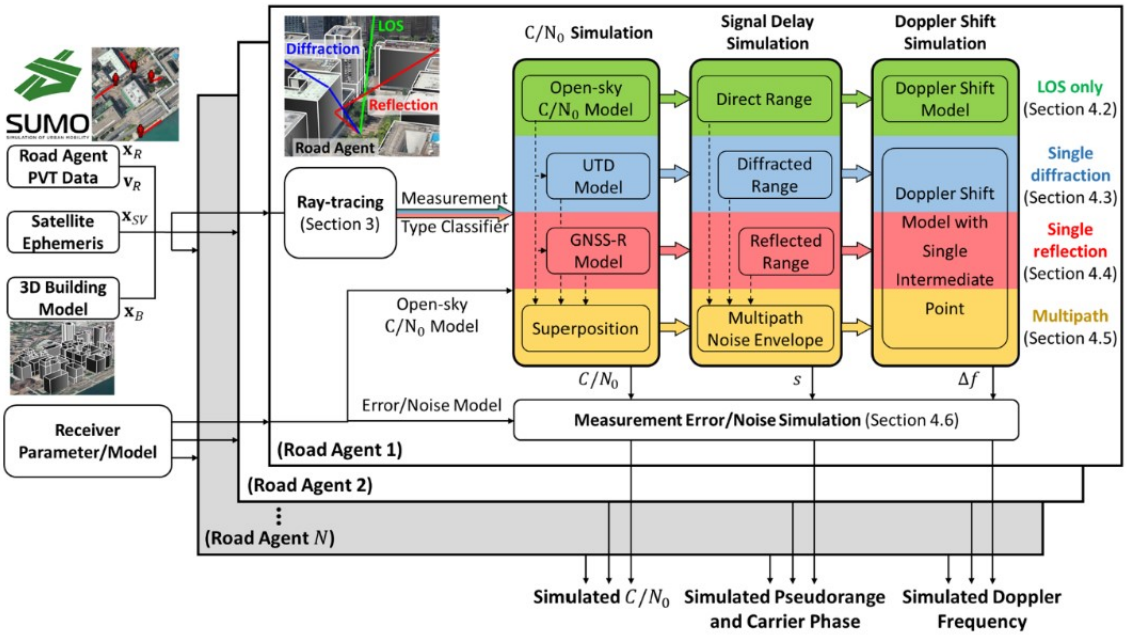


Figure 57: Diagram of realistic GNSS simulation in urban multi-agent context (GNSS RUMS) [12]

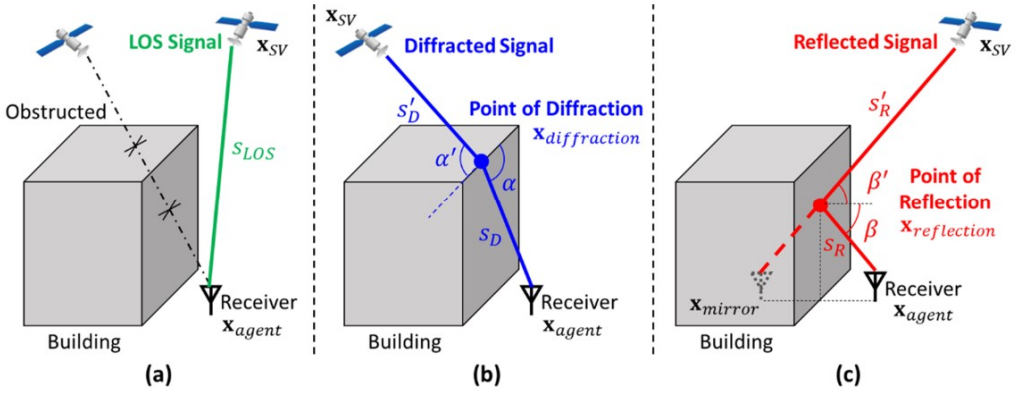


Figure 58: Main effect encountered in urban areas including (a) the LOS signal; (b) the diffracted signal; (c) the reflected signal [12]

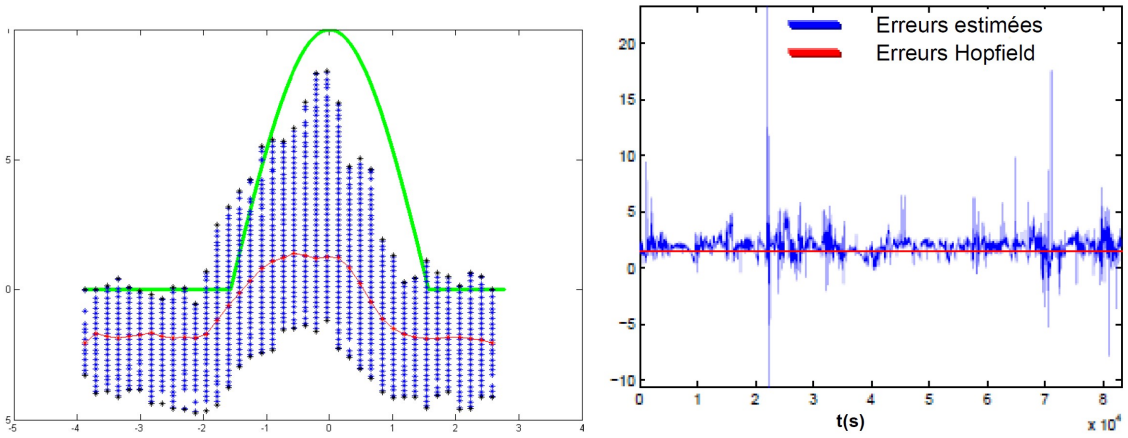


Figure 59: Left: Ionospheric normalized errors over 24 hours. In green, errors assessed by the Klobuchar model disseminated by the GPS system. In blue, EGNOS corrections obtained over several months. In red, a random and continuous representation of EGNOS corrections. Righ: Tropospheric errors over 24 hours. In blue, the measurements obtained with a UBLOX receiver. In red, the evaluation of these errors by the Hopfield model.

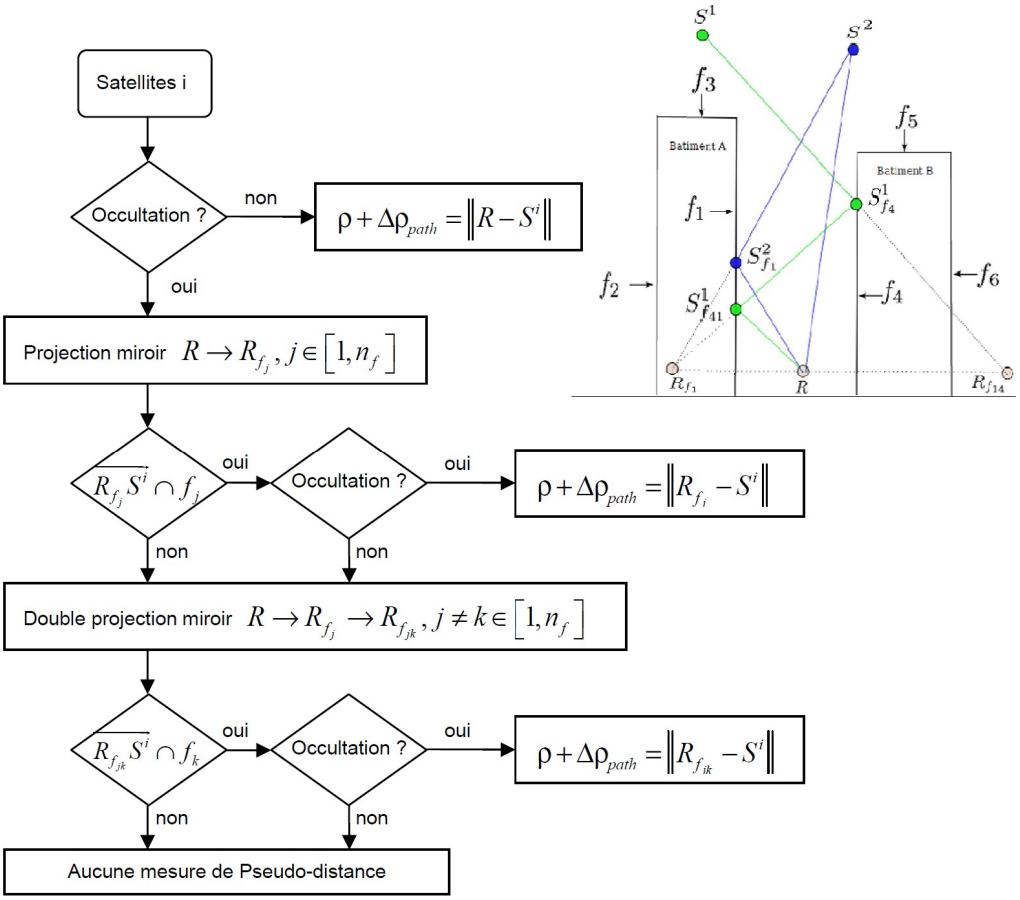


Figure 60: Calculations of possible pseudo-distances based on multi-paths.

Verification of possible components of a GPS Model:

- Atmospheric Layer Models:

  - Verify the implementation of models (e.g., ionospheric and tropospheric models) affecting GPS signals.The ionosphere is the uppermost layer of the atmosphere, extending from approximately 50 km to 750 km in altitude, with an average altitude of 350 km. It introduces delays in the transmission of electromagnetic waves that pass through it. These disturbances depend on the ionosphere's exposure to sunlight or solar radiation. Indeed, sunlight ionises the hydrogen molecules in this layer, causing variations in the Total Electron Content (TEC) and thus delaying the transmission times of signals emitted by satellites to varying degrees. The greater the solar radiation, the higher the TEC value, and the more significant the delay. Conversely, when this layer is not exposed to sunlight (during the night), this delay is modelled as constant and of low amplitude. These delays result in errors on the order of a few meters in the measurement of pseudo-distances. The most widely used model for representing and correcting ionospheric errors is the Klobuchar model (other models exist, such as the Global Harmonic Model (CODE) or local model (RGP), but they are much more complicated to implement). It provides an assessment of the propagation delay of a GPS signal induced by the passage through the ionosphere. It is also integrated into most GPS receivers and used to calculate correction grids for SBAS systems (EGNOS, WAAS, MSAS, etc.). The characterisation of ionospheric error behaviour should rely on this theory. Figure 59 depicts the normalised delay evaluated with the Klobuchar model disseminated by the GPS system (the green curve) and, sampled over time, the assessments obtained from EGNOS data collected over several months (the blue points). Taking into account these results and the general formula of the Klobuchar model, it is logically possible to formulate a mathematical model capable of simulating ionospheric errors. The troposphere is the lower layer of the atmosphere, in direct contact with the Earth's surface. Its thickness varies from 8 km (at the poles) to 17 km. It is a non-dispersive medium. Indeed, refraction (delay in signal propagation) is independent of frequency for electromagnetic waves with a frequency lower than 15 GHz. Taking into account Fermat's principle, the measurement error on the distance travelled by an electromagnetic wave in the troposphere can be defined. This model needs to take into account the Refractive index (2 parts: dry and wet aspects), the partial air pressure (dry and wet), the temperature, the air compressibility factor (dry and wet), and a set of empirically determined coefficients. The figure 59 is representative of the statistical studies conducted on a large set of experimental data collected in eMOTIVE project. This figure indicates that at a constant altitude, tropospheric errors can be modelled by summing: 1) the estimate evaluated by the Hopfield model, 2) a constant bias, and 3) non-white noise which can be modelled by a first-order autoregressive model. Note that the bias is not actually constant but largely depends on the altitude of the receiver. A study to characterize the bias as a function of altitude is necessary but proves to be challenging to implement. If we remain at an altitude that varies little, then this estimation is not justified.
  - Validate against known atmospheric conditions data using statistical metrics: Mean Absolute Error (MAE) of signal delays and Root Mean Square Error (RMSE) of signal delays.

- Calculation of Pseudo-Distances:

  - Verify algorithms for calculating pseudo-distances between satellites and the receiver. This algorithm is a well-known method. The main difficulties are related to the errors assessment used in the pseudo distance computation.
  - Validate pseudo-distances against known ground truth data or simulated scenarios: Check accuracy using Relative Positioning Error metrics and validate against ground truth data using RMSE of calculated pseudo-distances.

- Generation of NMEA Frames:

  - Verify the generation process of NMEA (National Marine Electronics Association) frames.
  - Validate NMEA frames against unexpected outputs: Check for compliance with NMEA standards (e.g., NMEA 0183) using parsing and validation tools. Verify correctness of parameters (latitude, longitude, time, etc.) within NMEA frames. In the NMEA 0183 standard, all data is transmitted in the form of ASCII characters, including all printable characters, as well as the [CR] Carriage Return and [LF] Line Feed characters, at a transmission speed of 4800 bauds. The data is sent in the form of frames (sentences, phrases). Each frame starts with the character $ followed by a group of 2 letters for the receiver identifier (GP, LC, OM, ...). Then the message includes a group of 3 letters for the frame identifier (type of frame). The main types of frames are: GGA (GPS Fix and Date), GLL (Geographic Positioning Longitude Latitude), GSA (DOP and active satellites), GSV (Visible satellites), VTG (Direction (course) and speed (in knots and km/h)), RMC (Minimum usable specific data). Generally, the first three types of messages are more commonly used. The frame is completed by a number of fields separated by a comma, and finally, an optional field called checksum preceded by the $*$ sign, which represents the exclusive OR of all characters between $ and $*$ (except for the $ and $*$ boundaries). Finally, the sequence is closed with a [CR][LF]. A total of 82 characters maximum are used for a frame.

- Multi-Reflection/multi-path Modelling:

  - Verify the modelling of multi-path effects in the GPS signals. The consequences of multi-path on measurements are twofold: signal attenuation and the introduction of a propagation delay with each reflection. While signal attenuation is observable and provided by the receiver as a signal-to-noise ratio, identifying the propagation delay of signals related to multi-path is challenging compared to delays introduced by other error sources. Moreover, these delays depend solely on the environment's configuration, making it impossible to characterise and therefore simulate them other than from a theoretical perspective. Generally, the simulation of multi-path effect involves 3 steps: 1) determining the possible paths through which signals emitted by satellites can reach the receiver, 2) selecting the path that is both the shortest and has undergone the least number of reflections (beyond 2 reflections, the signals are too attenuated, and therefore, the corresponding paths are excluded from consideration), and then 3) recalculating the theoretical distance between the receiver and visible satellites (see figure 60 developed in eMOTIVE project and implemented in Pro-SiVIC).

  - Validate against simulated scenarios or empirical data: Quantify multi-path error using metrics such as Signal-to-Noise Ratio (SNR) degradation. Validate against known multi-path scenarios using statistical methods.

- Satellite Orbit and Clock Models:
  - Validate the accuracy of satellite orbit models (ephemeris data) and clock models used for satellite position and time corrections. If the ephemeris data for a specific date are downloaded and used, then it is possible to obtain the corrections of the satellite's clocks.
  - Verify against known satellite positions and clock data provided by satellite agencies (like the International GNSS Service - IGS).
  - Metrics: Evaluate Mean Time Difference (MTD) or orbit deviations.

- Receiver Dynamics and Sensitivity:
  - Validate receiver dynamics and sensitivity to different signal strengths, frequencies, and environmental conditions.
  - Assess the receiver's ability to handle weak signal scenarios, signal acquisition time, and sensitivity to signal interference.When the time of flight is evaluated by the receiver, the time reference becomes that provided by the receiver's clock. Since the quartz in the receiver is much less stable than the atomic clocks on satellites, this time reference quickly diverges over time, introducing a bias in the measurement of time of flight and, consequently, in pseudo-distance measurements. It is noted that the receiver clock drift is considered as an unknown in the system of equations used to calculate the receiver's position. Based on results obtained during experiments in the eMOTIVE project, and except for discontinuities, it is easy to see that the clock drift over time can be represented by a linear drift (in our experiments, it was approximately 1 μs/s). It should be noted that the discontinuities are explained only by a self-correction of the receiver's clock, although this is not specified in the technical documentation of the used receiver.
  - Metrics: Sensitivity analysis, Time to First Fix (TTFF), Signal-to-Noise Ratio (SNR) thresholds.

- Error Sources and Correction Models:
  - Validate error sources such as clock biases, satellite ephemeris errors, and atmospheric delays.
  - Assess the effectiveness of correction models (like Differential GPS - DGPS) in reducing errors.
  - Metrics: Differential corrections applied, comparison of corrected vs. uncorrected positions.

- Signal Propagation and Signal Degradation Models:
  - Validate signal propagation models considering terrain, obstructions, and atmospheric effects.

  - Assess signal degradation due to foliage, buildings, or other environmental factors.
  - Metrics: Signal strength attenuation models, Probability of Line-of-Sight (LOS) obstruction.

- Real-Time Kinematic (RTK) and Precise Point Positioning (PPP):
  - Validate high-precision positioning techniques like RTK and PPP if implemented in the GPS model. This also means the RTK behaviour and operating in the case of the lost of signal and L1/L2 correlation issue (correlation between emitter and receiver)
  - Compare simulated RTK/PPP outputs against known high-precision reference data.
  - Metrics: Accuracy improvement achieved by RTK/PPP, convergence time.

- Integration with Vehicle Dynamics:
  - Validate the integration of GPS with vehicle dynamics and navigation algorithms.
  - Assess the impact of GPS outputs on vehicle navigation accuracy and stability.
  - Metrics: Navigation accuracy, impact on vehicle trajectory prediction.

To validate the operation of a GPS simulator, it is generally necessary to conduct several experiments. These involve a comparative study between the results obtained using the simulator and the results acquired with a real GPS receiver. These results depend on numerous parameters (receiver position, environment, satellite constellation, atmospheric conditions). In practice, it is not possible to evaluate each individual error source, and only their overall contribution is taken into consideration. Thus, the comparative study usually involves performing a statistical analysis of simulated and real positioning errors.

These experimental validations are carried out in two steps: firstly, with static positioning, and then with dynamic positioning. The dynamic experimentation is conducted by integrating the simulation computer, real GPS, and RTK GPS into a real vehicle. The RTK GPS serves as a reference and allows for controlling the position of both the ego-vehicle and the simulated receiver. Data collection is performed using a platform like RTMaps with synchronised data timestamps. It is also important to have a faithful and accurate digital twin to reproduce the effects of multipath reflection. For a more accurate evaluation and validation, We can propose this possible, and more difficult to implement, methodology for the validation, this protocol serves as a framework and an actual implementation and specific metrics may vary depending on the software, the model, the requirements and available resources:

1. **Simulation Environment Setup**: configure simulation parameters including vehicle dynamics, environmental conditions, and GPS constellation. This step is essential and needs to validate the Digital Twin of the environment with the capabilities to mimics multi-path effects. In this stage, we need to validate the availability of the ephemeris files.

2. **Model Implementation Verification**:
   - Conduct unit tests for individual components of the used GPS model (atmospheric models, pseudo-distance calculation, NMEA frame generation, multi-reflection, etc).
   - Verify algorithms against analytical solutions or well-established models.
   - Use code review and static analysis tools to ensure implementation correctness.

3. **Data-Driven Validation**: Collect real-world GPS data or use pre-existing datasets for validation purposes and compare simulation output against real-world data using established metrics (e.g., MAE, RMSE). To obtain a representative characterisation of GPS error behaviour, experimental data need to be collected over several weeks or months. The collect of this large amount of GPS data allow to meet various atmospheric conditions such as sunny weather, rain, snow, and frost which could have a significant effect on the GPS operating and performance. With a set of experimentation with a 24 hours duration, these acquisitions will be relevant to study the actual GPS error behaviour, particularly ionospheric errors during both day and night. To ensure experiment repeatability, the receiver need to be placed and kept in a high and clear viewpoint like the roof of a building overlooking its surroundings throughout all acquisitions. This configuration is necessary to avoid partial satellite occlusion or the introduction of multi-path effects that make it difficult to characterise other sources of errors. A dedicated dataset need to be built for the occlusion and multi-path effect. For all these dataset, it is mandatory to add ephemerid files for the specific time period.

4. **Scenario-Based Validation**: design specific scenarios to stress-test the GPS model (e.g., urban canyons, open fields, signal blockage) and validate model performance under these various scenarios and compare against expected outcomes.

5. **Cross-Validation**: do not hesitate to use several validation approaches to ensure consistency and reliability of results.

For instance, with the Pro-SiVIC GPS model and in a static position experimentation without multi-path effects (actual and simulated GPS at the same place), the horizontal errors have been estimated.
The horizontal errors h are defined as the norm of the horizontal components of the positioning results. We also assume that their statistical behavior follows a Rayleigh distribution knowing that hRMS is the root mean square of the errors horizontal. Figure 61(a) and Figure 61(b) show the dispersion of measured and simulated horizontal errors, respectively. Figure 61 (c) and Figure 61 (d) present their probability distributions as well as the theoretical Rayleigh distributions deduced from these same errors (curves in black). From these figures, qualitatively we can deduce that the measured and simulated errors do indeed follow a Rayleigh distribution.

horizontal errors without multi-path				
	measured	theoretical	simulated	theoretical
hMean	4.736	4.928	4.665	4.730
hRMS	5.562		5.338	
hCEP	4.205	4.617	4.294	4.431
hR95	10.130	9.734	9.460	9.342

Table 4: Horizontal error similarity test without multipath

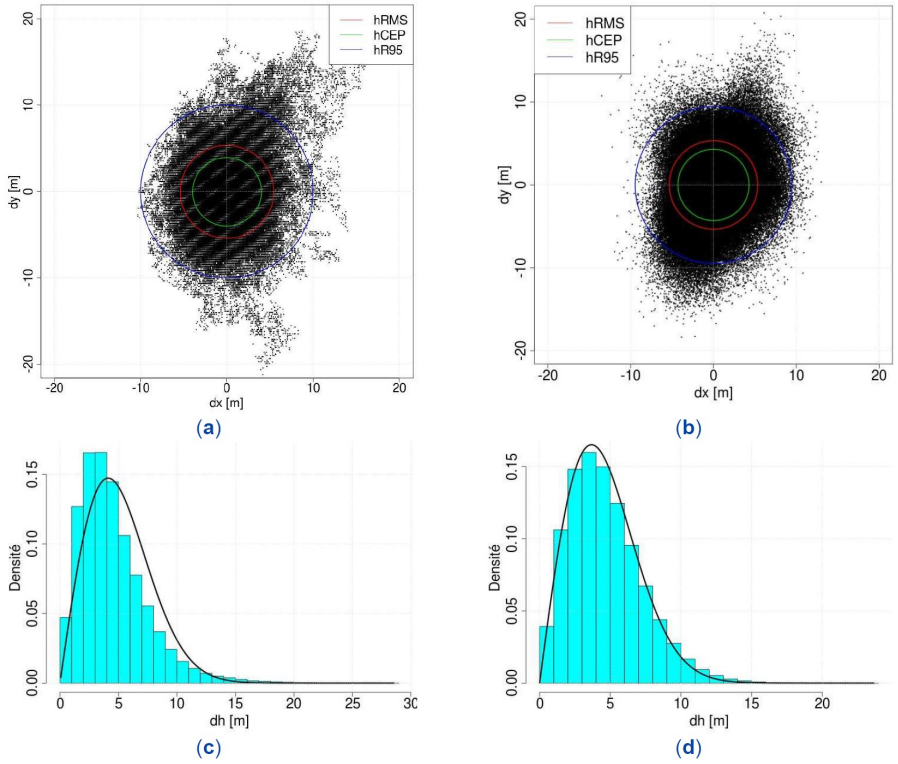


Figure 61: Horizontal errors without multi-path: (a) errors measured using a GPS receiver, (b) simulated errors, (c) probability distribution of measured errors, (d) probability distribution of simulated errors

Quantitatively, to carry out the similarity tests, we can choose the following evaluation criteria: the average error hMean, the circular probable error hCEP such that $P(h \leq hCEP) = 0.5$ and the probable error at 95 % hR95 such that $P(h \leq hCEP) = 0.95$. The results in Table 4 support this hypothesis.

The vertical errors v are defined as the absolute value of the vertical component of the positioning results. We also assume that their statistical behaviour follows a half-normal distribution. Figure 62(a) and Figure 62(b) respectively present the probability distributions of measured vertical errors and simulated as well as the theoretical half-normal distributions deduced from these errors (curves in black). From these figures, qualitatively we can deduce that the measured and simulated results follow a half-normal distribution.

Vertical errors without multi-path				
	measured	theoretical	simulated	theoretical
vMean	6.063	6.203	6.552	6.662
vRMS	7.774		8.350	
vR50	4,910	5.243	5.408	5.632
vR95	15.390	15.240	16.610	16.370

Table 5: Vertical error similarity test without multi-path

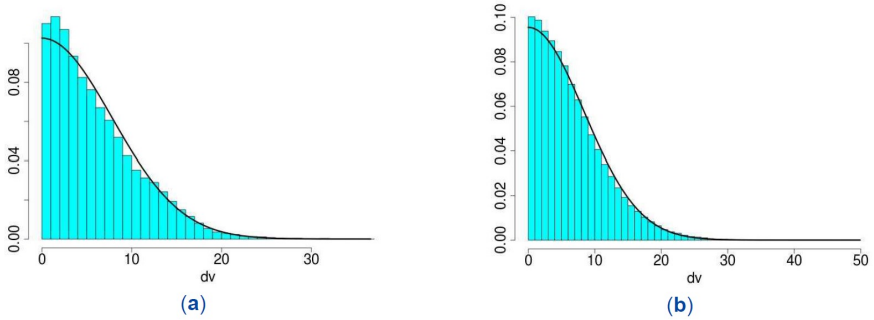


Figure 62: Vertical errors without multi-path: (a) probability distribution of measured errors, (b) probability distribution of simulated errors

Quantitatively, the following similarity criteria are used: the average error vMean, the probable error at 50% vR50 such that $P(v \leq vR50) = 0.5$ and the probable error at 95% vR95 such that $P(v \leq vCEP) = 0.95$ allows us to validate this hypothesis as shown in Table 5.

By taking into account the statistical study of horizontal and vertical errors, it is therefore easy to show that all of the errors measured and simulated follow the same laws of probability. The results provided by the GPS simulator are consistent with reality and therefore the simulator works correctly without multi-path.

In order to evaluate and to validate the quality of the GPS model, another experimental validation was also carried out in a specific configuration allowing multiple reflection of GPS signals. The GPS receiver was positioned in a parking area near a building in such a way that multi-paths would disrupt the positioning results. The experimental and simulated data are provided in an ENU frame of reference centred on the exact position of the receiver. The experimental plan was the same and the results are showed in the figure 63 for the horizontal error and in figure 64 for vertical errors.

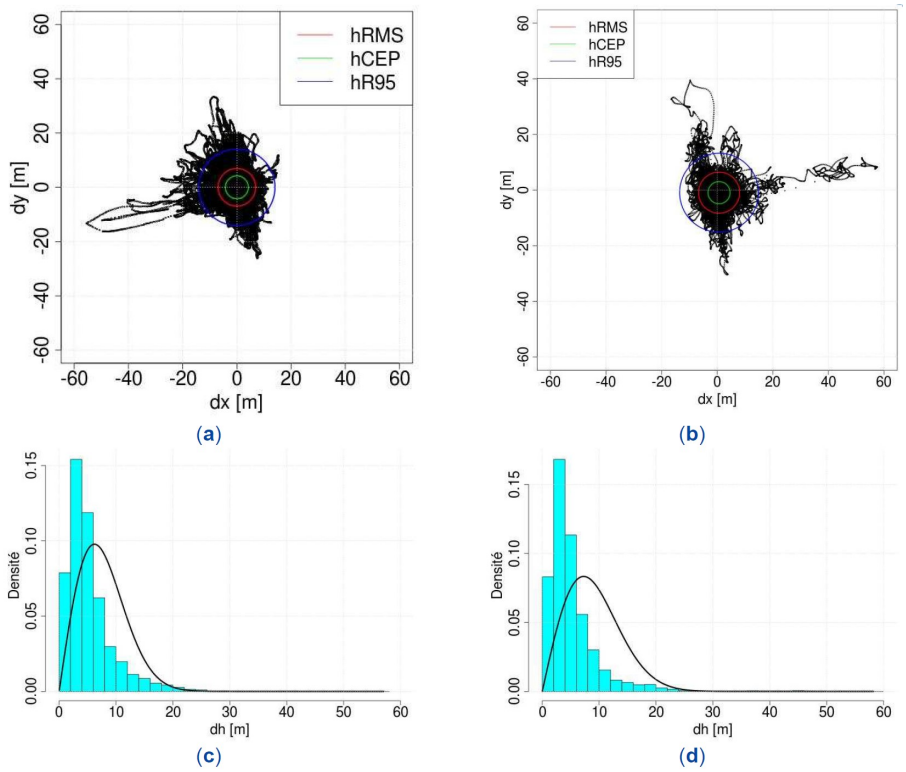


Figure 63: Horizontal errors with multi-path: (a) errors measured using a GPS receiver, (b) simulated errors, (c) probability distribution of measured errors, (d) probability distribution of simulated errors

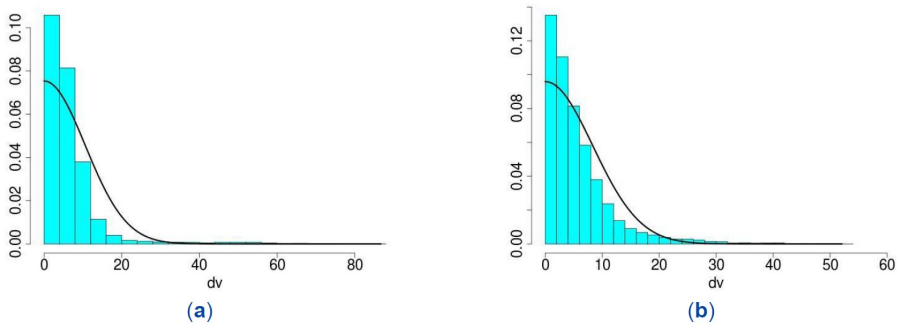


Figure 64: Vertical errors with multi-path: (a) probability distribution of measured errors, (b) probability distribution of simulated errors

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 [69] and [70] 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 65 and 66 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.

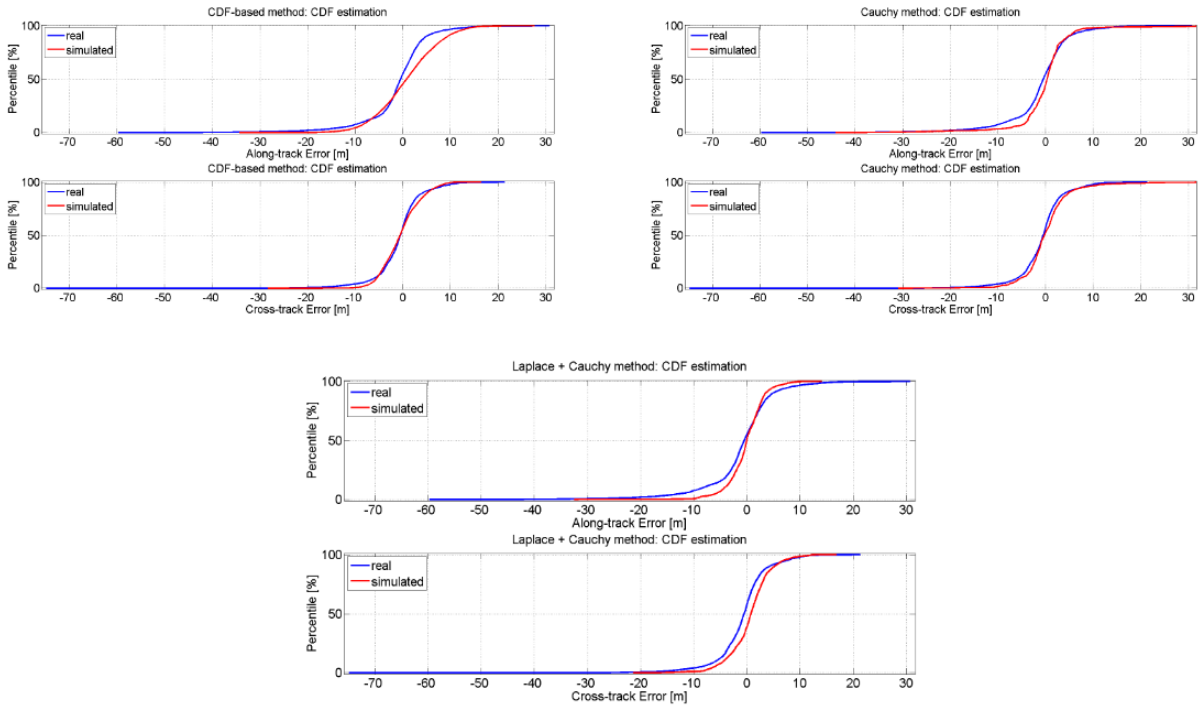


Figure 65: Actual and simulated along-track and cross-track CDFs for CDF-based, Cauchy and Laplace-Cauchy models for the Paris data set

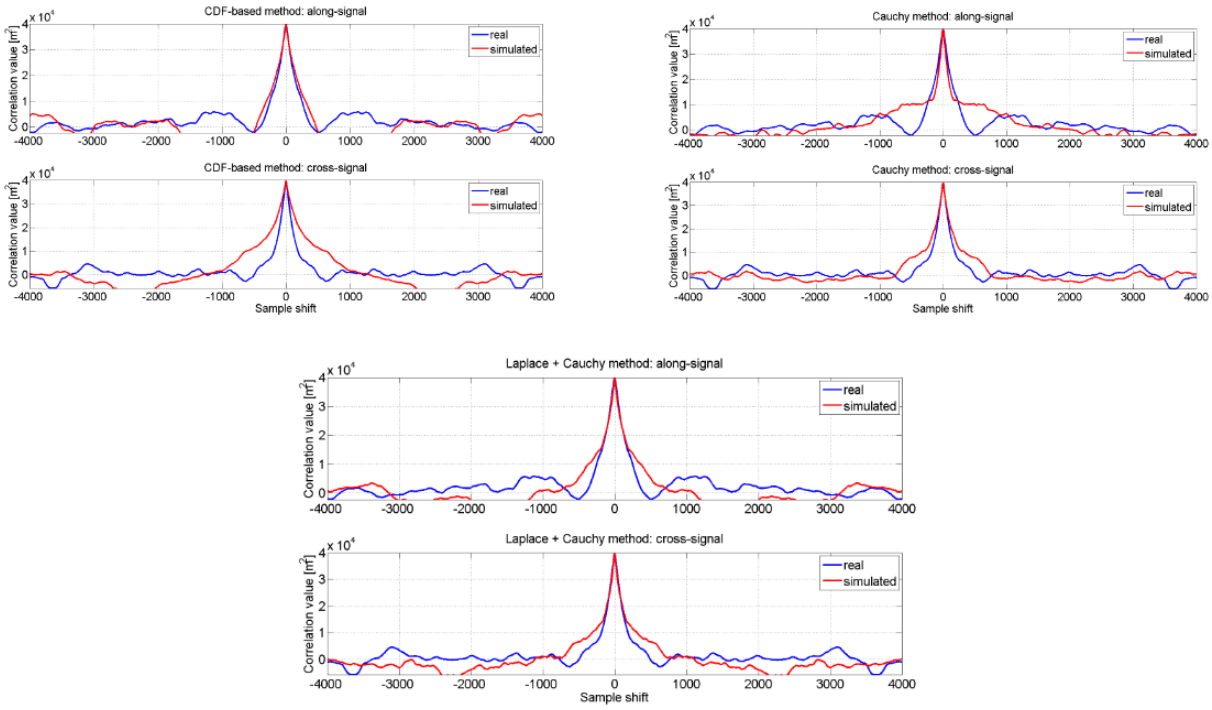


Figure 66: Actual and simulated along-track and cross-track auto-correlation functions for CDF-based, Cauchy and Laplace-Cauchy models for the Paris data set

2.4.3 Mobile dynamics

2.4.3.1 Definition of the types of models and components involved

Mobile dynamics have to be modelled for multiple purpose in simulation. Their behaviour should be described as they interact with sensors, with the other actors, with the environment and with the algorithms. That is why three main aspects should be defined for each mobile:

- their behaviour model : for traffic purpose this behaviour is more accurately described in the next section;
- their dynamic model which makes the behaviour of the mobile physically credible,
- their geometric model which describes how the mobile is viewed by the other actors or sensors.

These three aspects should be accurate enough regarding the validation purpose. That is why these models can be created at different scale regarding the type of mobile that needs to be defined. It is important to distinguish the ego-vehicle (which is the object of interest embedding the system under test), and the other static or dynamic ”actors”, as well as the ”figures”. Regarding the modelling of the dynamics of dynamic objects, it applies both to the ego-vehicle and potentially to mobile actors including dynamic road users. The ego-vehicle needs to have dynamic very realistic. But this is also the case for the actors who surround and interact with the ego-vehicle. Then we have the extras who populate the environment and who, in the majority of cases, do not need the same level of realism and fidelity.

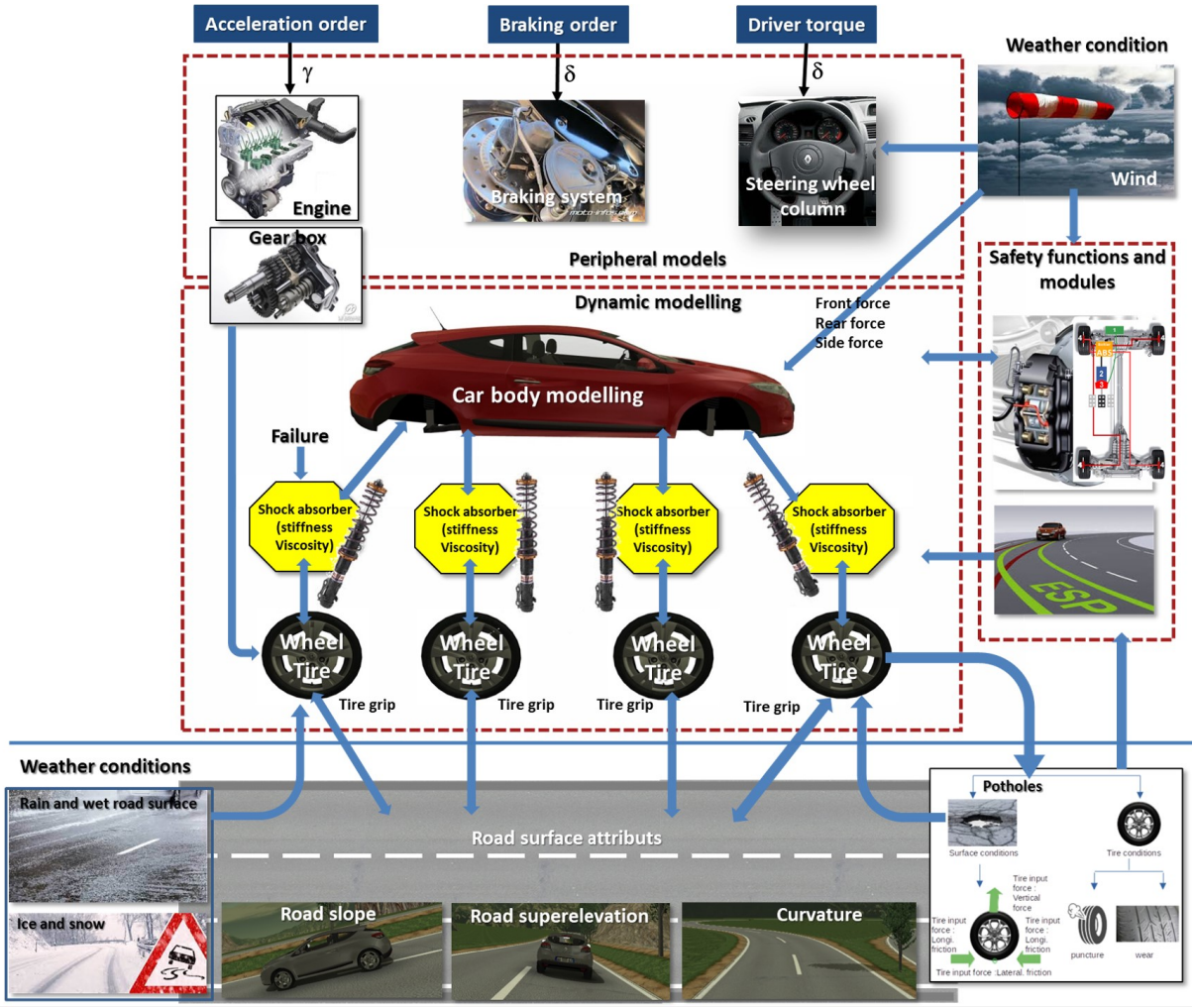


Figure 67: Different functions implemented in the dynamic modelling of a vehicle of Pro-SiVIC (UGE and ESI group)

Vehicles For any vehicle, the dynamic model can be defined as simple as the "Bicycle Model" [71]. The chosen dynamic model can be more accurate if it is able to take into account more parameters such as multi-axles, 4 road picking computation, suspensions, engine, transmission, steering, etc. Even more accurate model can be used in order to describe every mechanical part of the system. That is the case of models such as MADA model developed by Renault, or CALLAS model by AV Simulation [72]. Each type of vehicle then has its specificities.

- Cars: Most of them have two axles and can be described by the bicycle model in the case of a simple kinematic model. Usually, simple models are used to predict the traffic vehicles behaviours but for an ego vehicle, using a simple dynamic model will not be sufficient. Indeed, a correct control model should have an accurate dynamic model associated to have calibrated direction (depending on the steering column) and calibrated gas and brakes (depending on the whole mechanical parts of the vehicle). That is why such use cases rely on sophisticated models such as CALLAS model.

- Trucks: The articulation of the trailer adds a specificity to the dynamic model that can be

modelled with CALLAS model for example.

- Motorcycle: Motorcycles should take into account the different behaviour and rules and add the balancing aspect of the dynamic model.
- Bicycle: Bicycles have a different dynamic model and can be assimilated as pedestrians by some AIs.

Other objects or mobiles With other objects than vehicles, the dynamic models are different. Moreover, the geometry of these objects should be representative, in order to be correctly classified by any viewer or sensor. Particularly, a difficult part of the description comprises the pedestrian or animal object animation that must be realist in order to have a predictable behaviour for the object (a standing pedestrian will not have the same behaviour as a walking pedestrian, or a running pedestrian or a playing child for example).

- Pedestrian: Pedestrians are defined with different dynamic models to be able to predict the behaviour of any human on sidewalks or crossing the roads (example: WALKER-TRAFFIC model in SCANeR Studio)
- Other dynamic objects: The same kind of dynamic models as the Pedestrians models can be used to set the behaviour of any moving target (especially animals) to present diverse situations.

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

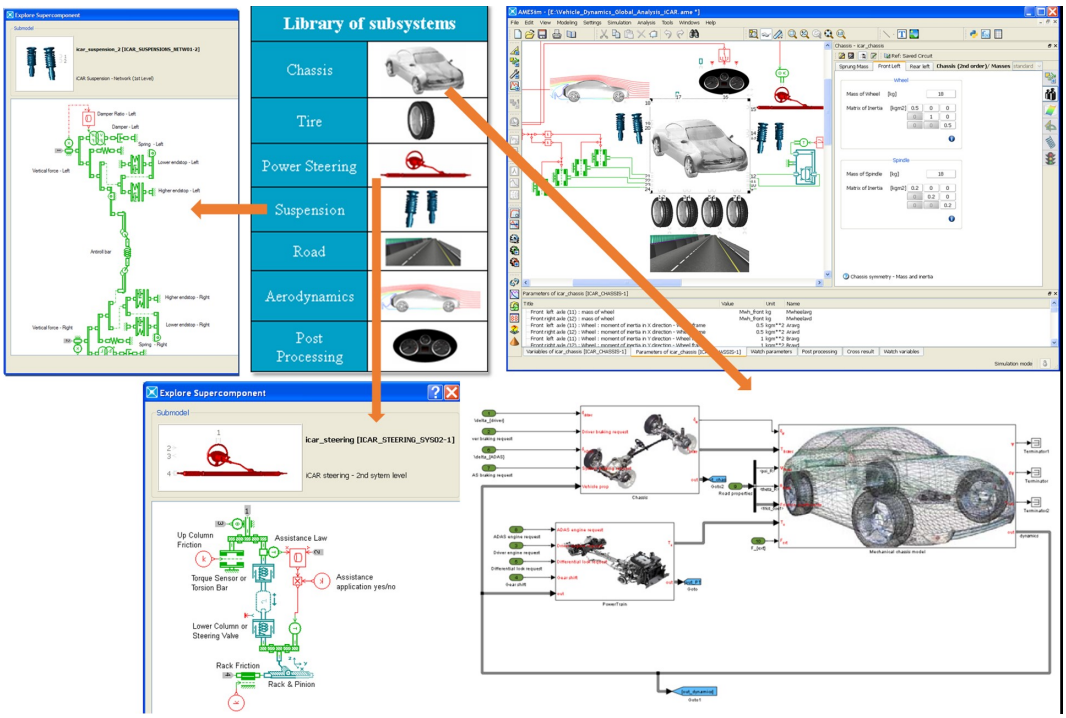


Figure 68: AMESim (SIEMENS) full vehicle model running on HiL platform [13]

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 modeling of vehicle.

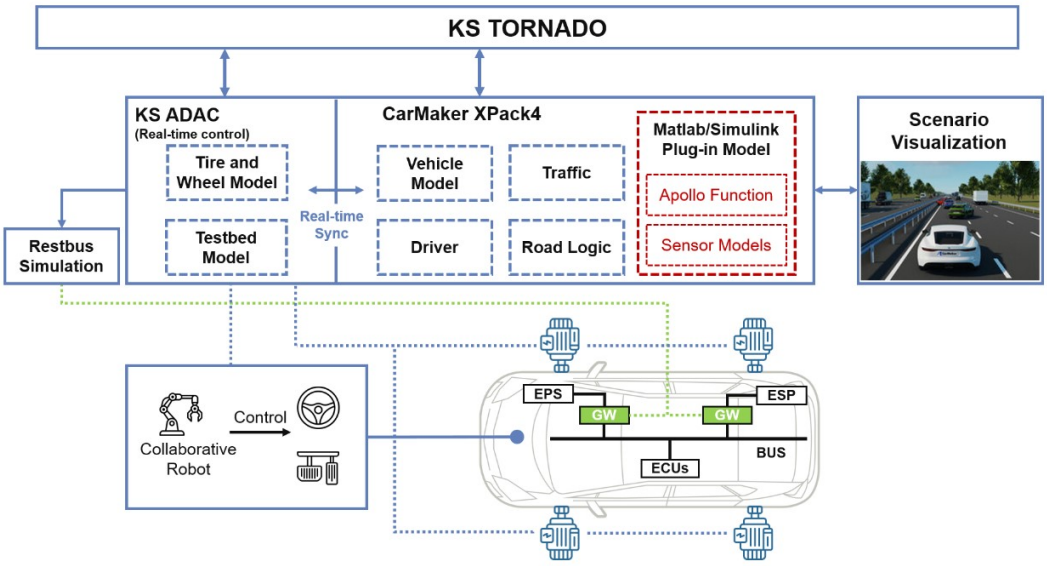


Figure 69: vehicle-in-the-Loop Testbed with Apollo platform and IPG CarMaker model [9]

2.4.3.2 Verification and validation methods and metrics

Verification and validation for mobile dynamics can be done at different scales :

- **Real testing:** Several type of validation methods such as conducting tests on closed tracks, in controlled environments, using a digital twin, various driving scenarios can be simulated. Otherwise dynamic validation on testing benches can be done at different level (Hardware in the Loop, Vehicle in the Loop, Human in the Loop).
- **Comparison with reference model:** Without real tests, reference model can be used to validate mobile dynamics. That is the case with CALLAS model that can be used as a dynamic reference models for other simple models.

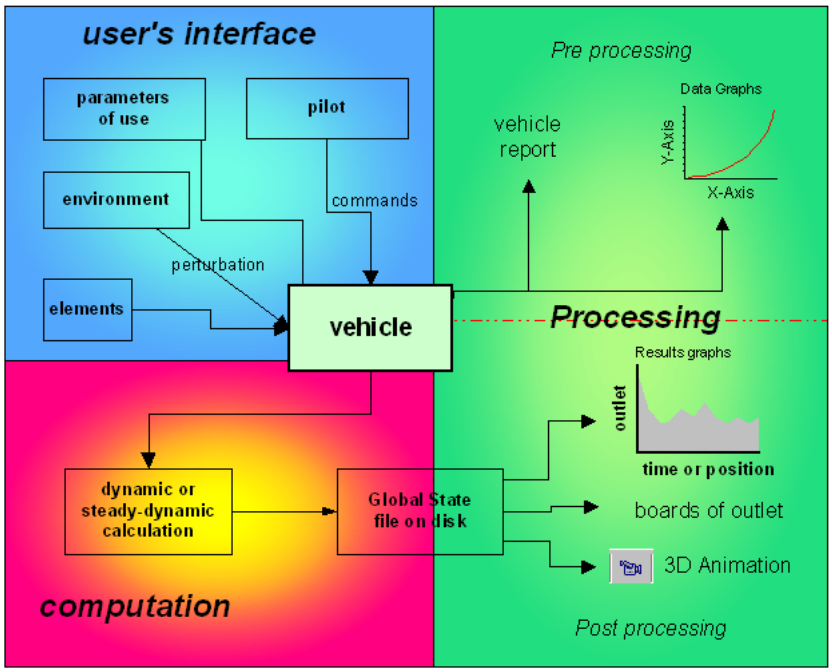


Figure 70: The CALLAS Model description

In [14], a framework is proposed for Vehicle Dynamics Model Validation. The authors started their study and framework proposal from the work presented in [73]. In this previous work, a set of five test cases are defined as primary validation manoeuvres for vehicle dynamic:

- steady-state lateral dynamics (low-frequency cornering)
- Transient lateral dynamics (wide frequency range steering input)
- Longitudinal acceleration (throttle inputs)
- Longitudinal deceleration (braking inputs)
- Road disruption input (suspension kinematics and ride dynamics

This verification and validation process was done to validate a 1997 Jeep Cherokee for the National Advanced Driving Simulator. A sixth group is defined as well, called 'other manoeuvres' - which are imitations of a real-life situations such as double lane change or a fishhook - but are not considered as a primary validation manoeuvres. The tests performed in [14] and [15] include steady-state, transient and frequency domain responses. The following maneuver sequence was followed:

- Quasi-steady-state
- Step response
- Pulse response (evaluated in frequency domain)
- Real-world like manoeuvre: lane change

Regarding data comparison time domain was used for steady-state, low-frequency, and non-linear effects, then frequency domain was used for high-frequency transient maneuvers. Wade-Allen et. al. [74] used both steady-state and transient manoeuvres:

- Quasi-steady-state steering wheel ramp input
- Pulse response (in frequency domain)
- Double lane change
- Fishhook maneuver

In [75], a flowchart of comparing system response quantity (SRQ) is proposed and obtained from both the simulation and real-life experiment. The SQR is defined as : The SRQ can be any type of physically measurable quantity, or it can be a quantity that is based on, or inferred from, measurements. For example, the SRQ can involve derivatives, integrals, or more complex data processing of computed or measured quantities such as the maximum or minimum of functional over a domain. When significant data processing is required to obtain an SRQ, it is important to process both the computational results and the experimentally measured quantities in the same manner.

[75] refers to validation metric, as the "mathematical procedure that operates on the computational and experimental SRQs." The main task to validation is comparing the validation metrics' results with the accuracy requirements (which depends on many factors) for the intended use of the model. In [75] three validation metrics approaches were reviewed:

- Parameter estimation and system identification,
- Hypothesis or significance testing,
- Bayesian analysis, or Bayesian statistical inference.

[75] proposed six properties that validation metrics should include to be useful in the engineering decision-making process. The six attributions are presented here in a manner that only includes the information relevant to our current vehicle simulation domain:

- Validation metrics should either:
  - explicitly include an estimation of numerical error in the SRQ resulting from the computational simulation - such as including the upper and lower estimated bound on the error in the SRQ, but it would add significant complexity to the calculation and interpretation of the metric.
  - Or exclude the numerical error in the SRQ, only if the numerical error was previously estimated to be small compared to the experimental uncertainty.
- The metrics should be a quantitative evaluation of the aggregate accuracy for a specific SRQ, including:
  - the combined modeling assumptions,
  - the physics approximations,
  - the physical parameters of the model.

- A metric should include, an estimate of the error resulting from post-processing of the experimental data that is compared to the simulation result.
- A metric should explicitly incorporate an estimate of the measurement errors in the experimental data for the SRQ that are the basis of comparison with the computational model. There are two types of measurement errors:
  - bias (systematic) errors
  - precision (random) errors

  The minimum requirement for validation metrics is to include an estimation of precision errors. As much as possible, the metrics should include an estimate of bias errors as well.
- A metric should take into account the number of experimental measurements. The authors emphasise the importance of multiple measurements and estimating the accuracy of the experimental result.
- A metric should not include any indications of the level of agreement between computational and experimental results (e.g. characterising them as ”good” or ”excellent”). Validation metrics should be used to assess the degree of agreement between computational models and experimental measurements. The metric should be kept separate from how satisfying the results are.

In [76], common measures are used to quantify the discrepancy between time histories in fields such as statistics, computational mechanics, signal processing, and data mining. The following existing metrics were evaluated:

- Vector norms,
- Average residual and its standard deviation,
- Coefficient of correlation and cross-correlation,
- Sprague and Geers (S&G) metric,
- Russell's error,
- Normalized Integral Square Error (NISE),
- Dynamic Time Warping (DTW).

Their proposal is a structured combination of some of these measurements. The new metrics classifies error components associated with three physically meaningful characteristics :phase, magnitude, and topology, and utilises norms, cross-correlations, and algorithms such as dynamic time warping to quantify discrepancies.

- For phase error, a cross-correlation method is used with a tunable penalty function to have a linear penalty for small time-step (local errors) and larger time-step (global errors) differences.
- Magnitude error is analysed after minimising the global local phase difference between the data sets and also the slope differences - because slope difference is a topological error, not a magnitude error.

- Dynamic Time Warping is used to reduce local phase and slope differences.
- After that L1 vector norm is used to measure the relative magnitude differences.
- The topology error - the measure of discrepancies in the slope - is calculated on the derivative of the time-shifted, warped channels. Then L1 norm is used to quantify the topology error.

In [14], a new more efficient and relevant validation architecture for dynamic vehicle modelling is proposed following the definition of the validity criteria. This framework is presented in Figure 71). In this context, individual parameter estimations for each important subsystem are performed to the most sufficient level, implying that the subsystems with a significant impact on vehicle behaviour in the investigation domain should be thoroughly measured. In addition to the individual parameter measurements, the measurement system constants - such as the position of the acceleration sensor in the chassis coordinate system - must be measured since they are critical for proper post-processing of the vehicle dynamics measurements. In the next block, vehicle dynamics measurements are carried out with a sophisticated measurement system - which is capable of measuring all the important movements, phenomenons such as vehicle body movement, suspension movement, tire forces and moments, side-slip angle, camber, etc. Then based on the measurements - in the previous two blocks - the vehicle model parameter identification is solved by an automated system (denoted by the dashed line on figure 71), as an output, it gives the base parameter set for the simulation environment, also after the post-processing the driver inputs (steering wheel angle, throttle position, brake pedal force/position, gear) and the measured system outputs (vary for each model, but usually vehicle speed, longitudinal and lateral acceleration, yaw velocity, etc.) are present for the comparison of simulation and real-world response quantities. Finally, the recursive process of validation (denoted by doted line on figure 71) is to be fulfilled by appropriate machine learning algorithm, the main steps of the iterative validation process are presented with the blue and red shapes on figure 71. As an output of the parameter estimation, the mean value of the parameters and the uncertainty of the estimation are given. This will define the base parameter set for the simulation. The iterative loop is the following:

- Running the simulation using the driver inputs from vehicle tests.
- Compare the two SRQ set by computing the previously defined validation metrics.
- Compare each metric with the belonging validity criteria.
- If the criteria is not met, then fine-tune the vehicle parameters, in the range given by the uncertainty of the estimation.
- The process ends when the validity criteria is met.

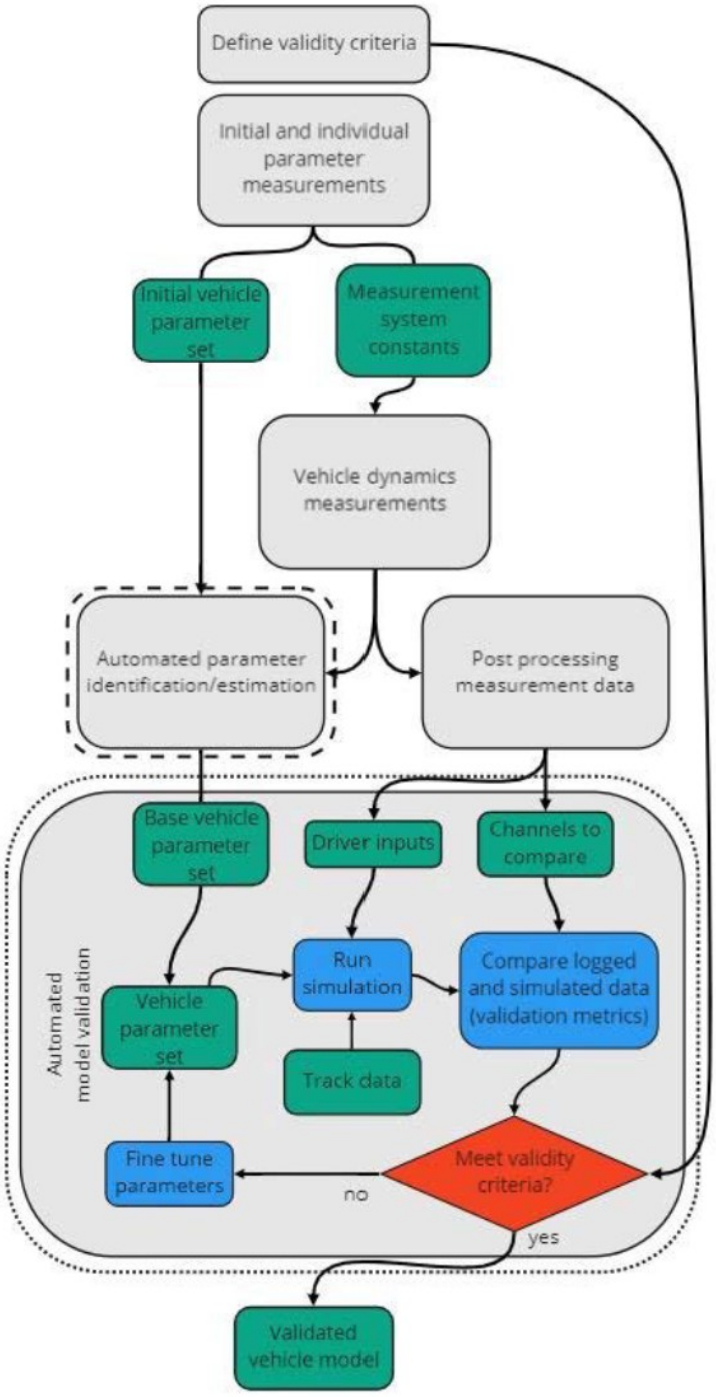


Figure 71: High level layout of recent framework for vehicle dynamics model validation. [14]

In [15], SAE proposed a definition and Application of a Standard Verification and Validation Process (figure 72) for Dynamic Vehicle Simulation Models.

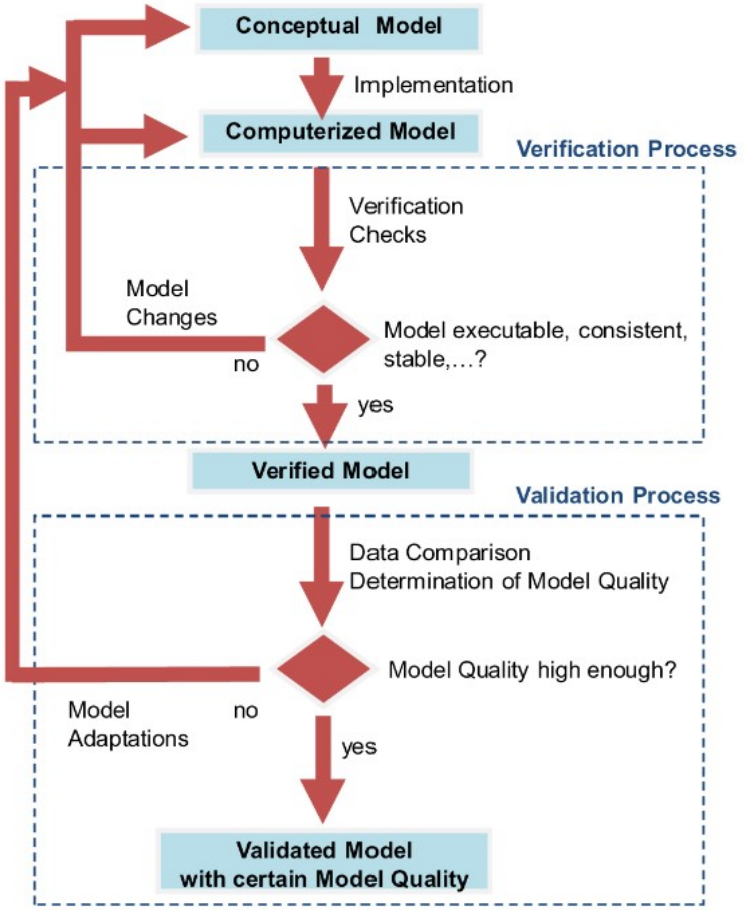


Figure 72: Overview of the verification and validation process proposed by SAE. [15]

In [77], an applied survey is proposed about simulation validation approaches for a vehicle dynamics Model.

In [78], Four standard manoeuvres have been considered, as defined below:

- Coast Down: this manoeuvre is one of the most frequent tests for motor vehicles. It consists of running the vehicle in a straight line, starting at a certain speed and letting it slow down until it stops. The main goal of this test is to evaluate the values of the resistant forces acting on the vehicle at a certain speed and road conditions, to validate the longitudinal dynamics model. Please, note that as the real tests have been carried out with different gears, it has been necessary to model the engine brake accordingly to the real ICE (internal combustion engine).

- Step Steer: the main objective of this test is to asses the lateral dynamic behaviour of a vehicle. Driving in a straight line at constant speed, the steering wheel is rotated as fast as possible to the target angle position, in which the vehicle's lateral acceleration will start to increase as it begins to turn.

- Ramp Steer: the goal of this test is to determine the steady-state circular driving characteristics of the test vehicle, increasing the lateral acceleration. This test is handled at a constant speed and increasing the lateral acceleration. We use a steering ramp input until the limit of adherence is reached.

- If the criteria is not met, then fine-tune the vehicle parameters, in the range given by the uncertainty of the estimation.
- Frequency response: This test aims is to determine the lateral transient response behaviour of the test vehicle in the frequency domain. The test covers a steering input frequency range between 0.1 - 4 Hz.

First, the vehicle is driven at the test speed on a straight line. Then, continuous inputs to the steering wheel at very low frequency are initiated. This input is increased until the maximum frequency. Additionally, the steering-wheel amplitude is maintained as constant as possible throughout the test, and each run is ended with a straight-line driving.

2.4.4 Communication means

In this section, we address the KPI and verification protocol for communication means (V2X). Even if some references are given about LTE and 5G protocols, the main protocol studied 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 propose relevant KPIs, we propose to seperate 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. For each of these sub-parts, we propose a short state of the art and to give a list of the main relevant KPIs. This study shows that the main KPIs applied are the ones identified in the propagation channel 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 73.
- **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 74.
- **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 74).
- **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.

2.4.4.1 Communication protocol, strategies, and OSI layers

In order to provide an aswer 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.

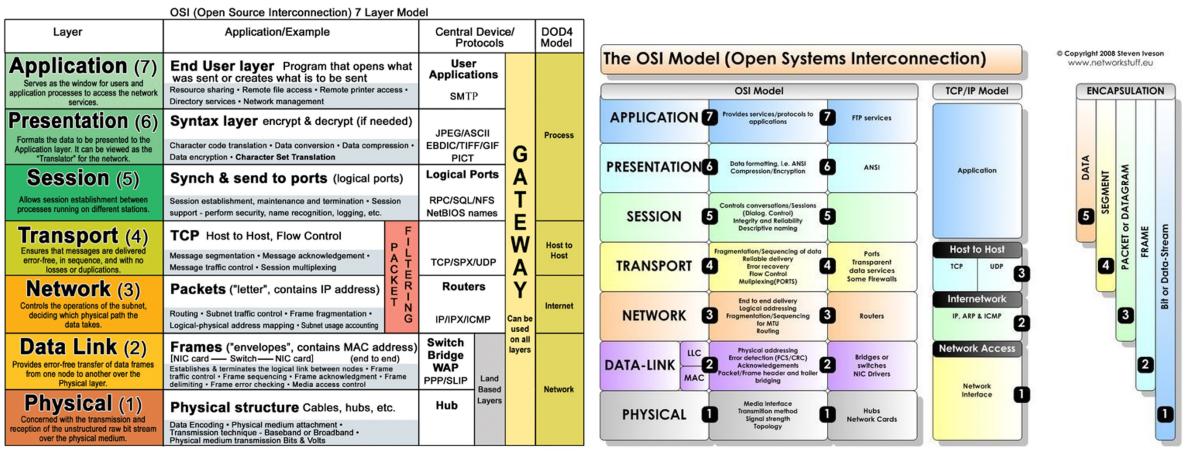


Figure 73: Overview of the OSI model and the fitting with TCP/IP.

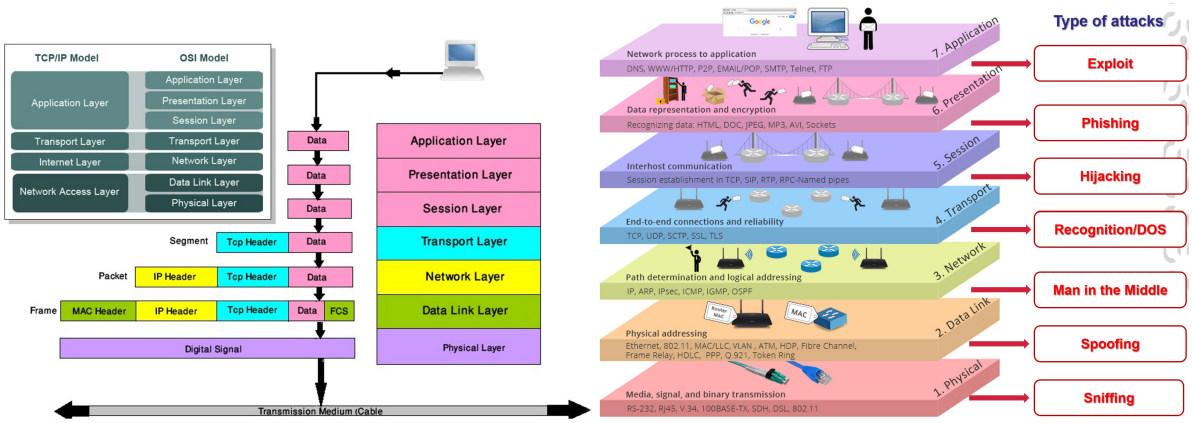


Figure 74: Overview of the different message format and possible cyber attacks by level of the OSI model

2.4.4.2 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.

2.4.4.3 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 75. 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 75: WiFi standard with the main parameters

2.4.4.4 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 [79], 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 [80], 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 [81], 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.

2.4.4.5 The performance metrics to evaluate the communication means

In addition to the different KPI already presented in the previous sub sections related to Communication protocol, strategies, and OSI layers, Emitter and receiver models, antenna diagram modelling, and Propagation channel modelling, we propose additional performance metrics divided into two parts. The fist one is dedicated to network performance and the second part is dedicated to the qualitative performance of the service.

For the first part, the relevant metrics are the following:

- **Network disconnection percentage**: the division of the total duration of network disconnections compared to the overall connection duration, including network availability and unavailability for the user.
- **Redundant messages**: the total number of messages re transmitted due to disconnection, message loss, malfunction, or re authentication.
- **Data age**: the quartiles of the time interval from the collection of information by the source vehicle to its reception by the concerned vehicles, specifying the distance of the receiving vehicles from the event.
- **Network overhead**: overhead refers to the ratio representing the quantity of useful data or application data compared to signaling data, which is used, for example, to announce the presence of vehicles on the network or to establish a specific routing path.

For the second part, we have the following metrics:

- **Service accessibility**: the percentage of requests processed by the server or vehicles, compared to the total number of received requests. This result is influenced by the number of vehicles equipped with a communicating module in the case of a Car2Car service because vehicles process their requests with each other. A processed request in a Car2Car application may involve informing vehicles affected by a certain event in a timely manner. For other services, it may be receiving a response from the server.
- **Server resilience and response time in increased load situations**: this parameter is very important in increased load situations, aiming to evaluate the server's capacity to handle a large load of requests and ensure continuous service under such conditions.
- **Quantity of data**: the total number of messages exchanged during service use.
- **Accuracy of exchanged data**: the percentage of correct data exchanged compared to the total number of messages, including false alarms as may be the case in a safety application, and outdated data not updated.
- **Security of exchanged data**: This parameter is essential to evaluate the integrity, authenticity, and non-repudiation of data exchanged by different peers on the network.
- **Quality of Experience (QoE)**: it is the perceived service quality by the user, which can also reflect the user's satisfaction level with the service.

2.4.4.6 verification and validation of simulation model

UGE Protocol extended for PRISSMA :

In UGE and in its experimental plan allowing to collect data about communication means, omnidirectional stick antenna (Antenna's reference: Doradus 50-1360, 5.0 to 5.9 GHz) has been used with a 8 dBi gain in the horizontal plane. The chipset's transmission power is lowered to 20 dBm. The Effective Isotropically Radiated Power (EIRP) can be used to approximate the actual power output at the antennas. It can be computed from equation:

$$EIRP = P_{tx} + G - L_c \tag{3}$$

where Ptx is the transmission power at the emitter (dBm); G is the antenna gain, relative to a (theoretical) isotropic reference antenna, in dBi; and Lc represents cable and connectors loss (in dB). In our architecture Lc is approximated from manufacturers' data to 2.5 dB, accounting for one pigtail connector, the efficiency of the antenna, and about 2 metres of coaxial cable. Thus, we have EIRP = 20 -2.5 + 8 = 25.5dBm for the power emission. The experimental protocol need to be kept voluntarily simple. We consider vehicle-to-vehicle and no relaying or rebroadcasting. The scenario has a receptor vehicle passing by a static emitter (static vehicle or RSU), which is located on the track's side. It is suggested by previous researches and PICSL's experience, that the absolute speed of 802.11 emitters and receptors in the environment's referential does not have much effects on IVC's (Inter Vehicular Communication) quality. On the other hand, the speed difference between the emitter and the receptor is a major parameter. As we consider a static emitter, the speed difference and speed relative to the environment are equivalent. We advice to use speed intervals. In previous experimentation, the following speeds were tested: 30, 50, 70, 130 and 170 km/h (approximately 20, 30, 45, 80 and 105 mph). Measurements need to be performed both on an actual test track and the simulation environment with the digital twin. For instance the Satory's speed track, the UTAC test track, or the TRANSPOLIS test track. The used speed track need to be isolated from regular traffic. The speed track needs to respect at least 2-kilometres long quasi-straight line, with 2 lanes, allowing for 1.4 kilometres of direct line of sight (LoS). The emitter is either located at the track's eastern end, or near the slight bend, so to have LoS with all the track's length. In order to reproduce motorway situation, the track's surroundings need to be largely open. Data must be collected on a great set of different days spread out on several month in order to be representative of a large set of uncountable conditions. In the proposed protocol "closing" will refer to items relevant to when the receptor vehicle is moving toward the emitter; "away" will refer to items relevant to when the receptor vehicle is moving away from the emitter. The real communication system analysis was made for the range of communication, the antenna quality, the latency, and the frame loss. The average transmission ranges for different speeds and for the driving direction is given by figure 76. We can quote that the communication range is clearly impacted by the relative speed between the transmitter and the receiver. With 30 km/h, the range is up to 700 meters, and with 170 km/h the range decrease to 300 meters. In order to have a better understanding of the energy propagation in the propagation channel and the antenna diagram, some additional experiments have been carried out. Figure 77 shows the SSI result and the antenna diagram. It is possible to observe that the omnidirectional antenne does not respect this datasheet. Also it is possible to observe the decrease of the SSI relatively to the range of communication.

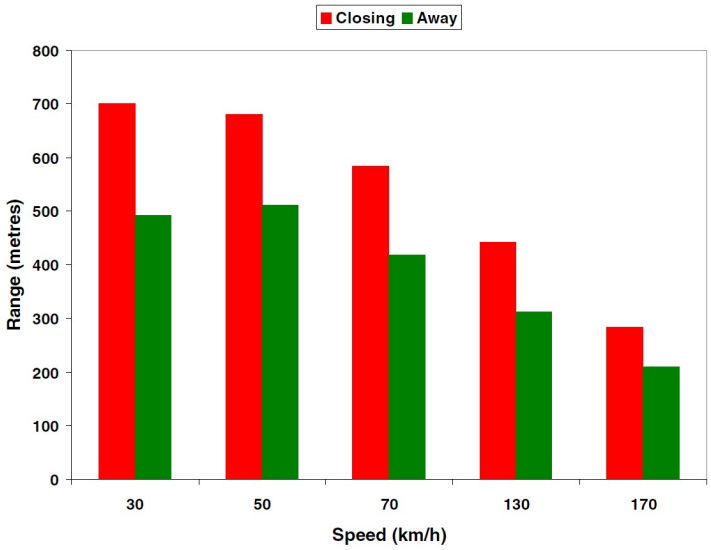


Figure 76: Average transmission ranges for different speeds, according to the direction of driving

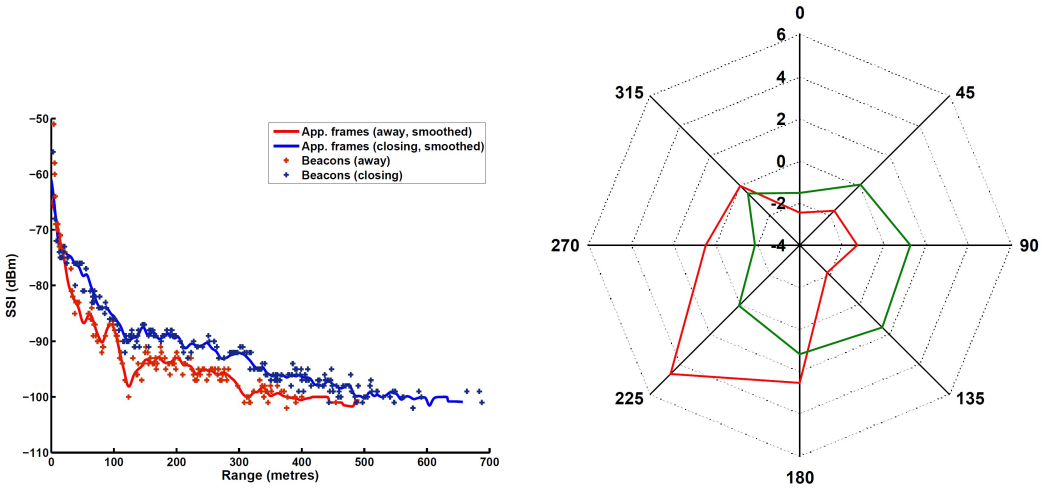


Figure 77: SSI according to direction of driving, and differences with the average SSI (in dBm) for 8 angular sectors (in degrees) of the receiving antenna (in red) and of the emitting antenna (in green)

For the frame loss, it is defined as the percentage of frames that are missed during a certain measurement interval, which we define temporarily and according to range. It is straightforward to deduce the actual bitrate from the nominal bitrate and the measured frame loss. The maximum range is an important indicator, but does not say anything about the quality of transmissions within this range. Typically, one could receive frames up to a thousand metres, yet have 90% frame loss beginning as soon as 400 metres away from the emitter. Thus, it is also important to measure the quality of transmission within the transmission range, typically via frame loss. Here, frame loss is viewed from the logging application point of view. That means that any frame that is not passed on to the IP layer is seen as lost. This include frames that are not received, or only partially received, but also corrupted frames.

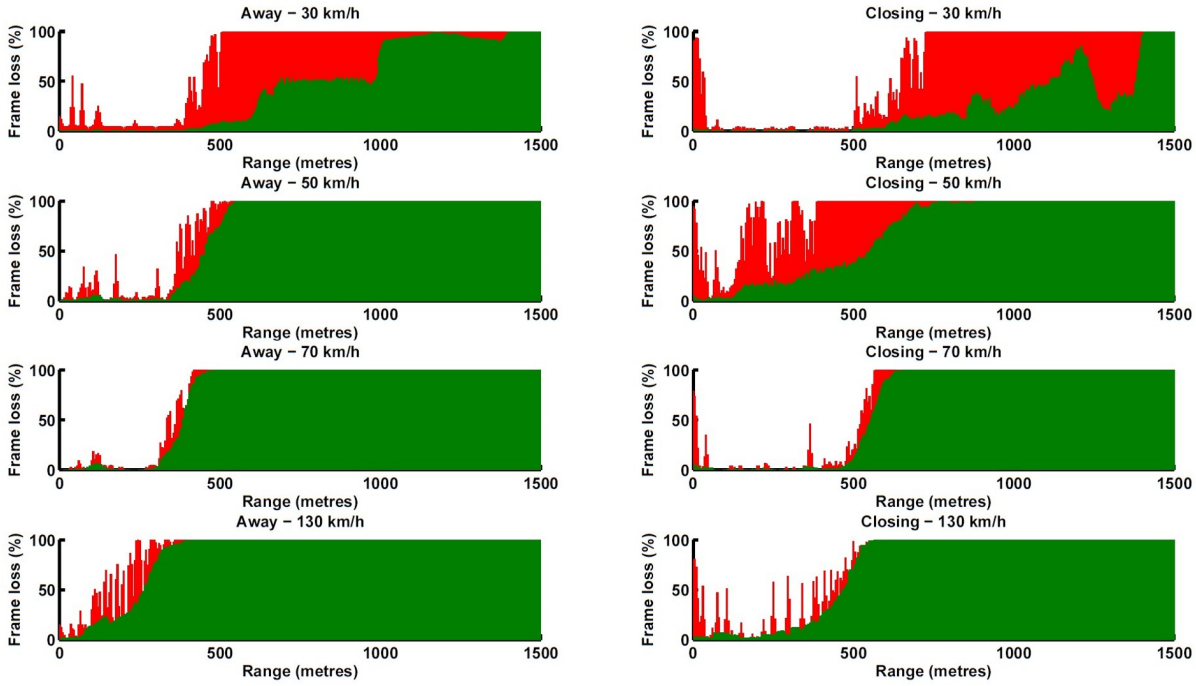


Figure 78: Detailed frame loss measurements for 30, 50, 70, and 130 km/h (5 metres intervals); red is the maximum value and green the average

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 79.

$$\tau = \max\left[A.\exp^{B.(d-C)^2}; \min\left(\max\left[D.d+E; F\right]; 1\right)\right]$$

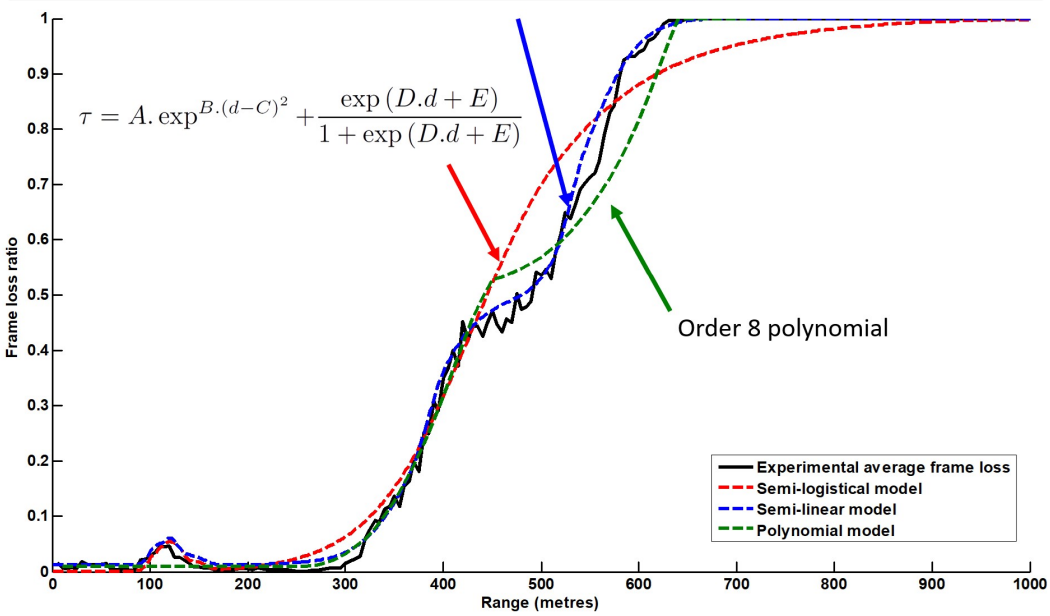


Figure 79: The 3 models of 802.11p communication standard proposed by UGE with the motorway dataset.

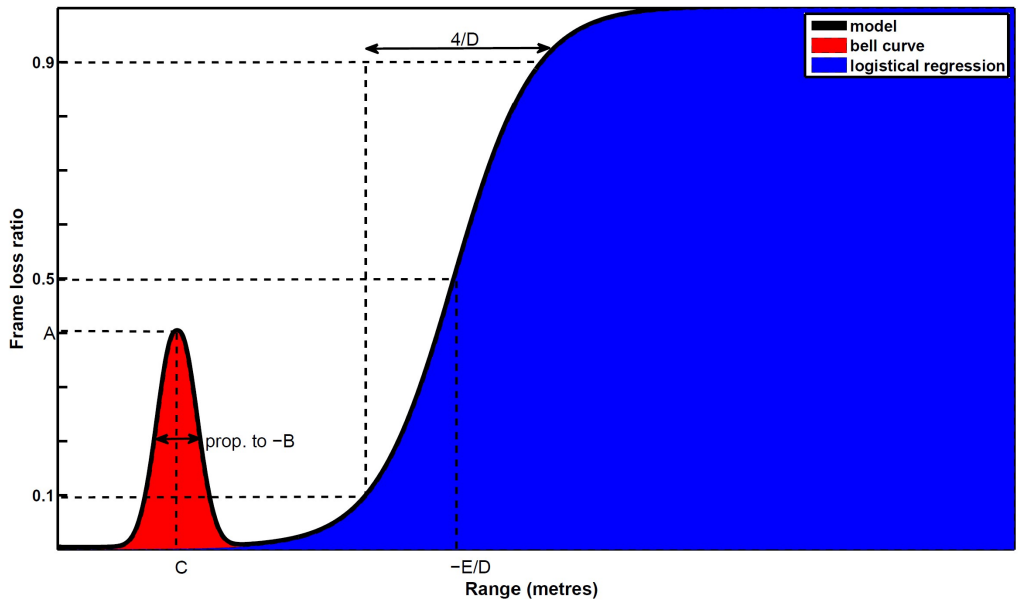


Figure 80: Decomposition of a frame loss profile with its parameters. This profile is share in 2 parts, the red one for the short range reflection interference given a significant loss of signal, the blue part representing the communication profile without interferences

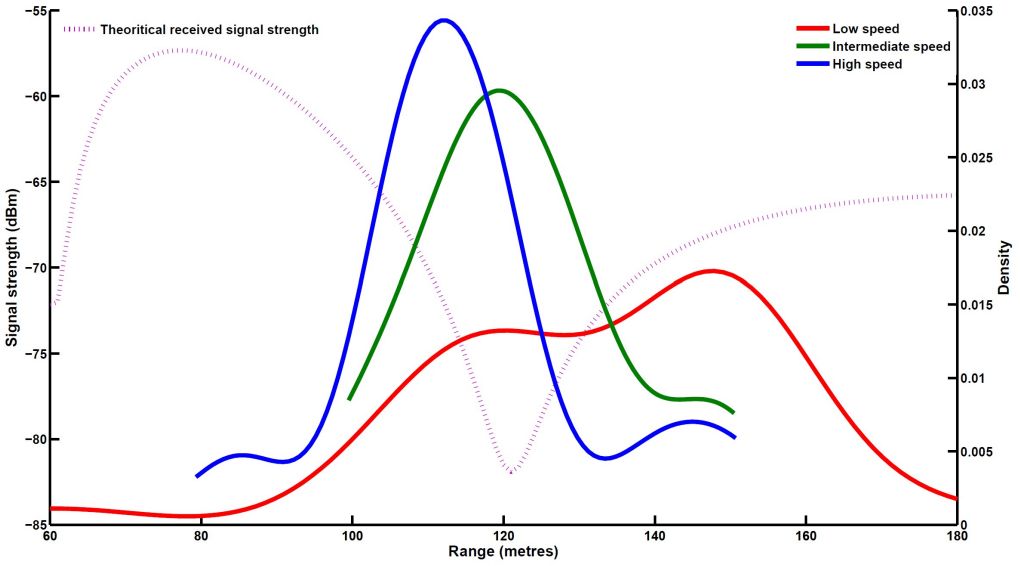


Figure 81: Distribution of parameter C (the central distance of strongest ground reflection interferences) for the three speed classes (right axis), compared to the received signal strength theoretical value (left axis)

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 80. 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 81, 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 82 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.

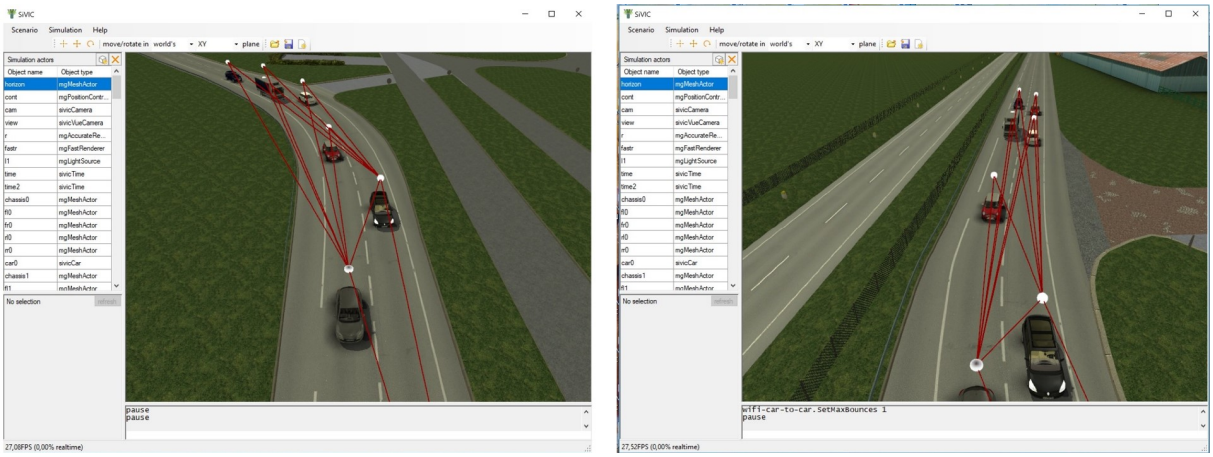


Figure 82: Simulation of communication: Emitter, receiver, antenna, and propagation channel

International Studies on telecommunication means performances

In [82], an experimental protocol is proposed for IEEE 802.11p Performance Evaluation in both Simulations and Real Experiments conditions. In this study, three key metrics are evaluated:

- the maximum range
- packet delivery rate (PDR)
- packet inter-reception time (PIR)

This study compared NS-3 simulation results with real device measurements for evaluating IEEE 802.11p in V2I and V2V communication scenarios. The impact of different modulations and speeds on packet delivery rate (PDR) and packet inter-reception time (PIR) was analysed, along with static experiments assessing the maximum range. Notably, only the 6 Mbps PHY rate modulation provided a satisfactory 700 m range. The study observed disparities between simulation and real-world results, especially in the impact of mobility. The findings underscore the need to enhance NS-3 models for vehicular network simulations. Future work will explore the effects of denser vehicular networks and compare results with the Veins simulator. But this study is mainly focus on the propagation channel effect on the effectiveness of 802.11p communication means.

In [83], The goal of the measurements was to assess communication performance between two moving vehicles using metrics such as Throughput, Jitter, and Packet Loss Rate. The study involved vehicles travelling on a straight path of around 350 meters under different conditions. Initially, vehicles moved in opposite directions at speeds of 15 km/h, 30 km/h, and 40 km/h. Subsequently, they moved in the same direction at a constant speed of 20 km/h. Measurements were taken at intervals of 200 ms, and contact time between vehicles varied inversely with travel speed.

In [84] a study is mage focusing on the IEEE 802.11p standard and the WAVE system in Vehicular Ad Hoc Networks (VANETs), particularly analysing the broadcast process for safety and non-safety applications. While a prior work proposed an analytical model for broadcast packet loss on the control channel, it doesn't encompass all phenomena affecting VANETs. This research conducts an extensive simulation campaign using the NS-3 simulator to study the

average loss rate of broadcast packets in various scenarios, comparing results with an analytical model proposed by [79]. The analysis explores factors like contention window, packet size, vehicle count, and additional influences on the loss rate, highlighting discrepancies with the analytical model, especially in channel switching-related losses. The research provides valuable insights into the loss process in VANETs, shedding light on factors beyond the existing analytical model's scope. This study addresses another interesting KPI related to Loss Rate depending of the number of communication nodes.

In [85], a study conducts an experimental analysis of two MAC protocols, IEEE802.11p and HCMAC (a hybrid MAC protocol). HCMAC is implemented on the IEEE802.11p physical layer using DSRC-compliant devices. The performance of beaconing messages in various highway scenarios is evaluated in terms of RSSI, PDR, and PIR. Results indicate that vehicle mobility minimally affects connection performance, but packet collisions significantly degrade beaconing. In multiple test scenarios, HCMAC outperforms IEEE802.11p by up to 88% in PDR and 47% in PIR. Cooperative transmission scheduling, exemplified by HCMAC, emerges as an efficient solution to mitigate collisions and enhance overall performance compared to IEEE802.11p.

In [86], a pilot platform based on IEEE-802.11p, LTE, and 5G test network (5GTN) is designed to leverage road traffic and weather information for enhanced road safety. Real-time and accurate data on road conditions are essential, and a combination of networks is employed for delivering updated information. The work compares vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) scenarios using IEEE 802.11p as the primary vehicular communication channel, supported by LTE and 5G networks. The analysis focuses on network performance, including latency, packet loss, and throughput in both V2V and V2I scenarios.

In [87], a comparison of IEEE 802.11p and LTE-V2X is made in order to provide an evaluation of the communication performances With Periodic and Aperiodic Messages of Constant and Variable Size. These aspects are important and need to be took into consideration in communication simulation and in the propagation channel model. The figure 83 presents their study about the range of communication and the percentage of message collision in dense traffic conditions.

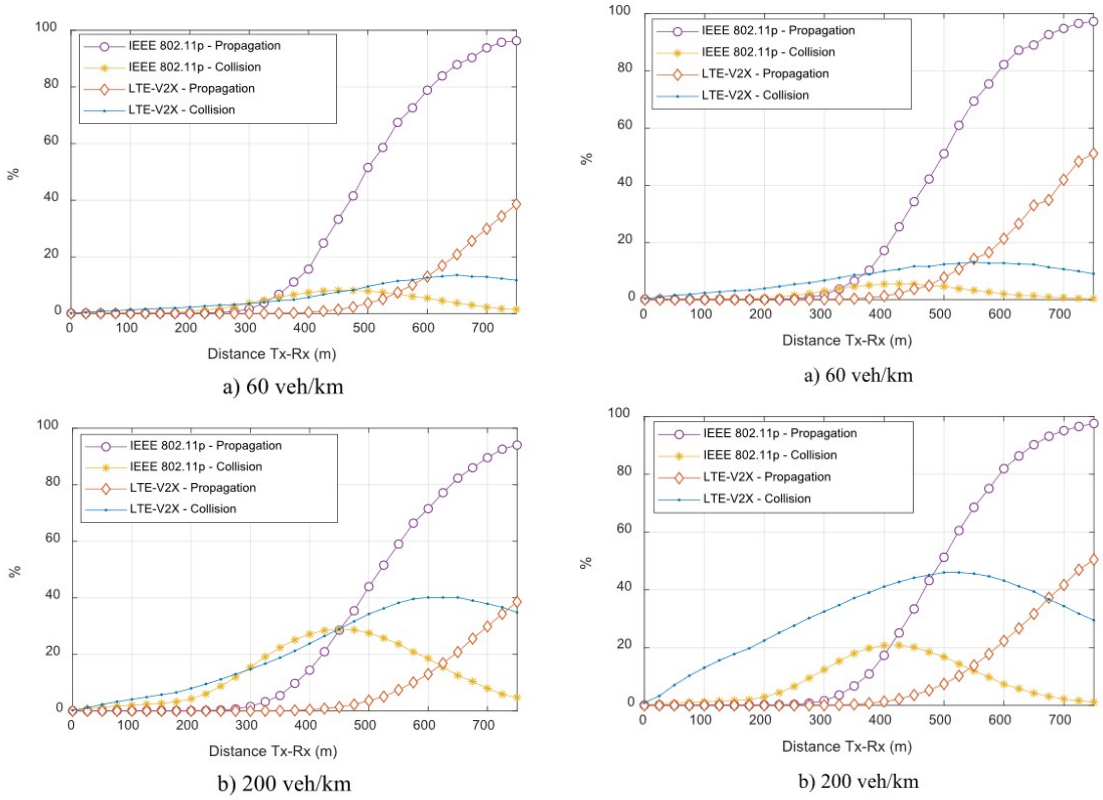


Figure 83: Percentage of packets lost due to propagation and collision errors when messages are (left figure) periodic and of constant size (simplified model), and (right figure) aperiodic and of variable size (empirical CAM model).

2.4.5 Traffic generation (AVS / UGE)

Modeling 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.

2.4.5.1 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 [88], while others aim at describing how a predetermined and nominal trajectory might be affected by human errors or approximations when driving a vehicle [89].

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. [90] 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. [91] distinguish the following car-following approaches:

  - Analytical models, covering the following approaches:
    * Gazis-Herman-Rothery (GHR) approach [92]: 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 MITSIM (MIcroscopic Traffic SIMulator) framework.
    * Safe distance [93] and psycho-physical [94] 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 [93] (by default in AIMSUN), Krauss models [95] (by default in SUMO)... Some were tested and compared by Brockfeld et al. [96]. 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 [94].
    * Intelligent Driver Model [97]: it is a time-continuous car-following model developed as an extension of Gipps' model [93] by Treiber [97] to cope with the losses of accuracies observed in the deterministic limits of Gipps' model.
    * Optimal Velocity approach: based on Bando et al. [98] 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 [99];

  - Cellular Automata approach [100]: 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 [101], 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. [89] 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.

2.4.5.2 Verification and validation methods and metrics

Most validation approaches for traffic simulation rely on comparing real data with simulated data. They are usually developed according to the model type under consideration (please refer to section 2.4.5.1. Several works have already been done and some examples are given thereafter:

- **Long drive simulation:** A first method relies on letting behavioural traffic models being simulated on long drive scenarios to combine different atomic situations in order to fuse them into a general traffic that can be analysed and supervised. A study of this supervised simulation as a traffic validation model has been done for example in the 3SA Project. [102]

- **Reference traffic model:** Validating the traffic generation can be achieved using a reference traffic model, like SUMO (Simulation of Urban MObility) that is conceived to handle large traffic networks.

- **Real validation data:** More and more real traffic data is used as traffic model validation and can also be used to train Artificial Intelligence Traffic model. That is the case, for example, with roads recorded with drones for several days to have a description of the different models

Beyond the approach adopted to validate the models, the usual validation processes systematically aim to reproduce indicators observed in the real world. The validation processes are usually formulated as an optimisation process with objective functions describing the gap between observations and simulation for a set of specific indicators. Consequently, the main features discriminating the approaches are the following:

- **the indicators** under consideration when comparing observation and simulation: this feature is the most critical in the context of validating traffic simulation models since it strongly depends on the purpose and the use of the traffic flow. Usually, the traffic simulation aims at reproducing accurately the traffic flow to dynamically mimic congestion propagation through the road network, while approximations might be tolerated regarding the microscopic behaviour of cars with each other, like approximations in the lane-change manoeuvres with instantaneous lane changes, etc. Consequently, the usual calibration processes applied to validate traffic simulators might not be relevant when the target is to measure safety indicators. For example, in the specific case of traffic generated through physic rule-based approaches, most metrics rely on aggregated and macroscopic traffic indicators, like traffic volume, traffic density, or Average Speed, which can be measured in real-world with classical loop sensors [96]. Such indicators enable us to validate that the traffic flow and congestion are consistent with real-world observations, but do not ensure an adequate reproduction of the safety indicators of paramount importance when validating automated vehicles. Only a few and recent studies explore the calibration of microscopic traffic simulators with the perspective of automated vehicle testing. Wang et al [16] develop an exhaustive literature review about the existing methods of car-following models in the literature (84). They adopt the perspective of the driver by drawing the analysis opposing Human versus Automated drivers with the driving behaviour as background. They further discuss the model calibration methods.

- **the formulation of the metrics** measuring the gap between observation and simulation: for this purpose, usual indicators are under consideration, like the classic Root-Mean-Square-Error (RMSE) or the Mean Absolute Error (MAE) and its percentage version: the Mean Absolute Percentage of Error (MAPE). However, Brockfeld et al. [96] use a specific error formulation dedicated to measuring errors between vehicle pairs, called Theil's U-value [103]. [17] found that 7 GOFs (goodness-of-fit function)(RMSE, RMSPE, Theil's U, MAE, MAPE, $\widetilde{RMSPE}$, NRMSE) and 4 MoPs (measure of performance)(i.e., s, v, a, σv) were used in the literature on CF modelling and their quality and the performance evaluation and validation. In total, 21 combinations of GoFs and MoPs, i.e., objective functions, were adopted. The description of these GOFs and MoPs is given in [17] and given in the figure 85.

- **the resolution procedure of the optimisation problem:** various approaches are observed, from direct optimisation, like the gradient-free optimisation method known as the "downhill simplex method" [96] to the use of heuristics, like Genetic Algorithms [104].

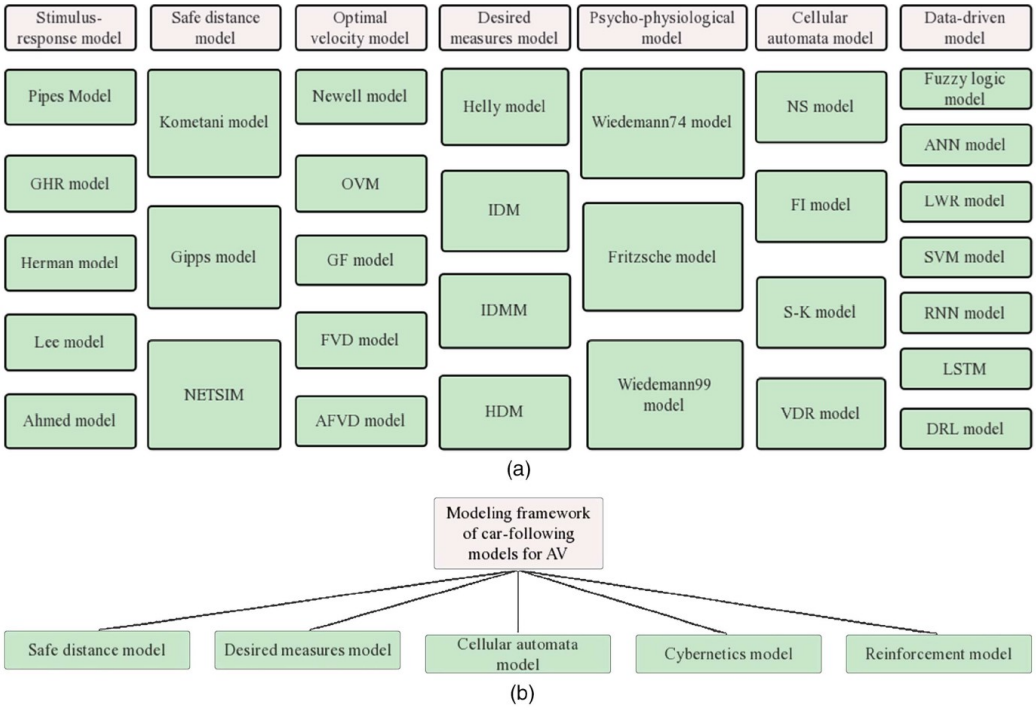


Figure 84: Summary of car-following models for HDVs and AVs: (a) development process of car-following models for HDVs; and (b) modelling framework of car-following (CF) models for AVs ([16])

GoF(MoP)

$$RMSE(s) = \sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[s^{sim}(t)-s^{obs}(t)\right]^2}$$
$$RMSE(v) = \sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[v^{sim}(t)-v^{obs}(t)\right]^2}$$
$$RMSE(a) = \sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[a^{sim}(t)-a^{obs}(t)\right]^2}$$
$$RMSE(\sigma_v) = \sqrt{\left(\sigma_v^{sim}-\sigma_v^{obs}\right)^2}$$
$$RMSPE(s) = \sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[\frac{s^{sim}(t)-s^{obs}(t)}{s^{obs}(t)}\right]^2}$$
$$RMSPE(v) = \sqrt{\frac{1}{|D|}\cdot\sum_{t\in D}\left[\frac{v^{sim}(t)-v^{obs}(t)}{v^{obs}(t)}\right]^2}, D=\left\{t\in[1,T]: v^{obs}(t)\geq v_{th}\right\}$$
$$RMSPE(a) = \sqrt{\frac{1}{|D|}\cdot\sum_{t\in D}\left[\frac{a^{sim}(t)-a^{obs}(t)}{a^{obs}(t)}\right]^2}, D=\left\{t\in[1,T]: \left|a^{obs}(t)\right|\geq a_{th}\right\}$$
$$RMSPE(\sigma_v) = \sqrt{\left[\left(\sigma_v^{sim}-\sigma_v^{obs}\right)/\sigma_v^{obs}\right]^2}$$
$$RMSPE(s,v) = \alpha\cdot RMSPE(s)+\beta\cdot RMSPE(v)$$
$$RMSPE(s,v,a) = \alpha\cdot RMSPE(s)+\beta\cdot RMSPE(v)+\gamma\cdot RMSPE(a)$$
$$RMSPE(s,\sigma_v) = \alpha\cdot RMSPE(s)+\beta\cdot RMSPE(\sigma_v)$$
$$U(s) = \frac{RMSE(s)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[s^{sim}(t)\right]^2}+\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[s^{obs}(t)\right]^2}}$$
$$U(v) = \frac{RMSE(v)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[v^{sim}(t)\right]^2}+\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[v^{obs}(t)\right]^2}}$$
$$U(a) = \frac{RMSE(a)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[a^{sim}(t)\right]^2}+\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[a^{obs}(t)\right]^2}}$$
$$U(s,v) = \alpha\cdot U(s)+\beta\cdot U(v)$$
$$U(s,v,a) = \alpha\cdot U(s)+\beta\cdot U(v)+\gamma\cdot U(a)$$

$$MAE(s) = \frac{1}{T}\cdot\sum_{t=1}^{T}\left|s^{sim}(t)-s^{obs}(t)\right|$$
$$MAE(v) = \frac{1}{T}\cdot\sum_{t=1}^{T}\left|v^{sim}(t)-v^{obs}(t)\right|$$
$$MAE(a) = \frac{1}{T}\cdot\sum_{t=1}^{T}\left|a^{sim}(t)-a^{obs}(t)\right|$$
$$MAPE(s) = \frac{1}{T}\cdot\sum_{t=1}^{T}\frac{\left|s^{sim}(t)-s^{obs}(t)\right|}{s^{obs}(t)}$$
$$MAPE(v) = \frac{1}{T}\cdot\sum_{t=1}^{T}\frac{\left|v^{sim}(t)-v^{obs}(t)\right|}{v^{obs}(t)}, D=\left\{t\in[1,T]: v^{obs}(t)\geq v^{th}\right\}$$
$$MAPE(a) = \frac{1}{T}\cdot\sum_{t=1}^{T}\frac{\left|a^{sim}(t)-a^{obs}(t)\right|}{\left|a^{obs}(t)\right|}, D=\left\{t\in[1,T]: \left|a^{obs}(t)\right|\geq a^{th}\right\}$$
$$MAPE(s,v) = \alpha\cdot MAPE(s)+\beta\cdot MAPE(v)$$
$$MAPE(s,v,a) = \alpha\cdot MAPE(s)+\beta\cdot MAPE(v)+\gamma\cdot MAPE(a)$$
$$\widetilde{RMSPE}(s) = \sqrt{\frac{1}{\sum_{t=1}^{T}s^{obs}(t)}\cdot\sum_{t=1}^{T}\frac{\left[s^{sim}(t)-s^{obs}(t)\right]^2}{s^{obs}(t)}}$$
$$\widetilde{RMSPE}(v) = \sqrt{\frac{T}{|D|\cdot\sum_{t=1}^{T}v^{obs}(t)}\cdot\sum_{t\in D}\left[\frac{v^{sim}(t)-v^{obs}(t)}{v^{obs}(t)}\right]^2}, D=\left\{t\in[1,T]: v^{obs}(t)\geq v_{th}\right\}$$
$$\widetilde{RMSPE}(s,v) = \alpha\cdot\widetilde{RMSPE}(s)+\beta\cdot\widetilde{RMSPE}(v)$$
$$NRMSE(s) = \frac{RMSE(s)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[s^{obs}(t)\right]^2}}$$
$$NRMSE(v) = \frac{RMSE(v)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[v^{obs}(t)\right]^2}}$$
$$NRMSE(a) = \frac{RMSE(a)}{\sqrt{\frac{1}{T}\cdot\sum_{t=1}^{T}\left[a^{obs}(t)\right]^2}}$$
$$NRMSE(s,v) = \alpha\cdot NRMSE(s)+\beta\cdot NRMSE(v)$$
$$NRMSE(s,v,a) = \alpha\cdot NRMSE(s)+\beta\cdot NRMSE(v)+\gamma\cdot NRMSE(a)$$

Figure 85: Formulas of possible combinations of the 7 GoFs and the 4 MoPs ([17])

Whatever the adopted approach, the analogy with fluids mechanics is made a several time, and lots of studies exist to present validation method [105].

In [18], a survey is proposed about Traffic Modelling Based on Data-Driven Method for Simulation Test of Autonomous Driving. In this recent work, Evaluation Metrics for Traffic Simulation. Moreover a framework is given and a survey is done about the evaluation of traffic simulation with 3 criteria:

- **Simulation realism**: Simulation realism metrics gauge the extent to which the simulation output aligns with input data, measuring the fidelity of the simulated environment to real-world scenarios. An ideal simulation system should adeptly replicate the behaviours of other agents as documented in the input data. Often referred to as controllability or reconstruction ability indicators, these metrics assess the system's capability to reconstruct genuine traffic scenarios accurately. Agents demonstrating controllability or reconstruction ability should generate trajectories or behaviours that closely mirror the distribution observed in real-world data.
- **Simulation reactivity**: Simulation reactivity metrics, on the other hand, focus on evaluating the ability of target vehicles to respond safely and effectively within the dynamic and intricate transportation environment. It is crucial to emphasise that reactivity indicators extend beyond hazardous situations, encompassing safety assessments throughout the entire driving process. These metrics provide insights into how well vehicles navigate diverse and challenging traffic conditions, contributing to a comprehensive evaluation of their performance.
- **Simulation diversity**: Simulation diversity metrics play a key role in evaluating the discriminability and coverage of agent policies within the simulation. Higher diversity is desirable, but its significance lies in the assurance that diverse samples maintain a comparable level of realism. This ensures that while the simulation encompasses a wide array of scenarios and behaviours, each scenario remains realistic and contributes meaningfully to the overall simulation landscape.

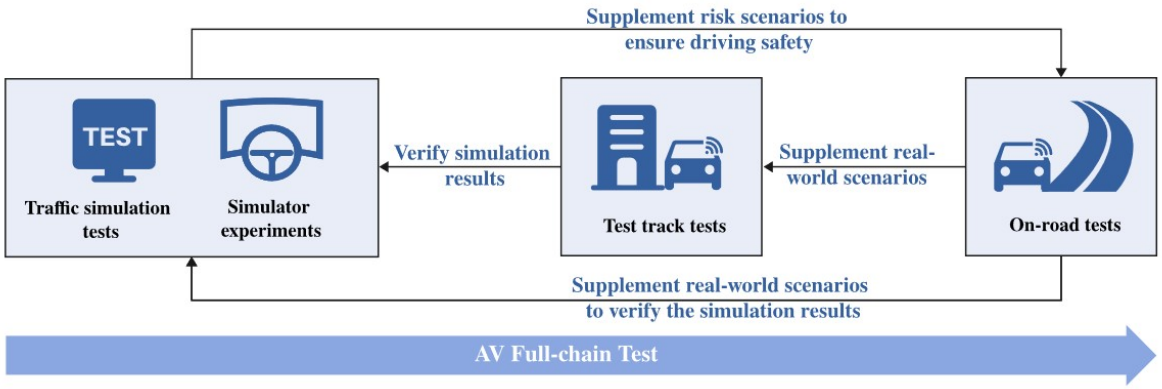


Figure 86: The full-chain test of AV including traffic simulation tests, simulator experiments, test track tests and on-road tests

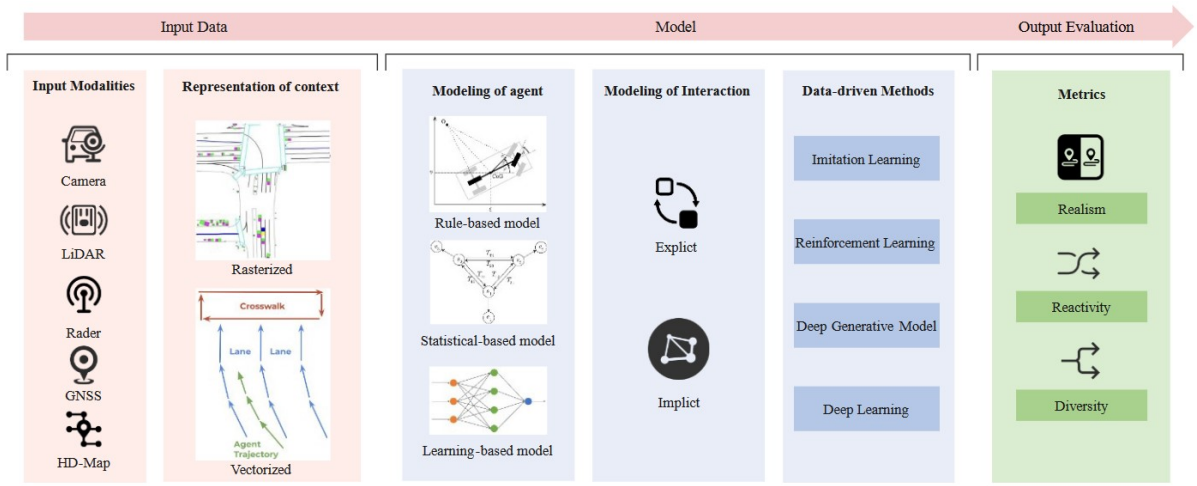


Figure 87: The framework for traffic simulation proposed by [18]

Simulation evaluation is commonly categorised into two main types: open-loop and closed-loop evaluation. In the open-loop evaluation, the simulation system is configured to emulate a human driver. During this process, the agent's predictions do not influence its own actions, meaning the network does not receive its own predictions as input. This setup eliminates agent interactions from consideration. Conversely, closed-loop evaluation involves the simulator generating a planned trajectory based on available information at each time-step, similar to open-loop evaluation. However, in closed-loop evaluation, the proposed trajectory serves as a reference for a controller. Consequently, the planning system undergoes gradual adjustments at each time-step, incorporating the updated state of the vehicle. Closed-loop evaluations are pivotal for accurately assessing the real performance of driving models, as open-loop evaluation alone can be misleading. It ensures a more dynamic and responsive evaluation by considering the interactive nature of driving scenarios. Evaluation metrics play a vital role in traffic simulation, offering a quantitative measure of the realism, reactivity, and diversity of the simulated traffic system. These metrics are specifically crafted to assess various aspects of the simulation and can be employed to compare simulated results with real-world data or other simulation models. The comprehensive evaluation metrics applied in traffic simulations are detailed in figure 88 and figure 89, providing a structured framework for gauging the effectiveness and fidelity of simulated traffic scenarios.

Perspectives	Metrics	Formula	Description
Realism	Final Displacement Error, FDE	$FDE = \lvert \hat{Y}_{end} Y_{end} \rvert$	The discrepancy between the predicted final location $\hat{Y}_{end}$ and the actual final location Y_{end} at the conclusion of the prediction horizon.
	Average Distance Erroe, ADE	$ADE = \frac{1}{T}\sum_{t=1}^{T} \lVert \hat{Y}_t - Y_t \rVert_2$	the distance between the predicted location $\hat{Y}$ and the actual location $Y_t$throughout the prediction horizon, which is defined as T
	minADE		The smallest ADE value among K predictions
	minFDE		The smallest FDE value among K predictions
	meanADE		The average ADE value among K predictions
	meanFDE		The average FDE value among K predictions
	Along-track Error, ATE		The lateral distance error between the vehicle's current position and the reference trajectory
	Cross-track Error, CTE		The perpendicular distance error between the vehicle's current position and the reference trajectory.
	Miss Rate, MR		The percentage of missed cases over all cases
	Maximum Mean Discrepancy, MMD	$MMD^2(p,q) = \mathbb{E}_{x,x' \sim p}[k(x,x')] + \mathbb{E}_{y,y' \sim q}[k(y,y')] - 2\mathbb{E}_{x \sim p, y \sim p}[k(x,y)]$	A measure of the distance between two distributions p and q
	Jensen-Shannon Divergence, JSD	$JS(P\|Q) = \frac{1}{2}[KL(P\|M) + KL(Q\|M)]$	A measure of the similarity between two distributions P and Q. M is the average of P and Q.
	Wasserstein distance, WD	$W(P,Q) = \inf_{\gamma \sim \Pi(P,Q)} E_{(x,y)\sim\gamma}[\lVert x - y \rVert]$	A measure of the distance between two distributions P and Q.
	Negative log-likelihood, NLL	$H(q,p) = E_{x \sim p} - \log(p(x))$	The trajectory distribution of the model is represented by p, whereas q denotes the distribution of the ground truth data.
	mean average precision, mAP	$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$	AP is the average precision for i
	Root Mean Square Error, RMSE	$RMSE = \frac{1}{T}\sum_{t=1}^{T} \sqrt{\frac{1}{M}\sum_{m=1}^{M} \left(s_t^{(m)} - \hat{s}_t^{(m)}\right)^2}$	It measures the difference between the recovered data and the actual data. $s_t^{(m)}$ and $\hat{s}_t^{(m)}$ are the true value and recovered value of the position or speed of vehicle m at time t respectively.
	Coverage	$Coverage = \frac{\sum_{f=-N}^{N} v_{model}(f)}{\sum_{f=-N}^{N} max(v_{model}(f), v_{human}(f))}$	$v(f)$ is the amplitude at frequency
	Intersection over union, IoU		The intersection over the union area of model and human data
	Fréchet Inception Distance, FID		The Fréchet inception distance between the synthesized images and the ground-truth images
	Fréchet Video Distance, FVD		The Fréchet video distance between the synthesized and the truth videos.

Figure 88: Evaluation metrics applied to traffic simulation and generation for realism aspect. Study made by [18] from a literature survey

Perspectives	Metrics	Formula	Description
Realism	Route Consistency, RC		It measures the path-to-path distance between an agent's desired route and the route reconstructed from its executed trajectory, averaged over agents and scenarios.
Reactivity	Collision Velocity, CV		The relative speed at the time of collision
	Collision rate, CR	$CR = \frac{Num(\ collision\ vehicles/scenarios/trajectories)}{Num(\ all\ vehicles/scenarios/trajectories)}$	The average percentage of collision vehicles in each scenario or collision scenarios or trajectories.
	Success Rate, SR		The ratio of episodes where the agent arrives at the destination
	Failure Rate		The average fraction of agents experiencing a critical failure
	Off-Road, OR		The percentage of time that an interactive agent spends off the road
	Average Collision Time, ACT		The average time between the detection of an obstacle and the occurrence of a collision.
	Average Collision Frequency Per Second, CPS		The average number of collisions that occur per second of driving time.
	Average Collision Frequency Per Meter, CPM		The average number of collisions that occur per meter of driving distance.
	Average Collision Distance, ACD		The average distance between the vehicle and the obstacle at the time of collision.
	Number of lane changes		The average of the number of lane changes made by the self-vehicle in each time period.
	Traffic Rule Violation Rate, TRV	$TRV = \frac{Num(traffic\ rule\ violations)}{Num(alltraffic\ rule\ observation)}$	It measures the percentage of time that a vehicle violates traffic rules or regulations during a given period.
	Progress		The total traveled distance of all simulated agents divided by the number of simulated agents.
Diversity	Map-aware Average Self-distance , MASD	$MASD = \max_{k,k' \in 1,\ldots,l} \frac{1}{NT_{\text{pred}}} \sum_{n=1}^{N} \sum_{t=1}^{T_{\text{pred}}} \left\| z_t^{n,k} - z_t^{n,k'} \right\|^2$	The average distance between the two most distinct samples that do not violate traffic rules
	Mean Wasserstein distance, meanWD	$meanWD = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{Wass}\left(\rho_i, \rho_j\right)$	The Wasserstein distance between the density profiles ρ
	Inter-policy Diversity, IPD	$D_{IP}(\Pi) = \frac{1}{\lvert\Pi\rvert(\lvert\Pi\rvert - 1)} \sum_{\pi \in \Pi} \sum_{\pi' \in \Pi} D_{IP}(\pi, \pi')$ where $D_{IP}(\pi, \pi') = \frac{1}{\lvert S_\pi \cap S_{\pi'} \rvert} \sum_{s \in S_\pi \cap S_{\pi'}} d\left(\tau_s(\pi), \tau_s(\pi')\right)$	The average distance of two policy pairs
	Overall Diversity, OAD	$D_{OA}^{\mathcal{T}}(\Pi) = \frac{1}{\lvert S \rvert} \sum_{s \in S} D_{OA}^{\mathcal{T},s}(\Pi_s)$ where $D_{OA}^{\mathcal{T},s}(\Pi) = \inf_{\gamma \in \Gamma(\tau_s(\Pi), \mathcal{T})} \mathbb{E}_{(\tau,\tau') \sim \gamma}[d(\tau, \tau')]$	The distance between the obtained trajectory and the expected trajectories

Figure 89: Evaluation metrics applied to traffic simulation and generation for reactivity and diversity aspects. Study made by [18] from a literature survey

2.4.6 Degraded and adverse conditions and disturbances

2.4.6.1 Definition of the types of models and components involved

Autonomous vehicles are expected to operate outdoors and in uncontrolled environments. This means that many uncontrollable environmental factors can impact the performance of an automated system. In addition to the parameters related to other objects on the road and road users (other vehicles, pedestrians, animals, etc.), which are dealt with in the figure 90, we will list the factors impacting performance (particularly perception):

- 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
- Loss of connectivity (urban canyon, tunnel...)

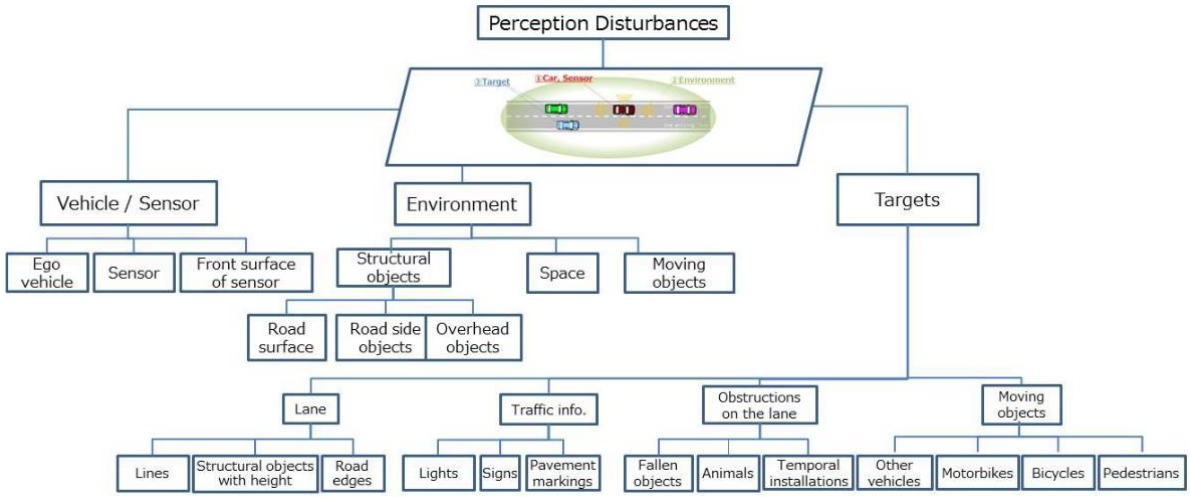


Figure 90: Type of disturbances.

In the following, we will focus on environmental disturbances other than those related to road users and connectivity (dealt with in dedicated sections).

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.

The World Meteorological Organization [106] 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.

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.

The World Meteorological Organization 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 most impact of fog is the contrast loss leading to more difficulties to detect or recognize objects. The rain affects perception of the road environment in different ways. Like fog, heavy rain can reduce transmission through the atmosphere, by scattering light. But it also has other effects: the impact of water on the windscreen, water sprayed up from the road by vehicles, light reflected from wet objects, etc. In the METAR terminology (METeorological Aerodrome Report) which is recognized by the World Meteorological Organization, 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 characterizing the impact of weather conditions of optical sensors. The 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 [106, 107]. According to this definition, the visibility MOR is related to the extinction coefficient β as follows:

$$MOR = \frac{-ln(0.05)}{\beta} \approx \frac{3}{\beta}, \tag{4}$$

It is important to mention that in Equation (4) β is considered in the visible band and is assumed constant by the WMO [106]. This is not relevant for infrared wavelengths [108].

In order to model perception sensors under harsh weather conditions, it is necessary to be able to model the path of a light ray interacting with the particles in the air. 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. Assuming unpolarized light, by using the Mie scattering theory [109], these wavelength-dependent parameters can be computed thanks to the particle size distribution (PSD) of the scatters. 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 $W/m^2/\mu/sr$. The RTE can be expressed as [109]:

$$\begin{aligned}&\frac{1}{c}\frac{\partial L_\lambda}{\partial t}(t, r, u) + u \cdot \nabla_r L_\lambda(t, r, u) = \\ &\quad - \sigma_\lambda(r) L_\lambda(t, r, u) - \kappa_\lambda(r) L_\lambda(t, r, u) + \frac{\sigma_\lambda(r)}{4\pi} \int_{\mathbb{S}^2} L_\lambda(t, r, v) \Phi_\lambda(r, v, u) dv + q(t, r, u),\end{aligned} \tag{5}$$

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 a space continuous source. 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 (5). For each light source occupying the space region S and emitting light from its surface ∂S, we have:

$$\forall\, r \in \partial S,\ \forall\, u \in \mathbb{S}^2,\ u \cdot n_r^S > 0,\ L_\lambda(t, r, u) = E_\lambda^S(t, r, u), \tag{6}$$

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:

$$\forall\, r \in \partial O,\ \forall\, u \in \mathbb{S}^2,\ u \cdot n_r^O > 0,\ L_\lambda(t, r, u) = \int_{v \in \mathbb{S}^2,\, v \cdot n_r^O < 0} L_\lambda(t, r, v) B_\lambda^O(r, v, u) dv, \tag{7}$$

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 (5) can be considered:

$$u \cdot \nabla_r L_\lambda(r, u) = -\beta_\lambda L_\lambda(r, u) + \frac{\sigma_\lambda}{4\pi} \int_{\mathbb{S}^2} L_\lambda(r, v) \Phi_\lambda(v \cdot u) dv + q(r, u), \tag{8}$$

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 (6) and (7) in which the time t is removed. 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). From an energy point of view, coefficients σ_λ and κ_λ define the amount of energy scattered and absorbed by the particles. These optical characteristics of the medium (σ_λ, κ_λ and Φ_λ) are determined according to Mie theory and PSD measurements, under the assumption of spherical scatters (this assumption can be avoided by using the T-matrix method).

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 91. 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 [109, 110]. On the other hand, for sphere radii beyond 0.5 μm, the influence of the wavelength is noticeable, and back-scattering gradually disappears. Finally, it should be noted that all of the curves presented in Figure 91 consider the variation in the complex refractive index of water according to the wavelength. This shows the importance of using accurate microscopic fog data (actually the measured DSD), not just approximate models simply recalibrated on macroscopic fog density, as already explained [111].

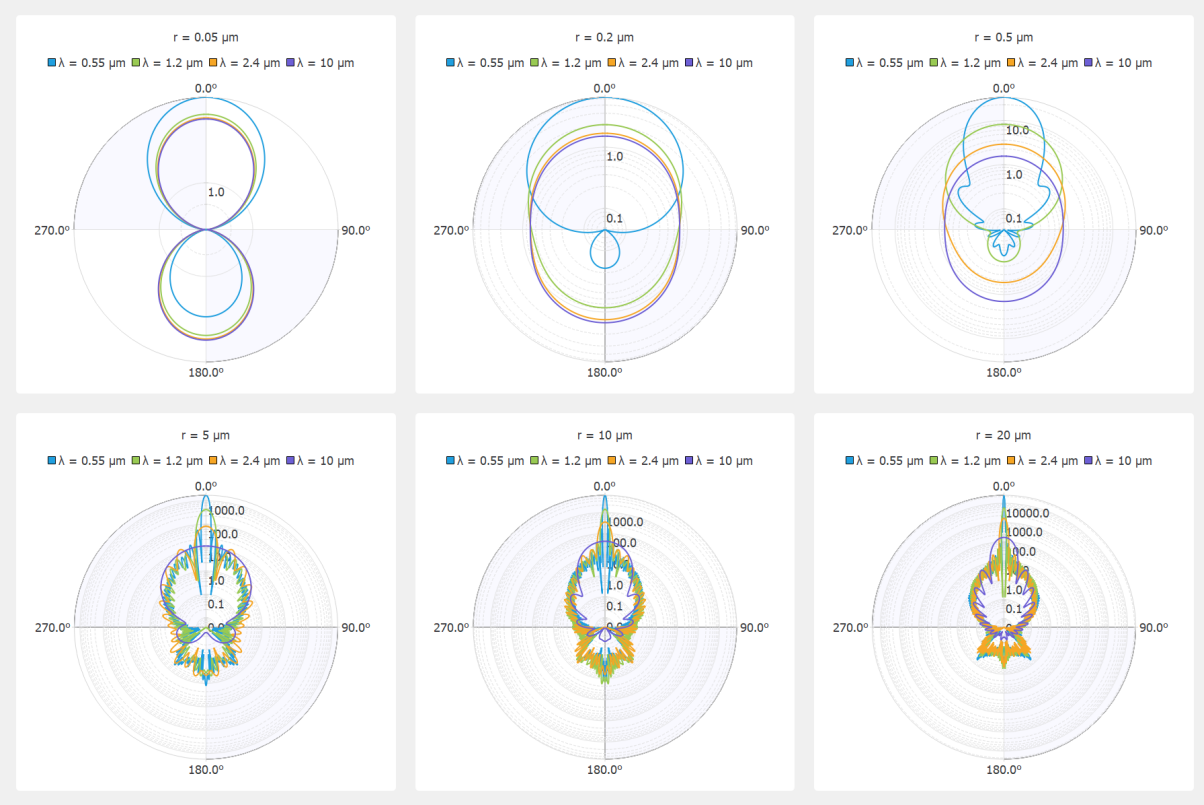


Figure 91: Polar representation of the phase function for six radii (r) of spherical particles and different wavelengths (λ).

The RTE equation 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 (8) is often used in a way to achieve an analytical solution. It consists of eliminating the collision (integral) term in (8) 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:

$$L_\lambda(r_0 + xu, u) = L_\lambda(r_0, u)e^{-\beta_\lambda x} + \frac{q}{\beta_\lambda}\left(1 - e^{-\beta_\lambda x}\right), \tag{9}$$

leading to the Beer–Lambert solution if $q = 0$:

$$L_\lambda(r_0 + xu, u) = L_\lambda(r_0, u)e^{-\beta_\lambda x}. \tag{10}$$

The simple case presented above corresponds to the framework of the Koschmieder theory [112, 113] 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:

$$I(x, y) = I_0(x, y)\, e^{-\beta d(x,y)} + I_s\left(1 - e^{-\beta d(x,y)}\right), \tag{11}$$

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 choice of the RTE modelling is important for the simulated fog added to a clear image. The images in Figure 92 are obtained with the Cerema Monte-Carlo based SWEET simulator [114] 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 92c, Koschmieder's model brightens the foggy image much more than SWEET (Figure 92b). 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).

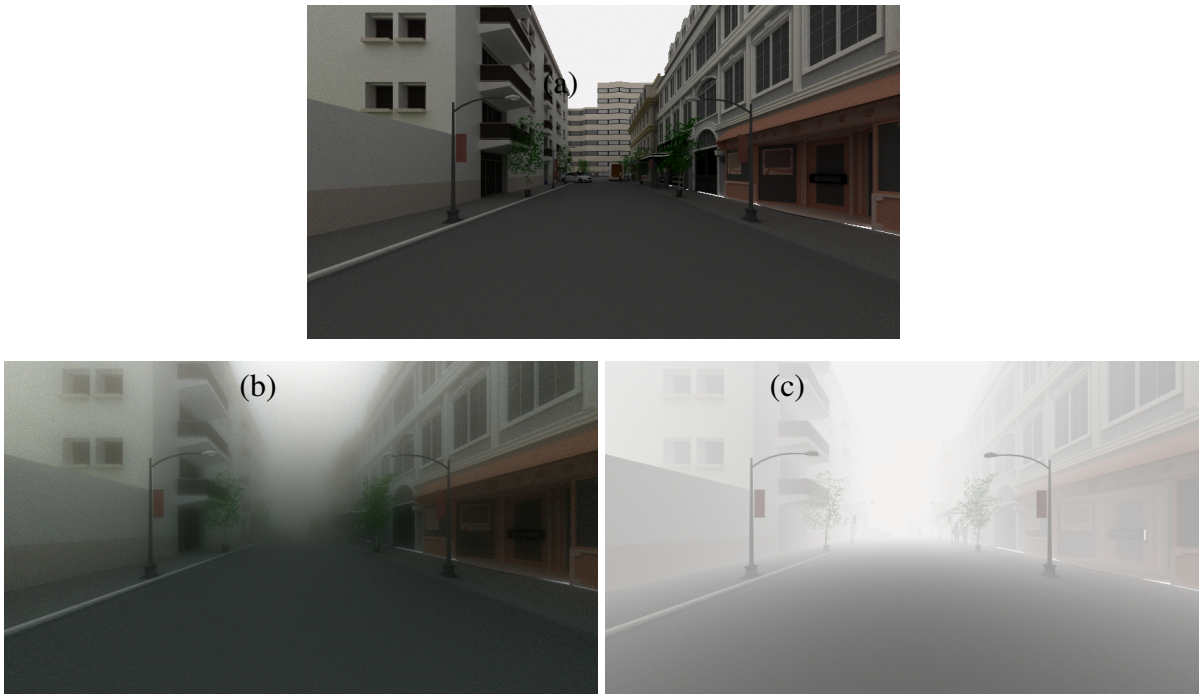


Figure 92: Simulated images for the intra-urban scene with the SWEET simulator without fog (**a**) and with fog (MOR = 20 m, (**b**)) and with the Koschmieder model (**c**) in day conditions.

The microphysics of the meteorological meteors have to take into account since they can impact light propagation. In [111], 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 (8) with two different PSD are showing in Figure 93). It can be observed different behavior of extinction for two fog PSD corresponding to the same meteorological visibility. We can remark that sensitivity depends on the wavelength (visible range at 0,55 μm) and thermal infrared range at 12 μm.

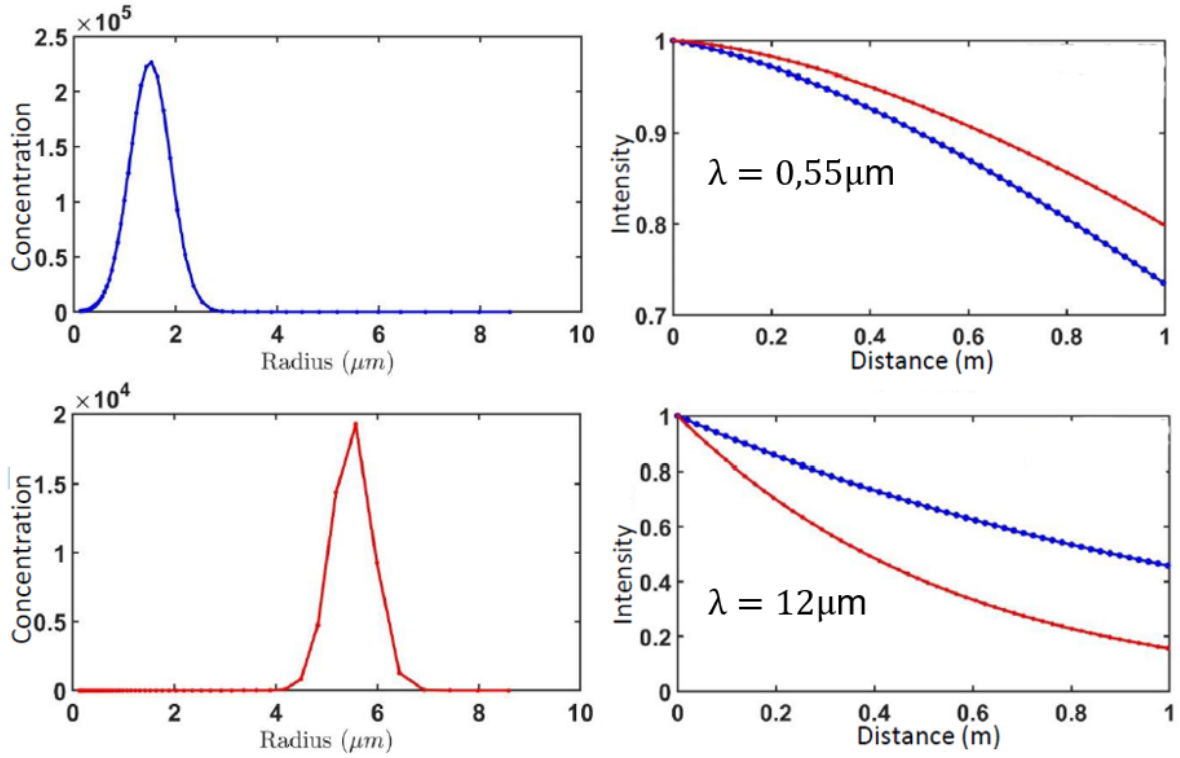


Figure 93: Simulated intensity w.r.t. the distance of a lambertian source for a fog with normalized visibility 0,75 m with small droplets (blue PSD on the left) and bigger droplets (red PSD on the left) at wavelength 0,55μm (top right) and 12μm (bottom right).

Rain affects perception of the road environment in different ways. Like fog or dust, heavy rain can reduce transmission through the atmosphere, by scattering light. 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 [115], a rain simulator was developed in [19] in order to add rain on "clear" images. The method is sketched in Figure 94.

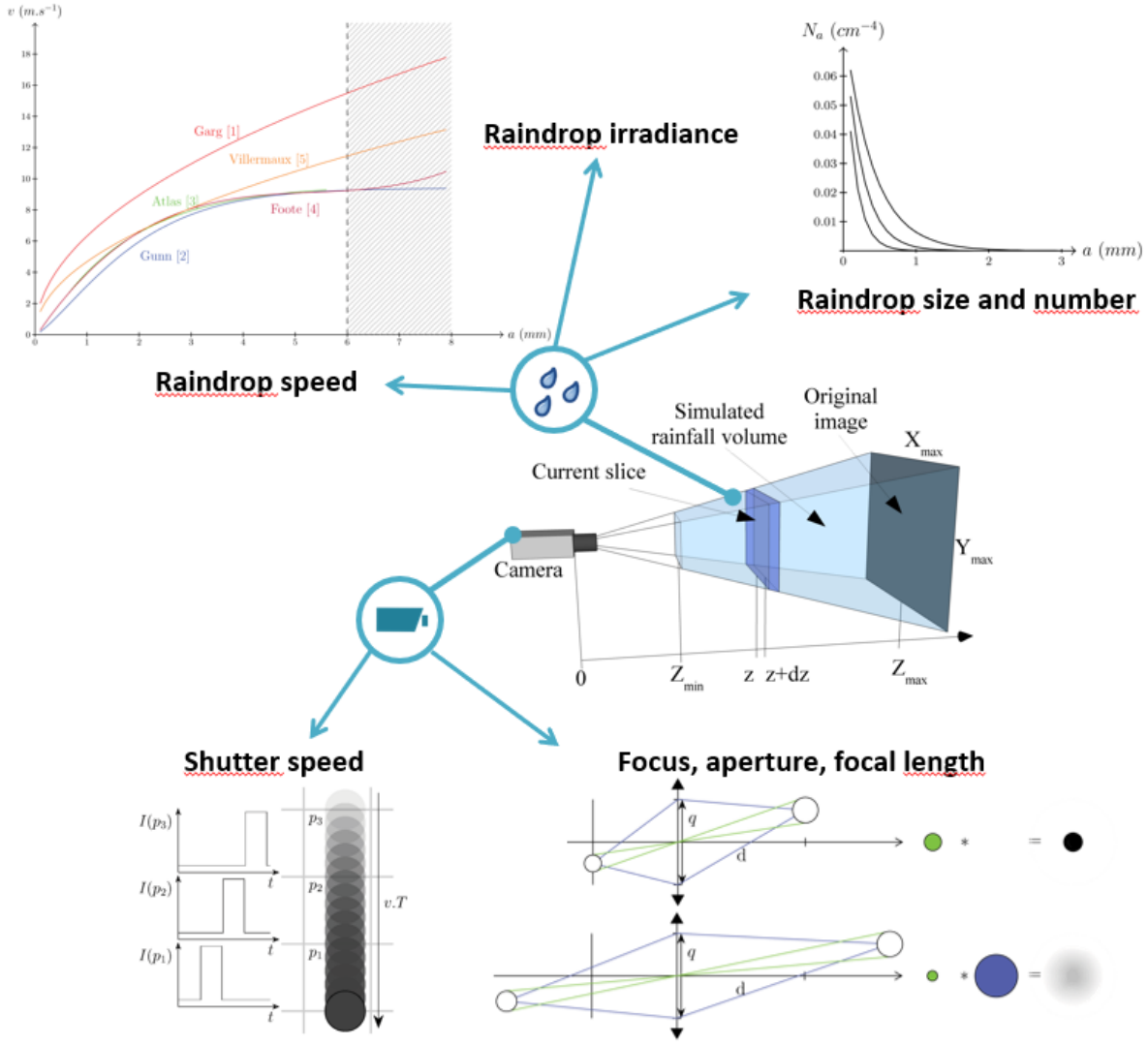


Figure 94: Scheme of the rain simulator developed in [19].

The simulator uses the most well-known laws of rain physics: PSD thanks to the Marshall and Palmer law [116], fall velocity depending on the size of the water drops parameterised by [117, 118], uniform distribution of raindrops in the air volume [115]. 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 Garg and Nayar (2007)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, [115] 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.

Rain directly affects the driver's perception (transmission through the atmosphere) but also produces changes in visibility through its action on vehicle headlights and windscreens. It also reduces the visual effectiveness of road markings and their contrast with the road by day and by night. An example of images generated by ProSivic are done in Figure 95.

Figure 95: Rain and snow effect in simulated images of ProSivic (specular reflection on wet materials, drops on windscreens and snow on roads) - source UGE.

The optical reflection characteristics of road environment materials (including wet materials) have to be considered in the simulators. We refer to [119] for a study on the spectral reflectance characterisation of the road environment to optimize the choice of autonomous vehicle sensors.

The LiDAR can be simulated for fog, rain and dust thanks to the well-know LiDAR equation. In [120, 121], the LiDAR equation for fog, rain or dust, for a target at a distance z from the sensor is expressed as:

$$P_r(z) = E_l \frac{c\rho(z)A_r}{2R^2} \tau_T \tau_R \beta_{\text{back}} exp(-2 \int_0^d \alpha(z')dz') \tag{12}$$

With P_r the power received by the LiDAR sensor in W, E_l the laser pulse energy in J, c the speed of light in $m \cdot s^{-1}$, $\rho(z)$ the back-scattering coefficient of the target, β_{back} the back-scattering coefficient of the medium, $\alpha_{z'}$ the scattering coefficient of the medium along the path of the target, A_r the effective receiver area in m^2 and, τ_T and τ_R the transmitter and receiver efficiencies. For the rain, β_{back} is approximated by 1 and the extinction coefficient α can be estimated based on a power law depending of the rainfall rate R [122]:

$$\alpha = aR^b, \tag{13}$$

where a and b are constant values which depend on the PSD and the droplet velocity.

For the RADAR, which are mainly impacted in case of heavy rain fall, the radar equation is very close to the LiDAR equation: - The attenuation effect, based on the RADAR equation :

$$P_r = \frac{P_t G^2 \lambda^2 \sigma_T}{(4\pi)^3 r^4} V^4 exp(-0.2\gamma r) \tag{14}$$

- The back-scatter effect :

$$(\frac{S_t}{S_B}) = \frac{8\sigma_t}{\tau c \theta_{BW}^2 \pi R_t^2 \sigma_i}. \tag{15}$$

Variables	Names
P_r	Signal Power
P_t	Transmission Power
G	Antenna gain
f	Radar frequency
T	Pulse duration
θ_{BW}	Antenna Beamwidth
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 back-scatter signal
c	Speed of light
V	Multi-path coefficient
γ	Rain attenuation coefficient
σ_i	Rain back-scatter coefficient
λ	Radar Wavelength
r	Distance between radar and target

Table 6: Parameters and variables of radar equation

The previous parameters and variables are presented in table 6.

Snow and mist can also affect RADAR signal with attenuation and back-scattering, it can be modelled following the same equation than rain but with attenuation and back-scatter coefficients corresponding to rain [123].

2.4.6.2 Adverse weather condition generation with learning-based methods

Using learning algorithms to generate images with different weather conditions is another effective way of obtaining large datasets with fog, rain or snow. For this purpose, image-to-image translation approaches are widely used. Image translation is the ability to learn a mapping function between an input image and a target image in order to transferred a domain to an another one. Many works have explored this task and has been applied to different applications such as style transfer, object transfiguration, image enhancement. The majority of domain adaptation methods focused on degraded weather conditions involve generating such images from clear images. Architectures such as GAN or CycleGAN [124] are most often used for image generation in degraded conditions. Some reliable study [20, 125], have proposed an image translation approach to facilitate the translation of various weather conditions, including sunny, cloudy, snowy, and rainy conditions. In [20], the Weather GAN architecture is composed of an initial translation module, an attention module and a segmented module to extract the distribution of weather cues. These weather cues enable the determination of weather conditions in an image by precisely marking the location, structure, and distribution of these cues. This aids in identifying the region of interest to facilitate the translation process. Some results are presented in Figure 96. Multi-Weather City [125] propose a method based on GAN and CycleGAN archi-

tectures to create seven variations of the data under different weather conditions. Additionally, they introduce an extension to further diversify the generated conditions, resulting in a total of 14 adverse weather conditions, all associated with a single ground truth.

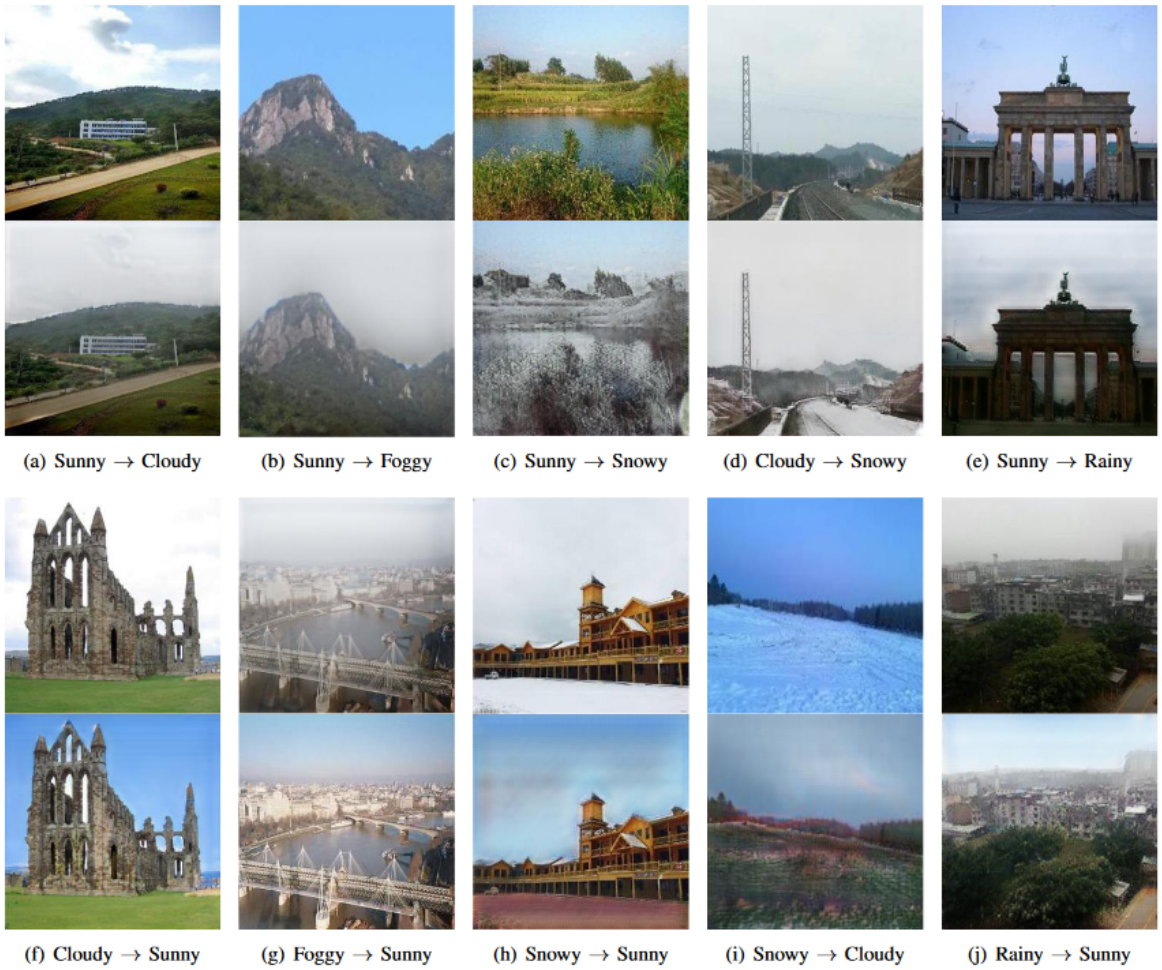


Figure 96: Weather translation results generated with Weather GAN [20].

Other works exclusively focused on fog generation have offered highly promising and encouraging results. As it is relatively straightforward to get synthetic foggy images, [126] proposed an analogical image translation between two domains, from real clear weather to real foggy images. This method, called AnalogicalGan, learns the correlation between synthetic clear images and synthetic foggy images and transferred it to the real domain. To achieve this, AnalogicalGan is composed of a supervised module in a synthetic domain, a cycle consistency module for the real domain and an adversarial training scheme between the two domains. [127] propose a Style Harmonisation for unpaired I2I translation (SHUNIT) method, that aims to generate a new style by harmonizing the target domain style (fog) obtained from a class memory and the original source image style (clear). Instead of a direct source-to-target style mapping as most frequently, SHUNIT focuses on harmonizing source and target styles. All these non-exhaustive works can contribute to the creation of synthetic image datasets under different meteorological conditions, with the aim of contributing to the evaluation of data under

adverse environmental conditions. In addition, these images can be evaluated using the metrics mentioned in the Image Generation paragraph 2.4.2.3.

2.4.6.3 Verification and validation methods and metrics

Verification and validation for degraded conditions in simulators can be done thanks to real testing allowing to compare simulated data and experimental data. A complementary approach would be to define a referenced simulator, for which a validation is done, and to then compare the outputs of the simulator under test to the ones of the referenced simulator.

For these two approaches, we can use the same metrics. The presence of degraded conditions in the road scene seen by the perceptive sensor (camera, LiDAR, RADAR) are supplementary conditions to the ”nominal” situations evaluated in the previous subsections. We can then use the metrics defined above in a degraded context. We detail here, thanks to the overview made in the previous paragraph, the key parameters to be set for the characterisation of the degraded conditions.

For fog, rain and snow, *macrophysics characteristics* respectively *meteorological visibility*, *rainfall rate* and *snow density* are the most influential parameters to be fixed for the validation testing. As seen above, *microphysics characteristics* (particle size distribution) have an impact on light propagation and they have to be taken into account. The optical reflectance characteristics of materials is of interest in the validation process since the radiance perceived by an optical sensor is the luminance received from sources after reflections on materials.

To summarize, the degraded conditions have to be defined by the parameters involved in the radiative transfer equation (8) and its source terms (6) and its boundary conditions (7):

- absorption coefficient depending on the wavelength;
- scattering coefficient depending on the wavelength;
- phase function depending on the wavelength;
- emitted radiance of the sources;
- materials BRDF.

The first three parameters can be obtained form the PSD of the medium thanks to the Mie theory (under assumption of spherical scatters).

We refer to [114] for respectively the assessment of a RGB camera simulator with foggy conditions by the comparison of contrast loss in virtual and real conditions.

2.5 General methodology and means to characterise the test resources used in simulation to validate AI-based systems

Validating a full vehicle simulation process involves ensuring that the simulated behaviour of the vehicle closely matches real-world performance across various and relevant scenarios and conditions. We have seen in previous sections how to qualify and evaluate each brick individually. To go further we can propose a general and generic methodology that can be applied to validate the entire vehicle simulation process.

PRISSMA methodology for validating the test resources used in simulation to validate AI-based systems

1. **Define validation objectives**: establish clear and specific objectives for validation, including the aspects of vehicle behaviour, performance metrics, ODD and scenarios to be validated.

2. **Gather real-world data**: collect comprehensive real-world data for comparison. This could include vehicle sensor data, road profiles, environmental conditions, and other relevant information. Real-world data serves as a benchmark for comparison. To qualify the ground truth data, validation datasets and associated annotations, please refer to the recommendations in Deliverable 1.6 of the PRISSMA project.

3. **Identify validation scenarios**: define a range of scenarios that the vehicle simulation should accurately represent. These scenarios should cover various driving conditions, environments, manoeuvres, and use cases specific to the intended application (e.g., urban driving, highway, off-road).

4. **Verify that calibration of all models** involved for the simulation have been done (see dedicated sections above). If this has not yet been done, calibrate the models concerned using ground-truth data and the model calibration procedures presented in the rest of the chapter.

5. **Simulation setup and execution**: configure the simulation environment to replicate real-world scenarios identified in the validation objectives. Run the simulation using the established scenarios and inputs.

6. **Data analysis and comparison**: analyse the simulation output data and compare it against corresponding real-world data. This comparison should include key performance indicators for all the sub-parts of the simulation framework, such as vehicle dynamics, sensor readings, control system responses, etc. All the sub components involved (all models for example) must be tested and check against appropriate validation metrics (see previous sections, depending on model type).

7. **Quantitative and qualitative Assessment**: perform quantitative analysis using statistical methods to quantify differences between simulated and real-world data use the appropriate comparison metrics. Then conduct qualitative assessments by visually comparing simulation outputs (e.g., visualisations, animations, photo-realism) against corresponding real-world scenarios.

8. **Iterative refinement**: identify discrepancies or areas where the simulation output deviates from real-world or expected data. Then iteratively refine the simulation models, parameters, algorithms, or input data based on the observed differences. Re-run simulations and re-assess until the simulated results align closely with the real-world data within an acceptable margin of error.

9. **Documentation and reporting**: document the validation process, including methodologies used, datasets employed, findings, modifications made to the simulation models, and the validation outcomes. Prepare a detailed report summarising the validation results, highlighting areas of alignment and any remaining disparities.

10. **Validation confidence and sensitivity analysis**: conduct sensitivity analyses to assess the simulation's robustness under variations in parameters, inputs, or scenarios. Evaluate

the confidence level in the validated simulation by considering uncertainties and limitations in the validation process.

11. **Continuous improvement**: implement a continuous improvement cycle by incorporating feedback from validation results into future iterations of the simulation process. This generic methodology aims to systematically validate and refine the vehicle simulation process, ensuring that it accurately represents real-world vehicle behaviour and performance across a wide range of scenarios and conditions.

3 Procedure and protocols for the evaluation and validation of AI-based systems through simulation tools

3.1 Introduction

3.1.1 Overall issues and objectives

This section aims to propose procedures and protocols for the evaluation and validation of critical AI-based systems and subsystems from the evaluation environments chosen in T2.3 and validated in T2.5 (particularly on the level of realism, the relevance of simulation tools, and the quality of ground truths in terms of evaluation references).

In addition, this section will address best practice procedures and implementation of experimentation plans using simulation and co-simulation platforms.

In order to enable the implementation of the evaluation and validation stages, we will propose definitions, criteria and evaluation metrics for AI-based systems (with the outputs of WP1, WP3, WP5 and WP8 as well as the previously mentioned collaboration with Pillar 1).

For the proposed simulation environments, an identification of the optimum scope of use (including dysfunctional through the injection of failures, etc.) will be carried out.

These actions will be tested with several POCs in parallel to those proposed in task 2.5 by adding a level of complexity linked to the applications (AI-based systems, system of systems, communication and cyber-security systems).

3.1.2 Classical high-level evaluation and validation procedures for critical systems

Artificial Intelligence (AI) is taking a more and more important part of Automated and Autonomous Driving Industry within the scope of the entire system and therefore, the robustness, reliability and safety of AI-based systems have taken centre stage in the global validation of the full system. Among these, critical AI-based systems, those that directly impact human lives, infrastructure, or decision-making, demand a heightened level of scrutiny in their evaluation and validation procedures. It is then imperative to establish comprehensive and rigorous processes to ensure their effectiveness and accuracy. This chapter digs into the pivotal realm of evaluation and validation procedures for critical AI-based systems, exploring and suggesting best practices that underline the development of AD focus on AI critical systems.

The high level process for the evaluation procedures for a critical system can be described using the following process flow :

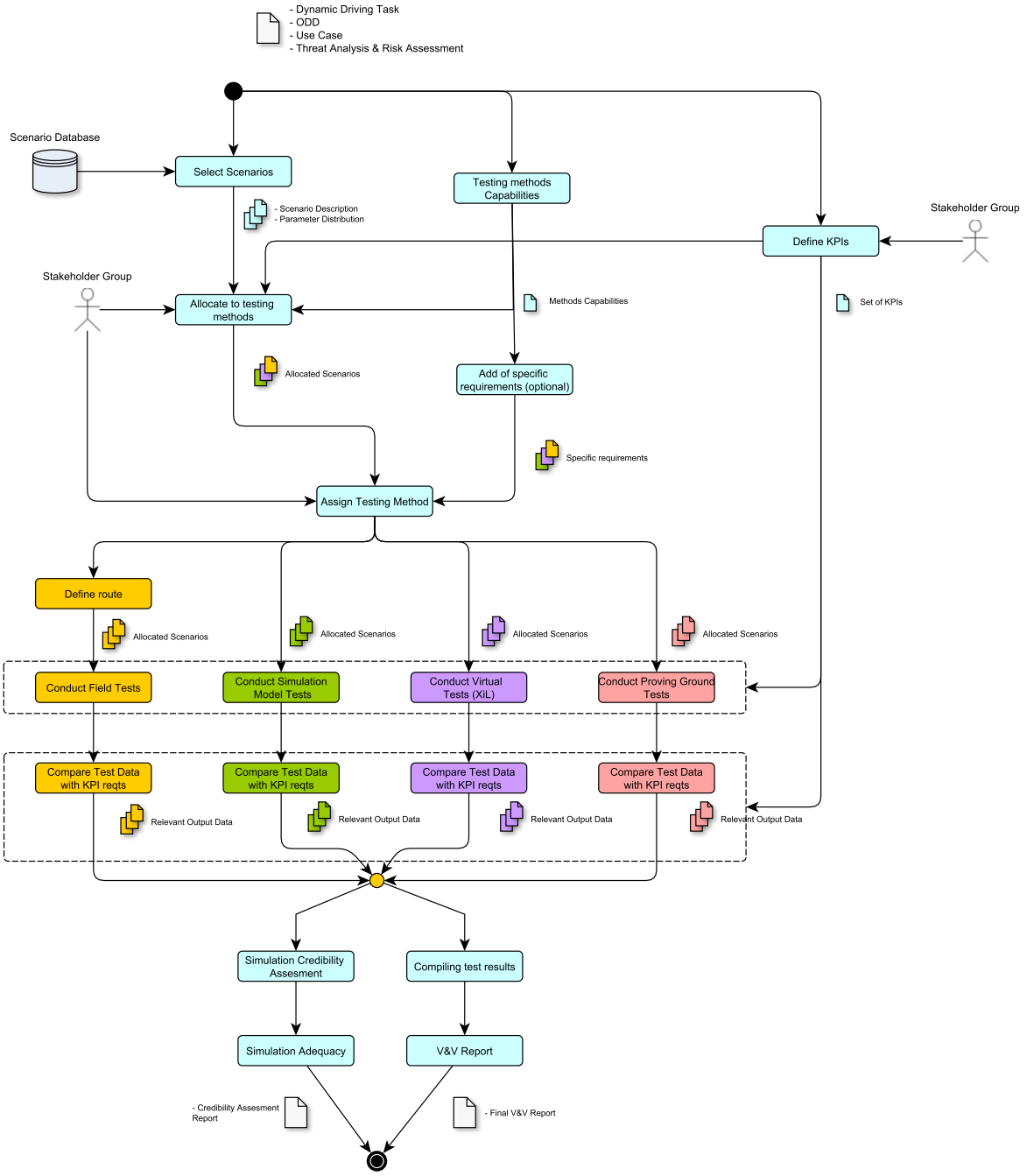


Figure 97: Global process for evaluating a critical AI-based system

As described in the graph above 97, the validation process of a critical Automated Driving System is based on three main steps :

- **Test Run Preparation phase** : This phase includes the definition of the objectives, the specification and the selection of the scenarios to be tested from the Database and the assignment of each test case to a specific test facility or defined simulation environment.
- **Test run execution** : Based on the specificity of each testing tool chain, execute the test as

per described on the concrete scenario (implemented test cases) given by the step before.

- **Test Results Compilation** : this task consists in extracting the results from the test execution and applying post-processing (like the creation of reference) and metrics and KPI to analyse the results. A final stage consists in compiling all test results into a unique document that is then distributed to required stakeholders (homologation body, auditing internal body, consumer testing if applicable etc...). A separate process here consists in assessing the adequacy of the testing method (i.e. Simulation, Open Road, Proving Ground) to the purpose of the test itself.

This broad outline should be borne in mind when writing the PRISSMA evaluation protocol dedicated to simulation, which will be covered in section 3.2.2.

3.1.3 Survey on recent techniques for testing system and modules of ADS in simulation, involving Cyber attacks

In [21] the authors endeavour to address the following research questions within the scope of a large survey:

- **RQ1: What are the techniques adopted for testing different modules of an Automated Driving System (ADS)?**: To provide insights into RQ1, a comprehensive survey is conducted about testing techniques applied to various modules of ADS. Notably, the technical distinctions inherent in these testing methods are studied, considering the diverse characteristics of different modules within the ADS (sensors, perception, path planning, control).
- **RQ2: What are the techniques adopted for system-level testing of ADS?**: In response to RQ2, Authors delve into system-level testing techniques, focusing on the following aspects:
  - Firstly, existing empirical studies addressing issues and bugs found in public reports and repositories are scrutinised. These studies unveil systemic problems encountered during development or deployment, offering a holistic perspective on potential issues without necessitating actual system execution.
  - Secondly, investigations into safety concerns at the system level are explored, particularly when distinct modules collaborate and interact during system operation.
  - Thirdly, attention of the authors turns to the disparity between simulation-based testing and real-world testing—an increasingly significant topic. This exploration aims to enhance our understanding of testing quality.
- **RQ3: What are the challenges and opportunities in the field of ADS testing?**: To tackle RQ3, challenges and potential research opportunities associated with ADS testing are pinpointed, drawing insights from our survey findings. This analysis contributes to a deeper understanding of the landscape and sheds light on areas where further research and development are warranted.

This very recent study (July 2023) address the ADAS testing from a general point of with with AI and Cyber-attack as well as in real condition that in simulation. This work gives an overview of the background of the ADS; then the main results of their survey is given in 3

different parts: empirical study on ADS testing; techniques on module-level ADS testing and answer RQ1; and techniques on system-level ADS testing and answer RQ2. Then statistics and analysis of the works is given and a summary of the challenges and potential research directions is provided and answer RQ3.

For the techniques on module-level ADS testing, three perspectives are identified, namely test methodology, test oracle, and test adequacy. Concretely, (i) test methodology introduces various methods or technical innovations for testing; (ii) test oracle defines metrics that can be used to judge whether the module behaves correctly; and (iii) test adequacy proposes coverage criteria that tell if the test cases in a test suite are sufficient.

The sensing module is the frontier module of an ADS and the performance of the physical sensors (e.g., camera, radar, LiDAR, GPS, IR camera, ...) in this module is critical to the safety and security of the whole ADS. Relevant studies on the test methodology of this module can be divided into *physical testing* and *deliberate attack* (mainly Jamming attack and Spoofing attack). Physical testing aims to test the performance of the sensors under different physical conditions, while deliberate attack interferes with the input signals of the sensors to diminish the sensing quality.

The perception module receives and processes sensor data; based on that, it perceives external environments. The literature collected in this work includes:

- **the test methodologies**: The Adversarial attack is the major approach for testing the DNN models.
- **the test oracles**: In this type of test, metrics are used to distinguish between the expected and unexpected behaviour of the system under test. The main types of test oracles that have been adopted for perception testing are ground-truth labelling, metamorphic testing [94–99], and formal specifications, to judge whether a bug exists in the perception module.
- **the test adequacy**: various metrics, analogous to the test adequacy criteria for programs, have been proposed; some of the metrics are for general DNN testing, while some are dedicated to ADS testing. Two typical adequacy criteria are Combinatorial coverage and Structural coverage.

The planning module takes the information from the perception module as input and produces a suitable driving trajectory as a reference for the control module to make decisions. In the planning module, the test methods are:

- **the test methodology**: provide traffic scenarios for an ADS and checking whether the planner module generates trajectories that satisfy properties such as safety, comfort, and low consumption
- **the test oracle**: Assessing the correctness of the output of the planning module, i.e., a planned trajectory. But this type of test needs to use a ground truth.
- **the test adequacy**: the inputs to the planning module involve both external parameters that identify a scenario and internal parameters of the ADS. Because there are infinitely many possible combinations of these parameters, generating test cases that are sufficiently diverse remains a great challenge.

The control module takes charge of the lateral and longitudinal control of the ADS. By using various control algorithms, such as MPC and PID control, it generates control signals, e.g., acceleration, deceleration, and steering angle, to the CAN bus for the control of the whole system. The testing of the control module mainly uses the test methodology (Fault injection, Sampling, Falsification) and the test oracle (design a model to generate an oracle area in the given scenario, which is the closest safe position ahead of the vehicle. A control decision is then considered as "the correct decision" if it could drive the vehicle close to the oracle area).

In figure 99, the different type of cyber-attacks are given by type of module involved in the ADS. In figure 98, the different methodologies and a short description of these methodologies are provides for the Simulation-based System-level Testing for Automated Driving System. In this very complete and interesting survey, an overview of the main simulation platforms for the evaluation and validation of ADS and their modules is given (see figure 102).

Methodology	Description	Literature	Test Objective	Environment
Search-based testing	Incorporating the perception input space and the whole system input space to accelerate search	[187]	AEB systems [261, 262]	Simulation
	Searching for unsafe feature interactions with decision trees	[188, 210]	Systems from IEE [263]	Simulation
	Generating virtual roads for testing lane keeping systems	[189–193]	Lane keeping system in BeamNG [15]	Simulation
	Searching for critical scenario boundaries	[194, 195]	Systems built based on simulators	Simulation
	Finding safety violations of an ADS in the dynamic environment	[196]	Apollo	Simulation
	Finding violations with a higher probability of occurrence	[199–201]	Systems such as BeamNG.AI [27]	Simulation
	Training surrogate models to accelerate testing	[202–205]	Systems like AEB system	Simulation
Adaptive stress testing	Assigning different priorities to the test cases	[211, 212, 214, 216]	Systems like the Intelligent Driver Model [264]	Simulation
Sampling	Sampling junction scenarios	[218]	Collision avoidance system	Simulation
	Sampling the simulation data in a traffic jam scenario	[221]	System in CarMaker	Simulation
	Combining STL with the sampling method	[219, 220]	Intelligent Driver Model	Simulation
	Identifying the relevant parameters of a logical scenario	[223]	Intelligent Driver Model	Simulation
	Sampling scenario space represented by feature model	[224]	AEB system	Simulation
	Adopt Importance Sampling to sample rare events	[226–234]	Systems like the ACC system [265]	Simulation
Adversarial attack	Generating attack patches as a camouflage of dirty roads	[235]	OpenPilot	Simulation and real world
	Generating adversarial perturbations under different weather	[236]	OpenPilot	Simulation
	Adding perturbations on the trajectories of NPCs	[238]	Driving models [266–268]	Digital dataset

Figure 98: A Survey on Simulation-based System-level Testing for Automated Driving System ([21])

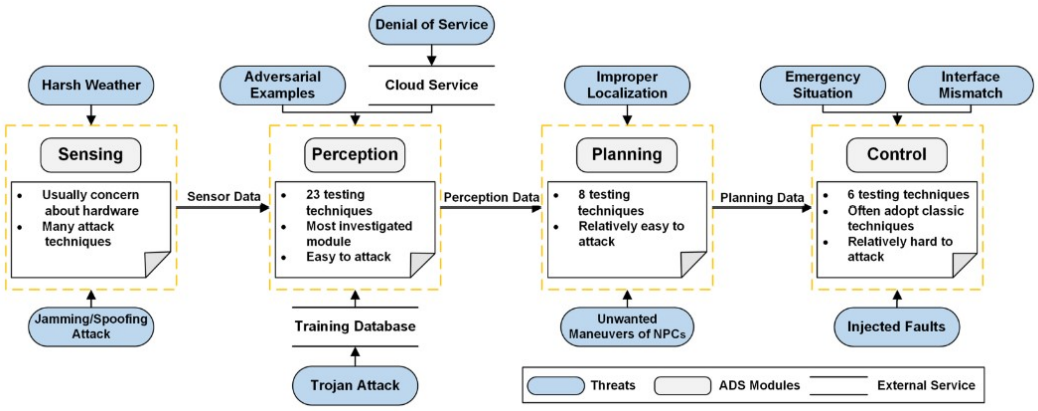


Figure 99: Threat model of ADS ([21])

For the System-level Testing with Simulators, the same 3 types of tests are performed:

- **the test methodologies**: The main used methods are search-based testing, adaptive stress testing, sampling-based methods, and adversarial attack.
- **the test oracles**: The oracles of the system-level testing of ADS are usually defined by safety metrics, such as time-to-collision (TTC) or Worst-Time-to-Collision (WTTC), which measures how far the ADS under test is from dangerous situations. The main used metrics are given in the figure 100 and 101. The Metamorphic method is also used in order to handle metamorphic relations to distinguish between real failures and false alarms. The metamorphic relation regulates that the behaviour of the ADS should be similar in slightly different scenarios. The third method is the formal specification which uses temporal logic languages to express the properties that the system should hold during the running.
- **the test adequacy**: the adequacy of testing is embodied by the diversity of the testing scenarios for the ADS. Two typical adequacy criteria are Combinatorial coverage and Scenario coverage.

Category	Name	Description
Temporal metrics	TTC [163]	The time until two objects collide with the current speed and path
	WTTC [241]	The time of the collision in the most likely accident scenario
	MprISM [242]	Estimating the TTC with the consideration of game interaction between vehicles
	THW [243]	The time between two objects reaching the same location
	MprISM [242]	The remaining time until the start of the last driving maneuver that can avoid collisions with all objects in the scenario
Non-Temporal metrics	SD [196]	The stopping distance of the vehicle at the maximum comfortable deceleration
	LP [191]	The distance between vehicle center and lane center
	DRAC [244]	The minimum deceleration rate required by a vehicle to avoid a crash
	SARR [245]	The number of steering angle reversals larger than a certain value

Figure 100: Commonly Used Safety Metrics ([21])

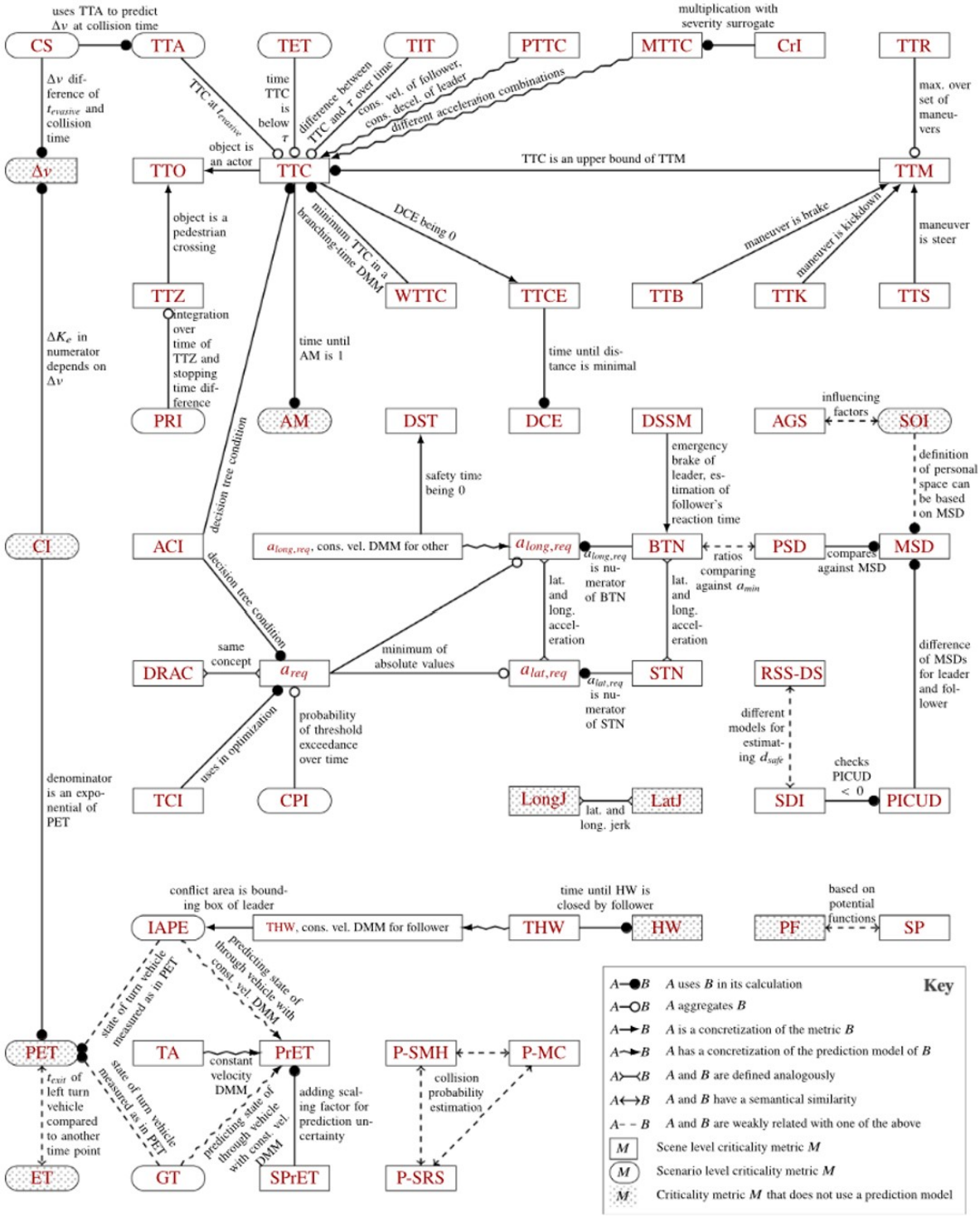


Figure 101: Inter relation and interaction between critial and safety metrics for ADS ([22])

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 102: Simulation Platforms for ADS Testing ([21])

3.2 PRISSMA generic experimentation plans

3.2.1 WP1 Requirements

Task WP1, through its deliverable 1.5, provides a list of requirements for the project as a whole, which provides an important initial set of best practices to be followed in the context of this document.

In the scope of the PRISSMA method, the main objective of the safety demonstration is to verify that the ARTS is at least as safe as human in equivalent situation (GAME principle). This demonstration shall therefore rely on an objective criteria that remains the same in all the

situations. The proposal is to demonstrate that the ARTS reaches an acceptable residual risk level. The severity of risk can result in severe or fatal injuries. To cover the SOTIF part, it can be verified that the ARTS has been operated on a large enough distance or duration (see PM-1051- Qualified safety objective metric) with appropriate justification of the coverage of the evaluation domain, which means the demonstration has been realized on many different situations (see PM-1053 - Statistical distribution justification).

The evaluation domain is the space resulting from the combination of the following spaces:

- The ODD, including:
  - The road infrastructure (Pathway) and the events that can reasonably occur on this pathway (both environmental conditions and actors events)
  - ARTS capabilities limitations with regards to the possible events and environmental conditions (for example, if the ARTS cannot be operated safely at night, then night utiliation is out of the ODD)
- ARTS functions and requirements
  - Object and Event Detection and Response (OEDR) (ARTS automatic driving requirement, being AI or not)
  - Other system events (risks, failures, functional insufficiency, triggering conditions)

The major applicable requirements are the following:

- PM-1051: Qualified safety objective metric - It specifies the criteria that the safety metric must meet to be considered valid and reliable for the evaluation process.
- PM-1065: Qualification of test runs - Specifies the objective of the test against the metric previously defined
- PM-1053: Statistical Justification for Variability Parameters - Ensures that the scenarios are statistically representative of real-world conditions, thereby making the evaluation results more reliable.
- PM-1070: ARTS AI function-of-interest Evaluation - Mandates that the ARTS is tested on simulation and test tracks, and produce the following metrics
  - The % of the evaluation domain covered by track tests (confidence level)
  - The % of the evaluation domain covered by tests in the operational environment (the % will be very low, and related to test permissions on the final route)
  - The % correlation between the real ARTS behavior versus the simulated ARTS behavior
  - The % correlation between the results obtained by the ARTS supplier in simulation and the results obtained during this evaluation

One of the most important part is the verification of the correlation between the behavior of the real system on test track compared to the simulated one, which is, after the audit of simulator and models, the final key to enable the usage of the simulation for qualification of the simulated systems, and not only a mean of analysis.

- PM-1068: Qualification of the simulator - focus on the fidelity and robustness of the simulation tools, ensuring that they can accurately replicate real-world conditions.
- PM-1075: Qualification of the test sequencer - specify that the test sequencers must be capable of executing the evaluation scenarios in a reliable and consistent manner.

The annex of the document 1.5 list the requirements for the following aspect which shall be particularly covered in the scope of simulation test:

- AI properties to be evaluated
  - Performance & convergence definition
  - Limits, Robustness & Resilience
  - Confidence index
  - Interpretability
  - Testability

3.2.2 PRISSMA Experimentation plan

Of course, before even starting to write the evaluation protocol, the first thing to do is to properly define the tasks and functionality that you want to evaluate. This task in itself may require discussions upstream of the evaluation with the supplier of the solution to be evaluated, but is not specific to WP2. In the rest of this section, we will present the main features of the PRISSMA protocol applied to the simulation framework.

3.2.2.1 Global overview of the evaluation/validation stages

The PRISSMA methodology provides a generic framework for evaluation/validation starting from the definition of generic automated mobility service to the final evaluation in specific conditions allowing to guaranty the safety of the service, and the involved systems and component. This generic framework of evaluation is presented in the figure 103.

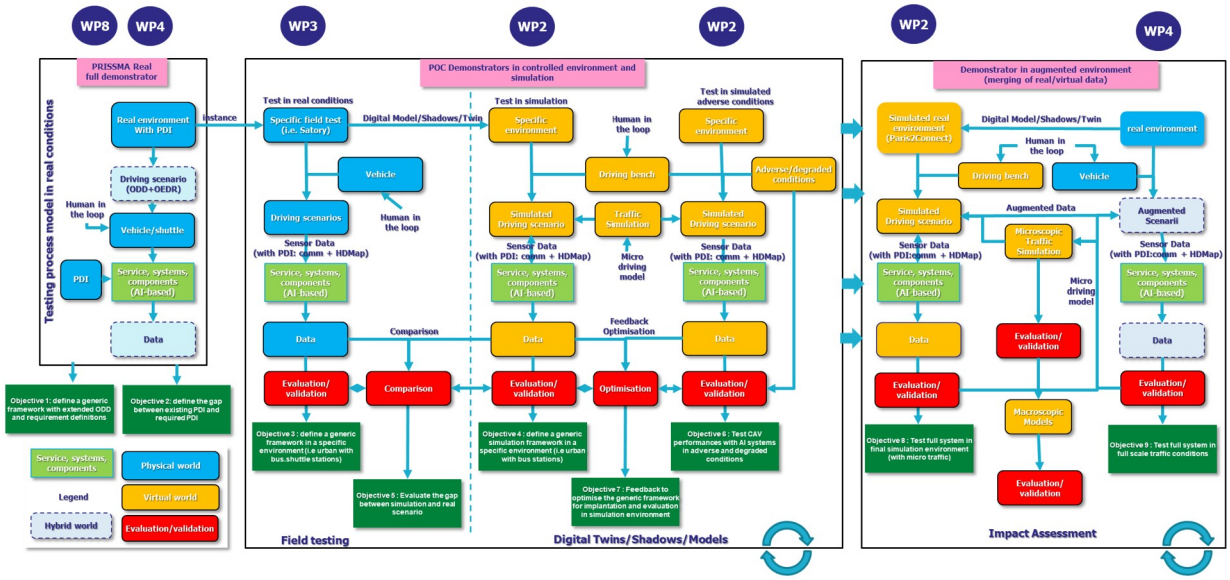


Figure 103: Generic Framework for evaluation process in simulation for ADS using AI-powered systems

The first part of this framework (see figure 104), in relation with WP8 and WP4, is dedicated to the definition of the operating space used by the service, system, or component under test. In the good practice of the evaluation and validation process, it is essential to know and to understand the environment (involving possible PDI), the conditions, and the constraints relative to the service (system under test) exploitation. So this first stage defines the following data, parameters, models, systems:

- Operational Domain (OD), ODD, OEDR based on extended PRISSMA's ODD taxonomy. The taxonomy need to be provided by the OEM, manusfacturer, or custumer and need to fit with the PRISSMA's taxonomy.
- scenarios based on scenes library build from the OD, ODD, and OEDR. This part needs to define the events and transition constraints between scenes.
- service, systems, and components for ADS with or without AI-based systems using. These information need to be provided by the OEM, manufacturer, customer.
- real vehicle embedded architecture (hardware and software) involving parameters. These information need to be provided by the OEM, manufacturer, customer.
- topology of sensors: types of sensor, Number of sensors, Position/orientation and perception coverage, Intrinsic and extrinsic parameters, Sensor calibration (provided by PRISSMA's team)
- data recording process: Define the format, Generate reference and ground truth, Generation of DataSets. Provided by PRISSMA's team
- evaluation and validation procedure: Define risk indicators, Define KPIs, Define metrics for component evaluation, Define criteria and threshold of validation. Provided by PRISSMA's team

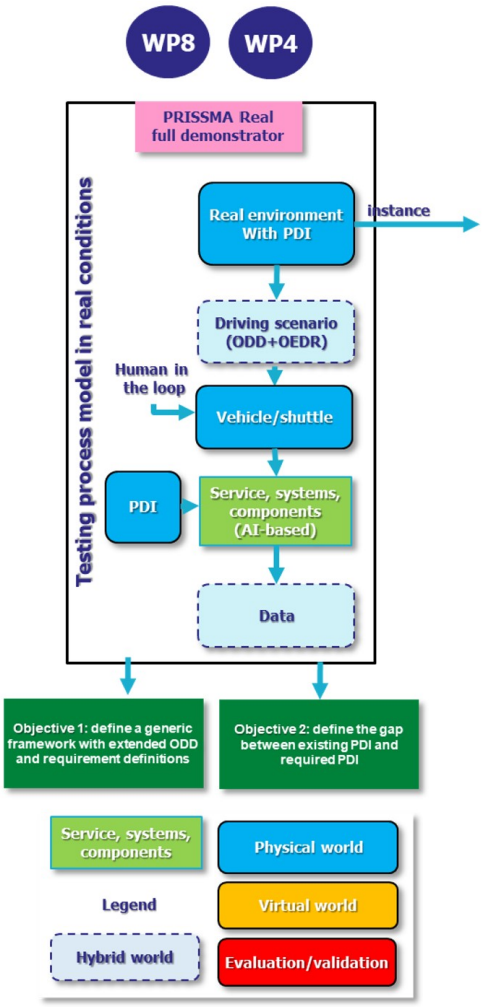


Figure 104: Generic Framework: Stage 1, the definition and description of the real service involving systems and components using AI-based applications and algorithms. In this framework, the service and system can use embedded components and sensors, and PDI (Physical and Digital Infrastructure) involving HD maps and communication means

The second stage is mainly dedicated to provide an instance of the real environment in a controlled environment allowing to test the system in specific critical and hazardous situation in order to assess the capability of the system to react to specific events and conditions defined in the OEDR without risk for the other road users. In PRISSMA, this step is addressed with POCs of WP2 and WP3 on the different test sites (UTAC, Transpolis, Satory).

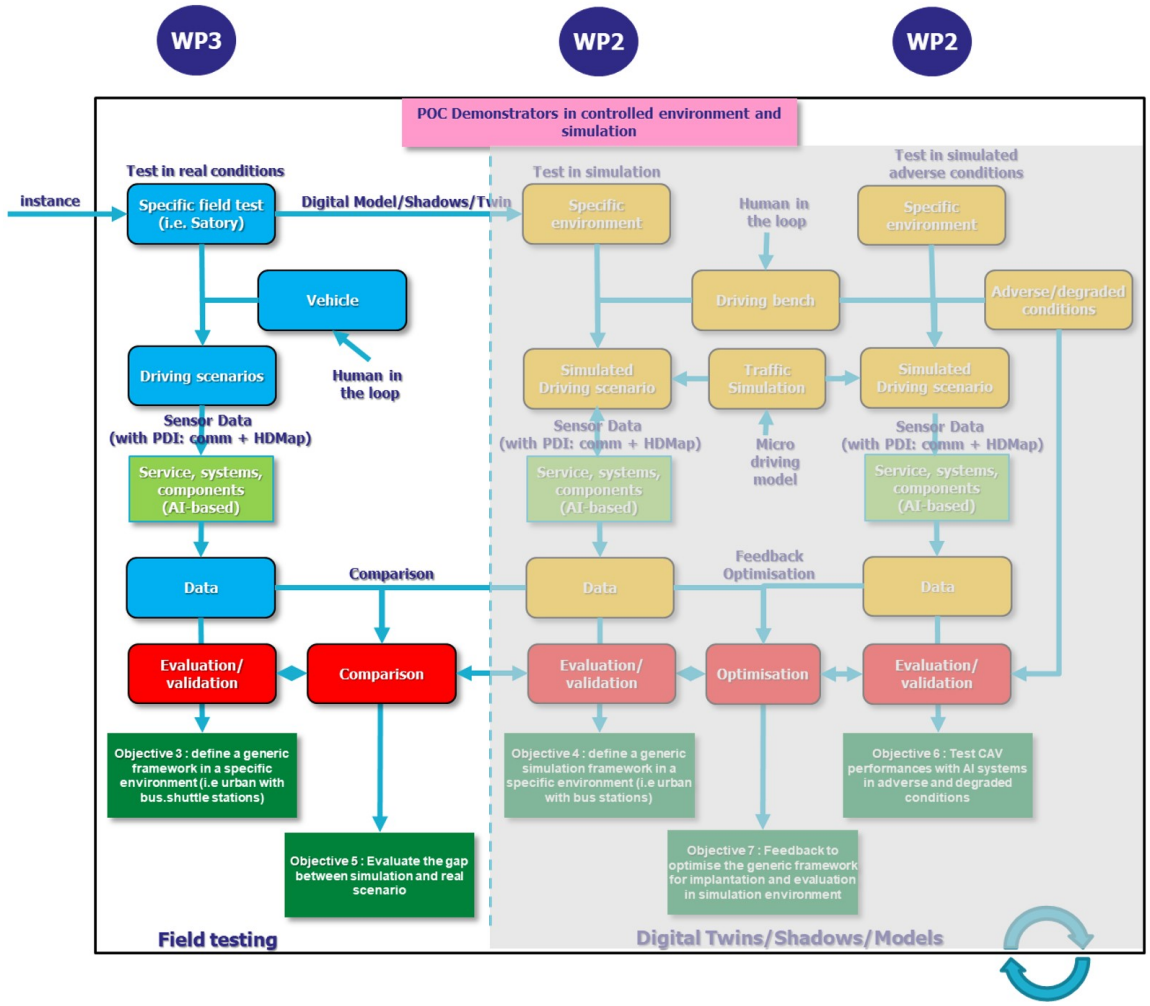


Figure 105: Generic Framework: Stage 2, implementation of the real system in a representative environment on a controlled test site

The third stage consist to implement for full application in a virtual environment using a digital twin of the controlled environment. In this virtual environment, we will apply a large set of representative scenarios representing specific situations having an impact on the mobility means safety. These scenarios represent crashes and near missed collisions situation, or adverse and degraded situations. A library of collision and near missed collision situations has been proposed in [128] and propose 3 classes (see figures 107, 108, 109) of situation with 10 different scenarios.

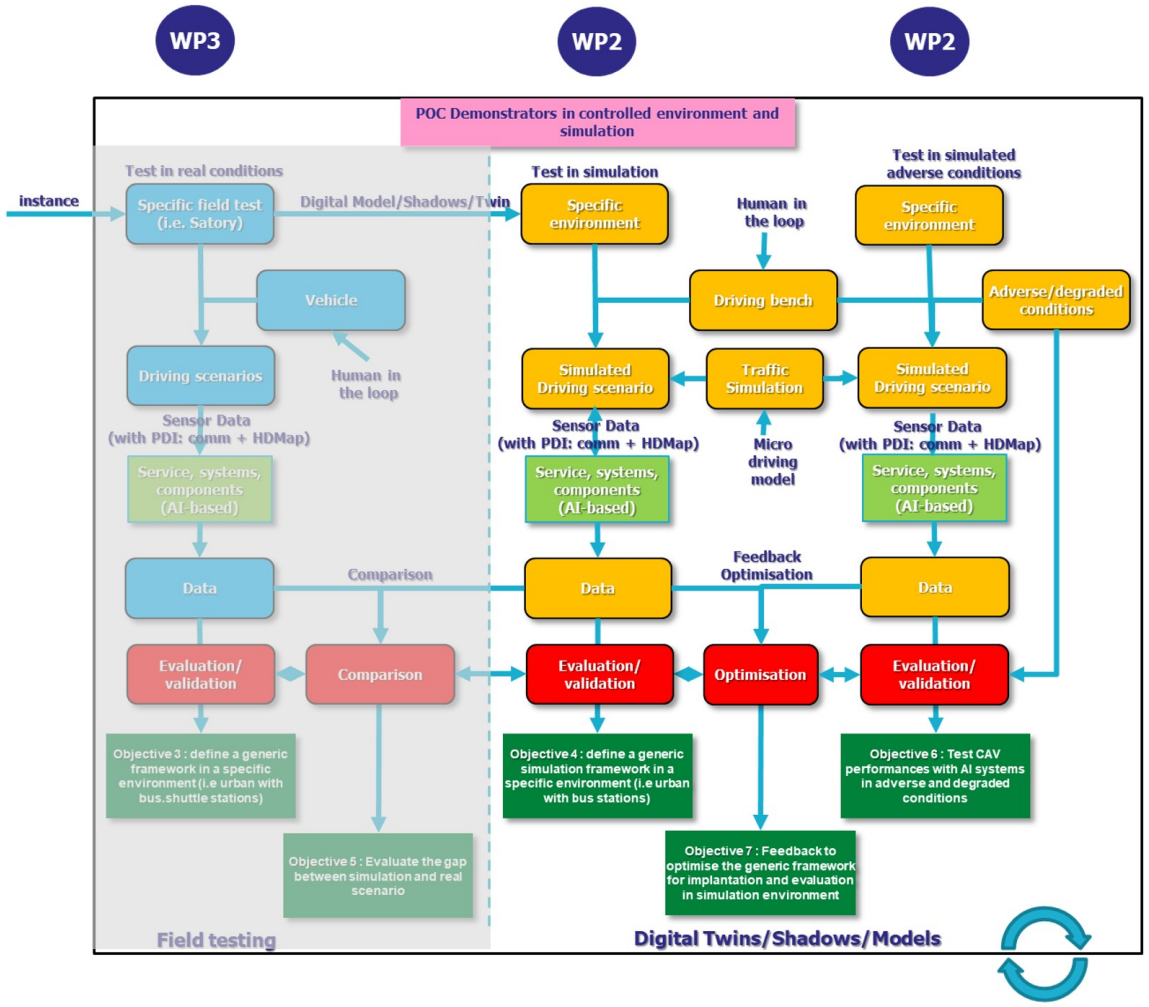
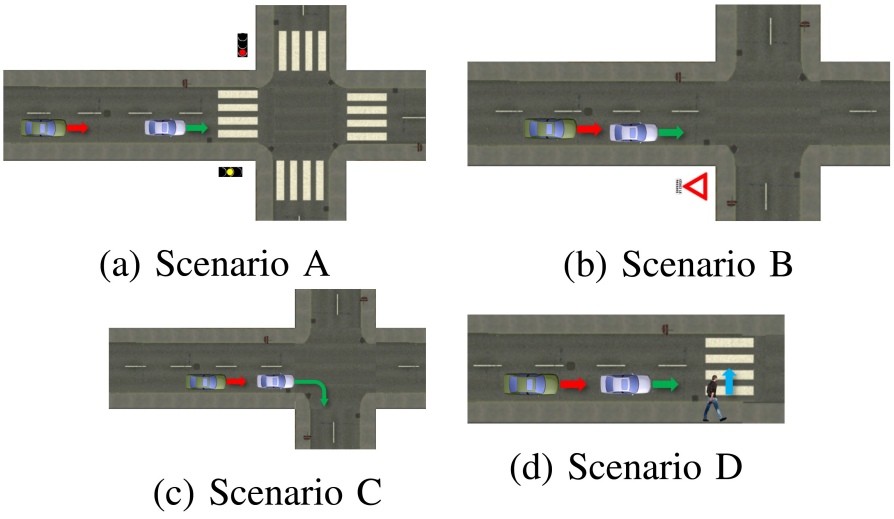


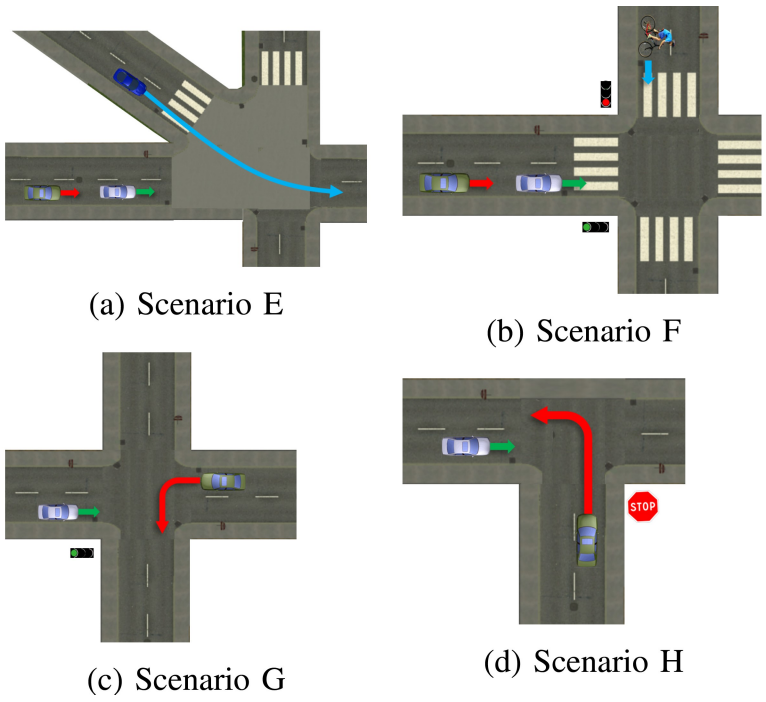
Figure 106: Generic Framework: Stage 3, implementation of the real system in the digital twin of the controlled test site



(a) Scenario A (b) Scenario B

(c) Scenario C (d) Scenario D

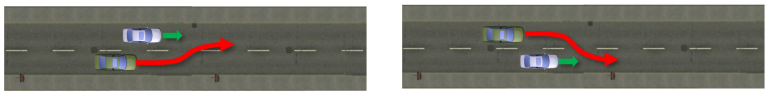
Figure 107: First class of collision scenario: Following Scenarios



(a) Scenario E (b) Scenario F

(c) Scenario G (d) Scenario H

Figure 108: Second class of collision scenario: Cross Paths Scenarios



(a) Scenario I (b) Scenario J

Figure 109: Third class of collision scenario: Lane Change Scenarios

In addition to these classes of scenarios, it is advised to use the scenarios proposed in Euro NCAP. Moreover, an additional set of scenarios could be dedicated to adverse and degraded conditions. These conditions have been described in [129],[130], and [23]. The adverse and degraded conditions impacting the safety of the automated vehicle and the AI-based systems are not only focused on the weather conditions but also on traffic, perception, and communication aspects. Traffic disturbances could be generated by the different configurations provided in the figure as showed in figure 110.

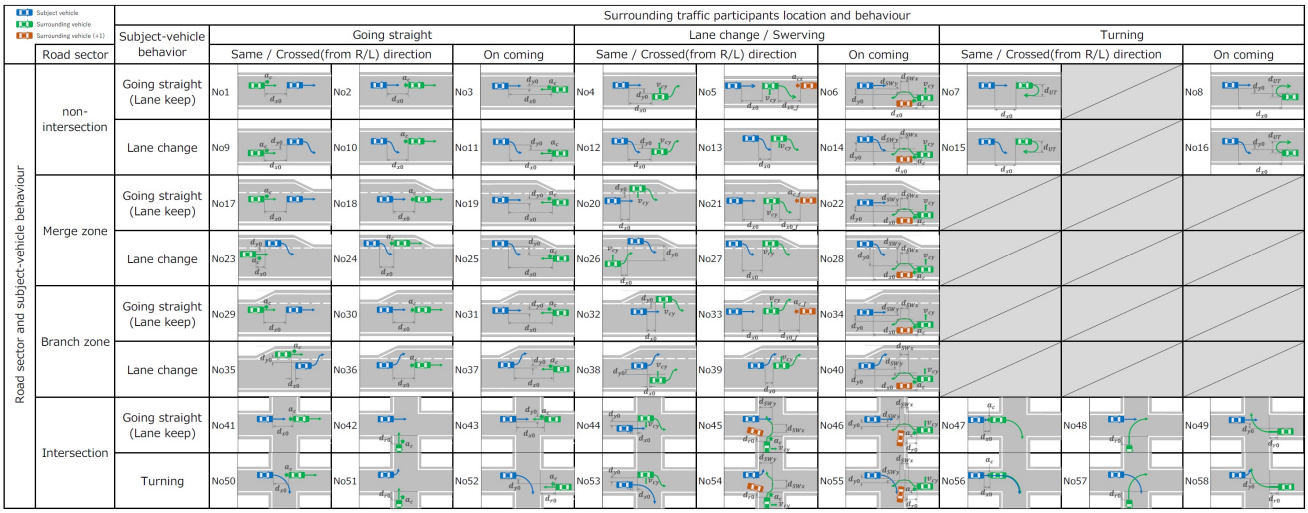


Figure 110: Traffic disturbance defined in [23]

Perception disturbances involve troubles and failures on the sensors behaviours impacting the data quality, issues of visibility and blind spots, and finally trouble and operating failure on the communication means (see figure 111).

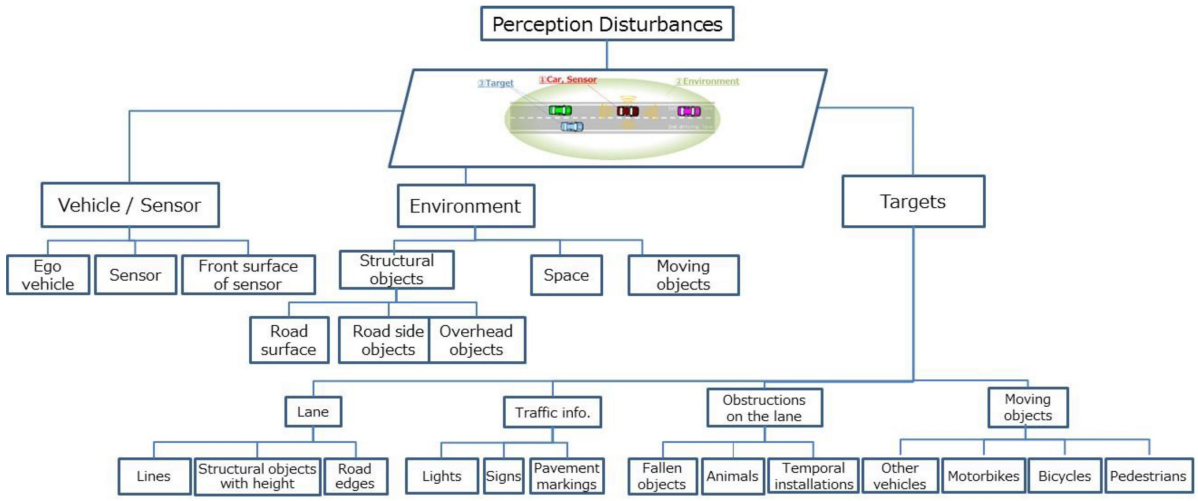


Figure 111: Perception disturbance defined in [23]

For instance, for the camera sensor, the possible disturbances are provided in the figures 112 and 113.

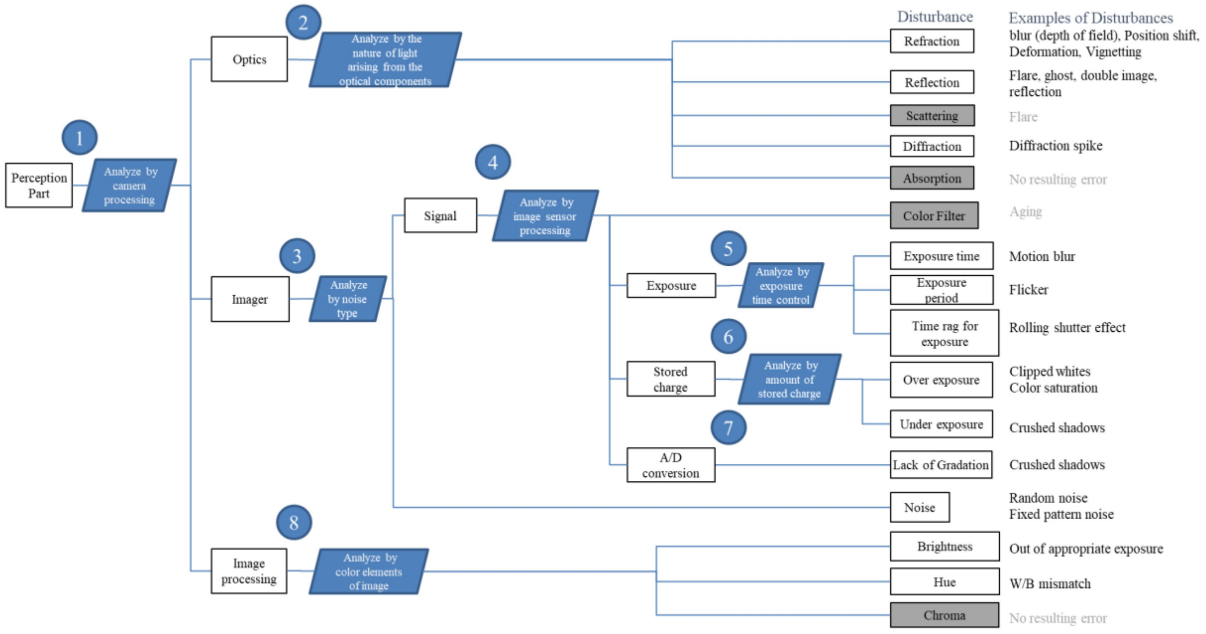


Figure 112: Disturbances on the detection part from camera data processing [23]

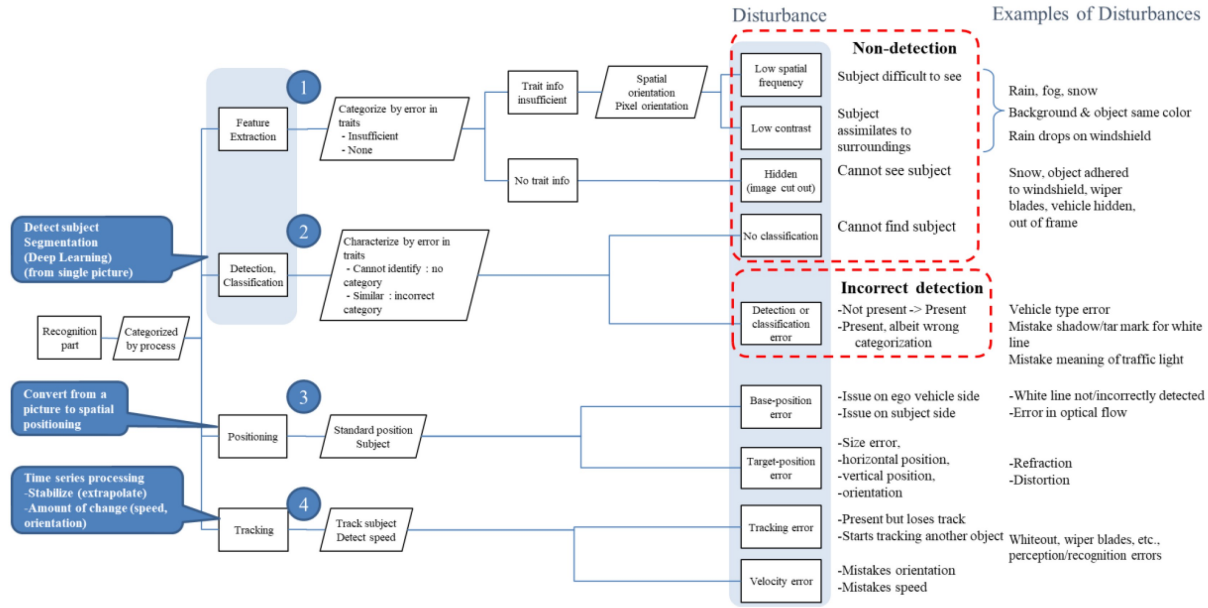


Figure 113: Disturbances on the recognition part from camera data processing [23]

All these conditions, by adding cyber attacks aspects, allow to build a set of representative scenarios for the evaluation and the validation of systems/applications/components performances from the safety point of view. Moreover, in this third evaluation stage, both the real controlled environment and the simulated environment can be used in same time in order to have a Vehicle in the Loop (ViL) evaluation process.

The fourth stage consists in generating the digital twin and the HD map of the real environment and to evaluate and validate the system and the AI-based methods and algorithms in this environment with the same ”semantic” scenario than in the third stage. An interesting aspect of this stage could be to generate virtual data for augmented scenarios in order to improve the complexity (level of entropy) of the scenario.

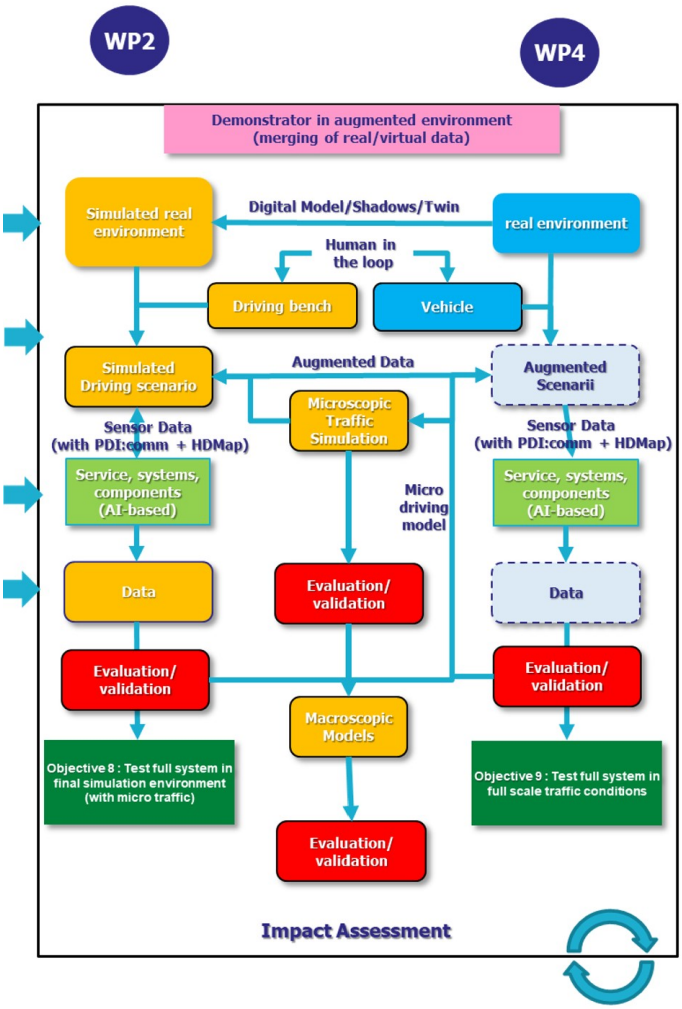


Figure 114: Generic Framework: Stage 4, implementation of the real system in the digital twin of the real environment with the capability to generate augmented reality in the two ways (from VR to Real, or from Real to VR)

3.2.2.2 The Generic PRISSMA's evaluation/validation framework

With all these constraints, big stages to follow and requirements, a generic evaluation/validation framework has been proposed in PRISSMA based on 3 layers.

1. **Layer 1**: This layer is mainly dedicated to the definition and the generation of the system under test, the simulation environment, the testing objectives and constraint. The testing objectives will define accurately the parameters and variables to take into account as well as the level of evaluation (service, system, component). In this layer, the OD, ODD and OEDR also have to be defined and built from the ODD taxonomy proposed in PRISSMA. The instantiation of the ODD of WP8 provides limits and thresholds to consider in the

scenes definition and by extension in the scenario building. Then, respecting defined ODD, a selection of the representative scenarios are built in order to cover the space of the testing objectives. The last process of this layer consists to select the AI method and algorithm under test to meet the desired tasks. In PRISSMA, several AI methods are tested in the different POCs (Perception, decision-making, path-planning). The information about system S, environment E and testing objectives are used in a parallel module allowing to select the relevant parameters for the generation of the ground truths.

2. **Layer 2**: The second layer consists to apply the scenarios on the system/component in the testing objective framework with constraints and limits provided by ODD in order to collect data about the behaviour of the system/components respecting the OEDR. In this layer, we take into account the constraints defined in the scenarios about the degraded and adverse conditions, the static and dynamic elements/components of the environment, and the capabilities of the ego-vehicle. The environment is generated by a dedicated simulation engine (currently an adapted graphical engine). The system/component runs in the environment and a set of observations are collected in order to feed the evaluation layer. Moreover, in same time, a set of references and ground truths are generated.

3. **Layer 3**: the last layer will use the 2 datasets generated by the previous layer in order to evaluate and validate the performances of the system in specific environments and conditions. The level of evaluation is adapted to the testing objectives defined in the first layers. Depending of these objectives, specific methods and metrics/KPI are use and apply in the evaluation process. At the end, a report and a document are generated.

These 3 layers are provided in the figure 115.

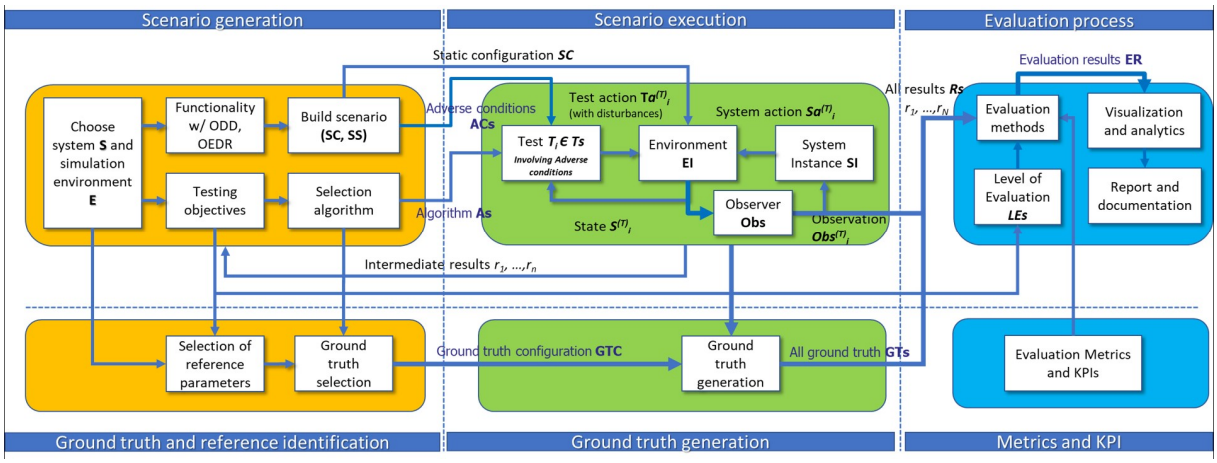


Figure 115: Simulation Framework for evaluation of AI-powered systems in STRA

The application of these 3 layers has been provided as the following pseudo code by [131].

3.2.2.3 Layer 1: the upstream process for definition and generation of the evaluation framework

The generator is crucial in building the framework, generating necessary configurations, and selecting algorithms for evaluation. As demonstrated in Algorithm 1, it is responsible for gen-

erating configurations of evaluation scenarios based on Operational Design Domain (ODD) and Object and Event Detection and Response (OEDR). It also selects candidates of AI algorithms for the framework according to specific objectives, then evaluates and validates them based on a representative real-world dataset. Moreover, the generator component generates the configuration of the ground truth for the executor based on the selected algorithms, ensuring the accuracy and reliability of the evaluation process.

Evaluation objectives

The evaluation objectives of an AI-powered system in ADS are derived from an analysis of the system and its operating environment. These objectives encompass multiple levels:

- At the system level, the overall performance and quality of the AI system are evaluated in simulated environments;
- The components/functionalities level focuses on evaluating specific functions and algorithms necessary to meet the expected functionalities of the system;
- Additionally, the scenarios level evaluates the system's capabilities within a defined ODD, including safe driving in different scenarios under varying conditions like non-optimal weather, traffic, and lighting.

Categorising the evaluation objectives into these levels facilitates a comprehensive evaluation of the system's performance, safety, and areas for improvement, offering valuable insights into its capabilities and limitations.

Scenario Definition

Scenario Definition involves the conceptualisation and specification of the fundamental elements:

- **Scene** contains the overall environment where the scene takes place, including
  - Dynamic elements which are objects capable of movement or state changes, such as vehicles, pedestrians, or cyclists;
  - Static elements, which are stationary objects in the scene, such as road infrastructure or buildings;
  - Environment factors, which refers to the surrounding conditions, such as weather or lighting, which can influence the behaviour of dynamic elements.
- **Event** represents incidents or occurrences that unfold during the scenario. These events can be pre-defined or dynamically generated and contribute to the scenario's progression. They include stimuli, triggers, or changes in the environment or state change of other objects (outside ego), shaping the sequence of actions and reactions within the scenario.
- **Action** pertains to the response or behaviour exhibited by the ego object in the scenario. It demonstrates how the ego object in the scene reacts to events or encountered conditions. Actions may include acceleration, braking, or changes in the direction of the ego vehicle.

- **Criteria** refers to the specific conditions or standards required for the simulation scenario to be deemed complete or successful. These criteria could include factors such as reaching a particular time limit, accomplishing predefined objectives, meeting specific performance metrics, satisfying safety requirements, or any other relevant measures that define the desired conclusion of the scenario.

Scenario Configuration

This part involves the implementation and customisation of a scenario based on the definition. This process focuses on the detailed setup and arrangement of specific elements, conditions, and variables within the scenario. An effective scenario configuration should be done within the defined boundaries of ODD and OEDR.

ODD contains the specific operating conditions and environments within which ADS is intended to function safely and effectively. By considering the ODD in the scenario configuration, the scenarios accurately reflect the real-world conditions that the system is designed to encounter. This involves defining geographic boundaries, traffic conditions, and factors that influence the system's operational limits, thus ensuring the scenario's relevance and accuracy.

OEDR focuses on the system's ability to detect and respond to specific objects and events within its operational environment. When configuring scenarios, it is imperative to define the types of objects the system should detect. Furthermore, the scenario should include events that the system should recognize and respond to, such as sudden lane changes, emergency braking, or any other relevant mapping. By incorporating these elements, the scenario enables the evaluation and improvement of the system's perception and response capabilities.

By aligning scenario configuration with the ODD and OEDR, the resulting simulations accurately represent the operating boundary and allow for a comprehensive evaluation of ADS.

Algorithms selection

To select algorithm candidates for an AI-powered system, it is essential to establish the domain of AI first. In the case of a visual perception system, deep learning methods like Convolutional Neural Networks (CNNs) have demonstrated promising results and are commonly used for image detection and segmentation tasks.

Once the domain is determined, specific tasks should be extracted based on the system's objectives. After an extensive investigation of algorithms suitable for these tasks within the chosen AI domain, potential candidates can be identified. These candidates will undergo training using relevant datasets, and if possible, the models will be fine-tuned. Subsequently, the performance of the trained models will be validated to ensure they meet the necessary criteria for further consideration.

Ground truth selection

In real-world environments, ground truth can be generated through manual annotation or by using calibrated and accurate sensors or devices to capture the actual values of the variables being measured. For example, the BDD100k dataset [132] has been widely used in visual perception research [133, 134] and provides ground truth labels for various tasks. In the proposed PRISSMA framework, several different types of references and ground truths must be used both

for the training stage and for the preliminary validation of the selected algorithm. This aspect will be detailed in another section.

Algorithm 1 GENERATOR

procedure GENERATE ▷ S: system, E: environment
  ODD , $OEDR$, $Objs \leftarrow$ DEFINE(S , E) ▷ Define ODD, OEDR, and also objectives $Objs$
  $SC \leftarrow$ CONFIGURE(ODD , $OEDR$) ▷ Generate configurations of scenarios SC
  $ACs \leftarrow$ GENERATE($Objs$) ▷ Generate adverse conditions ACs
  $As \leftarrow$ SELECT($Objs$, $Dataset$) ▷ Select the algorithms As based on objectives $Objs$, also the dataset
  $GTC \leftarrow$ CONFIGURE($Objs$, S , E , A_s) ▷ Generate configurations of ground truth GTC
  return SC, GTC, $Objs$, As ▷ Return configurations, objectives, adverse conditions, and selected algorithms
end procedure

3.2.2.4 Layer 2: execution of the system under test in the simulation environment

Execution process

The executor is responsible for executing the different test cases on the integrated platform and tools, which are built by the output from the generator component of the framework, and also generating different types of results. The module must ensure that the system is executing properly and that the intermediate results are being generated correctly, and then passed back to the generator component as feedback. This process aims to refine the parameters inside the configuration of scenarios and adjust ODD or OEDR if needed.

If there are any issues or errors in the execution, it needs to be resolved before passing on the final results to the evaluator component. Once the execution is complete, the final results are passed to the evaluator component for assessment against the different types of evaluation metrics.

The process of executor is expressed by the Algorithm 2.

Algorithm 2 EXECUTOR

procedure EXECUTE(SC, As, GTC, ACs) ▷ Outputs from generator as inputs of executor
  $Ts \leftarrow$ BUILD(As , ACs) ▷ Build the test cases with different algorithms As and adverse conditions ACs
  for $Ti \in Ts$ **do**
    SI , $EI \leftarrow$ INSTANTIATE(S with Ai in Ti , E with SC , P) ▷ Instantiate the system SI with algorithm Ai in test case Ti and environment EI with scenario configuration SC on the integrated platform P
    $Ta_i^T \leftarrow T$.GENERATE(S_i^T , ACi in Ti) ▷ Generate the test action Ta_i^T based on the environment state S_i^T and the adverse condition ACi in the test case Ti
    $Sa_i^T \leftarrow S$.GENERATE(Obs_i^T) ▷ Generate system action Sa_i^T based on the observation Obs_i^T
    S_i^T , $Obs_i^T \leftarrow E$.UPDATE(Ta_i^T , Sa_i^T) ▷ Environment updates based on system actions Sa_i^T and test actions Ta_i^T
  end for
  GTs , $Rs \leftarrow P$.GENRATE(Obs , GTC) ▷ P records final results Rs and ground truth GTs (based on ground truth configuration GTC) from the observer Obs
  return Rs, GTs ▷ Return final results and the ground truth
end procedure

Integration with tools and platforms

Integrating the evaluation framework with appropriate tools and platforms is crucial for the executor component to effectively perform its tasks. Fig. 11 illustrates a case of integration of two interconnected software, RTMaps™ and Pro-SiVIC™ in the presented generic framework. By utilising the capabilities of a simulation platform like Scaner-Studio™ or Pro-SiVIC™ to design realistic and complex virtual environments, developers can simulate various road, traffic,

and weather conditions that their systems might encounter. Meanwhile, RTMaps^{TM} provides a module-based environment to design different sub-systems for ADS, and also has real-time multi-sensor processing and data fusion capabilities to enable the effective management of sensor data. Besides, it also has the capability to record and replay the data from the observer, also providing an efficient means for the evaluation process.

As shown in Fig. 11, it is worth mentioning that the Data Distribution Service (Data Distribution Service (DDS)) as the communication mechanism is integrated within the evaluation framework, which offers an effective and inter-operable Application Programming Interface (API) for seamless data sharing and communication among the various components. It contributes to enhancing the framework's overall performance, scalability, and effectiveness in the evaluation process.

3.2.2.5 Layer 3: Evaluation and validation stage with the observation from the layer 2

The evaluator is responsible for evaluating the performance of the AI-powered systems. It applied the selected evaluation metrics to the output from the executor, and then hereby evaluate the results combining corresponding ground truth. The overall process can be abstracted as shown in Algorithm 3.

The metrics used in the framework are chosen based on the different levels inside the evaluation objectives of the system, such as component level, system level, and scenario level.

Algorithm 3 EVALUATOR

procedure EVALUATE ($Rs, GTs, Objs$) ▷ Evaluates the final results with ground truth
  $LEs \leftarrow$ LEVEL$(Objs)$ ▷ Define different levels of Evaluation (depending of criteria and objectives) LEs
  $Metrics \leftarrow Select(LEs)$ ▷ Select metrics and KPIs for different criteria evaluations
  $R_{metrics} \leftarrow Process(Rs\ ,\ GTs\ ,\ Metrics)$ ▷ Calculate the result of metrics $R_{metrics}$
  return $Visualize(R_{metrics})$, $Analyze(R_{metrics})$ ▷ Visualize and analyze the result of metrics $R_{metrics}$
end procedure

3.3 Criteria, metrics, and KPI

As explained in layer 3 of the previous section, one of the most important tasks in evaluation is to choose the right metrics for the tasks and factors to be evaluated. In fact, before evaluating, it is necessary to define the tasks to be evaluated (these could be classification or tracking tasks, for example) and to put them into perspective with the factors or risks selected (for example, are we going to simulate the impact of road conditions) and the scenarios selected. This was normally the case if the steps in the protocol were followed. In the remainder of this chapter, we will return to the selection of metrics and validation criteria.

3.3.1 Needed references and ground truth

Most AI metrics are in fact mathematical tools for comparing the hypotheses provided by the system under study with references that may either come from the ground truth (which will have to be qualified), or come from human annotations. Data labels (metadata or human annotations) must be qualified (presence of an annotation guide, inter- and intra-annotator comparisons, etc.), which may involve a qualification process for annotators in the case of human annotations. The processes for qualifying reference data are described in deliverable 1.6 of the PRISSMA project and should therefore be referred to as follows. Obtaining reference data and guaranteeing its quality is therefore an absolute necessity before using a metric.

Moreover, creating or obtaining references can be a real problem. Typically, references (not derived from a physical sensor) for an autonomous vehicle are binary (accident/non-accident, risk threshold crossed, etc.), and this is what is typically used in Euro NCAP-style tests or regulatory tests. However, even if this can be relevant from a purely safety point of view without AI and for a vehicle with an SAE level of less than 3 , it becomes problematic to judge an on-board AI solely on the basis of these binary criteria, for example how to judge the relevance of a trajectory taken in relation to the plurality of other possible trajectories with regard to these arbitrary criteria. Some approaches, such as [135], try to create the best possible trajectory for a given situation, taking into account safety criteria as well as vehicle comfort and dynamics. It will therefore be all the more important not to rely solely on the pass/fail used for the actual homologation scenarios, as the automated vehicle cannot rely on a driver to avoid an accident, unlike a vehicle at SAE level 2 or lower.

Another concern is that (if the annotation has to be done by experts) the quantity can be very large in the context of a campaign using simulation or recordings from driving databases on open roads, and this brings us back to the problem of cost.

The other major issue in simulation-based validation is the representativeness of simulated test data. Particular attention needs to be paid to this point if we are using simulated data (to study bias with respect to real data or ground truths, etc.) or transformed data (base amplification). The only way to guarantee this representativeness is to have ground truth data to calibrate our models, which gives a second function to the reference data, but this is a problem linked to task 2.5 (see chapter 2 of this document).

3.3.2 Presentation of the levels of evaluation and validation

Overall, there are two possible approaches to the evaluation of AI-based automotive systems. One is where the system is evaluated in its entirety, which will bring out more global metrics on safety (metrics often based on the notions of TTX i.e. Time to X) or global KPIs such as those proposed by the Euro NCAP protocols (success rate...) or the MAVEN or TRANSAID projects. The other approach is to focus directly on the AI component, which is much easier in simulation (much more difficult on the open road), and comes closer to more conventional AI evaluations based above all on the specific tasks to be evaluated.

3.3.3 KPI and Risk assessment for system level

3.3.3.1 System evaluation level

In order to evaluate the high-level quality of AI-powered system in ADS, such as a visual perception system, it is necessary to implement a full mobility service and propose relevant and representative scenarios involving an exhaustive set of conditions/configurations/situations allowing for quantification of the performances and the quality of the service. If an accident is detected during the simulation, accident-related metrics should not be overlooked (collision speed, number of vehicles involved, presence of over-accidents, comparison with real accident databases to characterise the type of injury severity etc.). In [136], an interesting study about injury severity is studied and assess with an ANN method. Moreover, in [137], the authors explore driver injury severity in Single-Vehicle Crashes under Foggy Weather and Clear Weather. In [138], Injury Severity for Hazard and Risk Analyses is addressed. This study is

interesting because it takes into account calculation of ISO 26262 S-parameter Values from Real-World Crash Data. In [139] Pre-crash Injury Risk (IR) assessment is proposed for guiding efforts toward active vehicle safety. This work aims to conduct crash severity assessment using pre-crash information and establish the intrinsic mechanism of IR with proper interpretation methods. The impulse–momentum theory is used to propose novel a priori formulations of several severity indicators, including velocity change (ΔV), energy equivalent speed (EES), crash momentum index (CMI), and crash severity index (CSI). Six IR models based on different machine learning methods were applied to a fusion dataset containing 24,082 vehicle-level samples. Prediction results in this work indicate that the pre-crash indicators (PCIs) are more influential than the commonly used basic crash information because the average accuracy of six models can be improved by 14.35% after utilising PCIs. Furthermore, the features' importance and their marginal effects are interpreted based on parameter estimation, Shapley additive explanation value, and partial dependence. The ΔV, EES, and CMI are identified as the determinant indicators of the potential IR, and their partial distributions are significantly influenced by the crash type and impact position. In [25], a set of Injury Probability Curves (IPC) as Functions of Delta V are provided based of US dataset for different road situations (crash data from the 2010–2015 National Automotive Sampling System Crash-worthiness Data System). This analysis established the univariant occupant injury risk predictive models (or injury probability curves) through logistic regression using delta V as the predictor. Delta V is a measure of crash severity. Occupant injury severity was based on the Maximum Abbreviated Injury Scale (MAIS). The injury probability curves were developed for three crash modes: all crashes, frontal crashes, and rear-end crashes (see 117).

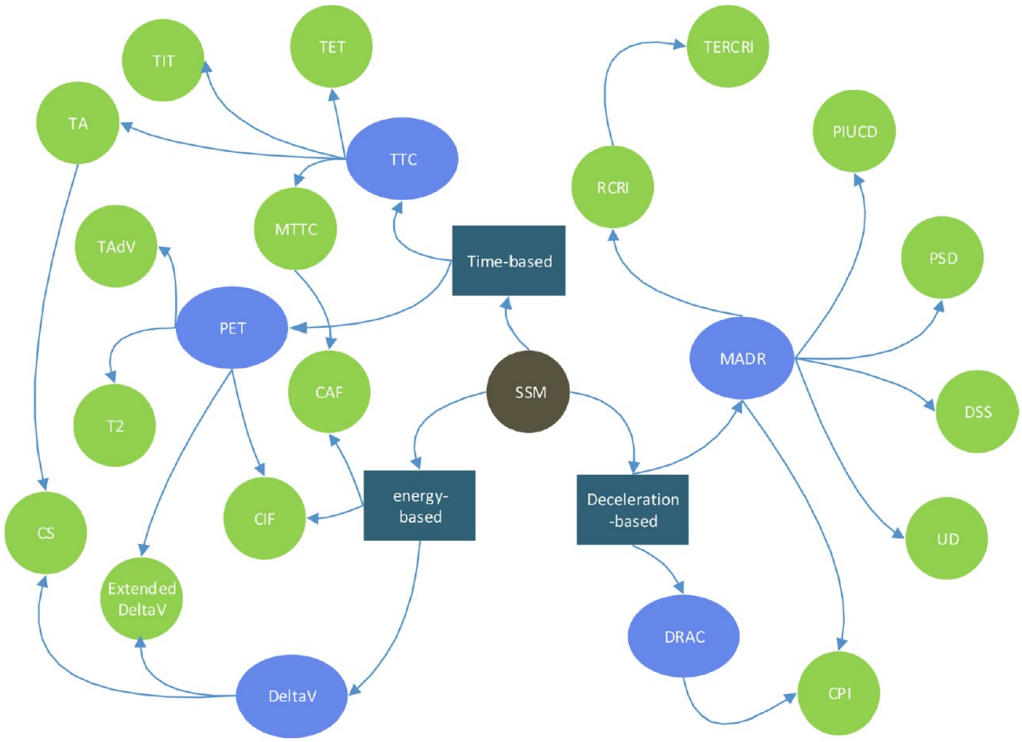


Figure 116: Surrogate safety measures for STRA and CAV [24]

The system evaluation level metrics (a case of visual perception system) can refer to a set of specific Key Performance Indicators (KPIs):

- **Risk specific**: Longitudinal and lateral distance, Time to collision (TTC), Time Exposed Time-to-Collision (TET), Deceleration Rate to Avoid a Crash (DRAC), etc.
- **Task (detection/tracking) specific**: Success rate, Loss, Distance, etc.
- **Time specific**: Frequency, Time to detect/track, False alarm frequency.
- **Severity specific**: Injury severity in function of the vehicle, the weather, the type of manoeuvres, the relative speed and EES. In [139], 6 Pre-crash Injury Risk (IR) models are proposed using different machine learning methods. These models include the use of velocity change (ΔV), EES, CMI, and CSI.

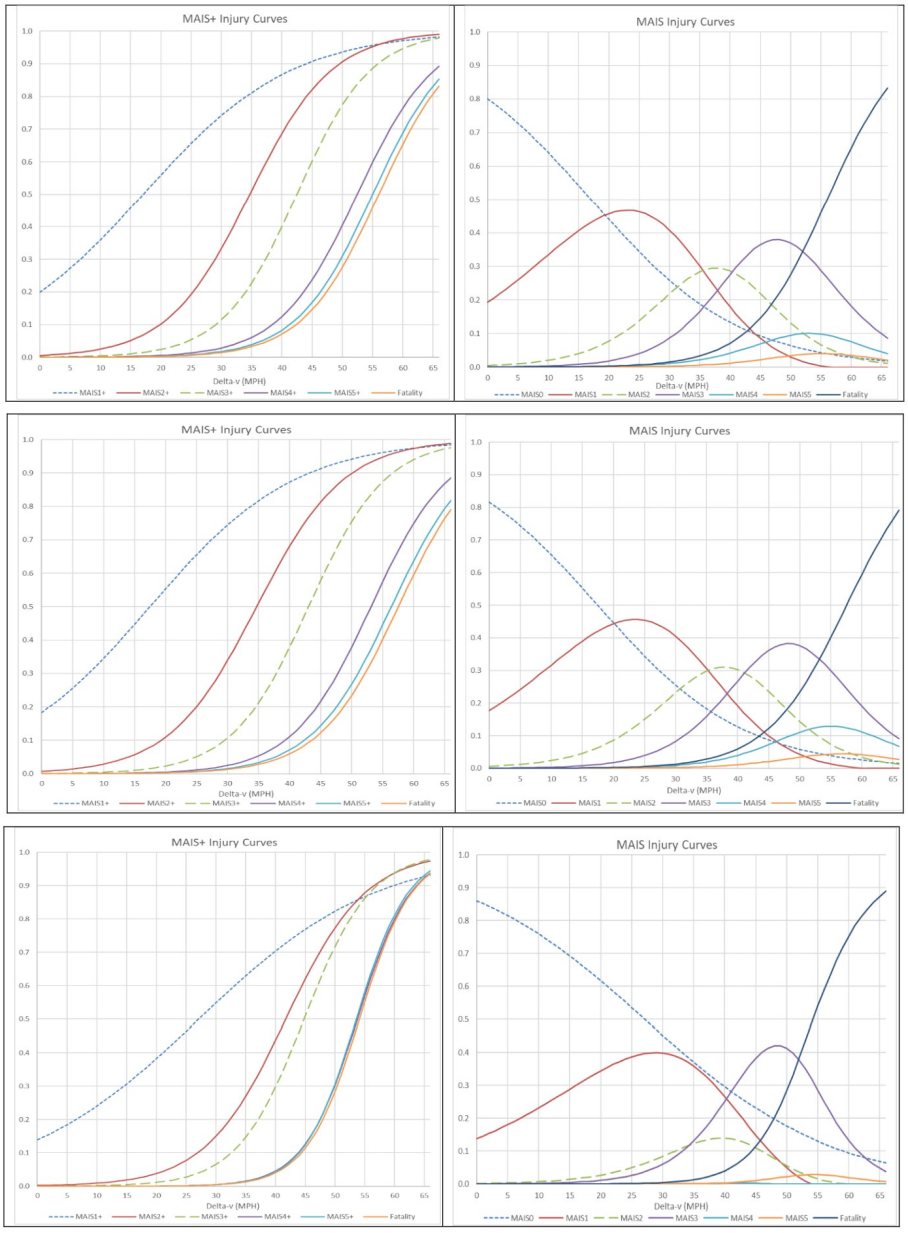


Figure 117: MAIS+(05/08) and MAIS(05/08) Injury Probability Curves by Crash Modes (top:all crashes; middle:frontal crashes; bottom:rear-end crashes)[25]

An exhaustive list of these criteria can be found in the PRISSMA WP2 state-of-the-art documents and PRISSMA WP1 deliverable about KPIs and metrics. An overview of these KPIs dedicated to risk calculation is provided in the figure 100 et figure 101. [24] also provides an overview of the Surrogate Safety Measures with their link to time, energy, and acceleration/deceleration aspects (see figure 116).

Indicator	Computation	Notations
Temporal proximal		
Time-to-Collision (TTC) [8] or Time-Measured-to-Collision (TMTC) [14]	$TTC_i(t) = \frac{X_{i-1}(t) - X_i(t) - l_i}{V_i(t) - V_{i-1}(t)}$	V: Vehicle speed X: Vehicle position l: Subject vehicle's length $X_{i-1}(t) - X_i(t)$: Relative distance $V_i(t) - V_{i-1}(t)$: Relative speed
Time Exposed Time-to-Collision (TET) [10]	$TET_i^* = \sum_{t=0}^{T} (_i(t) \cdot _{sc})$	τ_{sc}: Small time step δ: Switching variable (0 or 1)
Time Integrated Time-to-Collision (TIT) [10]	$TIT_i^* = \sum_{t=0}^{T} [TTC^* - TTC_i(t)] \cdot _{sc}$	$\forall\ 0 \leq TTC_i(t) \leq TTC^*$ TTC*: TTC value below the threshold value. $TTC_i(t)$: TTC subject vehicle
Modified Time-to-Collision (MTTC) [22–24]	$MTTC = \frac{-\Delta V \pm \sqrt{V^2 + 2\Delta a D}}{\Delta a}$	V_F: Following vehicle's speed (m/s); V_L: Leading vehicle's speed (m/s); a_F: Vehicle's acceleration (m/s^2); a_L: Leading vehicle's acceleration (m/s^2); ΔV: Relative speed (m/s) Δa: Relative Acceleration (m/s^2) D: Initial relative space gap (m);
Crash Index (CI) [22,24]	$CI = \frac{(V_F + a_F \cdot MTTC)^2 - (V_L + a_L \cdot MTTC)^2}{2} \times \frac{1}{MTTC}$	
Headway (H) [14,32]	$H = t_i - t_{i-1}$	t_i: Time (vehicle i passes a certain location) t_{i-1}: Time (vehicle ahead of vehicle i passes the same location).
Time-to-Accident (TA) [26,87]	$TA = 1.5 \times \frac{V_i}{16.7 \times \exp(-0.0306 \times 0.5 V_m)}$	V_i: initial speed V_m: Mean speed.
Post-Encroachment Time (PET) [36]	$PET = t_2 - t_1$	t_2: Coming time at conflict point t_1: Leaving time of conflict point.
Non-temporal proximal indicators		
Potential Index for Collision with Urgent Deceleration (PICUD) [40,41]	$PICUD(m) = \frac{V_1^2 - V_2^2}{2\alpha} + S_0 - V_2 \Delta t$	V_1, V_2: Velocity of leading car 1 and following car 2, respectively S_0: Distance between car 1 and 2 Δt: Driver's reaction time 1 α: Deceleration rate to stop
Proportion of Stopping Distance (PSD) [36,43]	$PSD = \frac{RD}{MSD}$	RD: Remaining distance to the potential point of collision (m) MSD: Minimum acceptable stopping distance (m).

Figure 118: Computational equations for major surrogate indicators [26]

Indicator	Computation	Notations
Unsafe Density (UD) [46,88]	$unsafety = \Delta S \cdot S \cdot R_b$ $Unsafety\ Density = \frac{\sum_{s=1}^{S_t} \sum_{v=1}^{V_t} unsafety_{v,s} \cdot d}{T \cdot L}$	S: Speed of the follower vehicle ΔS: Difference of speed at collision time R_b: unsafe parameter V_t: nb of vehicles in the link S_t: nb of simulation steps within aggregation period d: simulation step duration [s] T: aggregation period duration [s] L: section length [m]
Difference of Space distance and Stopping distance (DSS) [45]	$DSS = (\frac{v_1^2}{2\mu g} + d_2) - (v^2 \Delta t + \frac{v_1^2}{2\mu g})$	S: Space distance (m) Stop: Stop distance (m) v_1: Velocity of following vehicle (m/s); v_2: Velocity of leading vehicle (m/s) μ: Friction coefficient g: Gravity acceleration (m/s^2) d_2: Distance between leading vehicle and following vehicle (m) Δt: Reaction time
Deceleration Rate to Avoid the Crash (DRAC) [43,47,51]	$DRAC_{FV,t+1}^{REAR} = \frac{(V_{FV,t} - V_{SV,t})^2}{(X_{SV,t} - X_{FV,t}) - l_{SV,t}}$	X: Position of the vehicle (m) L = vehicle length (m) V = velocity (m/s)
Crash Potential Index (CPI) [42,51]	$CPI_i = \frac{\sum_{t=ti_i}^{tf_i} P(MADR^{(a_1, a_2, \ldots, a_n)} \leq DRAC_{i,t}) \cdot \Delta t \cdot b}{T_i}$	$DRAC_{i,t}$: Deceleration rate to avoid the crash (m/s^2) $MADR^{(a_1, a_2, \ldots, a_n)}$: Random variable following normal distribution for a given set of traffic and environmental attributes (a_1, a_2, ..., a_n) (m/s^2) ti_i: Initial simulated time interval for vehicle *i* tf_i: Final simulated time interval for vehicle *i* Δt: Simulation time interval T_i: Total travel time for vehicle *i* b: A binary state variable, 1 if a vehicle interaction exists and 0 otherwise.
Criticality Index Function (CIF) [54]	Criticality Index = V^2 / TTC (Time to Conflict)	V: Velocity (m/s)

Figure 119: Computational equations for major surrogate indicators [26]

Indicator	Definition	Limitations	Advantages	Suitability for collision type
Time-to-Collision (TTC) or Time-Measured-to-Collision (TMTC)	The time until a collision between the vehicles would occur if they continued on their present course at their present rates.	Assume consecutive vehicles will keep constant speeds; Ignore many potential conflicts due to acceleration or deceleration discrepancies; Can provide the magnitude of crashes but not their severity; Collision course must exists, TTC index cannot be estimated in a finite number where the leading vehicle is faster than following.	TTC is far more frequently used in practice than PET or TA due to theoretical issues; TTC was more informative than PET; Many automobile collision avoidance systems or driver assistance systems have used TTC as an important warning criterion; Applicable for Work Zone safety analysis, Applicable in post-processor such as SSAM	Rear-end, turning/weaving, hit objects/parked vehicle, crossing and hit pedestrian.
Time Exposed Time-to-Collision (TET)	Summation of all moments (over the considered time period) that a driver approaches a front vehicle with a TTC-value below the threshold value TTC*.[a]	Does not provide the variation severity levels of different TTC values below the threshold value; If TTC-value is lower than the threshold, does not affect the TET indicator value; Highly data-intensive and attainable only in a simulation environment.	Can be calculated separately for per user class, Can be applied in the comparison of a do-nothing case with an adapted situation; Suited for application in microscopic simulation studies of traffic; Easy to include small TTC value due to including of time-dependent TTC values of all subjects.	Same as TTC.
Time Integrated Time-to-Collision (TIT)	Integral of the TTC-profile during the time it is below the threshold	Difficult to interpret its meaning for complexity to determine; Not preferable to use in comparative studies in which simulation tools are applied to generate trajectories; Benefits is small due to the uncertainties in driver behaviour.	Level of safety of collision can be derived; Can be applied in the comparison of a do-nothing case with an adapted situation; suitable for microscopic simulation studies of traffic; Easy to include small TTC value due to including of time-dependent TTC values of all subjects.	Same as TTC.
Modified Time-to-Collision (MTTC)	Modified models which considered all of the potential longitudinal conflict scenarios due to acceleration or deceleration discrepancies.	Obtaining the field speed of both users and the distance gap in an evolution process is difficult and has to rely on other approaches; Not fit for lane changing or head-on collision; Does not reflect the severity of collision	More advance than TTC; Consider driving discrepancies; Severity of the collision could be weighted using CI indicators.	Vehicle-vehicle crash same as TTC
Crash Index (CI)	Influence of speed on kinetic energy involved in collisions.	Describes only the safety information about two vehicles at a certain time and place. Not fit for lane changing or head-on collision, has to rely on other approaches for data collection	Reflect the severity of a potential crash; describe the influence of speed on kinetic energy involved in collisions; Consider the elapsed time before the conflict occurred; Severity and the likelihood of a potential conflict could be interpreted.	Same as MTTC.
Headway (H)	The elapsed time between the front of the lead vehicle passing a point on the roadway and the front of the following vehicle passing the same point	Mainly applicable in conflicts related to follow up manoeuvre i.e. for read-end collision in lane based traffic environment; Do not take into account conflicts due to lateral movement particularly during lane changing or overtaking.	Easy to measure; Level of safety could be distinguished.	Rear-end mainly, other such as turning and hit objects/parked vehicle.
Time-to-Accident (TA)	Time-to-Accident (TA) is the time that remains to an accident from the moment that one of the road users starts an evasive action if they had continued with unchanged speed and directions	Often criticized for relying heavily on the subjective judgement of speed and distance. Mainly rely on the evasive action. Other same as TTC	Widely used; Easy to measure; Can be done by both manually or by Video analysis, Couple of manuals have been developed in different countries.	Same as TTC.
Post-Encroachment Time (PET)	The time between the moment that a road user (vehicle) leaves the area of potential collision and the other road user arrives collision area.	Only useful in the case of transversal (i.e. crossing) trajectories (Right angle collision); Cannot reflect changes with the dynamics of safety-critical events over a larger area; Levels of severity as well as impact of a conflict are not taken into account;	PET is more appropriate than TTC for intersecting conflicts; PET can be easily extracted; PETs can be easily estimated using photometric analysis in video or simulated environment; PET represents the driver behaviours.	Mainly for right angle or crossing crash, hit pedestrian. Merging/diverging, head on (to a certain extent).

[a] *Threshold value.

Figure 120: Summary of temporal proximal based indicators. [26]

Indicator	Definition	Limitations	Advantages	Suitability for collision type
Potential Index for Collision with Urgent Deceleration (PICUD)	Distance between the two vehicles considered when they completely stop.	Mainly applicable in lane changing condition when leading vehicle apply emergency break; Threshold value yet to be sated up; Do not take into account lateral conflicts.	PICUD is more suitable than TTC for evaluating the danger of collision of the consecutive vehicles with similar speeds. PICUD might detect the change in traffic condition and conflicts more sensitively than TTC.	Same as TTC.
Proportion of Stopping Distance (PSD)	Ratio between the remaining distance to the potential point of collision and the minimum acceptable stopping distance.	Based on evasive actions; PSD provide higher percentage of vehicles interaction and time exposure to conflict than TTC and DRAC, hence less focus on specific safety problem.	Single vehicle conflict with fixed or unfixed objects can be evaluated; Easy for observation and calculation.	Hit object (on road or road side), overturning.
Margin to Collision (MTC)	Ratio of the summation of the inter-vehicular distance and the stopping distance of the preceding vehicle divided by the stopping distance the following vehicle.	Same as Stopping distance. In addition, it does not consider a response delay of the following vehicle. A non-dimensional parameter.	Same as Stopping distance. It also provides the possibility of conflict when the preceding and following vehicle at the same time decelerate abruptly.	Same as Stopping distance.
Unsafe Density (UD)	Level of "unsafe" in the relation between two consecutive vehicles on the road for a determined simulation step.	The value of this parameter doesn't really have a sense in itself and must be used only for comparison purposes; fit for only rear-end collision analysis (identical trajectory); Applicable only for lane based traffic situation.	Gives more accurate information than typical micro-simulation outputs; Comparative study between link can be done.	Rear-end, merging and diverging or lane changing.
Difference of Space distance and Stopping distance (DSS)	DSS is defined by the difference of the space and stopping distance.	Provide information on the number of unsafe vehicle but cannot consider the degree of danger as well as the duration.	The calculation formula and dangerous threshold value are simple and clear.	Rear-end, hit object and turning.
Time Integrated DSS (TIDSS)	Total value of the time integrated value gap between DSS and the dangerous threshold value.	Mainly suitable for rear-end conflict.	Consider the degree and the duration of danger.	Same as DSS

Figure 121: Summary of distance based proximal indicators. [26]

Indicator	Definition	Limitations	Advantages	Suitability for collision type
Deceleration Rate to Avoid the Crash (DRAC)	Differential speed between a following/response vehicle and its corresponding subject/lead vehicle (SV) divided by their closing time.	Fails to accurately identify the potential traffic conflict situation; Not suitable for lateral movement.	Explicitly considers the role of differential speeds and decelerations in traffic flow.	Rear-end, Hit object/parked vehicle, Hit pedestrian, Merging and diverging manoeuvres.
Crash Potential Index (CPI)	Probability that a given vehicle DRAC exceeds its maximum available deceleration rate (MADR) during a given time interval.	Not suitable for lateral movement; mainly applicable at intersection.	Address some of the issues found in DRAC like vehicle braking capability for prevailing road and traffic conditions.	Same as DRAC.
Criticality Index Function (CIF)	Multiplication of vehicle speed with the required deceleration	Like TTC, it considers constant speed of consecutive vehicle; Further evaluation is needed using additional field date for validation.	Chance of occurrence and severity could be measured.	Turning accident, Right angle

Figure 122: Summary of deceleration based indicators [26]

3.3.3.2 Component evaluation level

This level of evaluation focuses on the performance of individual algorithms or functions within the AI-powered system. AI algorithms and specially the machine learning models are evaluated using metrics to analyse their performance on top of the loss function which is used to optimize the model during the training phase. Hence, different metrics can be used to evaluate both: the generalization performances to unseen data during training, and the performances stability and robustness toward data and/or environment changes.

AI metrics are strongly correlated with the tasks performed by the AI. These tasks include classification/detection, tracking, etc. The advantage of staying at the AI component level for evaluation is that the metrics are well defined in the literature and it's just a matter of staying at the state-of-the-art level when choosing one. The WP2 state-of-the-art deliverables (D2.1, D2.2 and D2.3) describe the major families of usable metrics (refer to this document for an exhaustive list).

Table 7: Metrics cross different functionalities

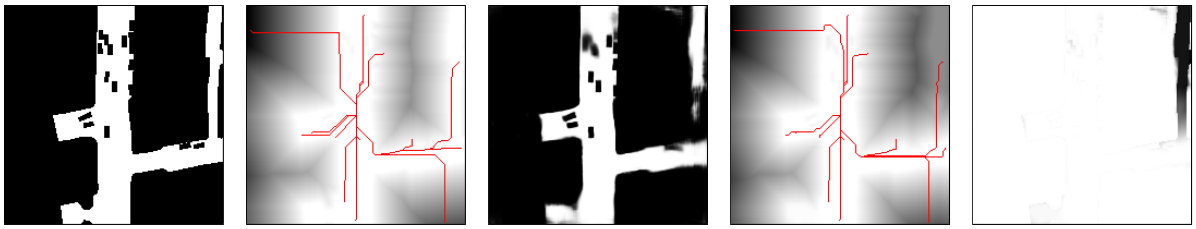
Perception Function	Explanation	Metrics
Detection	Identifying and localizing objects within an image or video frame using bounding boxes	False Positive Rate (FPR), False Negative Rate (FNR), True Negative Rate (TNR), True Detection Rate (TDR), Accuracy, Precision, Recall, F-Measure, Receiver Operating Characteristic (ROC Curve), Detection Error Trade-off Curve (DET Curve), Precision-Recall Curve (PR Curve), Average-Precision (AP), mean Average Precision (MAP), etc.
Segmentation	Partitioning an image or video frame into regions and assigning semantic labels to each pixel or region	Pixel Accuracy (PA), Class Pixel Accuracy (CPA), mean Pixel Accuracy (mPA), IoU, mean Intersection over Union (mIoU), etc.
Tracking	Following the movement and preserving the identity of an object or multiple objects over time in a video sequence	Object Tracking Time delay, Identification switch(IDSW), Multiple Object Tracking Accuracy (MOTA) [140], Multiple Object Tracking Precision (MOTP) [140], Higher Order Tracking Accuracy (HOTA) [30], etc.

The metrics are typically related to the functionalities of the system, as the metrics of the visual perception system shown in Table 7.

An other classical family of metrics is the qualitative metrics used strongly for the regression AI algorithms. Regression methods cover a wide range of algorithm families (e.g. SVMs, or certain neural networks) and deal with predicting a target value (that can be applied to object recognition or trajectory prediction, for example) using independent variables. The regression metrics are based on point distance computation methods. These metrics contain fundamental operations and the absolute or squared value of their result can be used to provide performance values. These metrics are part of the so-called Quantitative metrics presented in the state of the art. The most commonly used metrics are the RMSE and MAE. Most applications continue to incorporate traditional metrics as primary indicators of adequacy of the regression models. This seems to stem from our familiarity with these metrics, as opposed to others despite the limitations identified with these traditional metrics (for example, the RMSE can give similar results for a negligible constant small error and a punctual extreme error). This is why we recommend approaches based on multi-factor verification approaches, as proposed by [141, 142, 143], which couple traditional metrics with innovative approaches to overcome traditional limitations. For example, the Actual/Predicted correlation, that corresponds to the linear correlation (in the statistical sense) between the actual values and the predicted values for every value considered in a set, must be used with another metric, RMSE, MAE or Max Error.

Navigation-based evaluation metric for probabilistic occupancy grids

Probabilistic occupancy grids (OG) are widely used for autonomous driving to represent the environment surrounding the ego-vehicle. They are discrete and probabilistic representations where the value of each cell corresponds to the likelihood of being occupied, free, or still unknown. More broadly, OGs can also include semantic information, i.e. the probability of occupied cells to belong to a specific semantic class (e.g. vehicle, pedestrian, etc.). In the context of autonomous driving, OGs are generated by a perception system based on raw sensor data and used for navigation tasks such as Automated Driving Systems (ADS) and collision avoidance. Despite their importance and broad use in autonomous driving, most existing approaches to evaluate the reliability of probabilistic OGs are based on general-purpose metrics derived from the computer vision literature. In [27], the authors proposed a new metric dedicated to probabilistic OGs that evaluates the similarity of two OGs: a Ground Truth (GT) of the environment and the inference of the environment made by the perception system. This metric is specifically designed to assess the similarity between OGs by considering the behaviour of an ego-vehicle



(a) Ground Truth (GT) (b) GT cost grid (c) Inference (d) Inference cost grid (e) Distortion grid

Figure 123: Evaluation process of the metric from [27]. 123(a) and 123(b) are the Ground Truth and its cost grid, 123(c) and 123(d) are the inference and its cost grid. Examples of paths are drawn in red on both cost grid. The resulting distortion grid 123(e) is the pixel-wise absolute error between both cost grids, it is also weighed by the disjunctive probability of free occupancy on the GT or the inference. The metric score is obtained by doing an MSE pooling the distortion grid, the example scene metric score is $41,47$. The driving scene (ground truth and sensor data used to generate the inference) is taken from Nuscenes dataset, the AV is located at the centre of the grids.

navigating through the grids. The main postulate being that if a navigation algorithm generates similar trajectories using two OGs, then the two are alike for navigation purposes. The metric is computed using the following steps:

1. For each OG we simulate a navigation algorithm to generate a shortest-path tree, composed of all the shortest paths from the AV position to every cell of the OG (some example of paths are drawn in red in Figure 123(b) and 123(d)).

2. Both OGs are transformed into cost grids using the cumulative costs of the paths from their respective shortest-path trees (e.g. Figure 123(b) and Figure 123(d)).

3. An intermediate distortion grid is computed by performing the cell-wise absolute error between both cost grids (e.g. Figure 123(e)).

4. The metric is evaluated by computing the MSE of the distortion grid weighed by the disjunctive probability of free occupancy on both grids (i.e. probability of a cell to be free on either of the grids).

Simulating and comparing the navigation behaviour instead of doing cell-wise comparisons directly on the OGs gives this metric relevant properties. It is able to evaluate topological errors, it measures how occupancy errors on the inference changes the cost of the paths and how this affects the global topology of the OG. It puts emphasis on cells that are most crossed by paths since their occupancy is incorporated in more costs. In other words, these cells are topologically more important for navigation (e.g. areas closer to ego vehicle or bottlenecks). Furthermore, this metric is well fitted to evaluate uncertainty: paths tend to avoid uncertainty whenever possible or cross it otherwise, but in both cases, the navigation cost is increased.

3.3.4 Evaluation under complex scenarios

This level of evaluation involves testing the performance of an AI-powered system under various challenging conditions, such as adverse weather, low lighting, and unexpected obstacles. These complex scenarios can be difficult to replicate in real-world testing, which makes simulation tools and virtual environments more essential. The use of simulation allows for the creation of complex scenarios that can be repeatedly tested, analyzed, and modified to evaluate,

analyze, and improve the performance of the system. One example we used in the experiment is Pro-SiVIC^{TM}, which offers a range of adverse scenarios, as shown in Fig. 124.

Figure 124: Adverse scenarios from Pro-SiVIC^{TM}

The related criteria can vary depending on the specific application and system requirements. However, some common criteria for this level of evaluation include:

- **Robustness**: this criteria evaluates the ability of AI functionality to maintain compliance with expected requirements in the presence of input data within the intended use domain and the ability of the system to perform the intended function in the presence of abnormal or unknown inputs. It is usually reflected in various performance metrics, such as accuracy, precision, recall, and F1-score, etc. Robustness can be measured by analysing these performances in different scenarios and under different conditions, and also by assessing their ability to maintain performance levels over time. Robustness is the ability of a system to be little or not impacted by occasional errors.

- **Reliability**: This criteria refers to subsets of the notions of resilience and robustness notions and evaluates the system's ability to make reliable decisions in emergency situations or other adverse situations. In fact reliability characterises the validity of the operation of a sensor or a system according to its operating ranges (nominal operating range: ODD of the sensor or system). For a measurement, reliability is the probability that the measured parameter is measured correctly according to the nominal operation of the meter and is not an outlier. For a system, reliability is the probability that the measured behaviour/state is well measured according to the nominal operation of the system and is not an aberrant behaviour/state.

3.4 Optimum Evaluation Domain

This sub-section initiates a discussion on the optimum perimeter for evaluation of AI-based ARTS. The work herein shows why the notion of optimality is not straightforward and that the validation and certification procedures rather encompass the enlargement of the search spaces in order to ensure proper testing. This discussion is then focused on building an optimum evaluation domain and the process of sweeping this domain for testing in an optimal way is out of scope.

On the context of optimum evaluation domain and functional combinatorial explosion

Firstly, an evaluation space for ARTS (without yet addressing optimality) needs to be one that has been tested in the validation phase though the parameter coverage rate. This coverage rate is different according to whether the tests are performed by system's providers or by evaluators. In this sense, the level of detail depends on the party performing evaluation. However, for safety-related critical functions and specifically the Driving Dynamic Tasks, exhaustivity of tests is expected. For a conventional system, combination of inputs and expected results for outputs are known and the system can be validated by itself. For autonomous driving, the number and combination of input increase exponentially and are dependent from system environment; that is why scenario are required to describe testing environment. The scenario methodology presented in deliverable L2.6 [144] allows guaranteeing exhaustivity in the context of use of the system's functions. This context is constrained by the system's operation pathway, its ODD and its OEDR. The set of scenarios are then coherent with respect to this evaluation space.

On the context of the inclusion of AI specificities

In order to consider AI specificities, the test space is then enlarged (without touching on the notions of ODD/OD/OEDR of the system) when taking into account the hazards in the environment that can potentially introduce disturbances in the behaviour of the AI bricks. Otherwise stated, the functional complexity described in the previous paragraph is now combined with the complexity introduced by environmental hazards. To this day, 2 methods allow addressing this enlargement of the test space:

1. An approach scanning the evaluation domain including situations considered as outliers and spectators that are not taken into account in logical scenarios.
2. The SOTIF approach which identifies the triggering conditions and functional insufficiencies known by expert feedback that can potentially impact the system.

On the objective of building an optimum evaluation domain

All previous considerations only enlarge the test space conversely to narrowing it down to an optimum evaluation domain. Pre-requirements from a certification stand-point in a chronological manner:

1. The system provider will cover in an exhaustive manner the all scenarios stemming from parameter variability for critical functions.
2. The methodology employed by the systems provider must be properly documented for audit purposes.
3. The audit on the methodology performed by the systems provider allows focusing certification efforts in an optimum manner, this is:
    - Qualification of the digital models and test environments

- Depending on the scenario types (For more details on recommendations on audit efforts depending on scenario types, refer to [145]), this is nominal, nominal on degraded conditions, or critical, the type and therefore effort dedicated to audit must be adapted:
  - Generally speaking, *nominal scenarios* can be easily audited through light sampling for confirmation with the documentation,
  - audits on *nominal scenarios in degraded conditions* focus on the DDT fallback and the frontier between nominal and nominal in degraded conditions
  - the *critical scenarios* testing procedures are dependent on the accident being avoidable or not and the expected system's response.

3.5 Testing of POCs

In the remainder of this section, we will apply the PRISSMA test protocol for simulation tasks described in task 3.2.2 to 4 different tests (referred to hereafter as POC). This will show how, in practical terms, we can adapt this generic protocol to different situations and types of evaluation. Overall, five POCs using simulation exclusively or partially were carried out in the PRISSMA project, the fifth POC straddling WP3 between CEREMA and LNE was evaluated within the framework of WP3 exclusively (the simulation part of this POC nevertheless declined the approach proposed by WP2). The application to this POC can be found in deliverable 3.3 of the project.

3.5.1 POC 1

3.5.1.1 System and Environment

In our experimental setup, we use a visual perception system based on a monocular camera. This system is implemented as a module within the RTMaps^{TM} framework and is an extension of the driving system developed in the H2020 Trustonomy project [146]. The camera-based perception system is specifically designed to provide essential functionalities such as object identification, localization, and tracking.

For the purpose of our simulations, the module-based ADS operates within the Pro-SiVIC^{TM} rendering environment. To replicate real-world scenarios, we utilize the digital twin of the Satory test track within Pro-SiVIC^{TM}. This virtual representation of the test track has undergone validation using an existing perception system equipped with a set of various sensor types and technologies.

3.5.1.2 Scenarios generation

To evaluate our framework's genericity , we developed a full bus stop service in RTMaps and the full environment with vehicles, sensors, infrastructures, and building in Pro-SiVIC^{TM}. This set of models and modules is used for the evaluation of a visual perception system equipped with different AI models.

As shown in Fig. 125, the service and the situations that could be encountered are modelled with a set of 6 scenes. The continuous assembly of these scenes along a 3.4 km trajectory on the Satory's test track allows for the construction of a tree of possible scenarios. The vehicle trajectory includes 4 or 5 types of bus stations. The simulated environment comprises different

types of bends and straight lanes. The ego vehicle is restricted to driving on its ego-lane, while other traffic objects can perform cut-in and cut-out manoeuvres on different lanes.

The ODD is the following: Ego vehicle travels on the right lane of the predefined road at a maximum speed of 20 km/h, while maintaining a safe following distance and performing car-following manoeuvres when a front vehicle is detected. The vehicle's primary objective is to ensure the safety of passengers and road users by avoiding collisions and adapting its speed to traffic conditions, visibility caused by different weather conditions, and the environment. Additionally, the vehicle stops at the bus station to allow passengers to board and alight. After the stop, the vehicle utilizes dedicated longitudinal and lateral profiles to return to the centre of the right lane and resume driving.

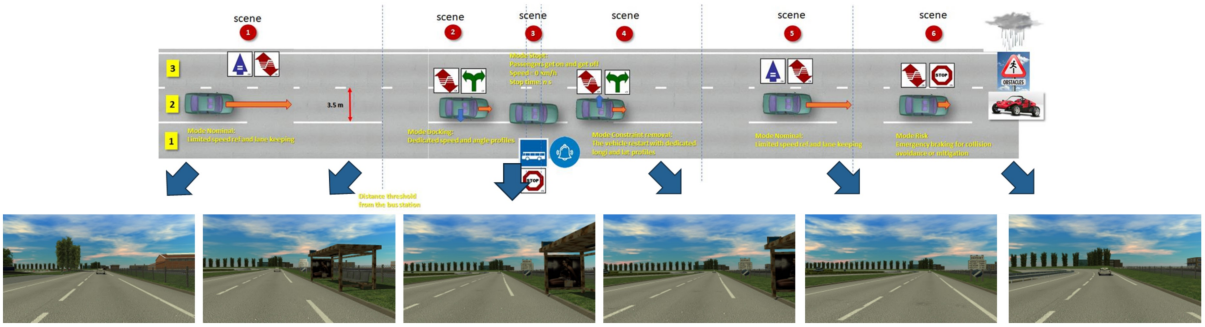


Figure 125: POC1: 6 scenes defined for the docking at a bus station

3.5.1.3 DataSet and data format

The dataset was built along the digital twin (as shown in the figure) of 3.4 km trajectory on the Satory test track (the road information was contained inside the corresponding trk file), with continuous assembly of different driving scenes. Seven filters was defined on the camera (inside Pro-SiVIC) in order to simulate the driving scenarios under different weather conditions. The dataset includes various types of simulated vehicles (totally 13 vehicles), such as car and trucks, navigating the Satory test track. The velocity is the range of 40 to 45 kmh. As shown in the figures, different camera perspectives were incorporated, including front and rear view of the ego vehicle, as well as close and distant perspectives.

The dataset comprises 7 weather conditions (as depicted in the figure below). Each condition's corresponding data has been stored in separate sub-datasets, and the overall dataset structure resembles that of the left folder.

The left folder illustrates the structure of the sub-dataset, which consists of two categories: "detection" and "segmentation". Each category contains 10,000 raw images collected from Pro-SiVIC. Additionally, corresponding label files are stored alongside the images to provide ground truth information, facilitating the evaluation of different functionalities.

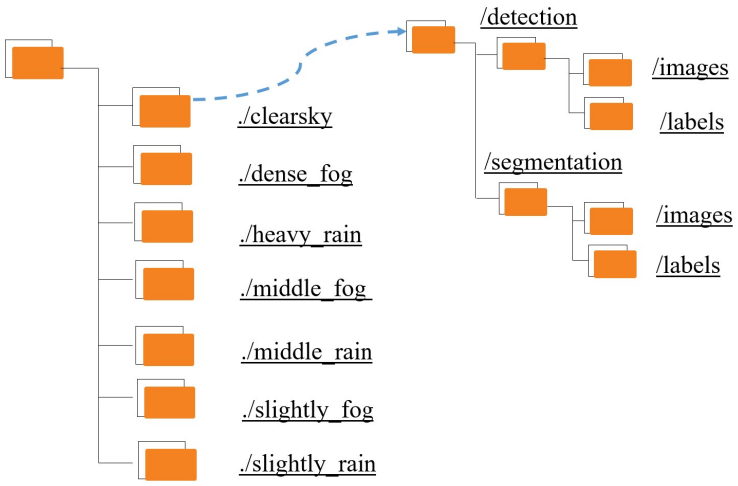


Figure 126: Structure of the dataset generated in BuSAS

Generation: Raw images and mask/reference images are generated automatically from Pro-SiVIC, representing the view of the real world and corresponding reference (here are vehicles, which could also contain the driving lane or other traffic objects) respectively.

- Raw images: JPG Files, Dimensions: 640 x 480, Resolution 96 dpi (both Horizontal and Vertical), Bit depth 24
- Reference images: JPG Files, Dimensions: 640 x 480, Resolution 96 dpi (both Horizontal and Vertical), Bit depth 24

Annotation: The information on ground truth will be extracted and annotated from the mask/reference image we collected, the relative position of every vehicle appearing in mask/reference images will be annotated within the real image, and also the corresponding boundary shape will be marked.

Representation: In order to record the information that has been annotated, 2 types of data formats are used.

- Bounding Box:
  - { Id, NormalizedCenterX, NormalizedCenterY, NormalizedBoxWidth, NormalizedBoxHeight } Where Id represents the categories of vehicle, 2 for vehicle and 4 for truck.
  - NormalizedCenterX = CenterX / W
  - NormalizedCenterY = CenterY / H
  - NormalizedBoxWidth = w / W
  - NormalizedBoxHeight = h / H
  - { CenterX, CenterY } is the center position of box, w and h are width and height of the box, W and H are the width and height of the raw image, the resolution is 640 x 480 pixels in this dataset
- Polygon:

- { Id, x0/W, y0/H, . . . xn/W, yn/H } Where Id represents the categories of vehicle, 2 for vehicle and 3 for truck.
- { x0, y0 } .. { xn, yn } are the coordinates of the polygon
- W and H are the width and height of the raw image, the resolution is 640 x 480 pixels in this dataset

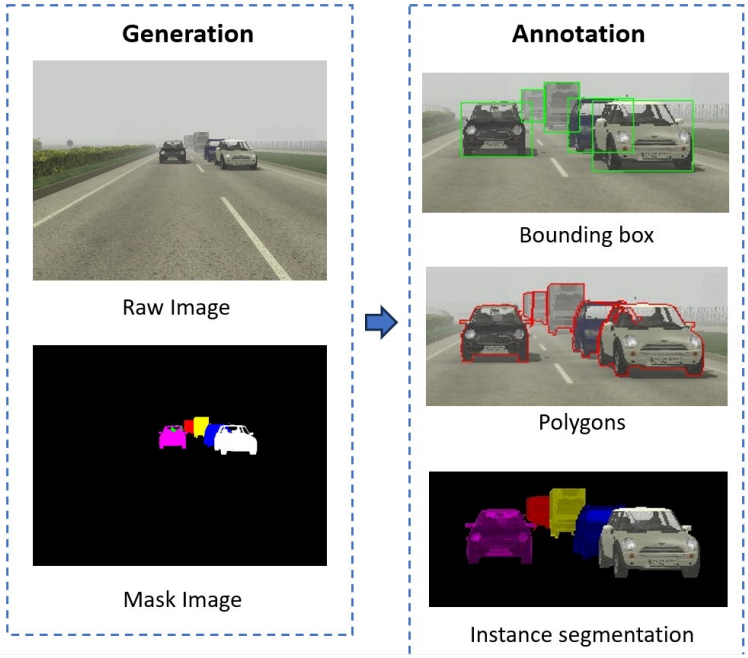


Figure 127: Ground thruth generation and annotation in BuSAS

3.5.1.4 Algorithms and models

To achieve the system functionalities, we have selected two renowned visual perception algorithms: YOLOv5 [147] and YOLOv8 [148], along with their respective models: v5s and v8s. In addition to the object detection models, we have also incorporated two corresponding instance segmentation models (v5s-seg and v8s-seg), aiming to provide more detailed information about object boundaries within an image.

To further enhance the system's capabilities for large-scale diverse driving, we retrained and fine-tuned the models using the BDD100K dataset based on the previous COCO dataset [149], the training phase is equipped with a 16-core Intel i9-12950HX 2.30 GHz CPU and NVIDIA-GTX A4500 GPU core graphics card.

3.5.1.5 Deployment and execution

After the training phase, we deployed the trained model into the software platform RTMaps^{TM} (C++ support), in order to perform the model as a functioning part of the perception system and interconnect with the simulator. During the deployment, we used TensorRT C++ API to optimize the model for inference (with precision FP16), which is based on NVIDIA's Compute Unified Device Architecture (CUDA). The setup environment and tools contain CUDA 10.2,

PyTorch 1.10.1, cuDNN v8.2.1 corresponding to CUDA, TensorRT 8.2, and Visual Studio 2017 (v141).

3.5.1.6 Metrics for AI-based evaluation and validation for POC1 object detection

The purpose of a detection-based metric is to get meaningful measures of the system's ability to perform object detection tasks. Metrics include the number of correctly detected objects, falsely detected objects, or mis-detected objects. Other widely used detection measures are detection rate/precision and sensitivity. The detection-based metrics (also called frame-based metrics) are used to evaluate the performance of a SUT (System under test) on individual frames from video sensor data. They do not take into account the identities of objects over the lifespan of the test. All the objects are individually validated to see if there is a corresponding match between SUT and GT (ground truth) systems for each frame during the test.When associating GT data with SUT-detected objects, six cases can occur: zero-to-one, one-to-zero, one-to-one, many-to-one, one-to-many, and many-to-many associations, which correspond to false alarms (the detected object has no correspondence), mis-detection (the GT data has no correspondence), correct detection (the detected object matches one and only one object), merge error (the detected object is associated with several GT objects), split error, and split-merge. The performances for each individual frame are then averaged over all the frames in the experiment to provide a performance evaluation measure.

The notation used for evaluation is as follows:

- **FP**: False positive, an object present in the SUT, but not in the GT (also called a False Alarm)
- **FN**: False negative, an object present in the GT, but not in the SUT (also called a Detection Failure)
- **TP**: True positive, an object present in the GT and the SUT (also called Correct Detection or one-to-one match)
- **TN**: True negative, an element present in neither the GT nor the SUT
- **CGT**: Complete Ground Truth is the total number of GT objects.

The quality of the object detection model can generally be evaluated from the following three aspects [28, 150, 151, 152]:

- **Accuracy of Classification**
  - **False Positive Rate (FPR)** is computed by FPR = FP/(FP + TN), representing the number of false positives relative to the sum of the number of false positives and true negatives. It is a measure of how well the system correctly rejects false positives.
  - **False Negative Rate (FNR)** is computed by FNR = FN/(TP+FN), representing the number of false negatives relative to the sum of the true positives and the false negatives. It is a measure of the likelihood that a target will be missed given the total number of actual targets.

- **True Negative Rate (TNR)** is computed by TNR = TN/ (TN + FP), representing the true false detections relative to the sum of the true false detections and the false positive. This provides a measure of the likelihood of a negative response given the total number of actual negative detections.
- **Detection Rate (DR)** is computed by DR = TP/ (TP + FN), representing the number of true positives relative to the sum of the true positives and the false negatives. It is a measure of the percentage of true targets that is detected.
- **Accuracy** is the proportion of all predictions that are correct ((TP+TN)/CGT). The accuracy rate is generally used to evaluate the global accuracy of the model, and cannot contain too much information to fully evaluate the performance of a model.
- **Precision** refers to the probability of correct detection among all detected objects. Precision (TP/ (TP + FP)) is defined in terms of predicted outcomes. It should be noted that Precision and Accuracy are not the same. Accuracy is for all samples, while Precision is only for the part of the samples that are detected (including false detections).
- **Recall** refers to the probability of correct detection among all positive samples. The P-R curve uses recall as the abscissa axis and precision as the ordinate. The change in the detection threshold will also cause the Precision and Recall values to change, thus obtaining the curve.
- **F-Measure** gives an estimate of the accuracy of the systems under test.
- **Receiver Operating Characteristic (ROC) Curve** is a graph of Detection rate vs. False Positive Rate (as in Figure 128). When the distribution of positive and negative samples in the test set changes, ROC curve can remain unchanged

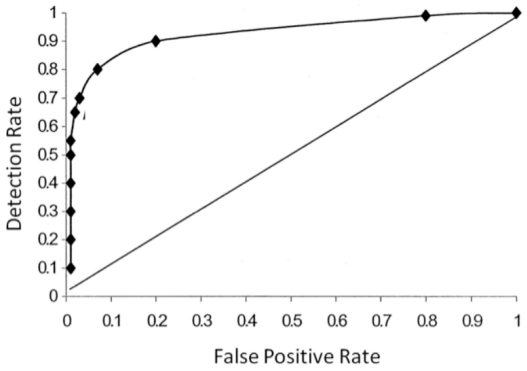


Figure 128: An example ROC curve (Detection Rate vs. False Positive Rate) [28]

- **Detection Error Trade-off Curve (DET Curve)** is a graph of Miss Rate (or False Negative Rate) vs. False Positive Rate. The DET curve is a plot of the error rate for a binary classification system.
- **Precision-Recall Curve (PR Curve)**: By varying the confidence value, it is also possible to create the PR curve. In pattern recognition and information retrieval, precision (also called positive predictive value) is the fraction of labeled or retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction

of labeled or relevant instances that are retrieved. Both precision and recall are therefore measures of relevance.

  - **Average-Precision** is the area under the P-R curve. Generally, the better the classifier, the higher the AP value. The common approaches are 11-point interpolation and Interpolating all points.
  - **mean Average Precision** is the average of multiple categories of AP, mAP must be in [0,1], the bigger the better. This metric is one of the most important in object detection algorithms.

- **Accuracy of Location**
  - **IoU** computes the ratio of the intersection and union of the "predicted bounding box" and the "true bounding box".
  - **Non-Maximum Suppression (NMS)** is to find a bounding box with a relatively high degree of confidence based on the coordinate information of the region and score matrix. For prediction boxes that overlap each other, only the one with the highest score is kept. NMS calculates the area of each bounding box, and then sorts it according to the score, and takes the bounding box with the largest score as the first object to be compared in the queue; Then calculate the IoU of the remaining bounding box and the current maximum score and box, remove the bounding box whose IoU is greater than the set threshold, and retain the prediction box with a small IoU; Then repeat the above process until the candidate bounding box is empty. Finally, there are two thresholds in the process of detecting the bounding box, one is the IoU, and the other is to remove the bounding box whose score is less than the threshold from the candidate bounding box after the process. It should be noted that Non-Maximum Suppression processes one category at a time. If there are N categories, Non-Maximum Suppression needs to be executed N times.

- **Performance of Detection** describe how fast the speed of the model can run. It can be the number of images that the model can process per second (fps).

3.5.1.7 Evaluation of Multi-Objects Tracking

The tracking-based metrics measure the ability of a SUT to track objects over time. The tracking-based metrics (also called object-based metrics) take the identity and the complete trajectory of each object separately over the test sequence and compare the GT tracks with the SUT tracks based on best correspondence. Then, based on these correspondences, various error rates, and performance metrics are computed.

Since the GT track(s) could correspond to more than one SUT track, a correspondence mapping has to be established first. Based on this mapping between the object tracks, the track-based metrics are computed. The correct match requires both spatial and temporal overlap between GT tracks and SUT tracks. Requirements for these metrics include:

- All objects that appear must be found in a timely manner(found a mapping);
- The object's position should be as consistent as possible with the position of the real object, where the precision with which the object's position was estimated should be determined.

- Each object should be assigned a unique ID, and the ID assigned to that object remains the same throughout the sequence.
- Making sure that the objects were tracked correctly over time. This includes checking that objects were not substituted for each other, for example when they passed close to each other and checking that a track was correctly recovered after it was lost, for example when an object was occluded.

The following metrics[140, 28, 153] are calculated:

- **Object Tracking Time delay**: This is the estimated delay between the SUT algorithm's detection of an object or person and that of the GT. It could be positive or negative.
- **Identification switch (IDSW)**: is the ID Switch (total number of ID switches, mismatches): in the above Figure 129, the switch from red to blue is recorded as an IDSW, the sum of the number of mismatches in the entire simulation.

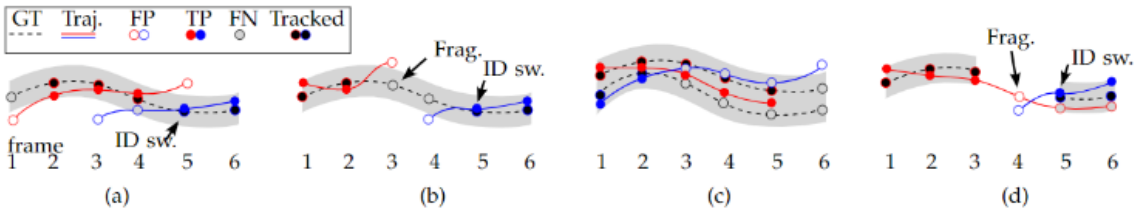


Figure 129: Parameters of MOT evaluation [29]

- **Track Matching Error (TME)** is the positional error between the SUT trajectory and the GT trajectory and measures the average distance error between the GT and SUT track. The smaller the TME number, the better the tracking accuracy.
- **Track Completeness (TC)** is defined as the time for which the SUT track overlapped with the GT track divided by the total duration of the GT track.
- **Occlusion success rate (OSR)**: OSR = Number of successful dynamic occlusions/ Total number of dynamic occlusions. A successful occlusion occurs when the track and object identity is not lost during the occlusion or are correctly recovered immediately following the occlusion. recovered immediately following the occlusion.
- **Mutiple Object Tracking Accuracy** is a metric to measure the accuracy of single-camera multi-objects tracking. The computation and the process are shown in Equation 1.

$$MOTA = 1 - \frac{FN + FP + IDSW}{GT} \in (-\infty, 1] \tag{16}$$

The closer the MOTA is to 1, the better the performance of the tracker. Due to the existence of the number of jumps, there may exist cases where MOTA is less than 0. MOTA mainly considers all object-matching errors in tracking, mainly FP, FN, and IDs. It is a very intuitive measure of tracking its performance in detecting objects and maintaining trajectories, and independent of object detection accuracy.

- **Mutiple Object Tracking Precision** is a metric to measure the position error of single-camera multi-objects tracking.
- **Mostly Tracked**: an Object is mostly tracked if it is successfully tracked for at least 80 percent of its life span. Note that it is irrelevant for this measure whether the ID remains the same throughout the track.
- **Mostly Lost**: If a track is only recovered for less than 20 percent of its total length, it is said to be mostly lost.
- **Partially Tracked**: Besides Mostly tracked and lost, others are all regarded as partially tracked.
- **Fragmentation**: Besides Mostly tracked and lost, others are all regarded as partially tracked. To that end, the number of track fragmentations(FM) counts how many times a ground truth trajectory is interrupted (untracked).In other words, a fragmentation is counted each time a trajectory changes its status from tracked to untracked and tracking of that same trajectory is resumed at a later point.
- **ID related metrics**: IDP (Identification Precision) represents the accuracy of pedestrian ID recognition in each pedestrian box; IDR (Identification Recall) represents the recall rate of pedestrian ID recognition in each pedestrian box; IDF1 (Identification F-Score) represents the F value of pedestrian ID recognition in each pedestrian box.

MOTA is not sufficient to measure the performance of multi-object tracking in some cases. The evaluation of MOTA overemphasizes the effect of detection. According to the calculation method of MOTA, an extreme case is that the performance of detection is very good, but all detected targets are not tracked, and are all assigned the same track id. MOTA will be very high because of IDsw=0. But obviously, the tracking performance for this extreme case is 0. By the way, MOTP is the same. IDF1 can evaluate the tracking, but still relies on MOTA.

- **HOTA (Higher Order Tracking Accuracy)** [30] is a new metric for evaluating the performance of multi-object tracking (MOT). It is designed to overcome many limitations of previous metrics such as MOTA, IDF1 and Track mAP. HOTA divides the task of evaluation tracking into three subtasks (detection, association, and localization), and uses the IoU (intersection pair union) formula (also known as the Jaccard index) to calculate a score for each subtask. It then combines these three IoU scores for each subtask into a final HOTA score (as in Figure 130).

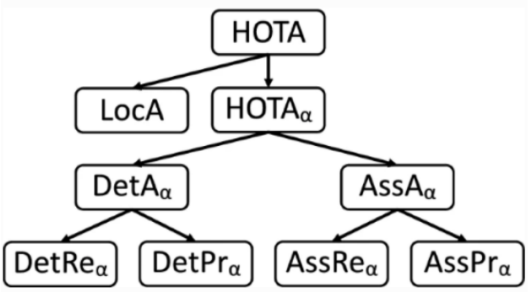


Figure 130: Sub-metrics of HOTA [30]

- **LocA (Localization Accuracy)**: Average localization similarity averaged over all matching detections and averaged over localization thresholds.
- **DetA(Detection Accuracy)**: Detection Jaccard index averaged over localization thresholds. DetA can be decomposed into two sub-metrics Detection Recall and Detection Precision.
- **DetRe(Detection Recall)**: TP/(TP+FN) averaged over localization thresholds.
- **DetPr(Detection Precision)**: TP/(TP+FP) averaged over localization thresholds.
- **AssA (Association Accuracy)**: Association Jaccard index averaged over all matching detections and then averaged over localization thresholds. AssA can be decomposed into two sub-metrics Association Recall and Association Precision.
- **AssRe(Association Recall)**: TPA/(TPA+FNA) (shown in Figure 131) averaged over all matching detections and then averaged over localization thresholds.
- **AssPr(Association Precision)**: TPA/(TPA+FPA) (shown in Figure 131) averaged over all matching detections and then averaged over localization thresholds.

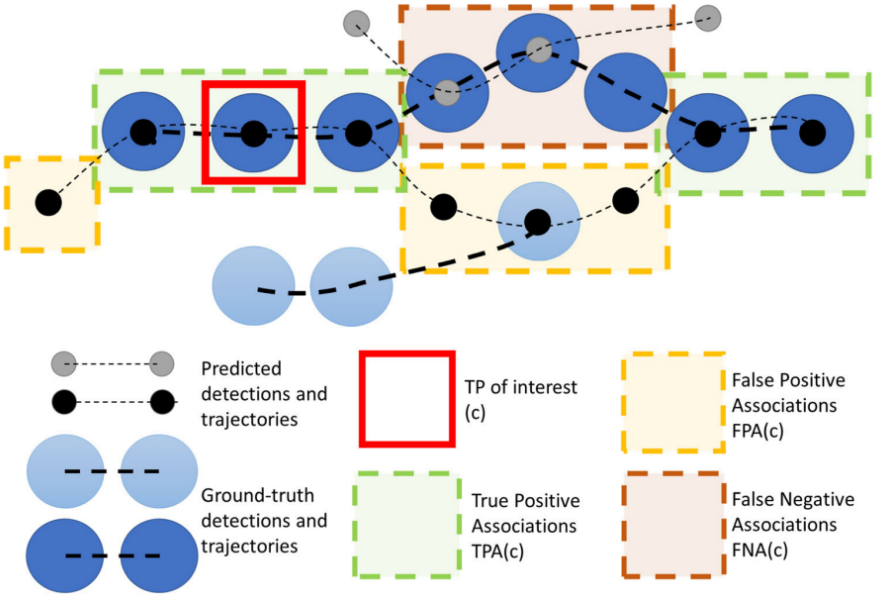


Figure 131: Parameters of ASSA [30]

3.5.1.8 Evaluation of Lane Detection

Essentially, Lane detection is a part of object detection,which means that the same metrics can also be applied to evaluate the module, such as Accuracy, F-Measure, and IoU, etc. Here we introduce more specific (lane-oriented) metrics [31]:

- **Accuracy of Lane Feature Extraction**: As shown in Figure 132, the detected lane (d) is determined by the lane features (b) that are extracted by a lane estimation algorithm. Inaccurate lane features will not fit the road model accurately, and hence the estimated

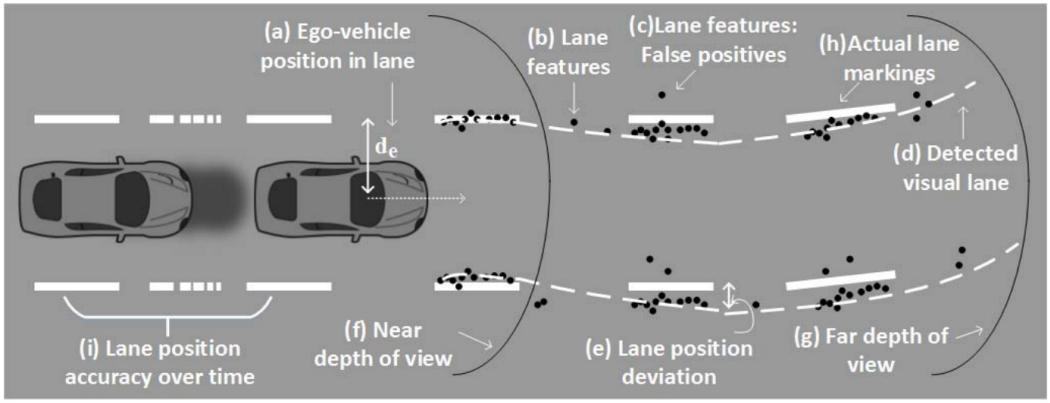
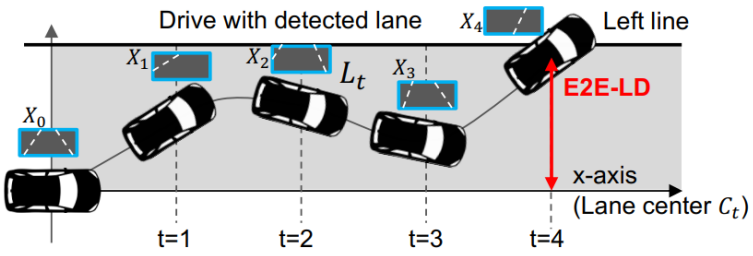


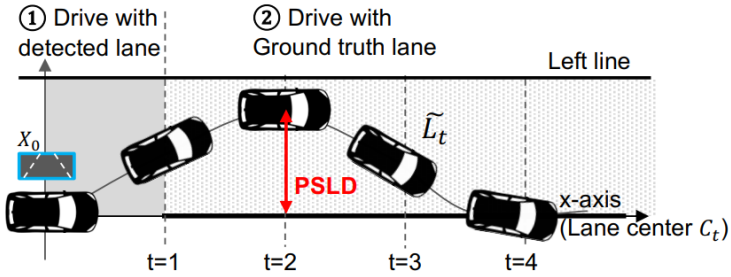
Figure 132: Illustration of the key parameters and performance metrics for evaluating lane analysis process [31]

lane will not follow the actual lane. Analyzing the accuracy of the lane feature extraction step aids in improving the performance of the entire lane detection process.

- **Ego-vehicle localization**: This metric uses distance d_e between the center of the camera on the ego-vehicle with respect to the lane markings detected by the algorithm. It is limited by lane detection in the far depth of view.
- **Lane Position Deviation**: ((e) in Fig. 132) measures the deviation of detected lane (d) from the actual lane that is obtained by joining the actual lane markings ((h) in Fig. 132). It captures the accuracy of the lane detection process in both the near and far depths of view of the ego vehicle.
- **Other Deviation Metrics**: These metrics(including ego-vehicle localization) represent the performance in the main real-world downstream application.



(a) End-to-End Lateral Deviation Metric [154]



(b) Per-frame Simulated Lateral Deviation Metric [154]

Figure 133: E2E-LD and PSLD

  - **End-to-End Lateral Deviation Metric** (in Fig. 133(a)) is the maximum lateral deviation from the lane center in continuous closed-loop perception and control, which is the ultimate downstream-task performance metric for lane detection. Such deviation is directly safety-critical as large lateral deviations can cause a fatal collision with other driving vehicles or roadside objects.
  - **Per-Frame Simulated Lateral Deviation Metric** (in Fig. 133(b)) simulates E2E−LD only with a single camera input at the current frame (X_0) and the geometry of the lane center.

- **Computation Efficiency and Accuracy**: Having a high accuracy of detection at the cost of high computational resources is not always desirable, a tradeoff between accuracy and computational efficiency solution should be evaluated.

- **Cumulative Deviation in Time**: studies how the accuracy of the lane estimation process varies in the last p seconds (indicated by (i) in Fig. 132). It helps to determine the maximum amount of time (also critical response time) for which a lane analysis method results in a "given" accuracy of lane deviation.

3.5.1.9 Evaluation and result

We proposed 3 levels of evaluation for this visual perception system equipped with different YOLO models. The results were produced by UGE and validated by LNE, using the AI matics evaluation software suite (LNE open-source software).

- **Scenarios level** : We have defined seven weather filters on the camera as shown in Fig. 134. The first scenario representing a clear sky is the reference. The remaining scenarios focus on simulating various weather conditions. We have categorised homogeneous fog

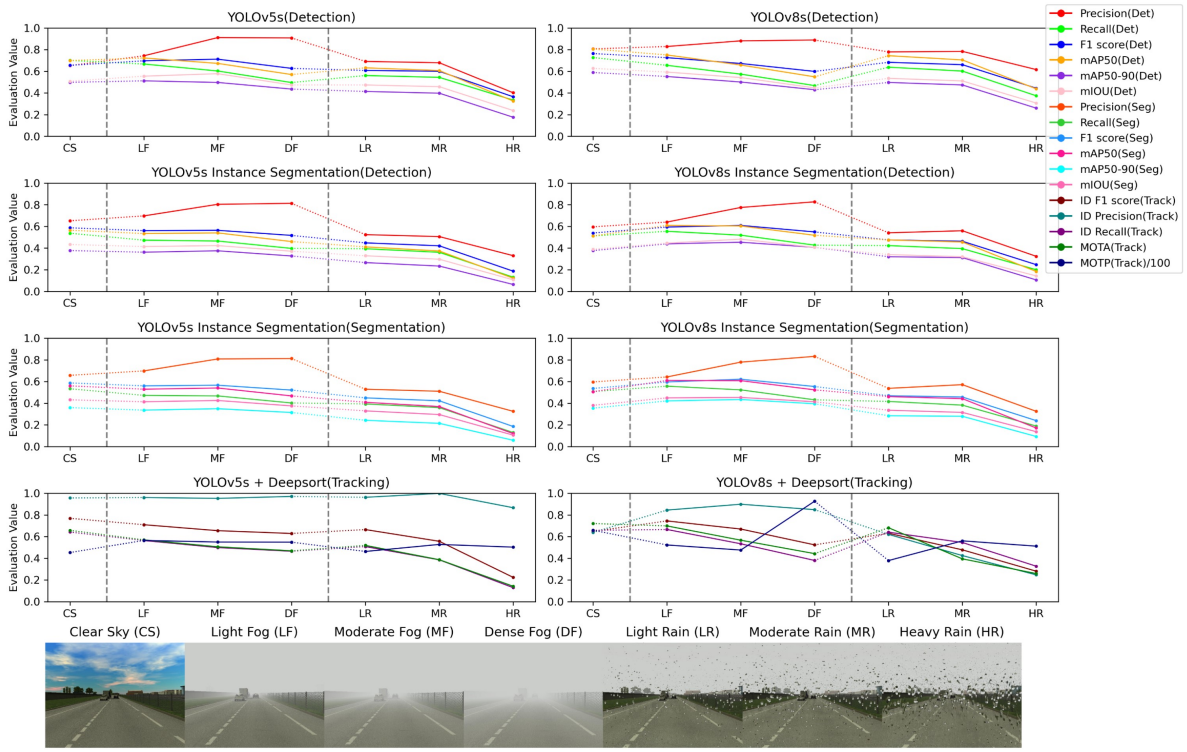


Figure 134: Result on different weather conditions at component level

into three levels using the Koschmieder law, ensuring an accurate representation. For rain, we have incorporated two filters. The first filter replicates a waterfall effect, while the second mimics raindrops on a camera lens or windshield. Similarly, we have created three different levels of intensity to simulate varying degrees of rain.

- **Component level** : We curated seven distinct groups of datasets, each consisting of approximately 10,000 images for detection and instance segmentation, organized based on predefined weather conditions. For tracking, the datasets are 7 groups of real-time videos. These datasets included various types of simulated vehicles, such as cars and trucks, navigating the Satory test track. Different camera perspectives were incorporated, including front and rear views of the ego vehicle, as well as close and distant perspectives. The performed results were evaluated using a set of metrics. The evaluation values are presented in Fig. 134. The results clearly demonstrate the influence of different weather conditions on the metrics. Adverse rain weather significantly impacts mAP, indicating difficulties in accurately detecting and localizing objects under heavy rain conditions. In contrast to the impact of adverse rain weather, varying levels of fog have a relatively lesser effect on the metrics. While there is still a decrease in the mAP with increasing fog levels, it is not as pronounced as the impact of rain. However, it is worth noting that there is a consistent increase in precision as the degree of fog increases. This suggests that fog has a discernible influence on improving precision by reducing background interference and minimizing the impact of distant objects. Comparing the performance of v8s and v5s, it is observed that v8s generally achieves slightly higher metric values. However, when it comes to object tracking, v5s demonstrates a lower MOTP than v8s. This indicates that v5s exhibits higher precision in tracking object locations. The disparity in tracking per-

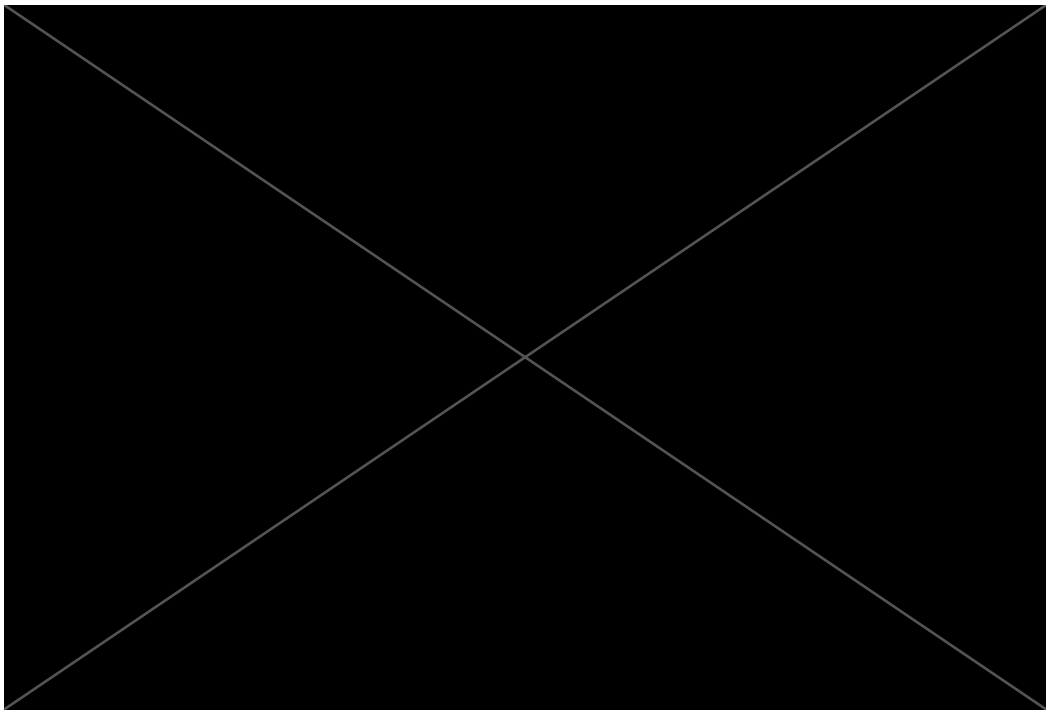
Table 8: Evaluation value of system metrics for the POC 1: BuSAS

		Clear Sky		Light Fog		Moderate Fog		Dense Fog		Light Rain		Moderate Rain		Heavy Rain	
		v5s	v8s	v5s	v8s	v5s	v8s	v5s	v8s	v5s	v8s	v5s	v8s	v5s	v8s
Perception Specific KPIs	Success Rate (%)	62.7	64.4	55.1	65.0	48.5	51.9	45.3	36.9	49.7	62.8	37.7	53.5	12.6	31.9
	Mean Tracking Error (m)	2.30	1.91	2.25	1.72	1.67	1.48	1.31	1.44	2.49	2.33	2.14	2.43	3.51	2.68
Time Specific KPIs	Detection time (ms)	20	36	21	37	19	36	19	36	19	36	19	36	20	37
	Tracking time (ms)	72	104	67	90	59	76	55	68	67	110	60	152	58	174
	System frequency (Hz)	16	11	17	13	17	14	18	16	16	11	17	8	18	8
Risk Specific KPIs	minimal TTC (s)	1.27	1.82	1.52	1.83	2.02	1.83	1.93	2.02	1.79	1.70	-	-	-	-
	Mean Collision Probability (%)	31.7	26.7	30.1	28.5	26.1	27.4	24.0	23.8	27.6	28.5	45.8	37.4	47.2	36.4
	Mean Risk (%)	3.77	2.75	3.23	2.76	3.03	2.75	2.85	2.78	2.90	2.78	8.51	4.84	8.93	5.04
	Crash	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✔	✔

formances may be attributed to the frequency, as v5s benefits from a higher update rate and more frequent predictions, leading to improved tracking precision.

- **System level** : In comparison to the instance segmentation model, both v5s and v8s models demonstrate higher accuracy in providing boundary information, as reflected by mIOU. Hence, we have incorporated these two models into our system and evaluated their performance under bus stop scenarios. Specific KPIs were calculated based on the real-time results obtained from the system, as in Tab. 8. While v8s demonstrates higher detection and tracking capabilities than v5s, it also exhibits an increase in tracking time under adverse rain weather. This suggests that more objects may be detected due to the presence of raindrops on the lens, and YOLOv8 proves to be more sensitive in this scenario. On the other hand, under foggy conditions, both system frequencies are reduced, indicating that fewer remote objects are detected due to the degree of fog. Tab. 8 also presents the evaluation results obtained using collision probability and risk of collision based on TTC [155]. We calculated the mean value specifically for critical scenarios. Notably, the risk across all groups was found to be low primarily because our scenarios involve low speeds, resulting in a low likelihood of injury. However, the risk can still indicate heightened safety concerns in moderate and heavy rain weather conditions, similar to the collision probability.

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

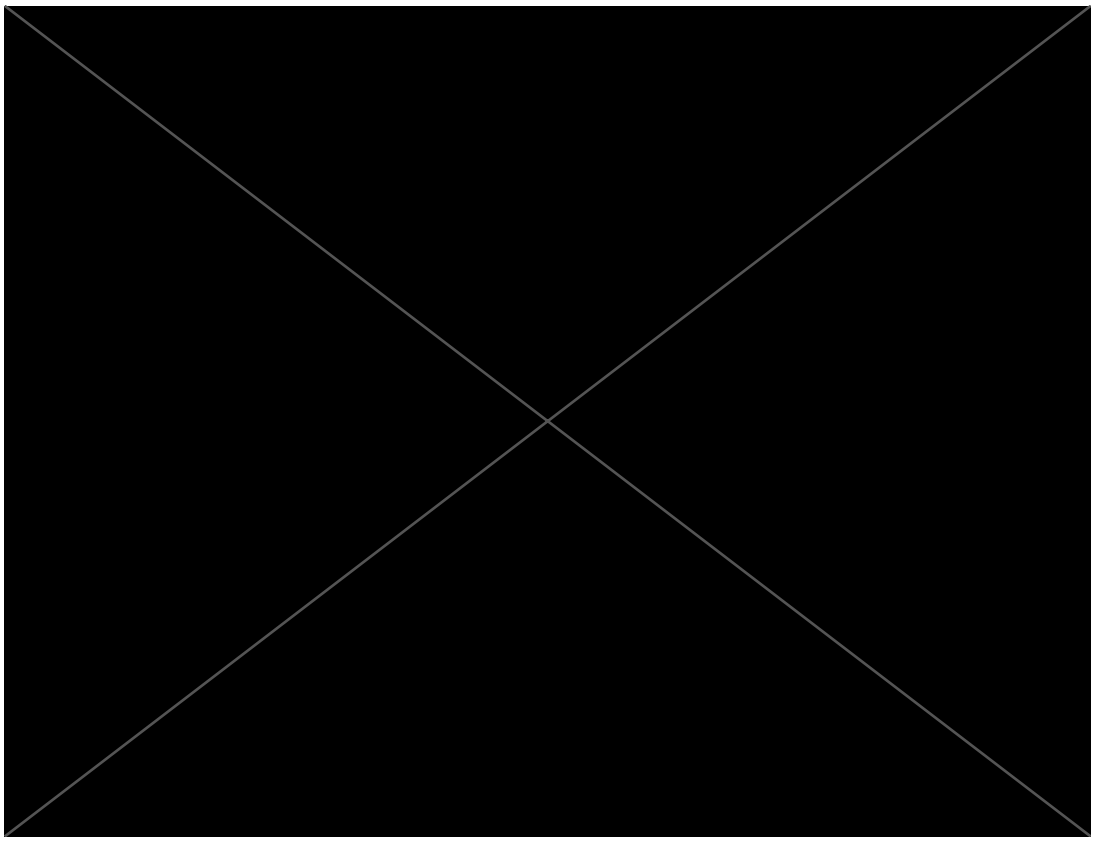
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

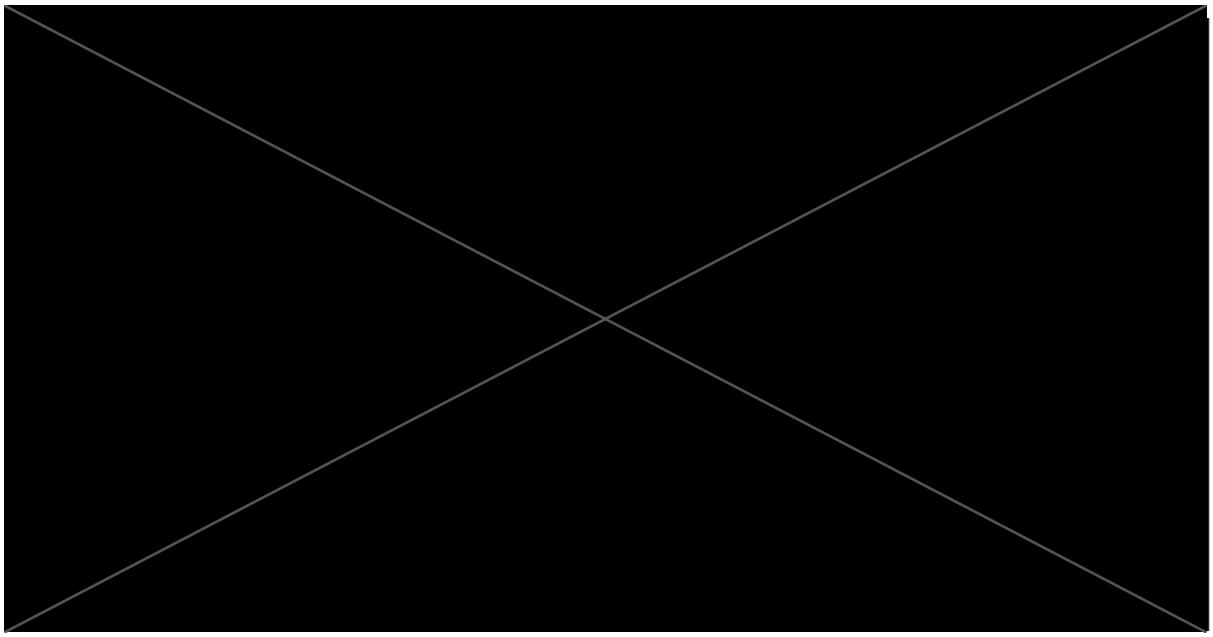
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



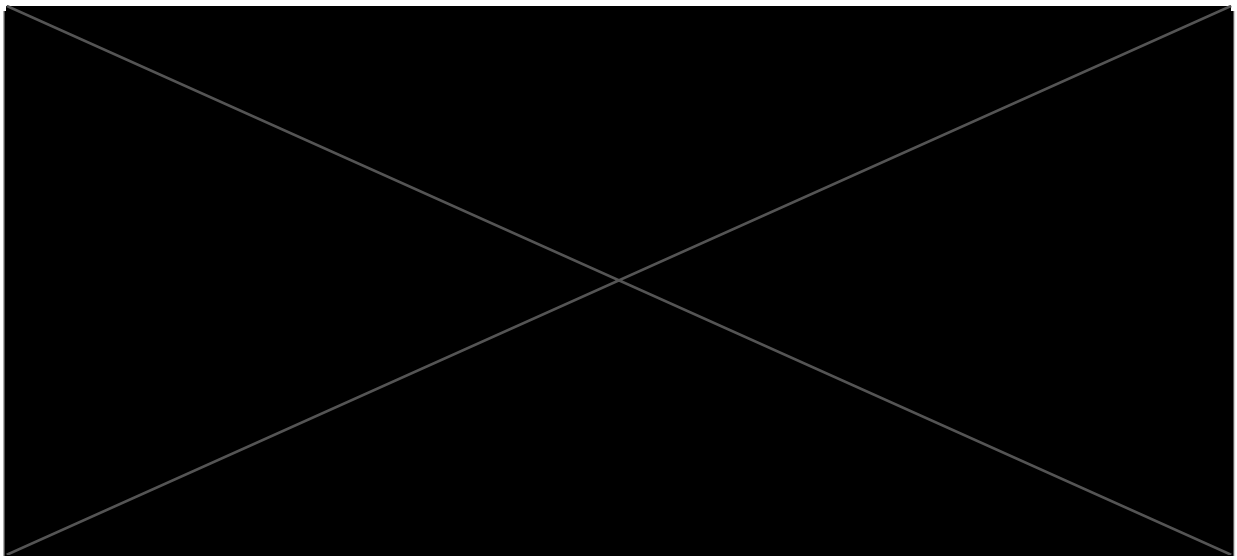
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

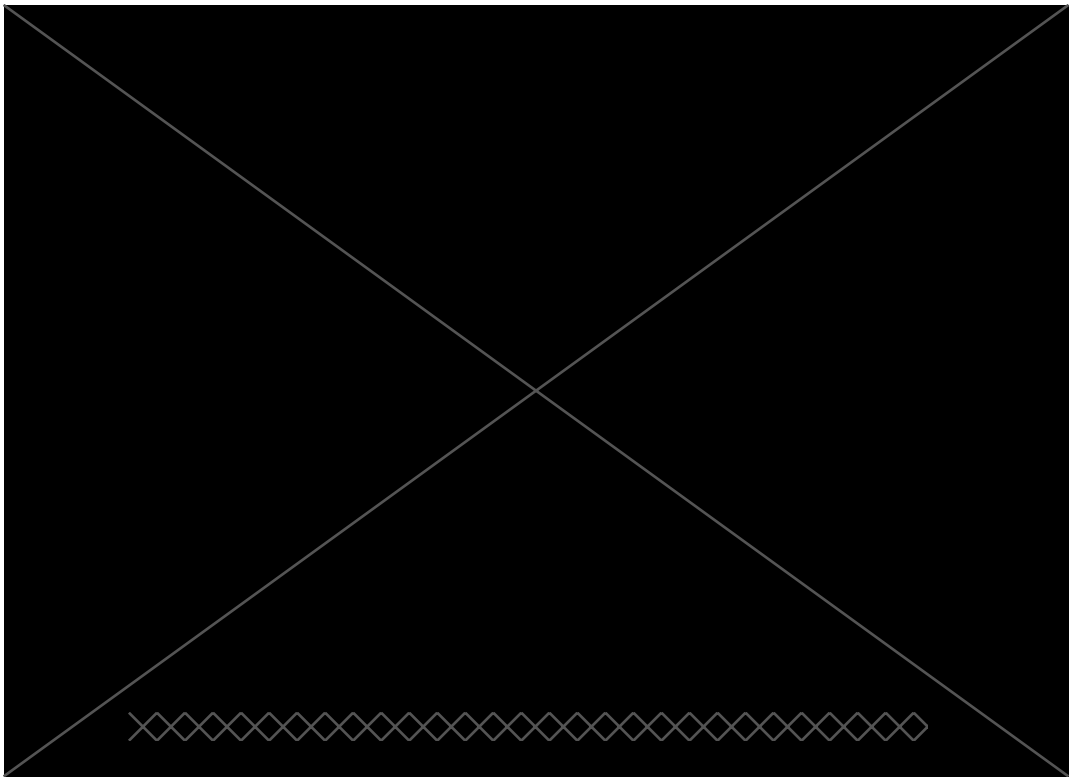
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

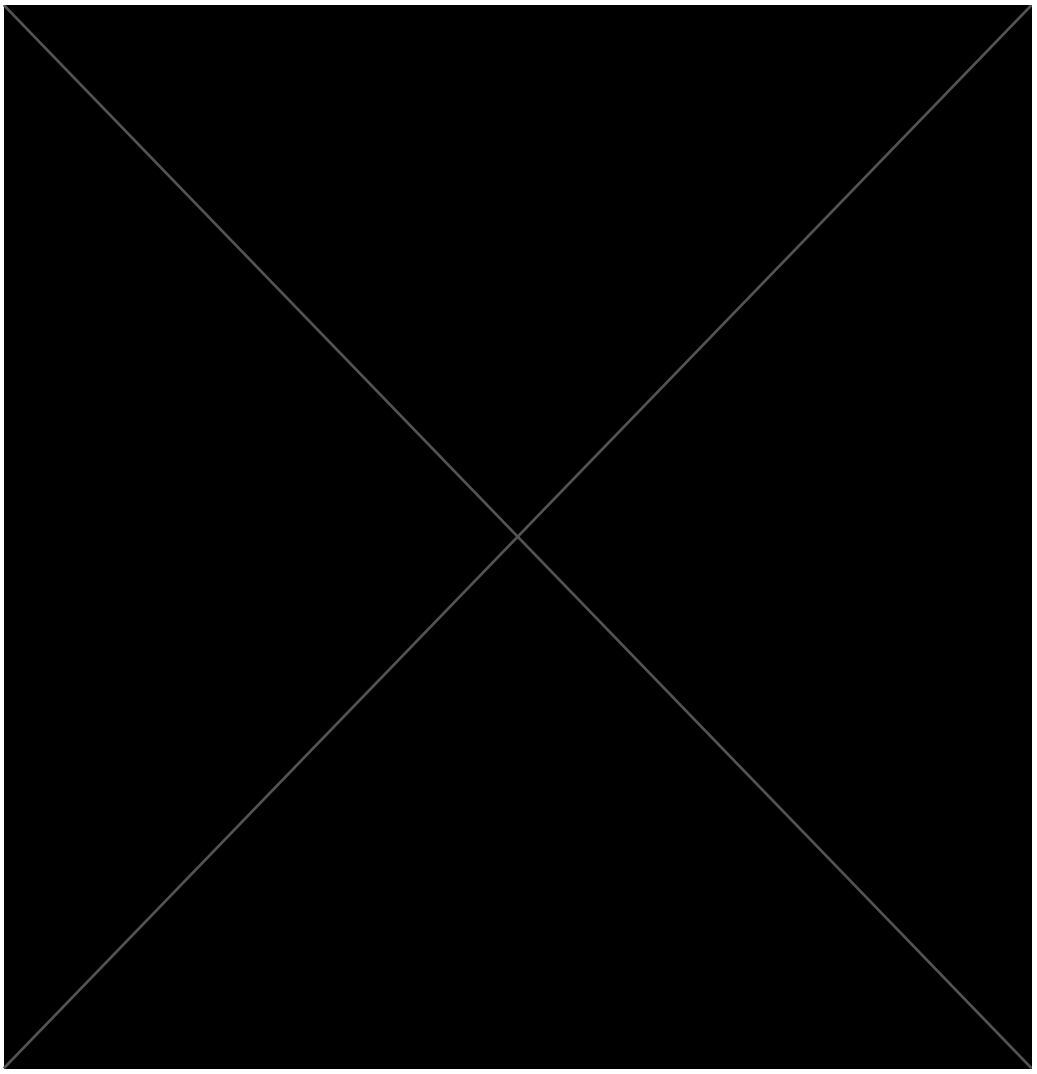


[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

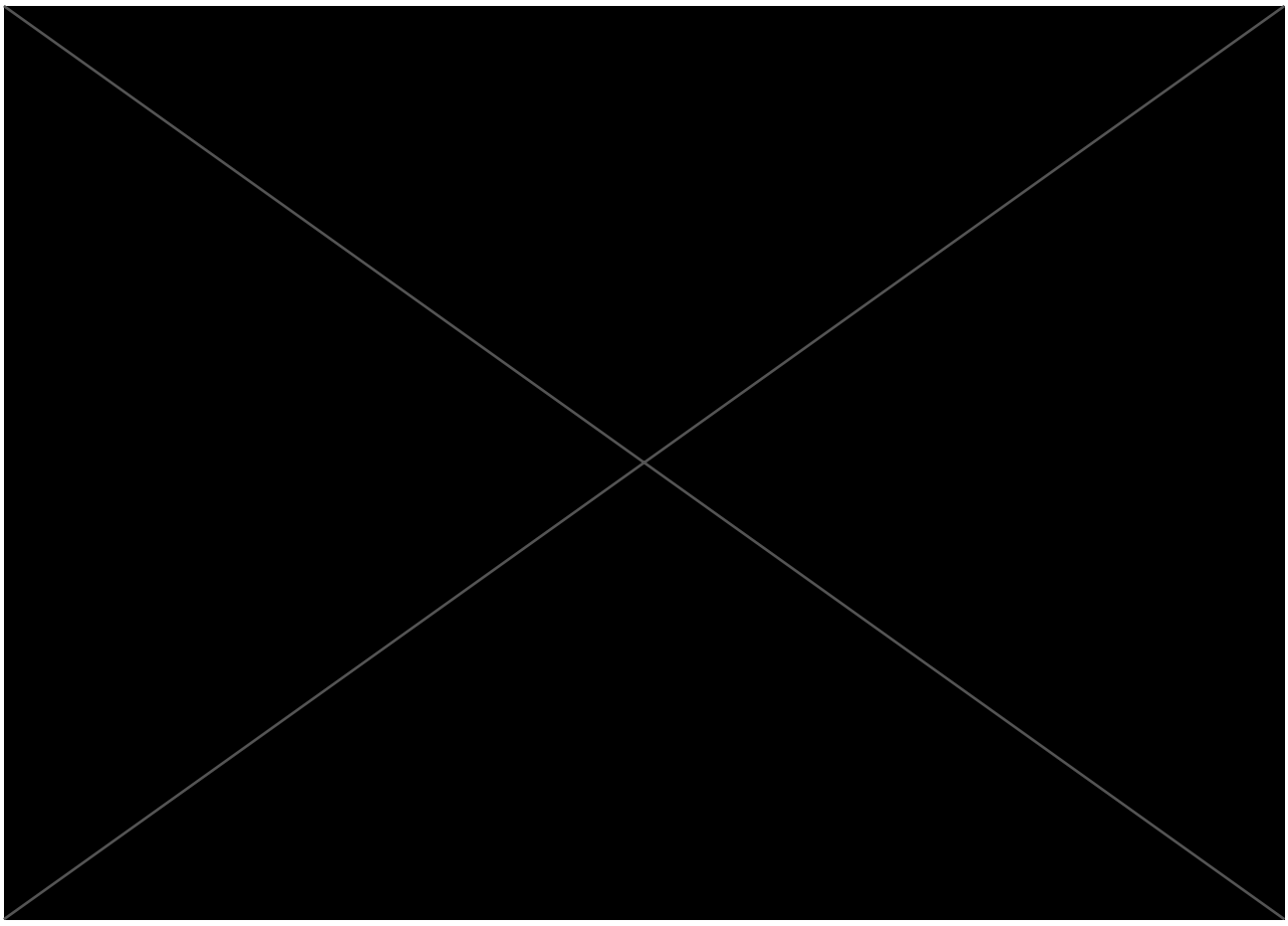
[redacted]

[redacted]

[redacted]

[redacted]

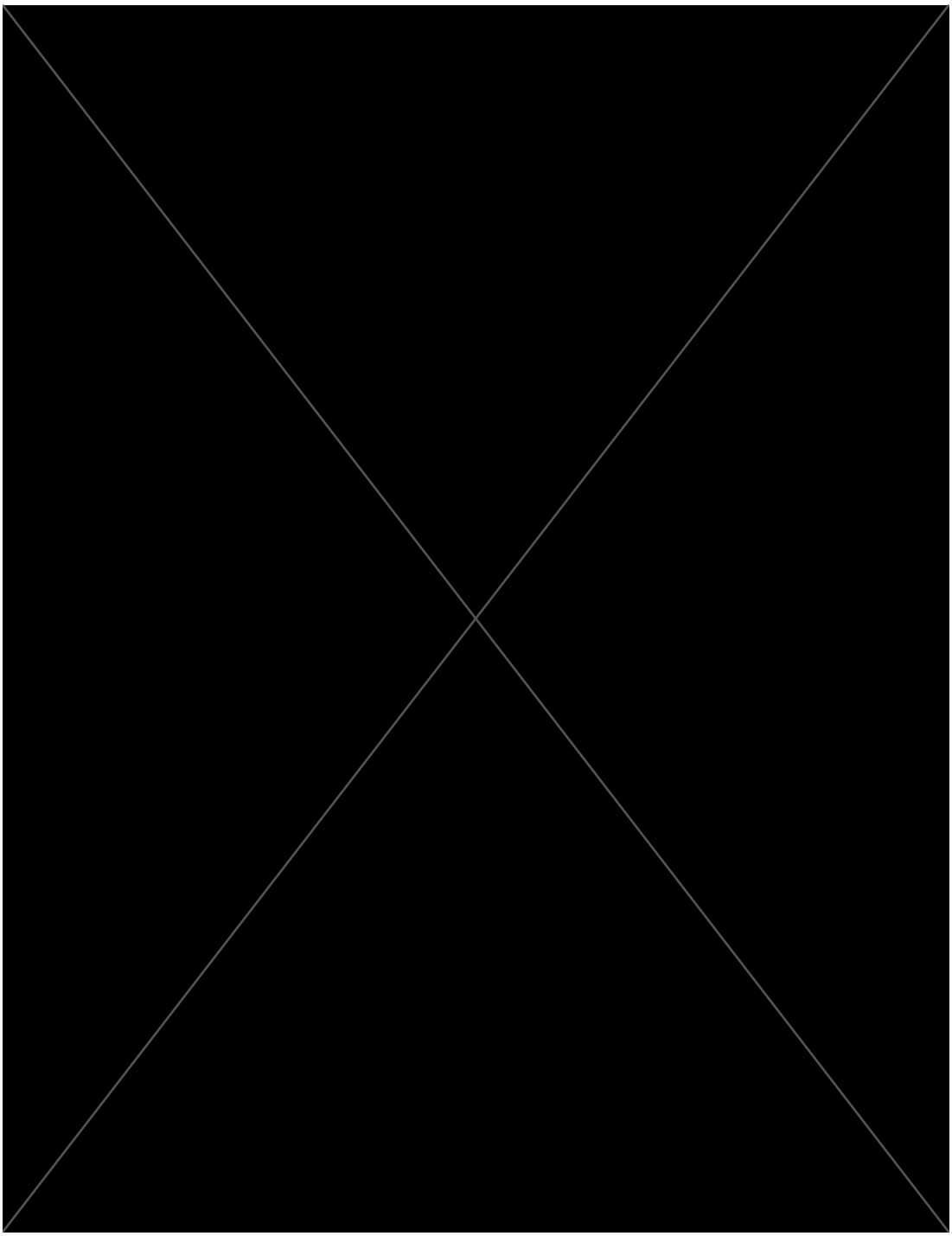
[redacted]



[redacted]

[redacted]

[redacted]



3.5.3 POC 3

UTAC and AVSimulation work together on the POC 3 which shows the use of two different virtual testing methods : the first one is model-in-the-Loop, fully virtual and the second one is the Vehicle-In-the-Loop (VIL) which combines virtual and physical. The aim of the POC is to see how VIL architectures simulation can be used and what they can bring in validation loop of AI-based system. Testing it with a POC helps discussing about methodology of validation, giving real examples as benefits of simulation and challenges the tools (software and hardware) to improve them so that they can better fit with the AI testing and validation issues. As UTAC and AVSimulation are not AI system developers, the study relies on testing an external AI sys-

tem. The one used in this POC is Openpilot from CommaAI which is an open-source automated driving software. Openpilot is an automated driving system of level 2. It has an Adaptive Cruise Control (ACC) and a Lane Centering Assist (LCA) to drive the vehicle, an AI algorithm is also embedded for object detection and the classification.

3.5.3.1 Context presentation and system environment

Openpilot presentation Openpilot software is built to be run on a small hardware device called Comma 3X (see Figure 143). By plugging this device to the CAN bus of any vehicle (that is in the list of the supported car models), the system is able to drive the vehicle relying only on camera sensors that are mounted on the device. To send driving commands, Openpilot sends steering wheel angle and gas/brake pedals command.

Figure 143: The Comma 3X hardware device

The goal of this POC is then to replace this device by a virtual solution that can connect to the VIL in order to ensure the same functionality : driving the virtual vehicle.

The Openpilot software can be interfaced with any simulator using a bridge. In the version 0.9.2 of Openpilot, this bridge relies on a messaging library called cereal. This library lets any system exchange information with Openpilot such as camera images, GPS location, speed of the vehicle, steering and pedal commands, etc. Using this messaging library, we can also get log information such as the detected targets and the estimated distances. This kind of information is an interesting way to evaluate the AI : getting any debugging information and comparing it with the real data can let us test our AI. This kind of testing is implemented in this POC.

Openpilot integration with SCANeR The first step of the implementation of this POC has been to integrate Openpilot to the simulation software SCANeR studio. In this case, SCANeR studio is used as a ”virtual vehicle” that can replace the real vehicle commonly used when using Comma 3X device. The figure below summarizes the used architecture to implement this Openpilot/SCANeR ”co-simulation”.

SCANeR studio relies on UXD Engine to generate images corresponding to a camera sensor. UXD Engine is a 3D photorealistic renderer based on Unreal to generate images. The photorealistic features of this engine let us test several cases based on different times or weather conditions in different environments.

Openpilot integration on a virtual platform A virtual platform comprising an emulated ARMv8 processor (at binary/instruction set level), similar to the one from Comma 3X device, was created, see Figure 147. The full software stack, from the operating system (Ubuntu/arm64)

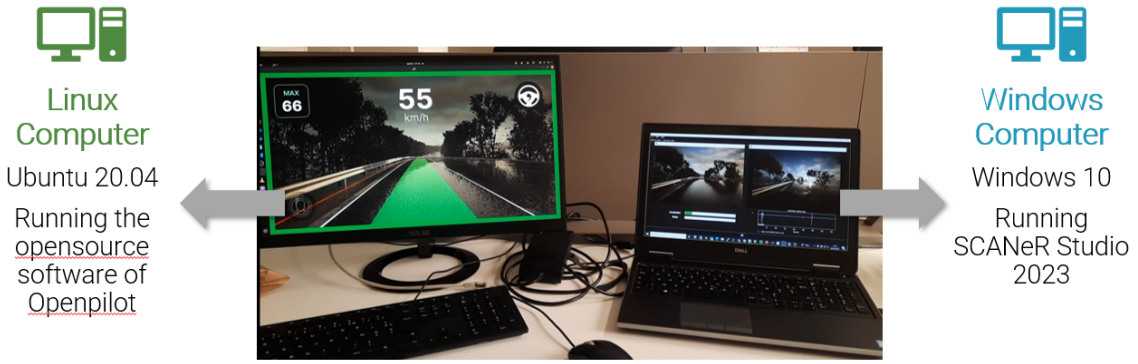


Figure 144: The simulation architecture relying on Ubuntu for Openpilot and on Windows for SCANeR studio

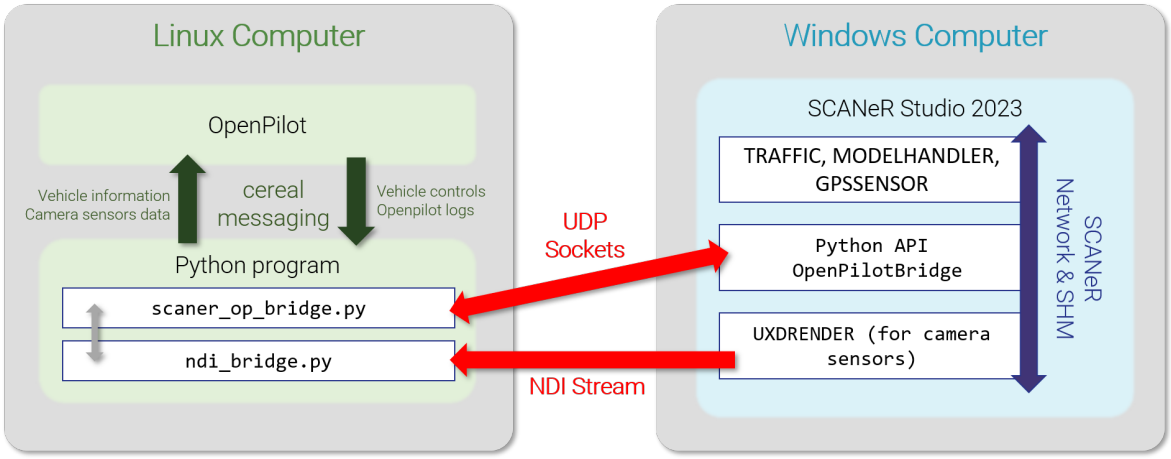


Figure 145: The interfaces between the two machines to establish the co-simulation between SCANeR and Openpilot

up to the Openpilot services (version 0.9.2 at the time of writing this document), runs within the virtual platform. Openpilot relies on a software bus, called `Cereal`, based on Cap'n Proto, which provides Openpilot services (daemons) with communication facilities. Cereal defines strongly-typed messages (structured objects) that services can publish or subscribe to. The two main daemons for autopilot are `Modeld`, the AI model for autonomous car driving, and `Controlsd` to generate car controls from the results of `Modeld` service (`modelV2`). In `Modeld`, the tinygrad models were selected at the expense of ONNX models because these are the models intended for embedded devices. However, it should be noted that the tinygrad models are nevertheless derived from the ONNX models at software build time. `Modeld` relies on OpenCL runtime to run the AI model and compute the predictions. An `Instrumenter` services, derived from the existing bridge for CARLA simulator, allows to feed `Modeld` service with inputs, see Figure 148, such as road video stream, and to record car controls (e.g. gas, brake, steer, ...) and car control state (for user's interface) from `Controlsd` service (`carControl` and `controlsState` respectively) in a trace file.

While a virtual platform models hardware device with a high level of details (e.g. using a bit accurate emulated processor), it has limited simulation speed not convenient for real-time

Figure 146: Tests of Openpilot in diverse weather and time conditions in SCANeR studio

co-simulation. As a consequence, even if independent tests can run in parallel, only short tests should run on such virtual platforms because minutes of simulated drive can easily become hours or days. However the virtual platform can assess the representativity of higher level simulation (i.e. whether the real system would react as expected with model-in-the-loop virtual testing method), before targeting true embedded devices. The virtual platform can also introduce low level errors on computations and monitor their influences on driving instructions to evaluate the sensitivity of models to errors.

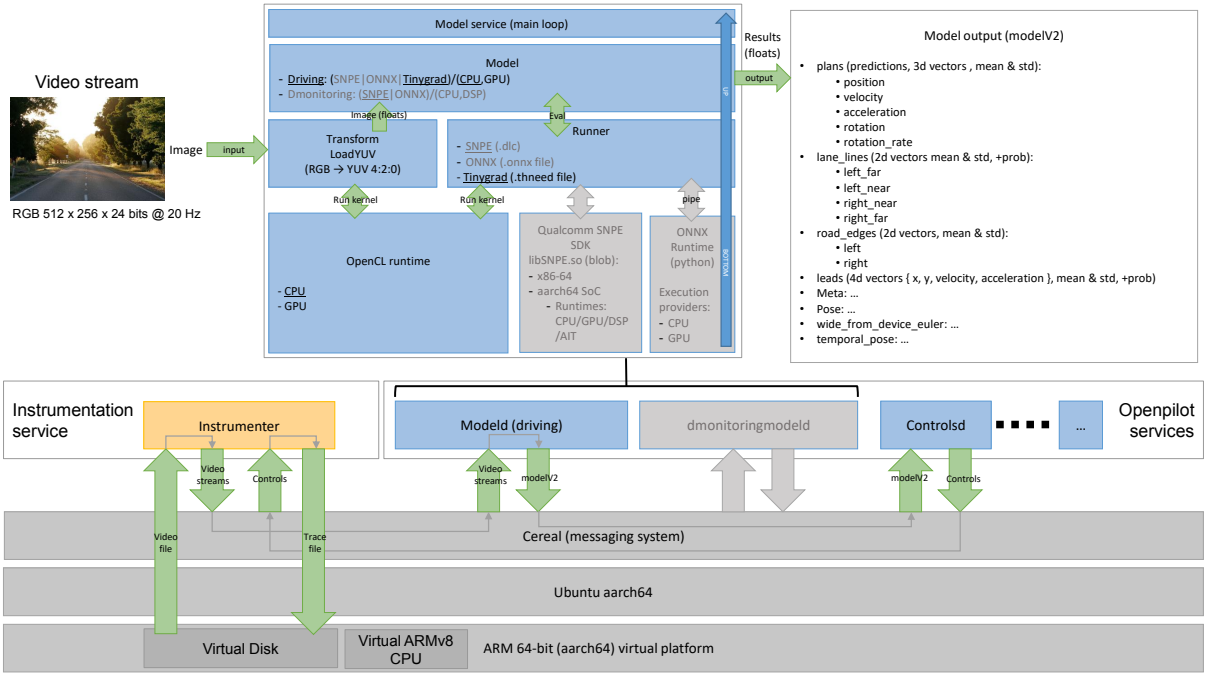


Figure 147: Virtual platform for Openpilot

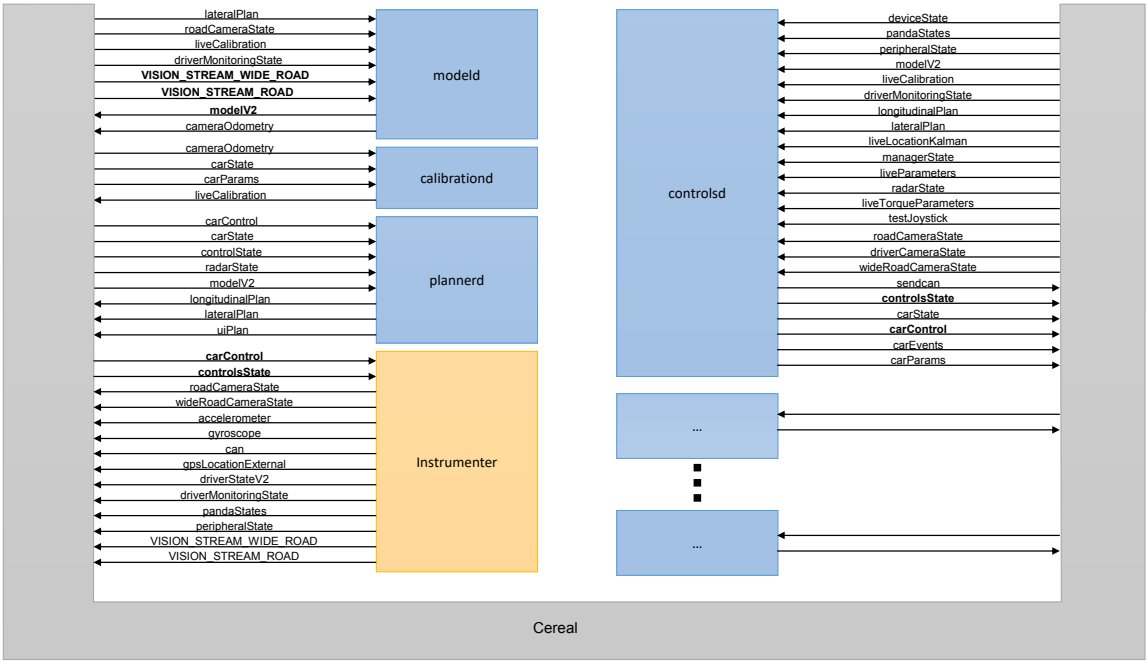


Figure 148: Openpilot services

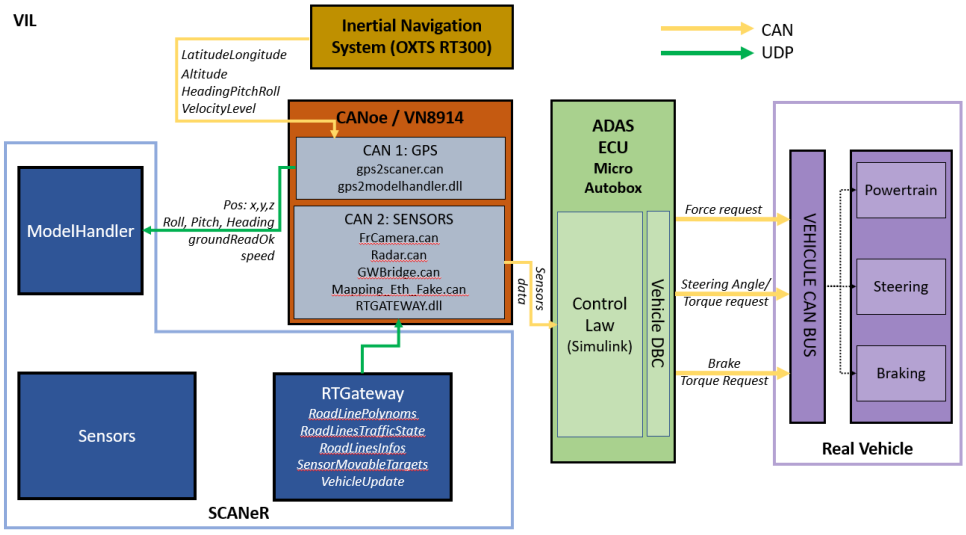


Figure 149: Initial VIL architecture

Vehicle-In-the-Loop at UTAC The VIL is a test facility that enables the ADAS of a real vehicle to be tested in a virtual environment. It can be used to test any type of driving aid (AEB, LSS). It makes it possible to carry out more complex and dangerous scenarios and to increase test productivity. An inertial measurement system is used to synchronise the position and speed of the virtual vehicle with those of the real vehicle. The CAN messages (longitude, latitude, altitude, etc.) are recovered and converted into x,y,z, etc. So that they can be read by SCANeR studio, the simulation software developed by AVSimulation. We then place virtual sensors on the virtual vehicle. The MicroAutoBox (MAB) from DSpace acts like an ADAS ECU and sends the requests directly to the vehicle's actuators.

Openpilot integration in the VIL After integrating Openpilot with SCANeR, the second part was to integrate Openpilot in the Vehicle-In-the-Loop test facility. Originally, VIL system is based on SCANeR studio for the virtual environment and a MicroAutoBox is in the trunk to allow to load control/ADAS systems we want to test. So the main issue is how we can send information from Openpilot to the MicroAutoBox.

We needed to create a new python script bridge similar to the one used in the previous chapter. In this script, messages and data from Openpilot are converted to CAN messages in order to send them directly to the MicroAutoBox. Then in the MAB, a simple Simulink model is in charge to transmit the right requests to the ECU.

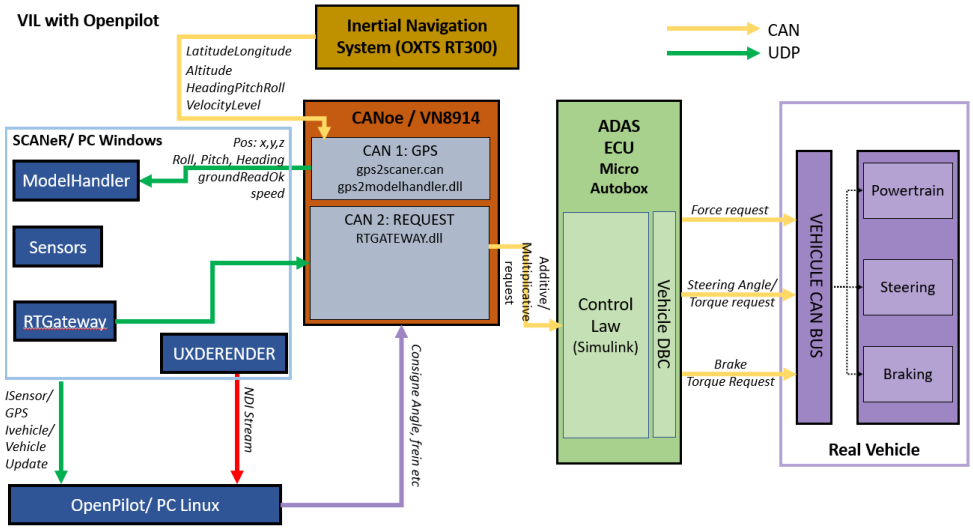


Figure 150: VIL architecture modified with Openpilot integration

3.5.3.2 Evaluation objectives

When implementing this POC, the main aim is about testing the methodology by analysing three main aspects:

- **Analysing the representativity of the methodology:** By comparing similar scenarios in fully virtual simulation and with the VIL, a score of representativity can be computed. Such criteria can be established by comparing measured data between virtual and real scenarios. These can be the trajectories of the ego, the gas and brake pedal command sent by Openpilot or the detected distances between targets and the ego.
- **Studying the repeatability of the algorithm:** Using controlled scenario by using fully virtual simulation or VIL architecture let us test the repeatability of Openpilot. Indeed the same scenario can be replayed and then the responses of Openpilot can be tested. As the AI relies sometimes on random calculation, we could expect that results obtained with Openpilot can have differences for a same scenario. These differences can be quantified.
- **Measuring the precision of the AI:** Since the AI of Openpilot is used to get perception and take decision, some of the steps of the algorithm can be evaluated. An example is comparing detected distances between what Openpilot detects and the real ground truth distance that can be obtained from SCANeR studio. The Openpilot algorithm works with a messaging library (cereal) that let us extract some log information (such as the measured distance) so that we can evaluate it to validate the system.

3.5.3.3 Scenario definition

In order to have a complete process of validation of the AI, there is a need of generating the corresponding scenarios to play with Openpilot. The generated scenarios aim at two main objectives :

- **Representativity:** The scenarios should be representative of a real situation that can occur for Openpilot and the response of the system should be the same as it would be in the real situation.

- **Coverage:** The generated scenarios should be as much as exhaustive for having the best coverage of the occurring situations (more precisely: the ODD (Operational Design Domain) of the system should be described and the chosen scenario should cover this ODD)

In the case of this POC, there has been three sources of scenario that are described below:

- **Test plan generated scenario:** Starting from a logical description of a simple situation that can occur, a scenario is repeated varying some parameters to create a wide range of concrete tests corresponding to this situation. An example would be the test of the emergency braking function of Openpilot that detects any obstacles on the path and brakes. Varying the type of obstacles, the speed of the vehicle and the weather conditions of the scenario can lead to have various scenario cases to test. This kind of scenario better fits to testing the robustness and the repeatability of the AI system.
- **Long drive scenario:** An other type of scenario that has been generated correspond to having the system in a wider environment with several external actors. The various actors trigger more complex situations that can challenge the AI system. This kind of scenario has been generated to test Openpilot in SCANeR Studio for several kilometers in order to detect some situations that would lead to accidents. This kind of scenario responds to the coverage need of the scenario generation.
- **Standard validation scenario (UTAC):** The third source of scenario used to cover as much as possible the ODD is the standard scenario issued from an official protocol test. The main advantage is scenarios given in these type of protocols are concrete, so few parameters can be modify and it is possible to evaluate the system according to a simple reference.

Table 11 summarises the scenario definition chosen in this POC regarding the Generic PRISSMA's evaluation framework with one example for each type of scenario.

Table 11: Scenario definition for each type of scenario in the POC 3

	Test plan scenario	**Long drive scenario**	**Standard validation scenario**
Scene	Infrastructure: Portion of N104 Actors: Ego and leading vehicles	Infrastructure : Straight long road Actors: Ego and diverse type of objects (cars, pedestrians, obstacles)	Infrastructure: Highway Actors: Ego and leading vehicle
Event	The leading vehicle does a cut-out manoeuvre at the middle of the scenario	Objects are crossing in front of the ego vehicle	The leading vehicle is braking from 50km/h to 0km/h
Action	The ego vehicle must correctly evaluate the distance to the targets and adapt the cruise speed	The ego vehicle should detect the objects and brake to avoid collision	The ego vehicle must brake to avoid collision
Criteria	The estimated distances and speed must be accurate and no collision should happen.		

3.5.3.4 Scenario configuration, algorithms and ground truth

- **Scenario configuration:** For each example of this POC, the scenarios have been configured in SCANeR Studio to respect a defined ODD and OEDR. The terrains designed in SCANeR Studio respect the highway ODD of the three cases. Varying the weather condition has been done using the weather parameters of SCANeR to chose between normal weather, snowy weather or foggy weather.
- **Algorithms:** Relying on Unreal Engine to compute the camera images lets us test the computer vision part of Openpilot and the robustness of the bridge between SCANeR Studio and Openpilot.
- **Ground truth:** The analyzing tools of SCANeR Studio (and using the export channels) lets us extract the correct position and distances between vehicles. These measurements are the core values that represent the ground truth which is compared with what Openpilot evaluates.

3.5.3.5 Scenario execution and evaluation metrics for the three generated scenarios

The following paragraphs summarize the work that has been done to execute (Layer 2 of the Generic PRISSMA's evaluation framework) each scenario defined in the previous paragraph and then the resulting evaluation metrics (Layer 3 of the Generic PRISSMA's evaluation framework)

Test plan generated scenario For the scope of this POC, we studied one scenario where the ego follows a vehicle that does a cut-out manoeuvre at the middle of the scenario. A specific architecture has been implemented to replay and activate Openpilot exactly the same way during several executions. The test plan replayed 7 times each scenario while varying the weather condition (one normal condition, one fog condition and one snow condition). The resulting detection shows disparity in distance precision regarding the weather condition.

Figure 151 shows the obtained result.

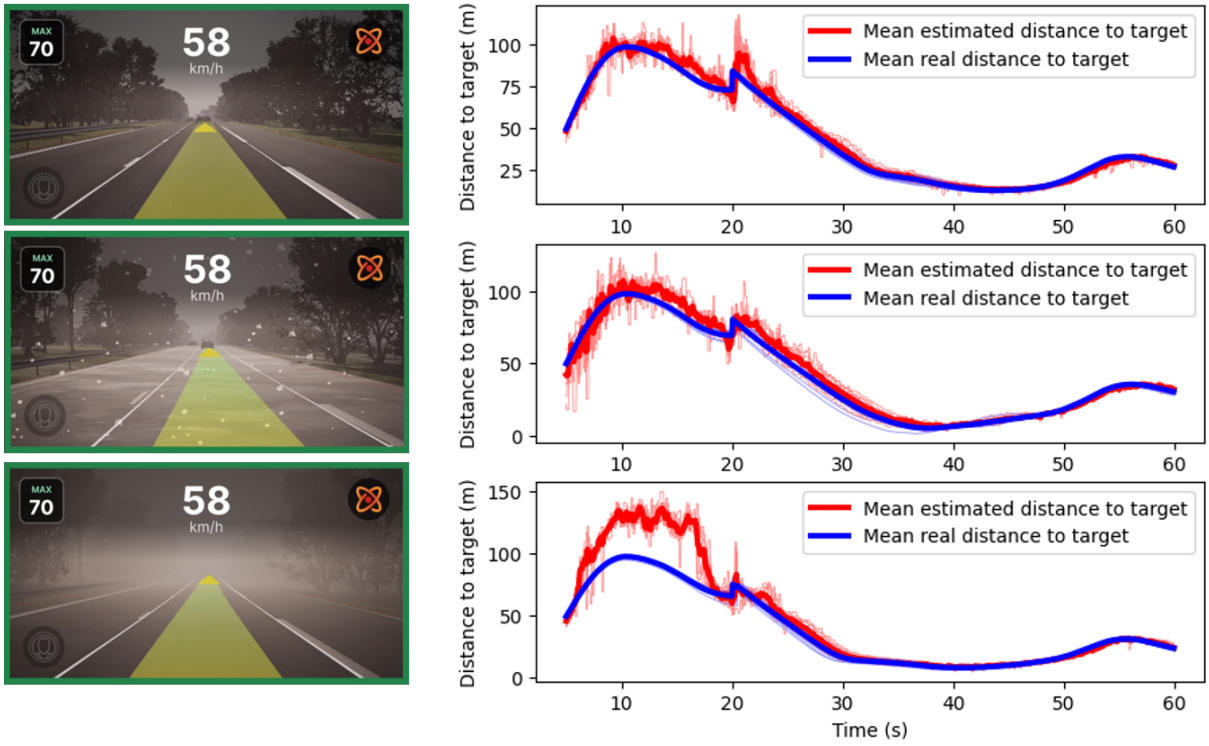


Figure 151: The measured distances by Openpilot for several replays (in blue : the real distance, in red : Openpilot's estimation)

When replaying 10 times the same scenario, the behaviour of Openpilot is slightly different, hence a demonstration of the lack of repeatability of the AI in this case. Table 12 shows the obtained metrics for 10 replays for each weather condition. This kind of obtained results should be tested for each scenario case to ensure a complete validation of the AI with fixed threshold to validate or not the system.

Table 12: Obtained metrics concerning the distance to target for a test plan generated scenarios with 10 replays (only the first 4 replays are displayed)

Environment	**Normal**					**Snow**					**Fog**				
Accuracy (%)	3.0	2.4	2.4	3.0	...	5.4	12.1	24.3	7.4	...	11.4	10.9	12.3	11.4	...
	Mean : 2.9					Mean : 10.7					Mean : 11.8				
RMSE (m)	4.6	3.6	4.7	4.8	...	6.3	6.7	5.6	6.7	...	16.5	16.1	15.5	17.1	...
	Mean : 4.55					Mean : 6.51					Mean : 16.35				
Max err (m)	20.5	18.9	25.2	29.9	...	29.0	46.4	30.1	52.4	...	52.0	60.2	57.4	58.3	...
	Mean : 25.9					Mean : 36.1					Mean : 57.5				
Std Dev (m)	2.52					4.12					3.52				

An important part of the work still is to establish the scenario database to do such tests (the ODD must be precisely defined for Openpilot and the coverage of the corresponding scenarios) and to fix the corresponding threshold to validate the system. Moreover, the number of replays to be done to have a reliable enough result should be correctly quantified. In the case of this POC, doing 7 replays is sufficient to show that the variability is wide.

Long drive scenario Another test has been done with Openpilot on a long drive (in the virtual N104 terrain) where the driving system has been tested in front of different actors to test the perception efficiency of Openpilot. Such a scenario lets the system explore the diversity of situations that can occur. In this case, the chosen test lets Openpilot meet different kind of objects

Table 13: Obtained metrics concerning detection behaviour of Openpilot for different objects

Detected object	Detection	Detection max distance	Braking decision	Distance to brake
Peugeot 308 (Grey)	✔	86 m	✘	-
Holden Astra (Red)	✔	87 m	✔	46 m
Opel Vivaro (Spring green)	✔	79 m	✔	52 m
Citroën C3 (Green)	✔	86 m	✔	51 m
Farm cow	✔	123 m	✘	-
Farm horse	✔	106 m	✘	-
Sewer plate	✔	120 m	✘	-
Tarmac hole	✘	-	✘	-
Speed limit sign	✘	-	✘	-
Speed bump	✘	-	✘	-
Crosswalk markings	✘	-	✘	-

Table 13 shows the kind of criterias to evaluate the perception of different objects. Through a long drive scenario, the list of objects to be detected can be diversified and the resulting detection distances can be more accurate through numerous replays.

Standard validation scenario Another scenario studied is from the Euro NCAP protocol about Assisted Driving Grading lately published in January 2023. We chose the Car-to-Car Rear braking scenario where a leading vehicle is driving at 50kph and the ego vehicle is following with the ACC engaged.

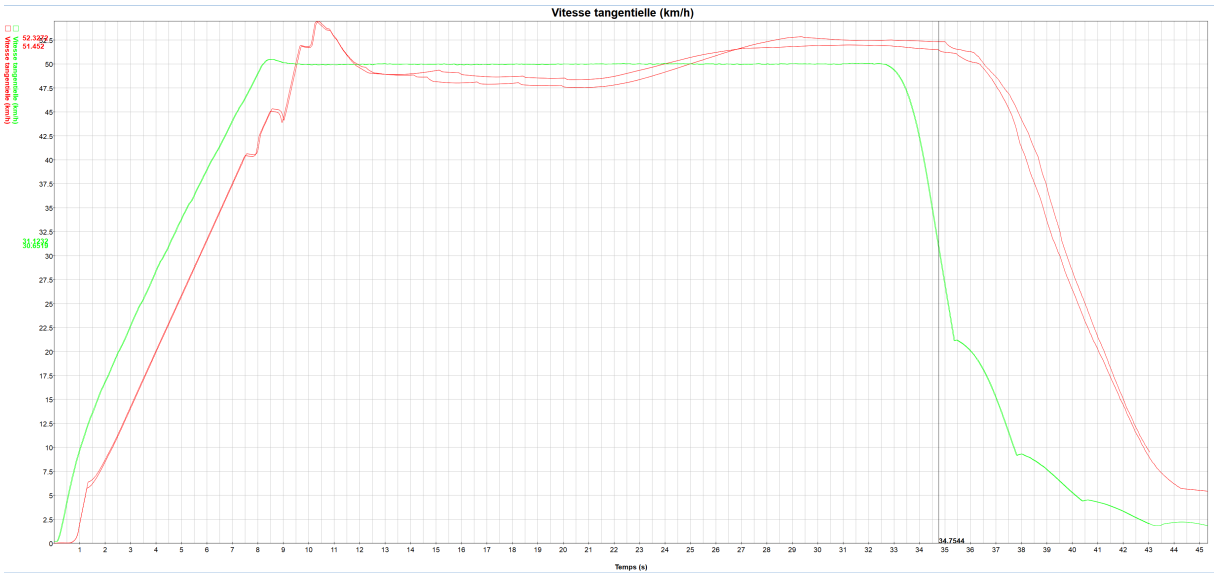


Figure 152: Ego Speed comparison in red, in green the target vehicle speed

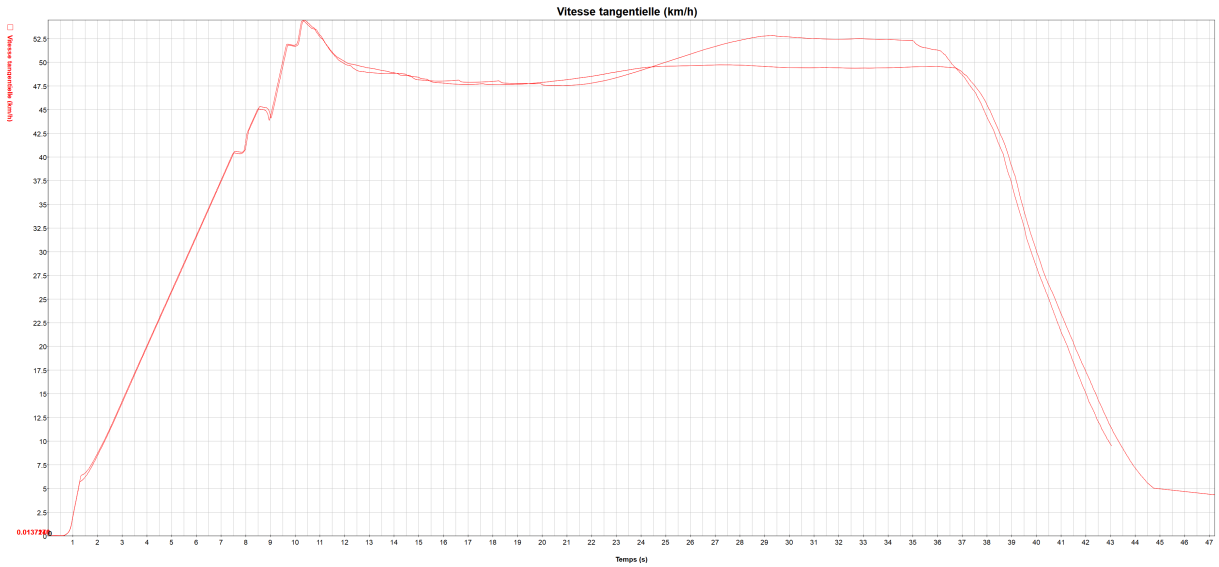


Figure 153: Ego speed comparison in red

On the figures above the ego speed is displayed in red, they represent three different runs. We can observe big differences in the figure 153 when the ego vehicle is approaching the leading vehicle, these differences lead to a lower impact speed for the best case. The same thing happens in figure 152 but the gap is smaller. Then, for the second test scenario, the Openpilot behavior is different despite the test case is exactly the same.

3.5.4 POC 4

In this POC, that involves INRIA and Transpolis, we present the use of a new framework based on Augmented Reality (AR) [156] as a tool to improve testing and validation of AI-based algorithms. In particular, this POC shows how simulated environments and road actors (vehicles, pedestrians, etc.) can play a fundamental role in the validation process by intervening also in real testing to enrich otherwise simple scenarios. Virtual pedestrians or vehicles are indeed easier to operate and can offer rich and active behaviours. Furthermore, replacing real road actors with their virtual counterparts allows considering safety critical situations and scenarios presenting high collision risks without the corresponding danger. Virtual scenarios are also repeatable and this is a key feature to reproduce experiments. Finally, as any element in the considered scenario can be either real or virtual, AR testing offers a smooth transition from simulation to actual testing. For these reasons, the proposed AR framework, by exploiting both simulated and real systems, can be a fundamental testing solution for the validation of advanced automotive software. In what follows we describe the different components of our POC, that closely follow the three layers of the protocol presented in section 3.2.2.

3.5.4.1 System and Environment

Augmented Reality module - Our augmented reality system comprises four essential modules:

1. **Virtual Environment**: This module features a replica of the experimental vehicle within a digital environment.

2. **Synchronization Module**: Responsible for constantly updating the virtual twin's position and state to match the real vehicle.
3. **Sensor Emulation**: This module generates simulated outputs from virtual sensors and seamlessly integrates them with the actual sensor outputs.
4. **Visualization Component**: Designed to assist testers in comprehending the AR scenario.

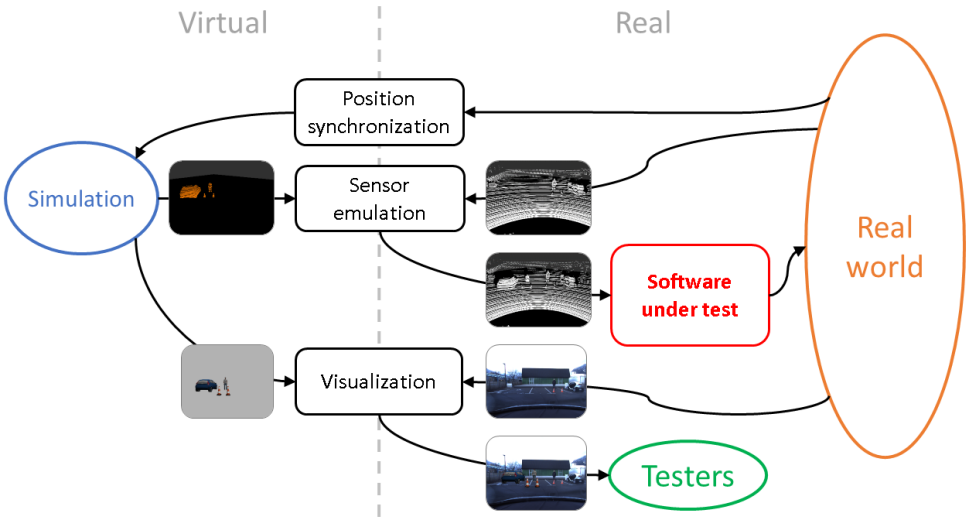


Figure 154: Augmented Reality framework.

Fig. 154 illustrates a schematic representation of our software framework. The real vehicle's sensor data dictates the pace of the virtual world. Consequently, all modules must operate in real-time, with execution times significantly shorter than the sensor data update frequency. This imposes a significant challenge on the design and implementation of the solution.

To begin, we create a virtual environment that is geo-spatially anchored using GPS coordinates. Within this environment, we include a virtual counterpart of the test vehicle and any virtual elements required for augmented reality. There are no restrictions on the nature, shape, and dynamic model of the virtual elements; they can be as intricate as necessary for testing purposes. Other than the virtual vehicle and test elements, the virtual environment remains empty, eliminating the need for a background, ground plane, or representation of the actual test site. This feature facilitates the deployment of our method in various locations with ease.

To maintain accurate synchronisation, the real-time position of the test vehicle must be continually estimated using a precise localisation system. This estimated position is then used to position the virtual twin within the virtual environment. This straightforward synchronisation approach offers flexibility and allows for the deployment of the AR system without requiring any physical installation.

The virtual twin of the vehicle is equipped with a set of sensors that emulate the behavior of the actual vehicle's sensors. Achieving accurate, realistic, and real-time sensor emulation is crucial. While the framework is versatile, our proof of concept (POC) focuses on LiDAR sensors. These emulated LiDARs must provide object detection data in a point cloud format for virtual objects. The generated point clouds are merged with the data from each corresponding physical sensor. The merging process is a pivotal element of our AR framework: it must operate in real-time despite the substantial data volume, adhere to a realistic sensor model, and faithfully reproduce occlusions between the real and virtual worlds. For each sensor, this merging process yields a new point cloud that represents the AR perception, which can then be transmitted to

the software of the vehicle under test, replacing the actual sensor's point cloud. This seamless integration ensures that the use of AR does not disrupt the software under test.

Additionally, the virtual twin of the vehicle is equipped with a set of cameras mimicking those on the actual vehicle. These virtual cameras capture images of virtual objects within the simulator. These images are subsequently combined with the output from each corresponding physical camera. This merging process produces a new image for each camera, representing the AR perception. This approach provides testers with a convenient means of visualising the AR scene. Although a photorealistic simulator and sophisticated image merging functions can be used for AR perception with cameras, a simpler simulator with basic graphics and merging procedures suffices for visualisation purposes.

INRIA's autonomous Zoé platform and Transpolis testing site - Tests were conducted with Inria's Renault Zoé autonomous vehicle (Figure 155). It features a Velodyne HDL-64 on the roof, 3 Ibeo Lux LiDARs in front and 1 in the rear, Spectra SP90 RTK Dual antenna GNSS, Xsens IMU for vehicle velocity and orientation, a stereo camera, and 2 IDS cameras. LiDAR data is synchronised using the IBEO fusion box. The perception system runs on a trunk-mounted PC with an Nvidia Titan X GPU, while automation processes are integrated into the vehicle.

Figure 155: Transpolis, headquartered at Les Fromentaux, is a cutting-edge testing ground for future urban mobility, where vehicles and infrastructures undergo daily trials with advanced equipment and technologies [32].

The tests were held at the Transpolis testing facility at Les Fromentaux, mainly on a long boulevard in the City area. This urban testing ground spans 30 hectares with a complex street network covering 12 kilometres, including two six-lane boulevards. It features sections with intersections, crossroads, parking slots, bus, and cycle lanes, accommodating diverse transportation needs. It corresponds to the aimed ODD of the validated AV: an urban area with velocities limited to 30 km/h.

The facility offers adjustable infrastructure, EV charging stations, dynamic signs, and various road features for comprehensive testing. The main boulevard intersection, where the experiment occurred, is a 6 by 6 lanes intersection. The experiments aimed to assess ego-vehicle behaviour, focusing on autonomous control, collision avoidance, and obstacle response.

3.5.4.2 Scenarios description

To test our system, we have defined some critical scenarios where the (real) ego-vehicle has to cross an intersection where the presence of other (virtual) vehicles may lead to collisions or near-collisions. In particular, occlusions problems have been reproduced to test the perception algorithms. More specifically, the scenarios include the ego-vehicle and four virtual actors: a

fire truck, a bus, and two cars. The test unfolds in two phases, as also depicted in Figures 156 and 157. The scene, event, action, and criteria of the scenarios are here detailed.

1. The first **scene** is the ego-vehicle approaching an intersection in an urban area while a bus and a car are simultaneously approaching from the right. Depending on initial conditions, the **event** is a collision or near-collision occurring at the intersection. Based on the information provided by the perception framework, the **action** that the ego-vehicle can take is an emergency brake to try avoiding the collision. The car is initially occluded by the bus, making it invisible to the LiDARs. If the ego-vehicle can avoid the bus without braking, it might successively collide with the car.

2. The second **scene** is the ego-vehicle approaching the same intersection from the opposite lane. Similarly to the first scene, a fire truck and a car approach from the right leading to an **event** that can be a collision or a near-collision depending on the timing. The possible **action** from the ego-vehicle is again represented by an emergency braking. However, this time, contrary to the first scene, an emergency brake in front of the fire truck could result in a collision with the car.

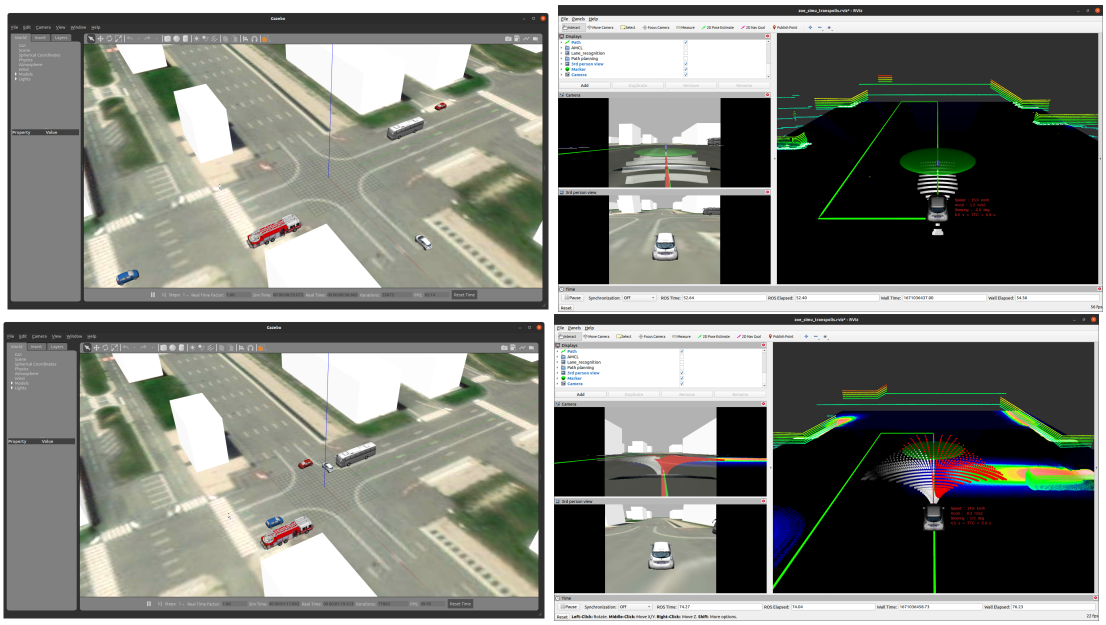


Figure 156: Gazebo (left) and ROS RViz (right) screenshots. The ego vehicle is the visible Renault Zoé crossing the intersection from bottom right to top left on Gazebo and at the center of the ROS RViz visualization.

It is important to note that the Zoé's behaviour is not predicted in advance while virtual obstacles follow controlled trajectories. Our **criteria** for a successful scenario is that the Zoé's behaviour is impacted by its interactions with other actors (collision avoidance, emergency breaking, collision).

3.5.4.3 Perception Algorithms

The depicted perception module relies on the use of probabilistic occupancy grids and Bayesian fusion techniques. These methods are employed to create precise and detailed representations of both occupancy and velocity.

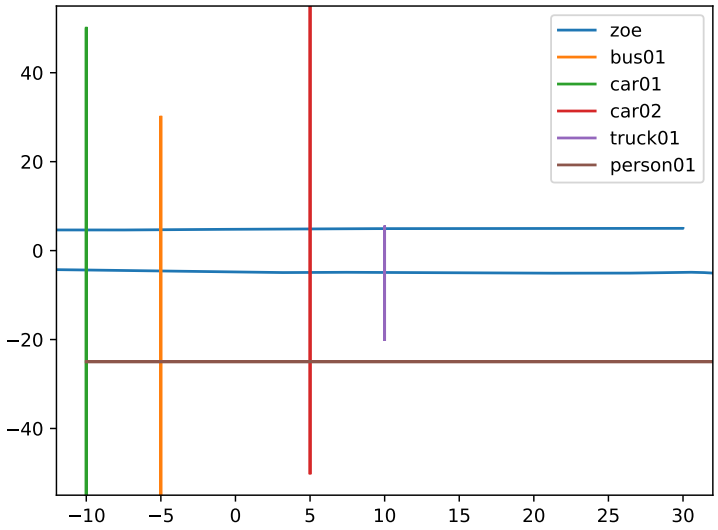


Figure 157: Plot of the actors position in the simulator during a scenario execution, abscissa is the x axis of the simulator world frame and ordinate is the y axis. The ego vehicle, the Zoé, crosses twice the intersection from right to left ($x = 30$ to $x < -10$, trajectory above) and left to right ($x < -10$ to $x = 30$, trajectory bellow). Bus01 and Car01 start their trajectories in the top side of the image ($y > 20$) and finish them at the bottom ($y < -40$). Car02 and truck01 drive in the opposite direction, starting bottom ($y < -40$) and finishing top ($y > 20$).

In our POC, we illustrate the perception module using the CMCDOT framework, developed by Inria [157]. CMCDOT serves as a comprehensive method inspired from Bayesian occupancy filter framework and integrates abstract states and a conditional Monte Carlo technique. This integration optimises velocity estimation focusing on relevant areas within the environment. It covers various states, including static, dynamic, free, and unknown states. Each of these states is associated with dedicated models, and the method explicitly takes into account uncertainties and sensor coverage.

The CMCDOT module operates with three primary inputs:

- LiDAR point cloud data
- Observed occupancy grids or multiple grids from various sensors
- Ego-vehicle odometry

As a result of processing these inputs, the CMCDOT module generates the following output grids:

1) **Filtered occupancy grid**: The instantaneous occupancy data from each sensor modality undergoes both temporal and spatial filtering. The CMCDOT occupancy filter, employing a Bayesian update model, conducts local occupancy filtering while also tracking changes in occupancy using a Monte Carlo approach. This process yields filtered occupancy grids and velocity grids.

2) **Velocity grid**: This grid provides a visual representation of stationary elements (shown in white) and dynamic obstacles (represented in various colors). The intensity of each color indicates the speed of the obstacle, while the color itself signifies its direction of motion.

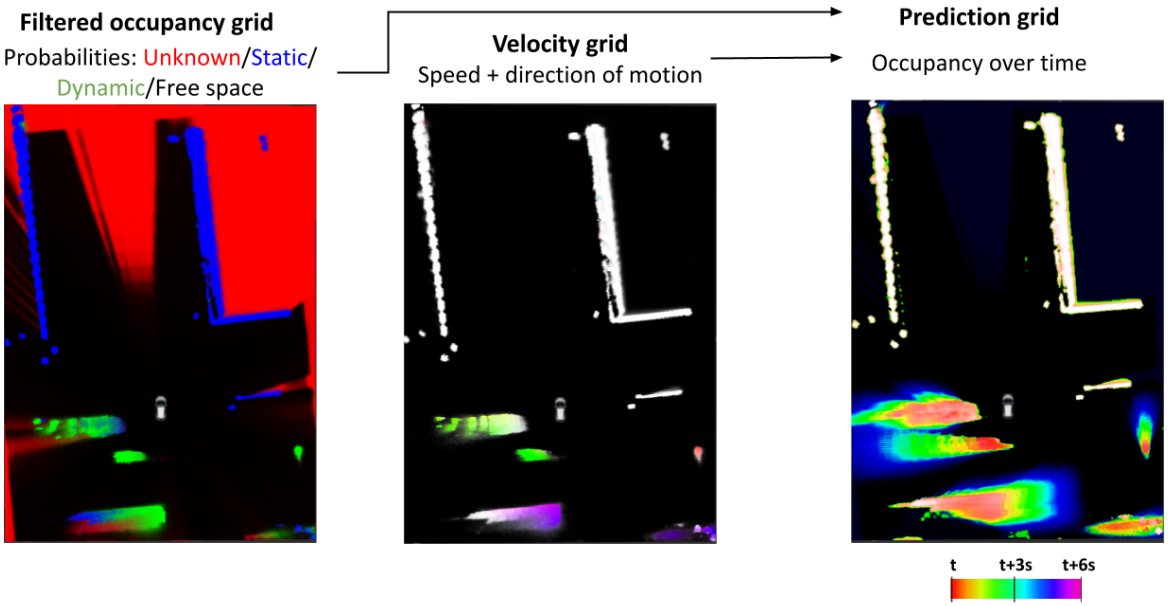


Figure 158: The DWA (Dynamic Window Approach) grid forecasts occupancy by projecting cells based on velocity, handling noise, and combining sensor data. It provides foresight into future occupancies, merging path planning and obstacle avoidance to enhance comprehension of dynamic environments.

3) **DWA (Dynamic Window Approach) grid**: This grid is a vital predictive model. It uses essential data, including occupancy probabilities and estimated velocity (Figure 158), to project cells based on velocity, representing movement. Cells are divided into particles with specific accelerations and angular velocities to handle noise and uncertainties.

This grid functions as a probabilistic distribution, offering insights into future occupancies within three seconds. It combines occupancy data from various sensors, creating a unified representation that evolves over time. The LiDAR-derived velocity grid remains the most accurate motion estimation. The DWA prediction grid visualises occupancy predictions over time, with static objects in white and moving ones in colored representations based on estimated arrival times. To ensure safety near moving objects, it introduces significant uncertainty, resulting in predicted occupancy clouds, accounting for potential variations and uncertainties in object movement.

3.5.4.4 Ground truth generation

The evaluation method of our results is based on a similarity evaluation of the perception output with a ground truth using the metric from [27]. We generated ground truth occupancy grids for the perception module (CMCDOT) using a satellite image of Transpolis facility and actors data from the AR simulator, stored in rosbags. By using the augmented reality framework, during the scenarios, all dynamic objects are virtual actors. They are controlled by the simulator, therefore, their state (position, orientation, speed, footprint) is known for each moment of the test. They are geolocalized on the satellite image of Transpolis facility and their footprint is drawn on it (coloured boxes in 159 and 160).

The remaining elements of the environment is the permanent environment of Transpolis.

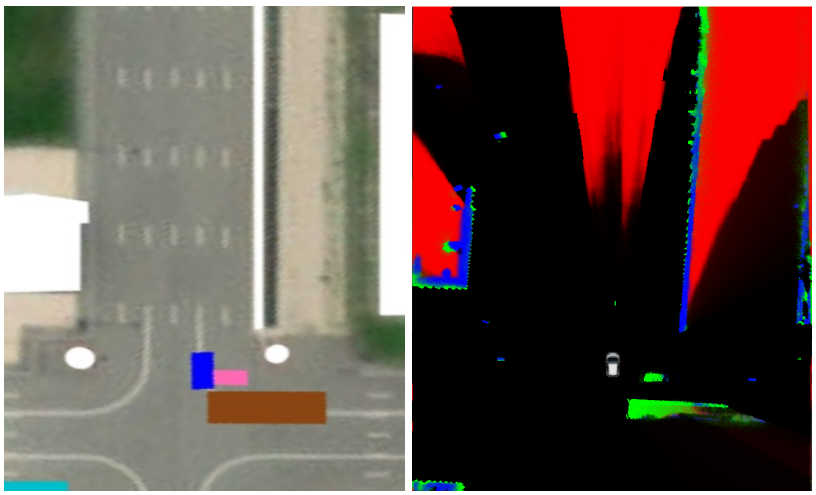


Figure 159: A bus approaches the ego-vehicle (blue box in the left image) but stops and avoids collision. However, an occluded car from behind this approaching bus collides with ego-vehicle. All dynamic objects in the scenario are virtual actors.

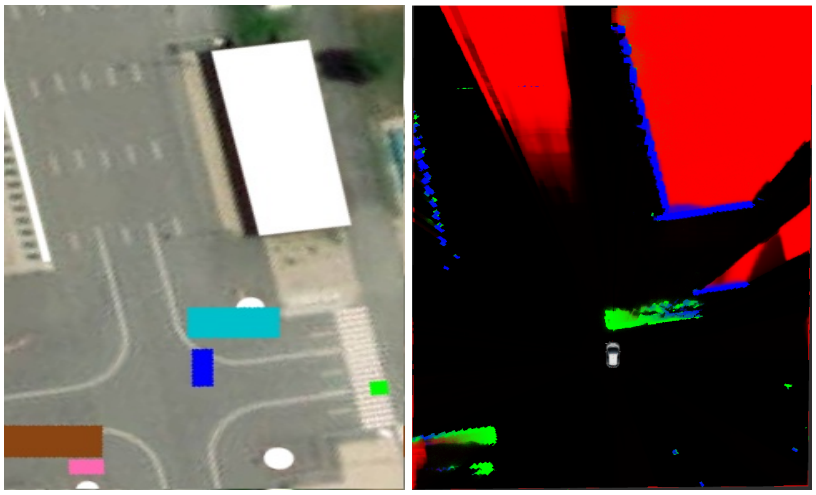


Figure 160: The ego-vehicle (blue box in the left image) applies emergency brakes and avoids collision with a fire truck. All dynamic objects in the scenario are virtual actors.

We manually labelled a satellite image of the facility to have a segmentation between static obstacles (white boxes in 159 and 160) and free space (satellite view background in 159 and 160)

In the same way we used the AR framework to merge a real environment with dynamic virtual actors during the test scenario, we generated a corresponding ground truth by merging a static ground truth of transpolis facility with the ground truth data of the actors from the simulator. Finally, using the Zoé geolocation during the test scenarios, an approximation of ground truth occupancy grid, fitting the dimensions of the CMCDOT grids, is cropped from the satellite image around the vehicle, corresponding to the specific timestamp in the experimentation.

During a scenario execution, the CMCDOT updates and publishes its inference of the environment every 100ms, we create a ground truth for each of this inference occupancy grids.

Topic name	Topic type	Description
/zoe/velodyne_points	sensor_msgs/PointCloud2	Point clouds of the Velodyne HDL-64 LiDAR
/zoe/lux_right /zoe/lux_center /zoe/lux_left /zoe/lux_rear	sensor_msgs/PointCloud2 sensor_msgs/PointCloud2 sensor_msgs/PointCloud2 sensor_msgs/PointCloud2	Point clouds of the front right, front center, front left and rear Ibeo Lux LiDARs
/temp/zoe/velodyne_packets	velodyne_msgs/VelodyneScan	Raw data measurements from the Velodyne HDL-64
/zoe/classified_cloud	sensor_msgs/PointCloud2	Merged point cloud from the 5 LiDARs with classification of ground
/zoe/us_right /zoe/us_center /zoe/us_left	sensor_msgs/Range sensor_msgs/Range sensor_msgs/Range	Front ultrasonic range sensors
/zoe/sp90_fix /zoe/sp90_time_reference /zoe/fix /zoe/fix_common /zoe/raw_fix	sensor_msgs/NavSatFix sensor_msgs/TimeReference sensor_msgs/NavSatFix gps_common/GPSFix sensor_msgs/NavSatFix	Satellite localization of the Zoé
/zoe/camera_front/image_rect_color	sensor_msgs/Image	Images stream of the front camera
/zoe/camera_front/camera_info	sensor_msgs/CameraInfo	Information about the camera and its calibration
/zoe/imu/mag	sensor_msgs/MagneticField	Magnetic compass of the Zoé IMU
/zoe/imu/data	sensor_msgs/Imu	IMU data (orientation, angular velocity and linear acceleration)
/navigation/dwa_result	dwa_dynamic_planner/Trajectory	Current trajectory of the Zoé generated by the local planner
/navigation/planner_result /navigation/planner_status	dwa_dynamic_planner/PlannerResult dwa_dynamic_planner/PlannerStatus	Status information on the local planner
/zoe/velocity_grid	e_motion_perception_msgs/VelocityGrid	Grid of velocity vectors of the dynamic cells
/zoe/state_grid	e_motion_perception_msgs/FloatOccupancyGrid	Grid of filtered probability of occupied, dynamic, static and unknown
/zoe/occ_grid	e_motion_perception_msgs/FloatOccupancyGrid	Grid from one LiDAR point cloud of probabilities of occupied and unknown. Output of the LiDAR sensor model
/zoe/control/refs	ros_zoe_msgs/ControlRefs	Throttle, brake and steering commands sent to the hardware controller of the Zoé for automated driving
/tf /tf_static	tf2_msgs/TFMessage tf2_msgs/TFMessage	Dynamic and static transforms of the frames of the Zoé
/zoe/velocity /zoe/speed	geometry_msgs/TwistStamped geometry_msgs/TwistStamped	Velocity of the Zoé
/zoe/pose	geometry_msgs/PoseWithCovarianceStamped	Filtered Pose of the Zoé by a Kalman filter. Relative to a *world* fixed frame
/gazebo/set_model_state /gazebo/link_states /gazebo/model_states	gazebo_msgs/ModelState gazebo_msgs/LinkStates gazebo_msgs/ModelStates	States and status of the virtual Actors in Gazebo
/gazebo_scenario/rosparam	std_msgs/String	JSON serialization of all ROS parameters of the Zoé
/gazebo_scenario/scenario	std_msgs/String	JSON serialization of the scenario description and parameters

Table 14: Topics recorded during the experiment using the tool rosbag.

3.5.4.5 Data recording

To record and store the data from the experiment, we used the tool rosbag designed for ROS. Firstly, The rosbag recorder listen to a list of requested topics and read every messages sent (i.e. messages exchanged by the ROS nodes such as the LiDAR driver, CMCDOT node, GPS driver, local planner node). It stores a timestamped serialization of each message in a binary file called bag. In a second time, the rosbag player can read the messages stored in the rosbag and publish them on their respective topics to replay the recording. The player uses the timestamp of the messages to publish them in order and at the right simulated time. Table 14 shows a list of the topics recorded during the experiments.

3.5.4.6 Integration with Robot Operating System (ROS) framework

The software architecture of the Zoé has been designed using the robotic framework ROS (Robot Operating System), under version *melodic*. When using ROS, the software components of a robotic system are separated in nodes that communicate by sending messages on typed topic. For example, the LiDAR driver sends point cloud messages on the lidar topic, then the CMCDOT node listen to this topic to read the LiDAR measurements and it publishes its output grids on their respective topics.

ROS provides several useful tools for data visualization, logging and debugging, package building, simulation (Gazebo) and data recording and storage.

3.5.4.7 Evaluation and validation of the perception

The third and last layer of the protocol proposed in Section 3.2.2 is the evaluation and validation stage based on the identified metrics. By exploiting the generated ground truth, the metric introduced in [27] and presented in Section 3.3, may be adopted to assess the correspondence

between the perception module's output and the ground truth. While conducting the experiments at Transpolis, we used the CMCDOT as perception module and all its occupancy grids were recorded, along with the necessary data for constructing the ground truth (obtained from the simulator). For each occupancy grid produced by the CMCDOT, we generate a corresponding ground truth with the same parameters as those applied to CMCDOT's occupancy grids. Subsequently, the metric from [27] can be used to evaluate the similarity between those grids. By performing this procedure on a significant number of scenarios we can statistically evaluate perception performance within the scenario context (i.e. crossing of an intersection in an urban area). As described in [27], the metric is suitable for a validation process. It assesses the similarity by considering the behavior of the AV navigating through the grids, effectively evaluating how closely driving using the perception grid aligns with driving using the ground truth.

4 Final Considerations

In conclusion, this deliverable constitutes a fundamental pillar of the PRISSMA project. It provides a comprehensive overview of the testing, evaluation, and validation aspects of functions, components, systems, and system-of-systems for automated driving. Beyond addressing the evaluation and validation of AI-based systems and components, this deliverable also delves into the complex subject of verifying and validating tools and models used in evaluation and validation platforms. The methodology proposed in this deliverable complements the state of the art and expands the spectrum of existing evaluation and validation methodologies.

The PRISSMA methodology is cross-cutting and encompasses the creation of a taxonomy for generating an Operational Design Domain (ODD) used to define relevant and representative test scenes and scenarios. This ensures a sufficient coverage of encountered situations. The evaluation protocol presented here serves as a structured framework designed to validate the simulation framework, providing an integrated approach to simulation in a homologation process that represents a significant departure from conventional procedures.

This methodology is implemented and applied to several use cases through five proof-of-concepts. This implementation demonstrates the practical application of the evaluation protocol, highlighting the adaptability and versatility of the simulation protocol across various scenarios. Each proof-of-concept illustrates the efficiency of the protocol in assessing the performance of different components and systems using AI, involved in automated mobility services. The application of the PRISSMA methodology aims to provide tools, models, metrics, and procedures ensuring the robustness, safety, and reliability of AI in diverse applications and use cases.

Furthermore, the validation proposal for each component of the simulation framework underscores the rigorous testing and validation procedures employed. The validation results serve as evidence of the framework's ability to accurately simulate real-world scenarios, reproducing the complexities and nuances encountered during open-road testing, XIL experiments, and track tests conducted in WP3.

In the future, this deliverable will serve as a foundation and a pathway for the improvement, optimisation, and extension of evaluation and validation protocols in simulation environments. The collaborative efforts involved in its development reflect our commitment to ensuring the safety, efficiency, and progress of automated and autonomous vehicles, promoting innovation while adhering to strict standards of quality and compliance.

References

[1] S. Hakuli and M. Krug, "Virtuelle integration," in Handbuch Fahrerassistenzsysteme, 2015, pp. 128–138. [Online]. Available: https://doi.org/10.1007/978-3-658-05734-3_8.

[2] 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

[3] R. G. Sargent, "Verification and validation of simulation models," in Proceedings of the 2010 Winter Simulation Conference, Baltimore Maryland, December 5 - 8, 2010, USA. IEEE, December 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5679166

[4] P. J. Durst, D. McInnis, J. Davis, and C. T. Goodin, "A novel framework for verification and validation of simulations of autonomous robots," Simulation Modelling Practice and Theory, vol. 117, February 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22000235

[5] P. A. Fonseca i Casas, "Continuous process for validation, verification, and accreditation of simulation models," Mathematics, vol. 11, February 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22000235

[6] B. Danquah, S. Riedmaier, and M. Lienkamp, "Potential of uncertainty quantification and statistical model verification and validation in automotive vehicle dynamics simulations: a review," Vehicle System Dynamics, International Journal of Vehicle Mechanics and Mobility, vol. 60, pp. 1292–1321, December 2020. [Online]. Available: https://www.tandfonline.com/doi/10.1080/00423114.2020.1854317

[7] A. M. Alaa, B. van Breugel, E. Saveliev, and M. van der Schaar, "How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models," July 2022. [Online]. Available: https://arxiv.org/pdf/2102.08921.pdf

[8] M. Jiralerspong, A. J. Bose, I. Gemp, and al, "Feature likelihood score: Evaluating the generalization of generative models using samples," in 37th Conference on Neural Information Processing Systems (NeurIPS 2023)., 2023. [Online]. Available: https://arxiv.org/pdf/1511.01844.pdf

[9] 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

[10] T. Chen, X. Yang, N. Li, T. Wang, and G. Ji, "Underwater image quality assessment method based on color space multi-feature fusion," Scientific Report, Nature, no. 16838, October 2023. [Online]. Available: https://www.nature.com/articles/s41598-023-44179-3

[11] 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

[12] G. Zhang, B. Xu, H.-F. Ng, and L.-T. Hsu, "GNSS RUMS: GNSS Realistic Urban Multi-agent Simulator for Collaborative Positioning Research," Remote Sensing, vol. 13, February 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/4/544

[13] 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

[14] A. WIDNER, V. TIHANYI, and T. TETTAMANTI, "Framework for Vehicle Dynamics Model Validation," IEEE ACCESS, vol. 10, pp. 35 422 – 35 436, April 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9730866

[15] J. Klemmer, J. Lauer, V. Formanski, R. Fontaine, P. Kilian, S. Sinsel, A. Erbes, and J. Zäpf, "Definition and Application of a Standard Verification and Validation Process for Dynamic Vehicle Simulation Models," SAE International Journal of Materials and Manufacturing, vol. 4, no. 1, pp. 743–758, 2011. [Online]. Available: https://www.jstor.org/stable/10.2307/26273812

[16] 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.

[17] V. Punzo, Z. Zheng, and M. Montanino, "About calibration of car-following dynamics of automated and human-driven vehicles: Methodology, guidelines and codes," Transportation Research Part C, vol. 128, no. 2, p. 1035, May 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21001832?via%3Dihub

[18] R. Zong, W. Deng, X. Bai, Y. Wang, and J. Ding, "Traffic modeling based on data-driven method for simulation test of autonomous driving," IEEE Transactions on Intelligent Transportation Systems, August 2023. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2310/2310.15975.pdf

[19] 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

[20] X. Li, K. Kou, and B. Zhao, "Weather gan: Multi-domain weather translation using generative adversarial networks," arXiv preprint arXiv:2103.05422, 2021.

[21] 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

[22] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, and E. Böde, "Criticality metrics for automated driving: A review and suitability analysis of the state of the art," Archives of Computational Methods in Engineering, June 2022. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2108/2108.02403.pdf

[23] A. D. S. Sectional Committee of AD Safety Evaluation, "Automated driving safety evaluation framework version 3.0," Japan Automobile Manufacturers Association, Inc., techreport, December 2022. [Online]. Available: https://www.jama.or.jp/english/reports/docs/Automated_Driving_Safety_Evaluation_Framework_Ver3.0.pdf

[24] G. Wang, J. H. Z. Li, and L. Li, "Harmonious lane changing via deep reinforcement learning," IEEE Transaction on Intelligent Transportation System, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9325948.

[25] J.-S. Wang, "Mais(05/08) injury probability curves as functions of delta v," United States. Report No. DOT HS 813 219.National Highway Traffic Safety Administration, Tech. Rep., May 2022.

[26] S. S. Mahmud, L. Ferreira, S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," IATSS Research, vol. 41, pp. 153–163, December 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0386111217300286?via%3Dihub

[27] J.-B. Horel, R. Baruffa, L. Rummelhard, A. Renzaglia, and C. Laugier, "A navigation-based evaluation metric for probabilistic occupancy grids: Pathfinding cost mean squared error," in IEEE International Conference on Intelligent Transportation Systems, 2023.

[28] A. Godil, R. Bostelman, W. Shackleford, T. Hong, M. Shneier et al., "Performance metrics for evaluating object and human detection and tracking systems," No. NIST Interagency/Internal Report (NISTIR), vol. 7972, 2014.

[29] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," arXiv preprint arXiv:2003.09003, 2020.

[30] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," International journal of computer vision, vol. 129, no. 2, pp. 548–578, 2021.

[31] R. K. Satzoda and M. M. Trivedi, "On performance evaluation metrics for lane estimation," in 2014 22nd International Conference on Pattern Recognition. IEEE, 2014, pp. 2625–2630.

[32] Transpolis, "Les fromentaux test center," 2021, https://transpolis.fr/les-fromentaux.

[33] PFA. (2020) Automated driving safety validation: proposals from the french eco-system. [Online]. Available: https://www.ecologie.gouv.fr/sites/default/files/2020%2001%2009%20-%20autonomous%20driving%20-%20safety%20validation%20-%20french%20views%20-%20Vdef.pdf

[34] 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.

[35] G. Chance, A. Ghobrial, K. McAreavey, S. Lemaignan, T. Pipe, and K. Eder, "On determinism of game engines used for simulation-based autonomous vehicle verification," IEEE Transactions on Intelligent Transportation Systems, June 2022.

[36] P. E. Strandberg, T. J. Ostrand, E. J. Weyuker, W. Afzal, and D. Sundmark, "Intermittently failing tests in the embedded systems domain," in Proceedings of 29th ACM SIGSOFT International Symposium on Software Testing and Analysis Virtual Event USA July 18 - 22, 2020, 2020.

[37] J. Khan, "Autonomous driving validation using game engine technology," Tata Elxsi, Tech. Rep., 2019. [Online]. Available: https://tataelxsi.com/storage/insights/March2021/BlySkLFzuJCrQolOvk1V.pdf

[38] 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

[39] 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.

[40] 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

[41] 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.

[42] J. Schumann, P. Gupta, and Y. Lui, "Application of neural networks in high assurance systems: A survey," in Applications of Neural Networks in High Assurance Systems. Springer, Berlin, Heidelberg, 2010, pp. 1–19.

[43] A. BARCZAK and H. WOŹNIAK, "Comparative study on game engines," Systems and information technology, 2019.

[44] Z. Zhou, J. Song, X. Xie, and al, "Towards building ai-cps with nvidia isaac sim: An industrial benchmark and case study for robotics manipulation," Pre print University of Alberta and NVIDIA, Tech. Rep., July 2023. [Online]. Available: https://arxiv.org/pdf/2308.00055.pdf

[45] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," Sensors, vol. 19, no. 3, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/3/648

[46] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," Sensors, vol. 21, no. 16, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/16/5397

[47] J. Jordon, L. Szpruch, F. Houssiau, A. Weller, and A. Weller, "Synthetic data - what, why and how?" Report from the Royal Society, May 2022. [Online]. Available: https://www.researchgate.net/publication/360462365_Synthetic_Data_--_what_why_and_how

[48] M. Boedihardjo, T. Strohmer, and R. Vershynin, "Covariance's loss is privacy's gain: Computationally efficient, private and accurate synthetic data," CoRR, vol. abs/2107.05824, 2021. [Online]. Available: https://arxiv.org/abs/2107.05824

[49] D. Gruyer, M. Grapinet, and P. Desouza, "Modeling and validation of a new generic virtual optical sensor for adas prototyping," in in IEEE Intelligent Vehicle symposium, 2012, Alcalá de Henares, June 3-7, Spain, 2012.

[50] M. Grapinet, P. Desouza, J. Smal, and J. Blosseville, "Characterization and simulation of optical sensors," in TRA 2012 conference, 23-26 April 2012, Athens, Greece., 2012.

[51] C. N. V. Maturana, A. L. S. Orozco, and L. J. G. Villalba, "Exploration of metrics and datasets to assess the fidelity of images generated by generative adversarial networks," MDPI Applied Sciences, vol. 13, no. 10637, pp. 1–37, September 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/19/10637

[52] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA., 2017. [Online]. Available: https://arxiv.org/pdf/1706.08500.pdf

[53] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," arXiv preprint arXiv:1801.01401, 2018.

[54] S. Ghazanfari, S. Garg, P. Krishnamurthy, F. Khorrami, and A. Araujo, "R-lpips: An adversarially robust perceptual similarity metric," in 2nd AdvML Frontiers workshop at 40 th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023., 2023. [Online]. Available: https://arxiv.org/pdf/2307.15157v2.pdf

[55] M. S. M. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," in 32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada., 2018. [Online]. Available: https://arxiv.org/pdf/1806.00035.pdf

[56] T. Kynkaanniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila., "Improved precision and recall metric for assessing generative models," in In Proceedings of the Thirty-sixth International conference on Machine Learning (ICML), Jun 9th - 15th, Long Beach, USA, 2019, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/0234c510bc6d908b28c70ff313743079-Paper.pdf

[57] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in Proceedings of the 37 th International

Conference on Machine Learning, Online, PMLR 119, 2020., 2020. [Online]. Available: http://proceedings.mlr.press/v119/naeem20a/naeem20a.pdf

[58] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in ICLR 2016, 2016. [Online]. Available: https://arxiv.org/pdf/1511.01844.pdf

[59] D. Brunet, E. R. Vrscay, and Z. Wang, "On the mathematical properties of the structural similarity index," IEEE TRANSACTIONS ON IMAGE PROCESSING, vol. 21, no. 4, pp. 1488–, April 2012. [Online]. Available: https://ece.uwaterloo.ca/~z70wang/publications/TIP_SSIM_MathProperties.pdf

[60] X. LI, H. XU, G. JIANG, M. YU, and T. LUO, "Underwater image quality assessment from synthetic to real-world: Dataset and objective method," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 20, p. 1–23, October 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3624983

[61] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," Proceedings of the IEEE, vol. 102, no. 10, pp. 1470–1484, 2014.

[62] J. Kaiser, J. C. Vasquez Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, R. Kohlhaas, R. Dillmann, and J. M. Zöllner, "Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks," in 2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2016, pp. 127–134.

[63] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in British Machine Vision Conference (BMVC), 2018.

[64] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," The International Journal of Robotics Research, vol. 36, no. 2, pp. 142–149, 2017.

[65] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," Oct. 2018.

[66] M. Hadj-Bachir, T. Bagheri, H. Toss, P. d. Souza, and M. Sanfridson, "Over-the-air automotive radars hardware-in-loop test for development and validation of active safety systems and autonomous cars," in 2023 IEEE International Workshop on Metrology for Automotive (MetroAutomotive), 2023, pp. 205–210.

[67] K. Schuler, D. Becker, and W. Wiesbeck, "Extraction of virtual scattering centers of vehicles by ray-tracing simulations," IEEE Transactions on Antennas and Propagation, vol. 56, no. 11, pp. 3543–3551, 2008.

[68] T. Gomes, R. Roriz, L. Cunha, A. Ganal, N. Soares, T. Araújo, and J. Monteiro, "Evaluation and testing system for automotive lidar sensors," Applied Sciences, vol. 12, no. 24, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/24/13003

[69] 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.

[70] SaPPART. (2017) Cost Action TU1302. Assessment of positioning performance in ITS applications. [Online]. Available: https://reflexscience.univ-gustave-eiffel.fr/lire/ouvrages/sappart-manuel

[71] T. D. Gillespie, "Fundamentals of vehicle dynamics," 1992. [Online]. Available: https://api.semanticscholar.org/CorpusID:107195969

[72] AVSimulation. (2023) Callas dynamic model. [Online]. Available: https://www.avsimulation.com/callas-edition/

[73] W. R. Garrott, P. A. Grygier, J. P. Chrstos, G. J. Heydinger, K. Salaani, J. G. Howe, and D. A. Guenther, "Methodology for validating the national advanced driving simulator's vehicle dynamics (NADSdyna)," SAE Tech., vol. 10, p. 882–894, January 1997. [Online]. Available: https://www.sae.org/publications/technical-papers/content/970562/

[74] R. Wade-Allen, J. P. Chrstos, G. Howe, D. H. Klyde, and T. J. Rosenthal, "Validation of a non-linear vehicle dynamics simulation for limit handling," Journal Automotile Engineering - Proc Instn Mech Engrs, vol. 216, p. 319–327, April 2002. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1243/0954407021529147

[75] W. L. Oberkampf and M. F. Barone, "Measures of agreement between computation and experiment: Validation metrics," Journal Computational Physics, vol. 217, no. 1, pp. 5–36, June 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999106001860

[76] H. Sarin, M. Kokkolaras, G. Hulbert, P. Papalambros, S. Barbat, and R.-J. Yang, "A comprehensive metric for comparing time histories in validation of simulation models with emphasis on vehicle safety applications," ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. 43253, no. 1, p. 1275–1286, August 2008. [Online]. Available: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-abstract/IDETC-CIE2008/43253/1275/330816

[77] K. Salaani, S. J. Rao, J. G. Howe, D. Elsasser, and S. Schnelle, "An applied review of simulation validation approaches on a vehicle dynamics model," NHTSA (National Highway Traffic Safety Administration) Report No. DOT HS 813 177, September 2021. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/57509

[78] A. PARRA, A. J. RODRíGUEZ, A. ZUBIZARRETA, and J. PÉREZ, "Validation of a Real-Time Capable Multibody Vehicle Dynamics Formulation for Automotive Testing Frameworks Based on Simulation," IEEE ACCESS, November 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9268939

[79] 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

[80] 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

[81] 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

[82] T. T. Almeida, L. de C. Gomes, F. M. Ortiz, J. G. R. Junior, and L. H. M. K. Costa, "Ieee 802.11p performance evaluation: Simulations vs. real experiments," in 21st IEEE International Conference on Intelligent Transportation Systems (IEEE ITSC 18), November 4-7, 2018, Maui, Hawaii, USA, 2018. [Online]. Available: https://www.gta.ufrj.br/ftp/gta/TechReports/itsc18.pdf

[83] P. Sandino, L. H. Gonsioroski, R. M. L. Silva, and A. dos Santos, "Performance of vehicle-to-vehicle communication using ieee 802.11a: A measurement study."

[84] S. Gonzalez and V. Ramos, "A simulation-based analysis of the loss process of broadcast packets in wave vehicular networks," Wireless Communications and Mobile Computing, vol. 2018, pp. 656–664, 2018. [Online]. Available: https://www.researchgate.net/publication/328341326_A_Simulation-Based_Analysis_of_the_Loss_Process_of_Broadcast_Packets_in_WAVE_Vehicular_Networks

[85] M. A. A. EL-GAWAD, M. ELSHARIEF, and H. KIM, "A comparative experimental analysis of channel access protocols in vehicular networks," IEEE ACCESS, October 2019. [Online]. Available: https://ieeexplore.ieee.org/ielx7/6287639/8600701/08868072.pdf?tag=1

[86] M. N. Tahir and M. Katz, "Performance evaluation of ieee 802.11p, lte and 5g in connected vehicles for cooperative awareness," Engineer Report, Wiley, October 2021. [Online]. Available: https://web.archive.org/web/20220528051737id_/https://onlinelibrary.wiley.com/doi/pdfdirect/10.1002/eng2.12467

[87] R. MOLINA-MASEGOSA, J. GOZALVEZ, and M. SEPULCRE, "Comparison of ieee 802.11p and lte-v2x: An evaluation with periodic and aperiodic messages of constant and variable size," IEEE ACCESS, July 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9133075

[88] 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.

[89] 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.

[90] 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.

[91] 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.

[92] R. Chandler, R. Herman, and E. Montroll, "Traffic dynamics: studies in car following, operation research," 1958.

[93] P. G. Gipps, "A behavioural car-following model for computer simulation," Transportation Research Part B: Methodological, vol. 15, no. 2, pp. 105–111, 1981.

[94] R. Wiedemann, "Simulation des straßenverkehrsflusses. schriftenreihe heft 8," Institute for Transportation Science, University of Karlsruhe, Germany, 1974.

[95] 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.

[96] 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.

[97] 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.

[98] 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.

[99] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," Transportation Research Record, pp. 82–82, 1992.

[100] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," Journal de physique I, vol. 2, no. 12, pp. 2221–2229, 1992.

[101] 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.

[102] J. Martin, A. Grandjean, S. Prabakaran, B. Dibaj, and B. Soualmi, "Addressing SOTIF requirement for AD/ADAS through long-drive simulations," in Driving Simulation Conference. Strasbourg, France: Driving Simulation Association, Sep. 2022. [Online]. Available: https://hal.science/hal-03603674

[103] K. I. Ahmed, "Modeling drivers' acceleration and lane changing behavior," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[104] K. Bhattacharyya, B. Maitra, and M. Boltze, "Calibration of micro-simulation model parameters for heterogeneous traffic using mode-specific performance measure," Transportation research record, vol. 2674, no. 1, pp. 135–147, 2020.

[105] S. Blandin, G. Bretti, A. Cutolo, and B. Piccoli, "Numerical simulations of traffic data via fluid dynamic approach," Applied Mathematics and Computation, vol. 210, no. 2, pp. 441–454, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300309000137

[106] World Meteorological Organization, Guide to Meteorological Instruments and Methods of Observation World Meteorological Organization, 2014. [Online]. Available: http://www.wmo.ch/pages/prog/www/IMOP/publications/WMO-8-Guide-contents.html

[107] 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

[108] 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, p. 491, 1957.

[109] M. Modest, "Radiative heat transfer," Elsevier Science, 01 2003.

[110] H. Van de Hulst, "Light scattering by small particles," 1957.

[111] 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

[112] H. Koschmieder, "Theorie der horizontalen sichtweite," Beiträge zurPhysik der freien Atmosphäre, vol. 12, pp. 33–55, 1924.

[113] 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

[114] 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

[115] 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

[116] 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

[117] 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

[118] 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, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.atmosres.2015.08.024

[119] 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.

[120] M. Hahner, C. Sakaridis, D. Dai, and L. V. Gool, "Fog simulation on real lidar point cloudsfor 3d object detection in adverse weather," in IEEE International Conference on Computer Vision (ICCV 2021), 2021. [Online]. Available: https://www.trace.ethz.ch/publications/2021/lidar_fog_simulation/HahnerICCV21.pdf

[121] C. Goodin, D. Carruth, M. Doude, and C. Hudson, "Predicting the influence of rain on lidar in adas," Electronics, MDPI, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/1/89/htm

[122] P. A. Lewandowski, W. E. Eichinger, A. Kruger, and W. F. Krajewski, "Lidar-based estimation of small-scale rainfall: Empirical evidence," Journal of Atmospheric and Oceanic Technology, vol. 26, pp. 656–664, march 2009. [Online]. Available: https://journals.ametsoc.org/view/journals/atot/26/3/2008jtecha1122_1.xml?tab_body=pdf

[123] S. Zang, M. Ding, D. Smith, P. Tyler, T. Rakotoarivelo, and M. A. Kaafar, "The impact of adversary weather conditions on autonomous vehicles," IEEE Vehicular Technology Magazine, 2019.

[124] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[125] I. Fursa, P. Newman, F. Cuzzolin, and A. Bradley, "Multi-weather city: Adverse weather stacking for autonomous driving. 2021 ieee," in CVF International Conference on Computer Vision Workshops (ICCVW), 2021, pp. 2906–2915.

[126] R. Gong, D. Dai, Y. Chen, W. Li, D. P. Paudel, and L. Van Gool, "Analogical image translation for fog generation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 2, 2021, pp. 1433–1441.

[127] S. Song, S. Lee, H. Seong, K. Min, and E. Kim, "Shunit: Style harmonization for unpaired image-to-image translation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 2, 2023, pp. 2292–2302.

[128] M. Ruchiga, R. Sainct, G. S. Pierre, and D. Gruyer, "Generic ontology and framework for critical scenario description and generation. applied to evaluation and validation of automated vehicle," Bilbao, Spain, September 2023.

[129] A. D. S. Sectional Committee of AD Safety Evaluation, "Automated driving safety evaluation framework version 1.0," Japan Automobile Manufacturers Association, Inc., techreport, December 2020. [Online]. Available: https://www.jama.or.jp/english/reports/docs/Automated_Driving_Safety_Evaluation_Framework_Ver1.0.pdf

[130] ——, "Automated driving safety evaluation framework version 2.0," Japan Automobile Manufacturers Association, Inc., techreport, December 2021. [Online]. Available: https://www.jama.or.jp/english/reports/docs/Automated_Driving_Safety_Evaluation_Framework_Ver2.0.pdf

[131] W. Xu, D. Gruyer, and S.-S. Ieng, "Generic simulation framework for evaluation process: Applied to ai-powered visual perception system in autonomous driving," Bilbao, Spain, September 2023.

[132] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[133] D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "Yolop: You only look once for panoptic driving perception," Machine Intelligence Research, pp. 1–13, 2022.

[134] J. Pang, L. Qiu, X. Li, H. Chen, Q. Li, T. Darrell, and F. Yu, "Quasi-dense similarity learning for multiple object tracking," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 164–173.

[135] R. R. Adel Djoudi, Loic Coquelin, "A simulation-based framework for functional testing of automated driving controllers)," The 23rd IEEE International Conference on Intelligent Transportation Systems, 2020.

[136] S. S. Haghshenas, G. Guido, A. Vitale, and V. Astarita, "Assessment of the level of road crash severity: Comparison of intelligence studies," Expert Systems With Applications, vol. 234, August 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423016202

[137] F. Wei, Z. Cai, P. Liu, Y. Guo, X. Li, and Q. Li, "Exploring driver injury severity in single-vehicle crashes under foggy weather and clear weather," Journal of Advanced Transportation, vol. 2021, August 2021. [Online]. Available: https://www.hindawi.com/journals/jat/2021/9939800/

[138] J. Krampea and M. Junge, "Injury severity for hazard risk analyses:calculation of iso 26262 s-parameter values from real-world crash data," Accident Analysis and Prevention, vol. 138, March 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000145751930380X?via%3Dihub

[139] C. Gu, J. Xu, S. Li, C. Gao, and Y. Ma, "Injury risk assessment and interpretation for roadway crashes based on pre-crash indicators and machine learning methods," Applied Sciences, vol. 13, no. 6983, June 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/12/6983

[140] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," EURASIP Journal on Image and Video Processing, vol. 2008, pp. 1–10, 2008.

[141] M. S. A.H. Gandomi, A.H. Alavi, "New formulation for compressive strength of cfrp confined concrete cylinders using linear genetic programming," Materials and Structures 43, p. 963–983, 2010.

[142] A. B. E.M. Golafshani, "Automatic regression methods for formulation of elastic modulus of recycled aggregate concrete," Applied Soft Computing 64, p. 377–400, 2018.

[143] D. P. M.-Y. Cheng, P.M. Firdausi, "High-performance concrete compressive strength prediction using genetic weighted pyramid operation tree (gwpot)," Engineering Applications of Artificial Intelligence 29, p. 104–113, 2014.

[144] Q. et al, "Definition of procedures of scenario management and results analysis – initial report," PRISSMA Deliverable, Tech. Rep., 2022.

[145] R. et al, "Tests and audit requirements - final report," PRISSMA Deliverable, Tech. Rep., 2023.

[146] W. Xu, R. Sainct, D. Gruyer, and O. Orfila, "Safe vehicle trajectory planning in an autonomous decision support framework for emergency situations." Applied Sciences journal, vol. 11, pp. 1–31, July 2021, special Issue "Human-Computer Interaction: Theory and Practice".

[147] G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[148] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[149] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer, 2014, pp. 740–755.

[150] V. Y. Mariano, J. Min, J.-H. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer, "Performance evaluation of object detection algorithms," in 2002 International Conference on Pattern Recognition, vol. 3. IEEE, 2002, pp. 965–969.

[151] J. C. Nascimento and J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance," IEEE Transactions on Multimedia, vol. 8, no. 4, pp. 761–774, 2006.

[152] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in 2020 international conference on systems, signals and image processing (IWSSIP). IEEE, 2020, pp. 237–242.

[153] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "Motchallenge: A benchmark for single-camera multiple target tracking," International Journal of Computer Vision, vol. 129, no. 4, pp. 845–881, 2021.

[154] T. Sato and Q. A. Chen, "Towards driving-oriented metric for lane detection models," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 17 153–17 162.

[155] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," IEEE Transactions on intelligent transportation systems, vol. 11, no. 3, pp. 589–606, 2010.

[156] T. Genevois, J.-B. Horel, A. Renzaglia, and C. Laugier, "Augmented Reality on LiDAR data: Going beyond Vehicle-in-the-Loop for Automotive Software Validation," in IV 2022 - 33rd IEEE Intelligent Vehicles Symposium IV. Aachen, Germany: IEEE, Jun. 2022, pp. 1–6. [Online]. Available: https://hal.inria.fr/hal-03703227

[157] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional Monte Carlo Dense Occupancy Tracker," in 18th IEEE International Conference on Intelligent Transportation Systems, Las Palmas, Spain, Sep. 2015. [Online]. Available: https://hal.inria.fr/hal-01205298