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

[DELIVERABLE 2.9] PROOFS-OF-CONCEPT FINAL REPORT: DEVELOPMENT OF PLATFORMS MEETING THE DESIRED OBJECTIVES OF EVALUATING MEANS OF AUTOMATED MOBILITY

[LIVRABLE 2.9] POCS FINAUX: DÉVELOPPEMENT DES PLATEFORMES RÉPONDANT AUX OBJECTIFS RECHERCHÉS D'ÉVALUATION DES MOYENS DE MOBILITÉ AUTOMATISÉE

Main authors : D. Gruyer (UGE), R. Régnier (LNE), G. Durand (AVSimulation), C. Chaves (AVSimulation), K. Quintero (IRT SystemX), Keilatt Andriantavison (Valeo), Wei Xu (UGE), Sio-Song Ieng (UGE), Alexandra Duminil (UGE), Jean-Baptiste Horel (INRIA), Cedric Gava (SPHEREA), Charlotte Segonne (Cerema) ...

Keywords: Simulation, evaluation, AI systems, validation plan and requirements, validation methods, verification process, evaluation protocol, scenario generation, metrics, KPI, risk assessment, performance indicators, simulation tools, sensor models, traffic simulation, graphical and physic engines, simulation environments, poc, proof-of-concept

Abstract. This document describes the final state of the implementation of proofs-of-concept (POC) that aim at demonstrating the use of simulation tests during the homologation and certification processes of autonomous vehicles. Several POC has been developed within the PRISSMA project and their particular ongoing work is presented separately.

Résumé. Ce document décrit l'état final de la mise en œuvre des preuves de concept (POC) qui visent à démontrer l'utilisation des tests de simulation lors des processus d'homologation et de certification des véhicules autonomes. Plusieurs POC ont été développés dans le cadre du projet PRISSMA et les résultats obtenus sont présentés séparément dans ce document.

Contents

List of Figures

List of Tables

1 Introduction (ALL)

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 CAV (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 Driving System)or an ADS (Advanced Driving System)as shown in Figure 1 ([16]). 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 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.

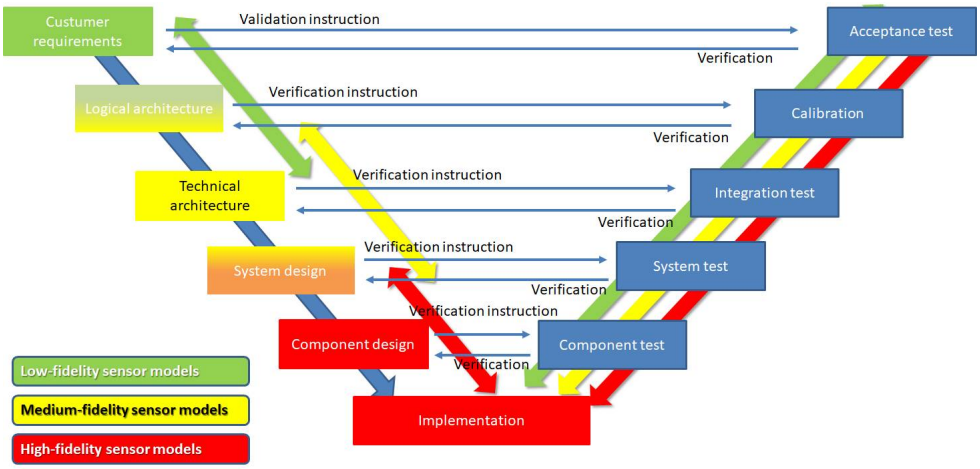


Figure 1: V-cycle for virtual prototyping, test, evaluation, and validation([1])

The validation phases go from the component tests to the functional test of the full system in its ODD (Operational Design Domain). At the end of a CAV or an ADAS validation process, approval and certification usually rely on physical tests. However, simulation results are cited in the list of elements that can contribute to the safety demonstration for the authorisation of a Highly Automated Vehicle to be operated on its ODD (French Decree n° 2021-873 du 29 June 2021, Art. R. 3152-6.-I ([17])).

To demonstrate feasibility on the use of simulation tools for testing AI-based systems related to CAV within the PRISSMA project, five different POCs have been proposed so far, as shown in the Figure 2. As this table indicates, different groups of partners have been composed to work on specific systems and simulation environments. The table also summaries the type of AI algorithm present in the system being tested with the simulation tools, as well as the equivalent physical site where physical tests may be conducted in other work packages of the PRISSMA project.

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)	POC 5: Fog simulation (perception)
Types of AI methods	Detection/identification/tracking: yoloV5 + DeepSort++; YoloP, YoloV8 Road marking: Ultra-Fast-Lane-Detection, YoloP		Openpilot	Probabilistic grid method	YOLO
Sensors	RGB camera, LiDAR	LiDAR, RGB camera, RADAR	RGB camera	LiDAR	RGB camera
Test sites	Satory	Créteil (Paris2Connect)	UTAC	Transpolis	PAVIN
AVS			X		
UGE	X				
LNE					X
AIRBUS PROTECT					
ANSYS		X			
CEREMA					X
ESI	X				
INRIA				X	
STRMTG					
SPHEREA	X				
SYSTEMX	X	X			
TRANSPOLIS				X	
UTAC			X		
VALEO		X			
CEA			X		

Figure 2: Table summarising the five final POCs implemented in PRISSMA.

The following sessions describe individually and in further details the aforementioned POCs.

2 POC 1: Bus Station Automated Service (BuSAS) on real-life track and simulation (UGE, SPHEREA)

2.1 Presentation

2.1.1 Overall Goals

The first main goal of this POC called BuSAS is to implement a full Bus station automated desert service with some function needing to implement IA based function (for detection, identification, and tracking of obstacles) both in real and virtual conditions. The second objective will be to propose a complete and generic enough simulation architecture in order to test/evaluate/validate the full system, the AI-based perception component, and to verify the validity and the performance of the simulation tools. The third goal consists to develop a generic validation methodology with dedicated and representative metrics. At the end, the last objective is to implement in the real prototype and on the real test track the same validation methodology. The use of both actual and virtual environment with the access to the digital twin will allow to implement a relevant comparison of our evaluation/validation methodology from the representativeness point of view. The question will be to identify the level of simulation we need to generate and to use in order to obtain an acceptable enough evaluation/validation. Moreover in order to evaluate the genericity of our methodology, we have proposed an ODD also similar to the POC dedicated to the automated serves of the 4 train stations in Paris (POC Paris2Connect). Depending of the available time, the architecture implemented in Satory could be also implemented on the Paris2Connect's journey, especially the journey between the Austerlitz and the Lyon train stations (Van Gogh street till the intersection with the Bercy street). In this framework, the implementation of the full service in both real and virtual environment thanks to the digital twin using will allow to apply comparison process between real and virtual data obtained by the system under test and by the full system. This methodology is presented in the figure 3 for an simple overview and in figure 4 for a more detailed view with the identification of the main objectives addressed for each layers. It is also interesting to mention that in PRISSMA, UGE has funded with additional intern budget the building of 2 additional Digital Models. The first one is the Paris2Connect area (co-funded with VALEO), and the second one is the Transpolis area. These 2 additional digital models will be presented in a next section.

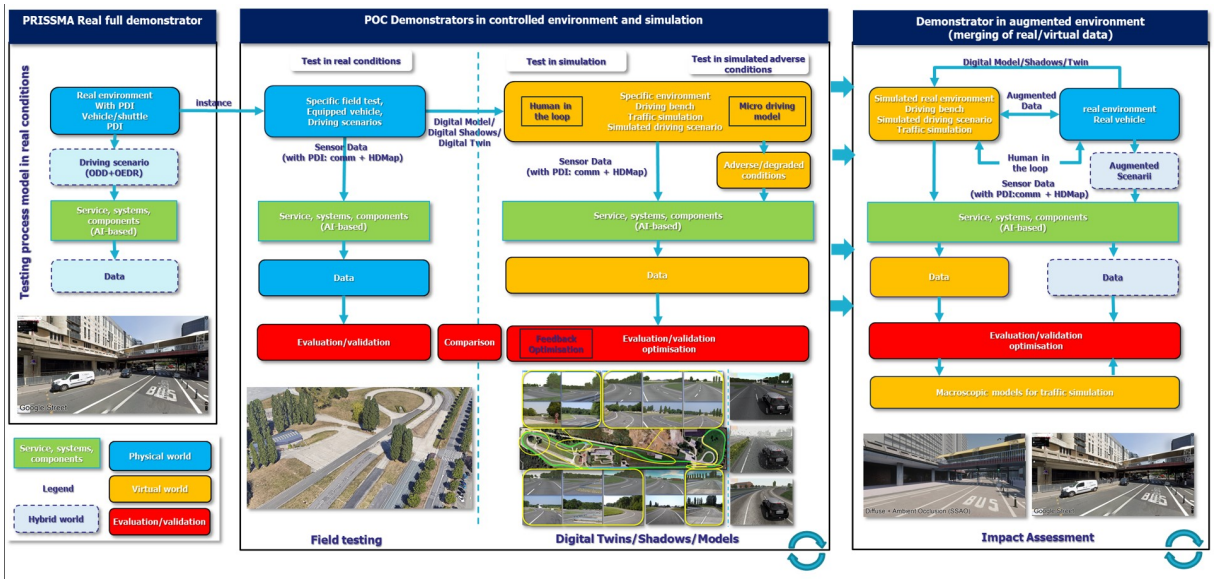


Figure 3: POC BuSAS, a generic way to compare real and virtual scenarios in same environment (From Paris2Connect real environment to Paris2Connect virtual Digital Model passing through Controlled environment (real and virtual) (Source UGE)

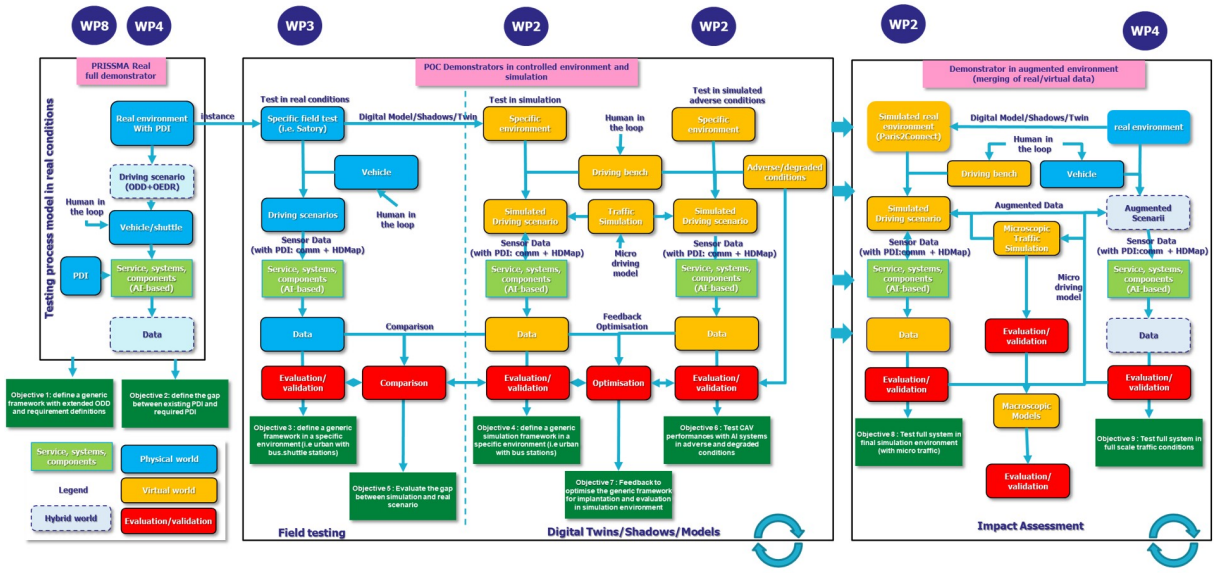


Figure 4: POC BuSAS, the detailed view of PRISSMA methodology applied to POC 1 BuSAS with objectives for each stages and layers of modelling and implementation (real and virtual) (Source UGE)

In order to reach these objectives, in the evaluation and validation processes, several levels have been defined and need to be addressed. The application of this methodology merging real, controlled, and virtual environment allows to provide a first way to build answers to a set of questions concerning evaluation and validation process for AI-based applications.

- **System under test**: The first one is the level *System (or service)*. This level needs to implement the full mobility service and to propose relevant and representative scene and

scenario involving an exhaustive set of conditions/configurations/situations allowing to quantify the performances and the quality of the service in the conditions defined in the ODD. This level needs to involve the car in the loop. The figure 23 shows the needed functional modules and more accurately the AI-based modules which could be under test.

- **Component under test**: The second level focuses on the *component* aspect. In this level, component means the algorithms, the functions, the ECU (Electronic Control Units) and all aspect involving a functionality with identified limits. For instance, this level could test, evaluate, and validate a perception module, a decision module, or a path planning module. The validation can be done with an open code or a closed code with only an access to the inputs and outputs of this black box. This level don't need to use the full service of mobility and it is not mandatory to have the car in the loop. Moreover, in the PRISSMA framework, we have to focus on AI-based module and components.

- **Verification and validation of tools and models**: The third level will address the verification and validation of the *models and tools* used in the full simulation architecture in order to validate the system or the component. In this level, we have to propose some experimental plan built from the WP1 outputs in order to take into account the scenes and the scenarios involving disturbances/failures/attacks/interferences impacting the AI-based components. We have to prove the capability of our tools and models to generate these adverse/degraded conditions with good enough behaviour/effect/rendering.

- **Evaluation/validation process and methodology**: The fourth level concerns the proposal of a methodology of evaluation/validation in the system and component levels with a set of appropriate metrics. In this part, we need to address the recording of the data, the format of the recorded data, and the development of the stages needed in the analysis of the data (definition, implementation, and use of KPI (Key Performance Indicators) and metrics). This aspect is addressed and processed in the WP1 of the PRISSMA project.

- **Scenarios for adverse conditions**: The last one concerns the capability to generate representative and accurate/realistic enough scenes and scenarios involving the static physical environment, the environment conditions, and the dynamic physical environment involving dynamic actors and the ego-vehicle. This level is clearly in relation with the *"Verification and validation of tools and models"* part.

In this POC BuSAS, these 3 levels will be addressed. In order to reach this challenge, several requirements have to be tackled:

- The full mobility service need to be designed, implemented, and working in real/virtual conditions

- ODD needs to be defined very accurately.

- The evaluation platform needs to involve the different tools and models allowing to reproduce the full environment and the main important situation and conditions.

- The first and second level of evaluation needs to have an access to ground truth

- In the simulation platform, the availability of a physical and realistic digital twin is essential and probably mandatory.

This POC clearly faces a strong challenge. About the first requirement, we choose to design and implement in a real prototype and in a virtual platform/environment a full Bus station automated desert service. It is interesting to quote that the Bus station automated desert service will apply 6 different scenes continuously on a trajectory covered 3,4 km on the Satory's test track with 4 or 5 bus stations. The environment will have bends with very small radius of curvature and intersections that may contain vehicles entering the main road and the ego-vehicle traffic lane. Moreover, with apply a strong constraint on the ego-vehicle. It does not have the capability to make lane changing and collision avoidance by lane changing. The ego-vehicle will only keep the same lane during all the evaluation and validation process.

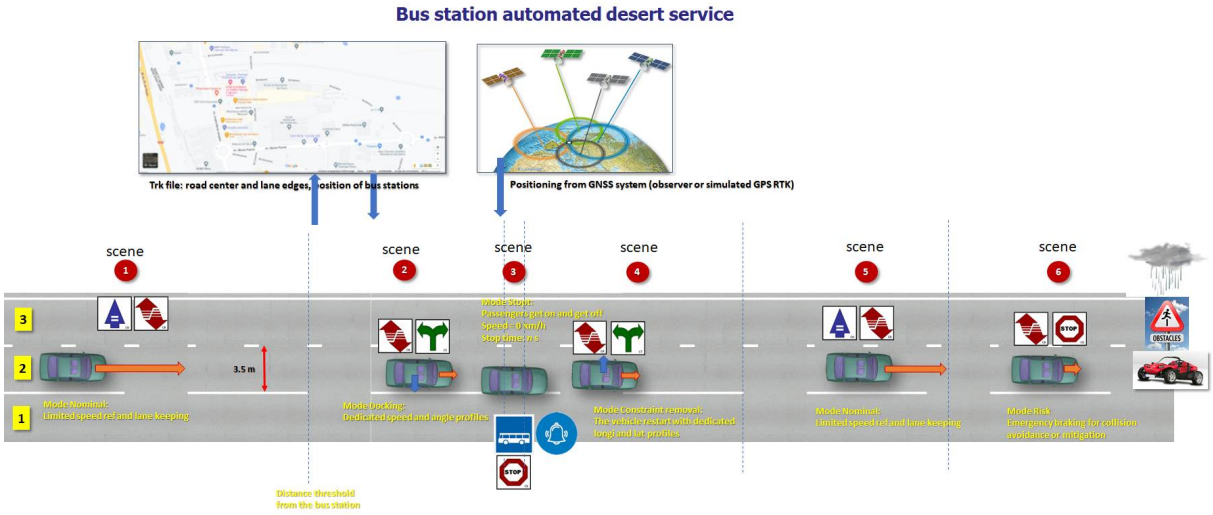


Figure 5: POC about Bus Station Automated Service (BuSAS). A view of the different scenes (Source UGE).

The POC BuSAS will be modelled by at least 6 possible scenes. The figure 5 represents the upstream and downstream scenes for the docking at a bus station:

- **The first scene** corresponds to the nominal driving mode. In this mode the shuttle or automated vehicle is driving on the right lane with a max speed up to 20 km/h and, in case of front moving vehicle, it will manage the inter-distance and apply a car following manoeuvre.
- **The second scene** occurs when the distance between the ego-vehicle and the bus station is lower than a threshold and the bus station is identified by the perception system. In this second scene, the vehicle will apply dedicated both longitudinal and lateral profiles in order to reach the bus station with constraints of ego-vehicle speed and lateral deviation.
- **The third scene** concerns the stop period to the bus station allowing passengers to get on and off.
- **The fourth scene** focuses on the restarting from the bus station with dedicated longitudinal and lateral profiles allowing to reach the centre on the right lane.
- **The fifth scene** is similar to the first one.

- **The last scene** concerns the reaction on a critical event like an object stopped on the right lane. In this situation, the ego-vehicle will have to apply an Emergency Braking allowing to avoid or to mitigate the collision. In this scene, it is possible to have a combination of elements building the risk situation. For instance, dense fog or string rain could reduce strongly the visibility distance and avoid to detection an obstacle on the traffic lane.

About the second requirement, an ODD has been proposed in order to fit not only with the service deployed on the Satory's test track but also to fit with the Paris 2 Connect POC (automated shuttle for the train station desert in Paris). Next section will present the designed ODD. The third requirement needs to implement a great set of tools, models, facilities, algorithms in order to validate the performance of the service in both real condition and virtual environment. In this part, a Renault Zoé has been equipped with hardware and software equipment. The same virtual model (Renault Zoé 3D) has been implemented in Pro-SiVIC. Moreover, a set/topology of sensors have been implemented. This two aspects will be presented in both sections 2.2.1 and 2.2.2 The next requirement was the access to a ground truth. In this context, the evaluation and validation process will be done with the same sensors and observers. In both, the actual Renault Zoé and virtual Renault Zéo, an RTK GPS will be available, the data about vehicle state will be collected on the CAN bus and by an observer in the simulation environment, the road marking positioning will be available and could be recorded from a *"trk" file* (centimetre measurement of the road marking coordinates, curvatures, angles, covering distance from the track origin). In the simulation platform, additional ground-truths will be generated like the object segmentation, the road segmentation, the accurate information about the weather conditions, the observers on pedestrians and other physical objects, and the current events.

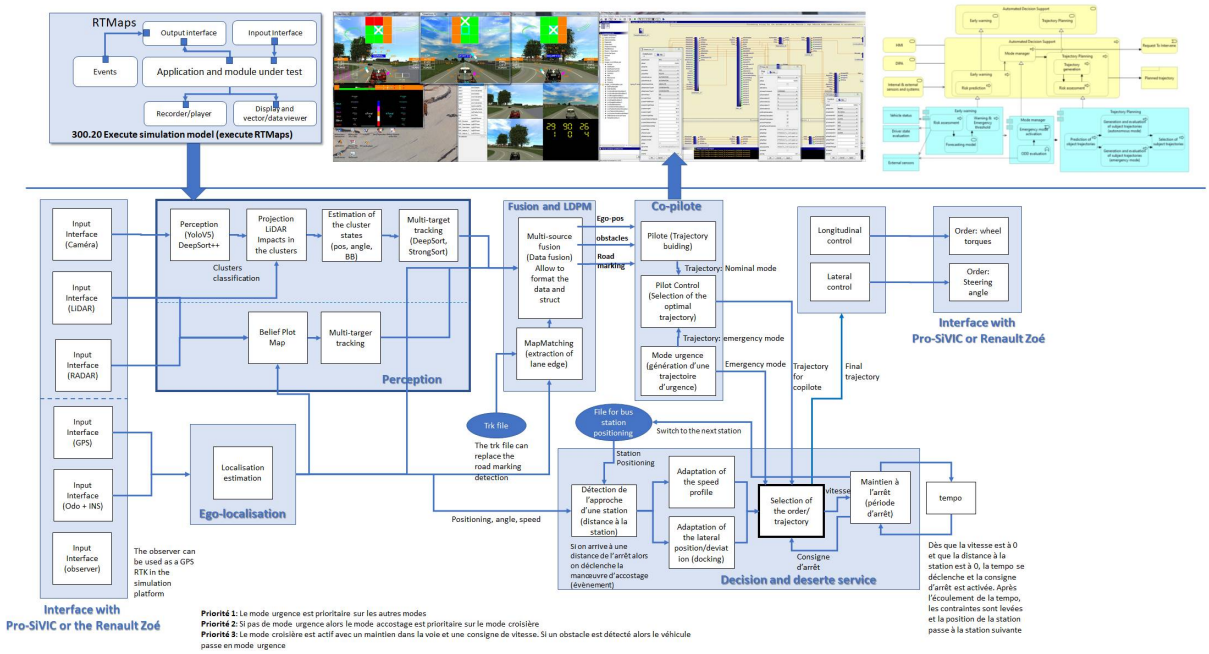


Figure 6: Functional architecture of the Bus Station Automated Desert Service implemented in RTMaps environment (Source UGE).

2.1.2 Design Domain

The section intends to discuss the definition of the ODD, OEDR, and main requirements about BuSAS. For more detail, it is recommended to read the deliverable 8.12 where a section is fully focused on the application of PRISSMA's taxonomy in order to generate the detailed ODD for BuSAS.

The tested system in the framework of the POC BuSAS should at least cover the following Operational Domain:

- Urban area
- Narrowing / narrow roads
- Ego speed range up to 20 kph
- Fluid and congested traffic conditions
- Roadway edges & markings : all possible in urban
- Signage : all traffic signs/road markings/traffic lights in urban
- Objects : all mobile objects in urban (non-classified/classified)
- Large/small static objects
- All weather (i.e light/intense rain) & light (day/night) conditions

Nevertheless, in this POC, the OD will be limited to the Satory's test track. Of course, in the simulation platform, features and furnitures will be added. It is important to highlight that in nominal conditions, the test track is fully involved in the ODD of the tested system.

2.1.2.1 Functional capacities of the system under test

The following description is an adaptation of the ADS Tactical and Operational Manoeuvres proposed by NHTSA.

ODD description and constraints			
Tactical and Operational Manoeuvre	Covered	Remark	ID
Parking	Out of ODD	Not in the PoC scope	NA
Stop on bus station	Yes	Action started when the vehicle is in the stopping zone indicated by a specific marking (to be defined) and a speed of $0km/h$. Activation of the parking brake. Stop for a fixed period + event	FC_01
Docking of the bus station	yes	Longitudinal and lateral profile modifications (speed and lateral distance). Convergence: stopping zone signaled on the ground (stopping criterion). The stopping zone is to the right of the traffic lane	FC_02
Restarting the shuttle after stopping at a station	Yes	Do not restart if an obstacle (vulnerable or non-vulnerable) is present on the restart path of the vehicle or intersecting an obstacle vehicle path (vehicle arriving from behind). Activate turn signal. Release the parking brake.	FC_03
Maintain speed	yes	NA	FC_04
Car following	yes	If a moving vehicle is present in the traffic lane, the speed and distance are adapted to guarantee safety.	FC_05
Vulnerable user following	yes	If a vulnerable mobile is present in the traffic lane, the speed and the distance are adapted to guarantee safety.	FC_06
Lane centring	yes	In nominal traffic mode, the vehicle remains in the centre of its lane. He does not make a lane change and he does not try to overtake in his lane.	FC_07
Lane switching / overtaking	Out of ODD	see requirements	NA
Enhancing visibility	Yes	To be defined (specific marking or template or sign: experimental Vehicle, flashing-ligth, LED, . . .)	FC_08
Obstacle avoidance	Yes	For an obstacle with acceptable dynamics, the vehicle will anticipate arrival at the obstacle and will apply "comfort" braking at a TTC of 2s (then following manoeuvre if possible). In the event of risky and sudden behaviour by the obstacle (sudden braking and/or reversing manoeuvre), the vehicle will apply emergency braking. If the vehicle is stationary and the obstacle continues to reverse then we are out of ODD.	FC_09
Low-speed merge	Out of ODD	NA	NA
High-speed merge	Out of ODD	NA	NA
Navigate on/off ramps	Out of ODD	NA	NA
Right of way decisions	Out of ODD	NA	NA
Navigate round-about	Out of ODD	NA	NA
Navigate intersection	Yes	Traffic lights	FC_10
Navigate working zone	Out of ODD	NA	NA
N-point turn	Out of ODD	NA	NA
U turn	Out of ODD	NA	NA
Route planning	Limited	One route. No other choice. Single lane route (bus lane). Signaling of the route by continuous and discontinuous markings, and a sidewalk on the right.	FC_11

About the ODD and the application of ODD to the developed service under test, a set of requirements has been identified:

- **Requirement 1**: For the moment, we do not take into account the opening of the doors, the closing of the doors, the signalling (alert) of starting the vehicle.
- **Requirement 2** : Do not restart if an obstacle (vulnerable or non-vulnerable) is present in the vehicle's restart path.

- **Requirement 3** : The vehicle moves on the bus lane (always the same lane).
- **Requirement 4** : No overtaking manoeuvre. The vehicle always stays on the same lane (bus/bike lane). The vehicle is using the far right lane.
- **Requirement 5** : The road surface and road material conditions are: asphalt, cobblestone, concrete. No snowy surface. For the paved road, it is possible to take into account the vibration of the tires and the shock absorbers. Taking into account the pavement roughness and high frequencies produced by the pavement. These vibrations have an impact on both the vehicle and the sensor behaviour.
- **Requirement 6** : The types of users that the vehicle may encounter: car, bus, scooter, motorcycle, bicycle, pedestrian, van (i.e. moving company, delivery).
- **Requirement 7** : The vehicle does not change lanes. It stays in its lane.
- **Requirement 8** : The vehicle does not overtake an object in its lane.
- **Requirement 9** : The vehicle can not cross the speed limit fixed to $20km/h$
- **Requirement 10** : The vehicle can move backward (reversing speed).
- **Requirement 11** : The vehicle in nominal mode cannot apply accelerations of more than $3m/s^2$ and decelerations of more than $3m/s^2$
- **Requirement 12** : In critical situation ($TTC <= 1s$), the vehicle must apply an emergency braking ($1G : 9.81m/s^2$)

2.1.2.2 Response mapping (OEDR) of the system under test

It is possible to accurately describe the responses that will be considered for different types of events, as described by the NHTSA, by highlighting the events and responses outside the ODD for this POC.

Definition of event: An event is a specific fact building a specific situation or condition for a set of scene elements. The event is more the observation of the realisation of a configuration/conjuncture with possible conditions. In fact, an event represents anything that happens in an instant of time (frame). Therefore, any instantaneous change of state caused by an Object or an element of the context can be defined as an Event. These changes usually cause a new occurrence and, depending on the duration, this can be defined as a new Event or an Action.

In order to define the different types of events, we have decided to share the events in function of the concerned environment key components: obstacles, road, ego-vehicle, environment.

The following table presents the events encountering with an input from static and dynamic physical obstacle:

OEDR and event description for Obstacles		
Event	Response	Remark
Lead vehicle is decelerating	Depending of the speed difference, the ego vehicle could apply a following, decelerate, stop	In the service deployed in this POC, the ego-vehicle keep the right lane (no lane change, no overtaking)
Lead vehicle is stopped	The ego vehicle decelerate, stop in order to avoid the collision or mitigate it	In the service deployed in this POC, the ego-vehicle keep the right lane (no lane change, no overtaking)
Lead vehicle is accelerating	Depending of the speed difference, the ego vehicle could apply a following, accelerate up to the speed limit defined in the ODD	In the service deployed in this POC, the ego-vehicle keep the right lane with a speed limit (20 km/h)
Adjacent vehicle apply a cut in	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC.	NA
Adjacent vehicle encroaching	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC.	In this condition, in order to respect the safety constraint, the ego-vehicle considers that the adjacent vehicle is driving on the ego-lane
Opposite vehicle encroaching	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC. If the opposition vehicle continue its driving between to traffic lane, the ego-vehicle is stopping	In this condition, in order to respect the safety constraint, the ego-vehicle considers that the opposite vehicle also is driving on the ego-lane
Lead vehicle cutting out	Depending of the speed difference, the ego vehicle could apply an accelerate up to the speed limit defined in the ODD	The acceleration is apply on when the lead vehicle will reach fully the other traffic lane without encroaching
Lead vehicle apply a parking manoeuvre	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC. If the lead vehicle has not performed its manoeuvre, then the ego-vehicle is stopping	In the service deployed in this POC, the ego-vehicle keep the right lane (no lane change, no overtaking)
Pedestrian is crossing the road	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC. If the pedestrian is yet on the traffic lane under a TTC=1s, then the ego-vehicle is stopping	In the service deployed in this POC, the ego-vehicle keep the right lane (no lane change, no overtaking)
Cyclist is riding on the traffic lane	The ego vehicle adapts its speed in order to respect the correct inter-distance and TTC. If the TTC is lower or equal to 1s then the ego-vehicle is stopping	In the service deployed in this POC, the ego-vehicle keep the right lane (no lane change, no overtaking)

The following table presents the events encountering with an input from the ego-vehicle:

OEDR and event description for Ego-vehicle		
Event	Response	Remark
Ego-vehicle operating outside the ODD	Must be defined, may be generate a request to intervene (fallback-ready user)	Must be defined
To be define	Must be defined	Must be defined
To be define	Must be defined	Must be defined

The following table presents the events encountering with an input from the road:

OEDR and event description for Road		
Event	Response	Remark
Presence of a speed bumper	Must be defined	Must be defined
Presence of a negative obstacle (pothole)	Must be defined	Must be defined
Pedestrian crossing way	Must be defined	Must be defined
Change of road marking type	Must be defined	Must be defined
Change of road curvature (bend)	Must be defined	Must be defined
Degradation of the visibility level / readability of road marking (soiling, ...)	Must be defined	Must be defined
Lack of marking	sidewalk edge detection	Available information (other embedded system, HD-Map)
Disturbers on the road surface (sand, gravel, soil, leaf, ...)	To be define	Possible occlusion of marking, production of artefact at the detection level
Speed limit sign	If necessary, the ego vehicle adapts its speed in order to respect the constraint. If the ego-speed is under the speed limit then the ego-vehicle keep the same behaviour	The max speed of the ego-vehicle is provided by the ODD

The following table presents the events encountering with an input from the Environment:

OEDR and event description for Environment		
Event	Response	Remark
Traffic light red	Decelerate and stop	Traffic light with red light. We have logical information (IoT)
Traffic light orange	Decelerate and stop	We have the logical information about the traffic ligth state
Traffic light green	Drive with the recommended speed or apply following maneuver. If the current maneuver consists to turn right then switch on the blinker (flashing right indicator).	We have the logical information about the traffic light state
Traffic light black	Drive and check the level of risk (detection of an obstacle close to the ego-vehicle : $TTC < 2s$)	Traffic light with no lights lit or covered with an out-of-order bag. We have the logical information about the traffic light state
Speed limit sign	Apply if possible (accelerate, decelerate)	If no specific instruction are asked, then the speed limit of the automated vehicle defined by the ODD is applied
No way sign	Out of ODD	Forbidden way
Rain condition	depending on the intensity of the rain and the level of visibility, the vehicle must adapt its speed	we consider that we have this information via a rain sensor and IoT information
Fog condition	depending on the density of the fog and the level of visibility, the vehicle must adapt its speed	we consider that we have this information via a fog sensor and IoT information
Light condition - Day - sunset/sunrise	To be define	generation of dazzle
Light condition - Night	switch on the head lights	Autonomous mechanism activated by an embedded ADAS. This system is available and independent of the system under test

About the OEDR and the different events identified in the 4 previous tabs, a set of requirements has been identified:

- **Requirement 13**: In degraded weather conditions, the vehicle has data sources to know the intensity of the event (rain, fog) and the visibility distance (for impacted sensors).

This information is therefore produced by another actor.

- **Requirement 14**: The vehicle lighting system is managed automatically according to the conditions present in the environment.
- **Requirement 15**: The semantic state of traffic lights is considered known and available (IoT or embedded system allowing this state to be detected). This information is therefore produced by another actor.

2.1.3 Tests in Simulation and Expected Results

In the simulation environment, the expected results are the following:

- To define a generic and inter-operable simulation architecture and framework allowing to replace, to add, to update tools and models (vehicles, sensors, environment, ...) needed for specific evaluation and validation procedures with a generic method.
- To generate a set of accurate and relevant ground truth (segmentation of the environment, observers, depth map, ...) in order to feed the evaluation and validation process.
- To develop an efficient and easy way/procedure to use scenario manager involving a clear and scalable description of scenario generation based on a generic ODD framework.
- To propose a library of metrics (levels system, component, tool and model) and the generic process to use it in an evaluation and validation process.
- To provide the capability to compare the real and virtual test process with the challenge to prove the representativness of the simulation in comparison to the real-life. This task could be done by using new metrics and scores allowing to assess the level of fidelity of the simulated data coming from the simulated sensors.
- To develop a ViL platform with the capability to merge real and virtual data and environment
- To propose some models and ways in order to take into account degraded and adverse conditions, failures, and cyber-attacks
- To propose a template of scenarios allowing to test specifically the performances of AI-based systems

For the POC BuSAS, a part of these objectives and expected results are presented in the figures 3, 4, and 8 . At this end, with the final version of the POC BuSAS, we expected to obtain and to provide a fully operating ViL platform with the Top-Down design method allowing to evaluate and validate a service/system/application/component involving IA-based system. Unfortunately, the ViL platform is yet under development and should be operational in 2025. Nevertheless the global architecture with all the needed functions and modules have been defined and presented in the section 3 dedicated to the generic framework. Moreover, a first set of methods and metrics have been developed by UGE in order to address the Verification and Validation stages allowing to guarantee the performance of the models and tools used in the evaluation/validation processes. In addition to the metrics and scores about the level of fidelity, a classification of the tools and models capability has been proposed with 3 main levels: High fidelity, medium fidelity, low fidelity.

2.2 Perimeter Definition

In the DataSets generated by UGE, a set of conditions have been defined and feed the scenarios. Among this conditions and their parameters, we have:

- Day conditions: rain, fog, with road reflection (wet road), with HDR images and light effects
- Night conditions: head light without degrade weather conditions
- Sensors: disturbances on sensors

All these conditions and parameters are synthesised in the Figure 7. In this figure we see the interactions between sensors (active and passive), the propagation channel, the energy sources (like the sun), and the material properties. In this study mage by UGE, 4 classes of disturbers and disturbances have been proposed. The first one is dedicated to the Atmospheric disturbances, the second one is focused on the material disturbers, the third one addresses the spectral and electromagnetic disturbers, and finally the last one defines the physical particles disturbers. In the scenario generated in PRISSMA, we have mainly focused our efforts on the last class dedicated to the particles and more accurately on the rain and fog disturbances.

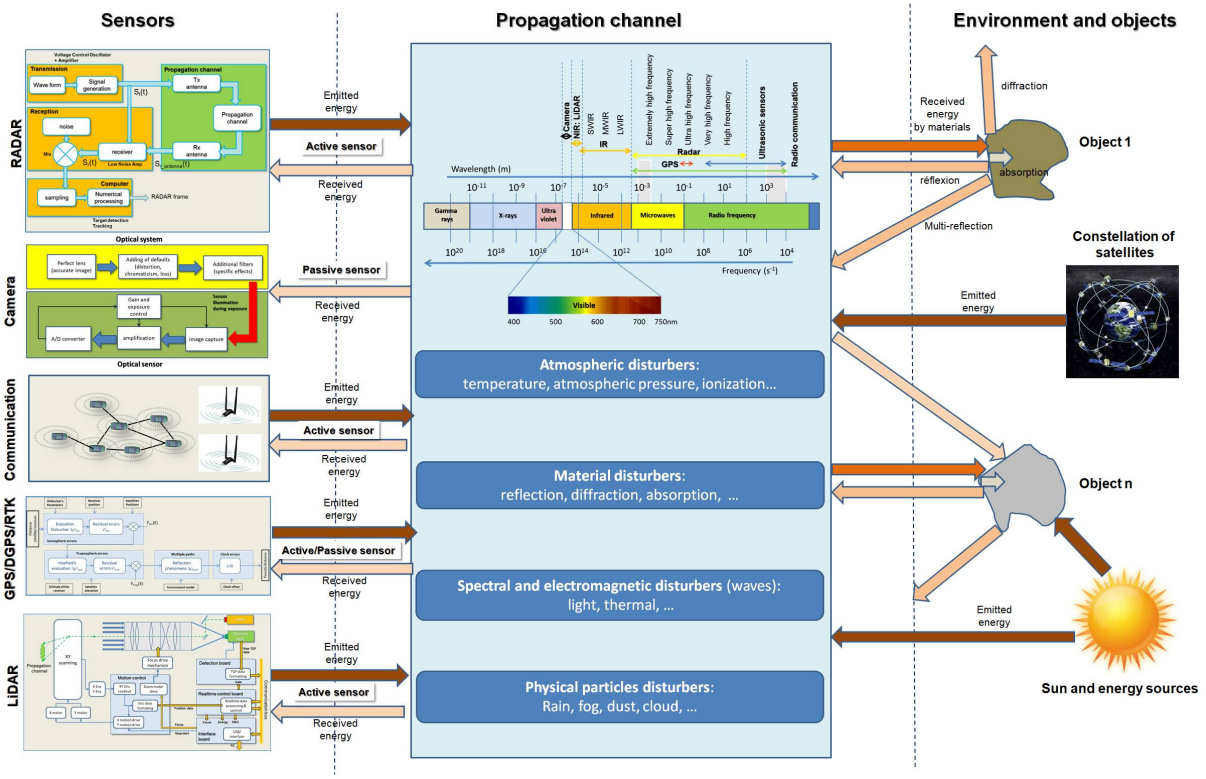


Figure 7: Classification of the type of disturbers in the propagation channel impacting the quality of the sensor data (Source UGE).

2.2.1 Simulation Environment and generic methodology

In the POC 1, the general architecture is shared in 3 main part:

- The preparation and building of scenarios
- The scenario management involving the events management
- The simulation platform allowing to run scenarios and to model vehicles-environment-sensors
- The application environment involving the system and component under test
- The recording, analysing and evaluation/validation modules

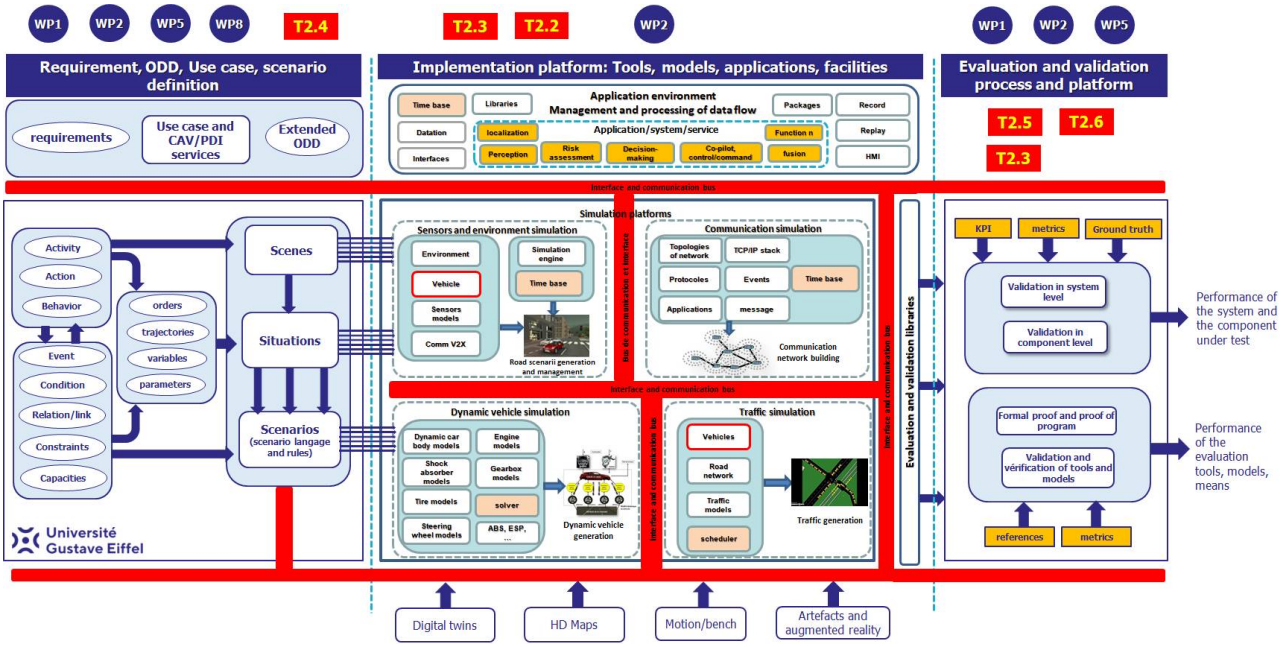


Figure 8: General architecture for simulation environment in POC 1 (called BuSAS) with link with WPs and WP2 tasks (Source UGE)

2.2.1.1 Upstream tools: prepare and build trajectories, scene, and scenarios

The first part of tools and models concerns mainly the scenario description, implementation, management, and use. In this upstream part, a set of tools are usable and partially used. In this tools, we can mention MOSAR from SystemX and some tools developed in University Gustave Eiffel like ROADS (see figure 11), Grotex (see figure 10), and Path Edit (see figure 9).

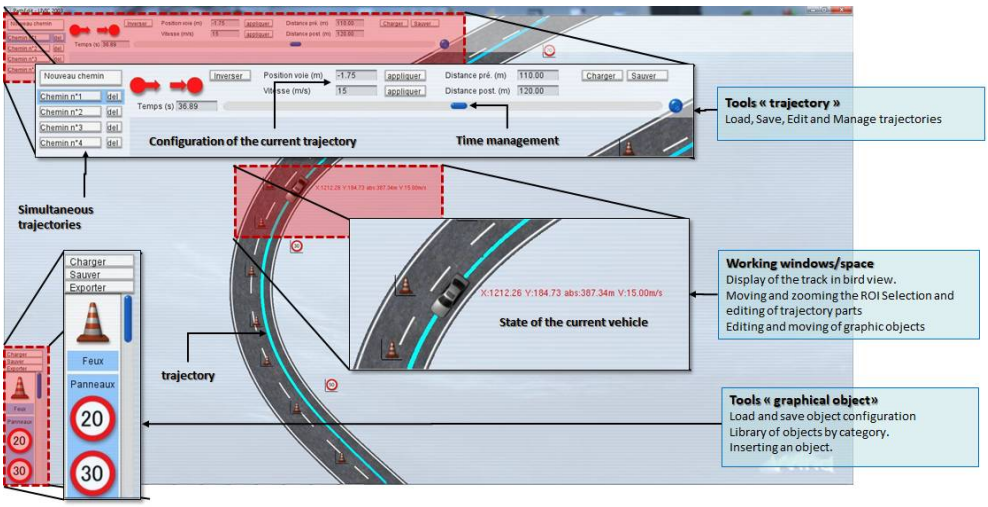


Figure 9: Path Edit: A platform for the trajectory generation and object positioning (Source UGE)

GRoTex software ([18]) is a efficient tool allowing to generate physico-realistic road surfaces involving road markings respected French standard. Among its functionalities, Grotex provides a large number of degradation effects and generate a road marking mask usable as a ground truth. If we look at the figure 10, a road texture is presented with some degradations. For instance on the top left image, the road marking has large holes without too much wear but a little dirty. Moreover, on the crack, the painting has not reached the bottom of the bitumen (Uniform wear). On the rougher medium, you can clearly see the painting holes. The noise on road marking edges is clearly visible on the zoom of the image. On the bottom left image, we can see some holes but, above all, uniform wear which has reduced the intensity of the marking. With this software, it is possible to re-product the similar rendering and effects observed in real conditions on real road marking.

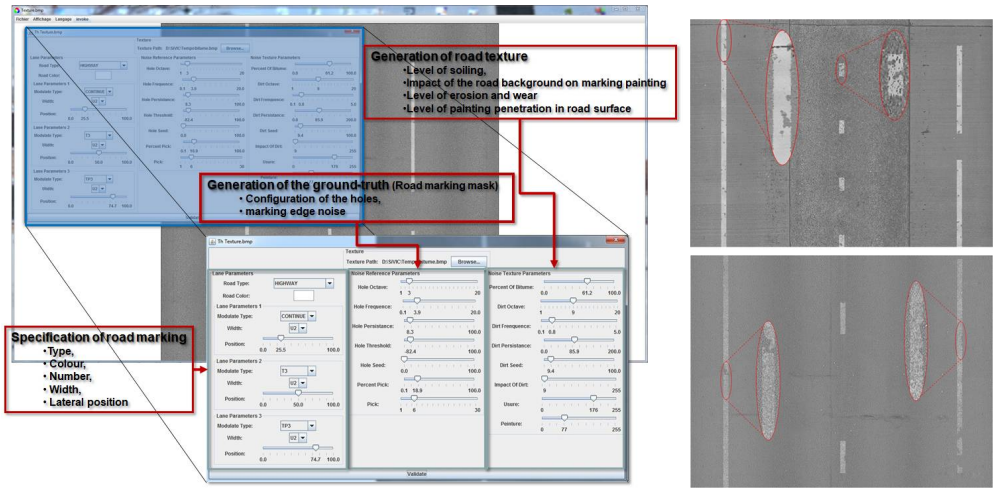


Figure 10: GRoTex: A generator of road texture with a set of filters to generate degraded conditions (Source UGE)

ROADS ([19]) is a software allowing to build and to update OpenDrive road network with the capability to generate the 3D objects corresponding to the road network and usable in the Pro-SiVIC platform. Grotex is a software with graphical environment allowing to generate road texture with road marking and a set of possible degradation of the road marking. Path Edit

is more dedicated to the Satory test track and allows to generate trajectories for the vehicles involve in the scenarios. Moreover PathEdit allows to put a set of object (road sign, traffic light, plot, ...) in the virtual environment and to generate the Pro-SiVIC script.

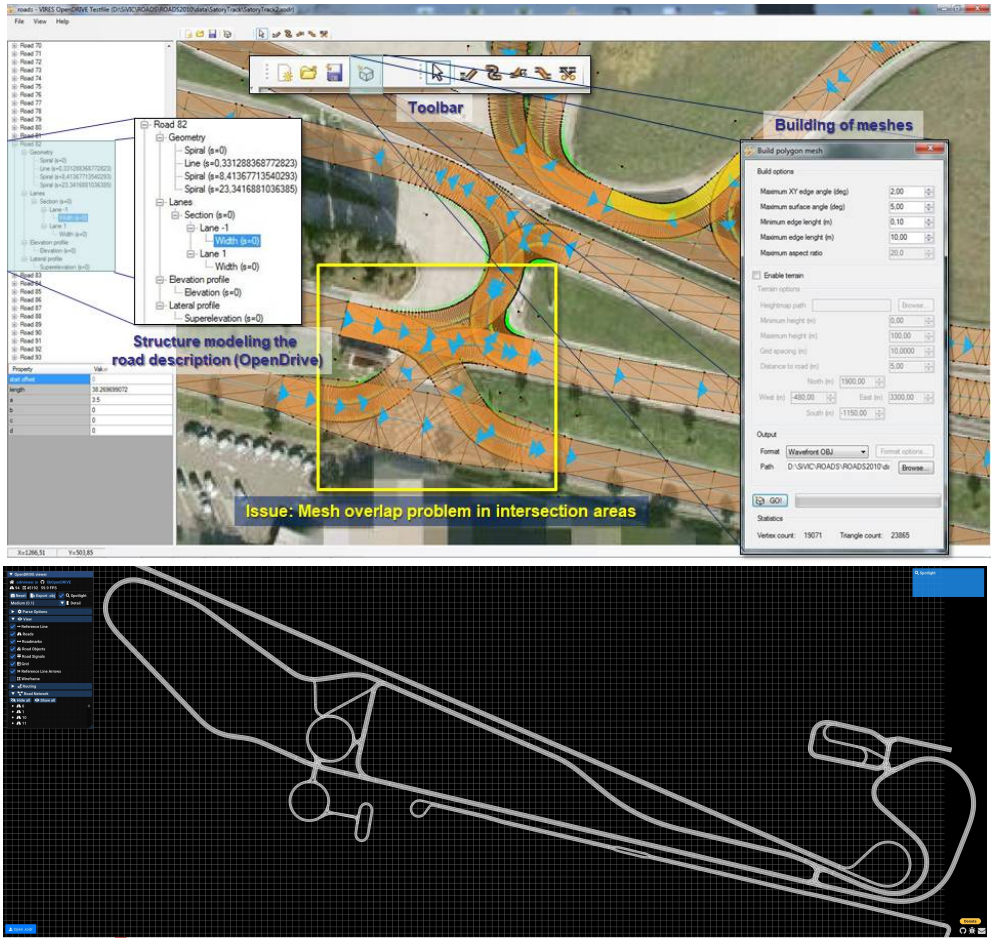


Figure 11: ROADS: An OpenDrive road network builder with a mesh generator (Source UGE).

2.2.1.2 Simulation platform: Pro-SiVIC capabilities and third party software

The core of the simulation platform is provided by the Pro-SiVIC platform. This software provide physical realistic models of sensors, of vehicle dynamics, and environment conditions (ligth, weathers, ...). The existing sensors are similar than the ones embedded in the real prototype. In the current configuration we will provide RADAR ([20], [21]), LiDAR ([22], [23]), camera ([24]), GPS, INS, Odometry and communication means ([25],). Figure 12 presents a set of sensor's outputs. In the figure 8, this part is presented in the tools and models sub part and in the top left of the architecture.

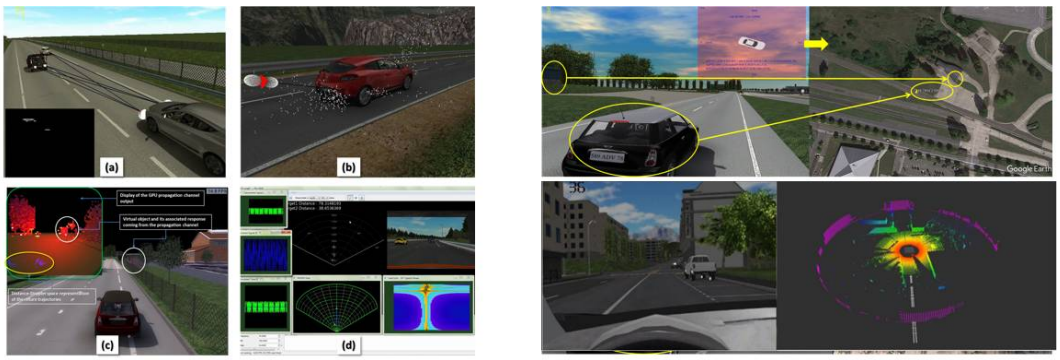


Figure 12: Pro-SiVIC: A dedicated simulation platform for realistic sensor simulation (Source UGE and ESI)

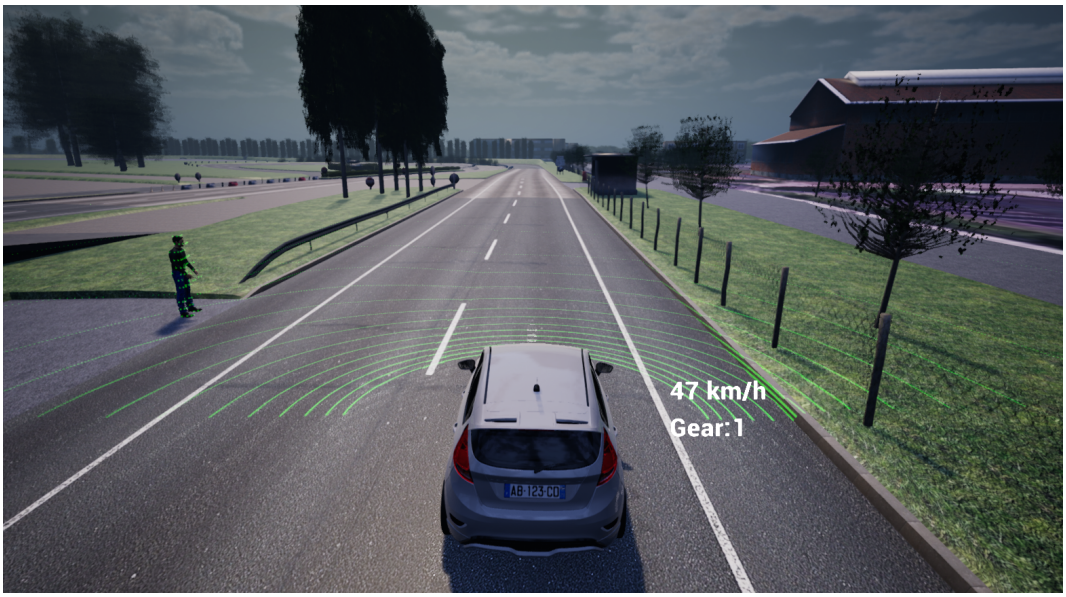


Figure 13: Pro-SiVIC: Sensor Simulation using Unreal Engine (Source ESI)

In order to generate the disturbances impacting the sensors and vehicle behaviour, a set of filter has been developed. Among this filter we propose Light filter (head ligth with ligth map) and weather condition filters like rain (rain fall and rain drops), fog, snow (snow fall). Figure 14 presents some screenshots of these light and weather filters.

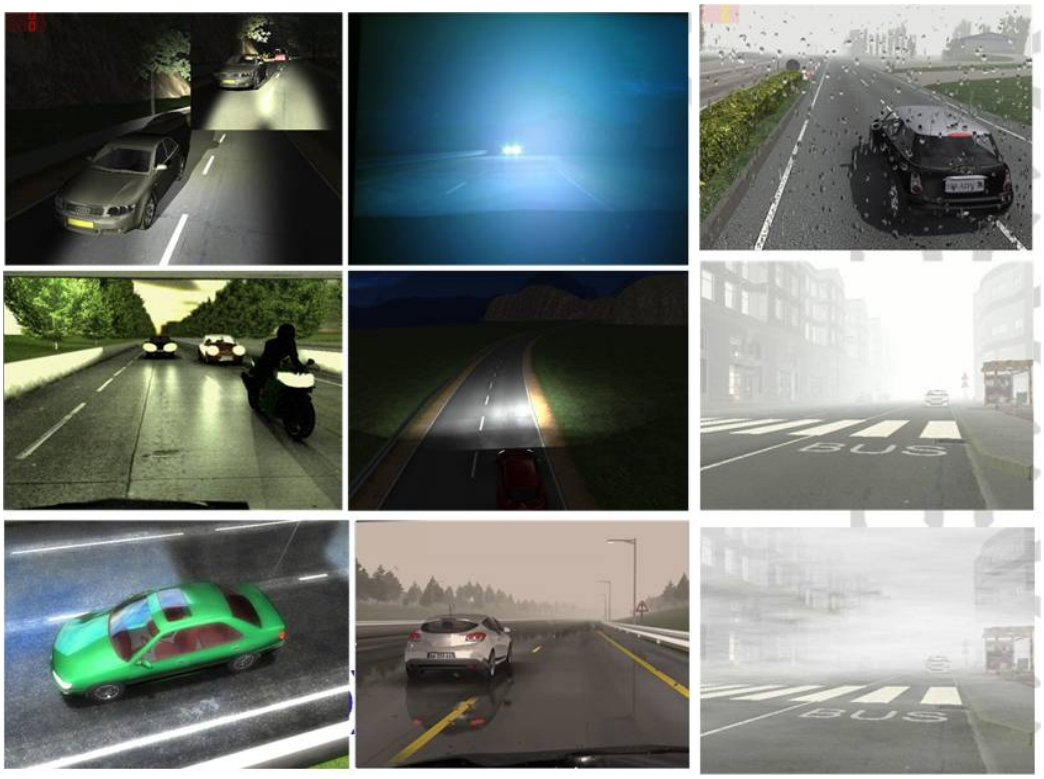


Figure 14: Pro-SiVIC: Filters and mechanisms for the light and the atmospheric disturbances modeling, generation, and management using mgEngine (Source UGE and ESI)

The dynamic modelling of the vehicle is done by a complex modelling involving car body, shock absorber, wheels and tires, powertrain, and the steering wheel. The interaction of the complex model with the ground and other objects is done by using a raytracing engine implemented in the simulation engine. In order to control the vehicle and pedestrian models, several modes are provided like ”human control”, ”trajectory following”, ”control/command”, ”control from RTMaps”. Moreover, for a great density of vehicle, it is possible to use the coupling/interconnection of Symuvia (traffic generator) and Pro-SiVIC. This coupling is done using DDsL and the OpenDrive modeling of the Satory’s test track. In this configuration, it will be possible to manage in real time up to 500 vehicles.

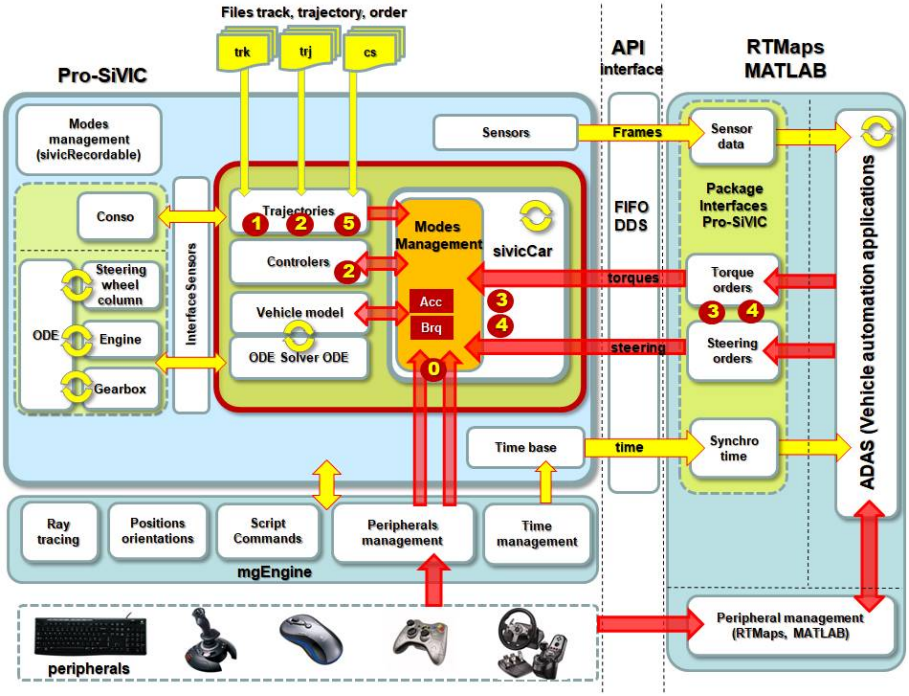


Figure 15: Pro-SiVIC: Existing interfaces and peripherals controllers (Source UGE)

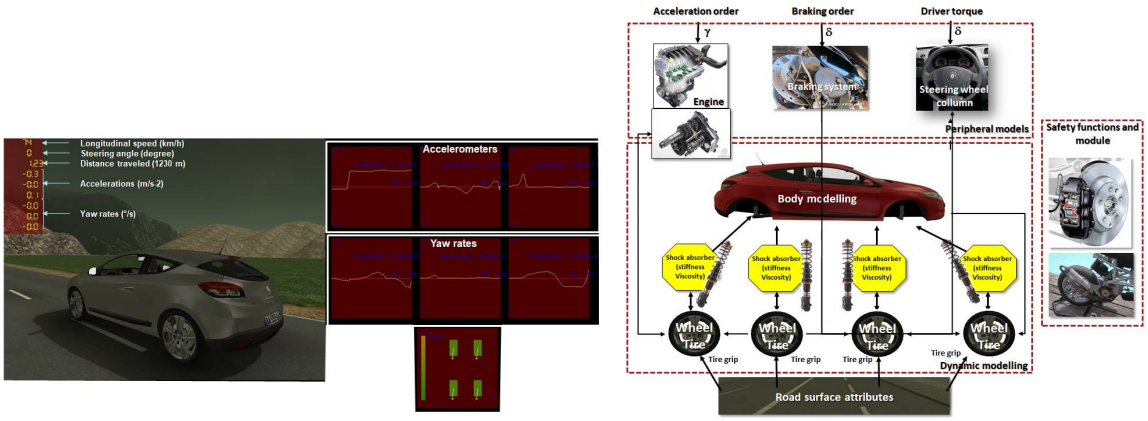


Figure 16: Dynamic model of vehicles in Pro-SiVIC (Source UGE)

In order to develop similar experimental plan in real and virtual conditions, a Digital Model of the Satory's test tracks has been developed. This Digital Model and the 2 other ones (Transpolis and Paris2Connect) will be presented in the section dedicated to the generic methodology allowing to generate the Digital Models. The figure 17 present a set of additional Digital Models build and implemented in Pro-SIVIC.

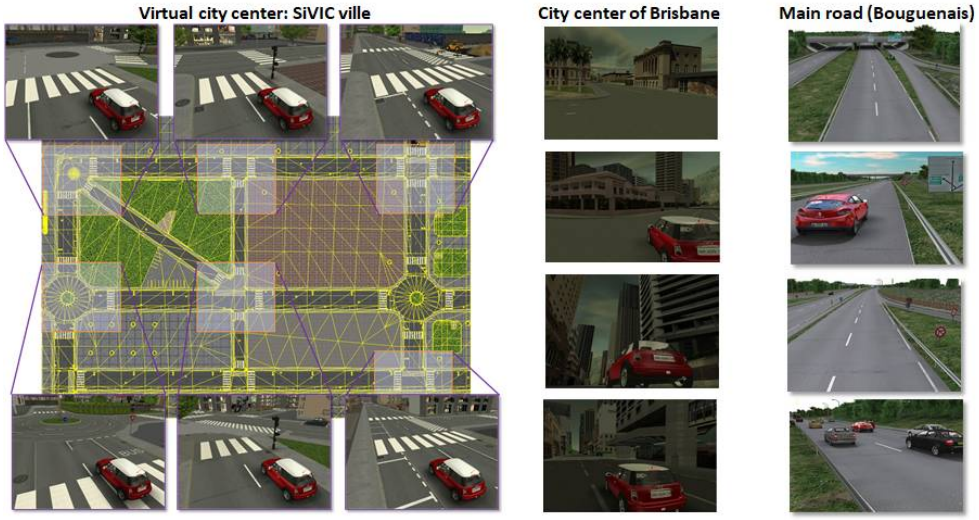


Figure 17: Some other Digital Models implemented in Pro-SiVIC. The left DM represent a generic city centre with the main meetable intersections. The screen shot in the centre provides some view of the centre of Brisbane (Australia). The pictures in the right part provide a part of the main road located in Bouguenais (near Nantes). (Source UGE).

2.2.1.3 The application environment included the system and components under test

In order to implement the full system under test, we use the RTMaps software. Initially, RTMaps has been developed to manage, record, and replay data flow coming from sensors. Added to this initial functions, a mechanism has been implemented in order to develop its own packages. A packages is similar to a DLL with a set of included modules. Each module is built from a generic template made from MACRO (management and definition of inputs, outputs, properties, actions) and methods (birth, death, core). The implemented code is in C or C++. In the work window, it is possible to build complex diagram with a set of module given a full system. All the data handle in RTMaps are timestamped. A set of generic packages are available. These packages concern the play and record function, the sensor drivers, the information display (image viewer, data viewer, and oscilloscope). Added to these packages, we have developed a specific package allowing the interconnection between Pro-SiVIC and RTMaps in both direction in order to get the sensors/observers data and in order to send vehicle orders, object control, and events. RTMaps has been used both in the real prototype (ImPACT 3D VA: Renault Zoé) and in the simulation platform (ImPACT 3D VR&Motion). In this efficient environment and in this context, we will use the same application codes and the same type of sensors data. So the comparison between real and virtual test case will be easy and representative.

This environment allows to process and to merge data in real time and provide an efficient environment for the prototyping and first test of an application. Moreover data processing can be performed in real time or during a playback session. Each RTMaps component embeds its own functionalities and settings. The RTMaps engine deals with data flow between the components and multi-threading management.

RTMaps also allows the operation of distributed and synchronised platform on several machines. A ”master” system manages the whole application. A single clock supervises and synchronises those of the various ”slaves”. The ”Master” clock can be the one of the ”master” host or coming from an external source: clock of an acquisition board, GPS clock or Pro-SiVIC (see figure 20). RTMaps technology is independent of the used OS, even in a distributed con-

figuration.

Pro-SiVIC platform has been interconnected with RTMaps in order to generate sequences of virtual data homogeneous with the real data processed on the vehicle. With this architecture, the algorithms used on the vehicle and with Pro-SiVIC are strictly the same ones. This is very interesting in the stages of driving assistance systems evaluation with particular conditions and with some very long or numerous scenarios. Moreover, the ADAS algorithms used in RTMaps are exactly the same when working with either real data (acquired on real vehicles), or simulated data coming from Pro-SiVIC. This approach drastically decreases the translation and adaptation stages between the simulated and the embedded algorithms. Furthermore, this global architecture (Pro-SiVIC and RTMaps) allows to build "reference scenarios" and constitutes an efficient platform for the tests and the validations of the embedded algorithms. The figure 18 shows a full application for driving automation with modules dedicated to the perception, the decision and path planning, and the control. This diagram use the replay of a sensor data-set.

RTMaps is not the only way to implement and to record data. It is also possible to interface Pro-SiVIC with ROS or a other third-part application.

In the current POC BuSAS, we will use the last version of RTMaps (4.8) in real and virtual prototypes.

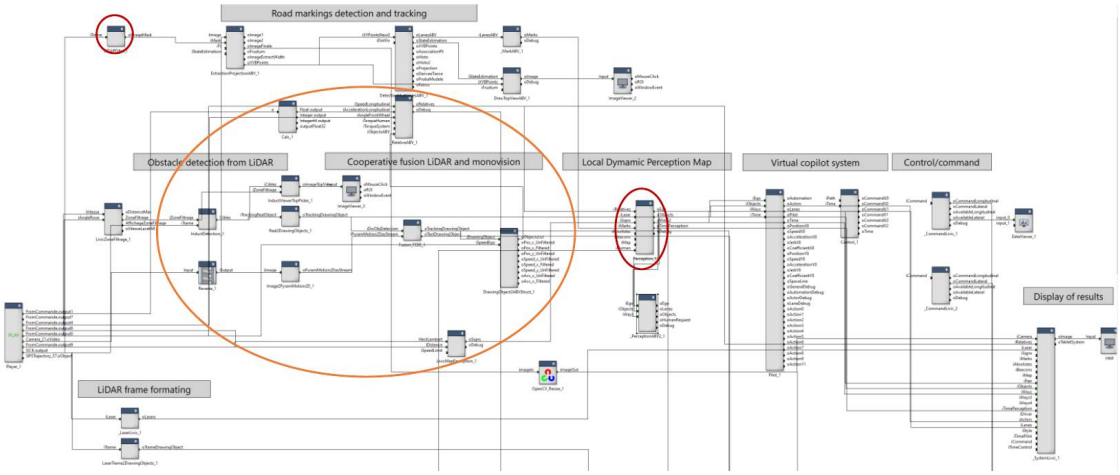


Figure 18: Example of full driving automation application in a RTMaps diagram with models for Perception/Decision/Action (Source UGE).

2.2.1.4 The recording, analysing and evaluation/validation modules

Several way are available to record the data with an accurate timestamping. The first solution consists to use the RTMaps recorder. This recorder allows the simultaneous recording of various tracks of information. Information is stored in STDB, Synchronised Time stamped DataBases. When replayed, the sequence is reproduced identically thanks to the data timestamps. It is possible to play information at the desired speed: accelerated, slowed down, step by step. The data can be recorded in different format (bin, matlab, ...). These record and replay functions are well done for the DataSet generation and the offline analysing of data. The second way is to use the Record mode in Pro-SiVIC in order to generate CSV files for the observers. The record of images can be done with a set of different formats and resolutions. The third way consists to use the interface libraries in third-part software like ROS or a proprietary application. The last

way consist to use the MATLAB mode in Pro-SiVIC in order to send the data toward Matlab. In this context, the data could be recording and/or plotted in Matlab.

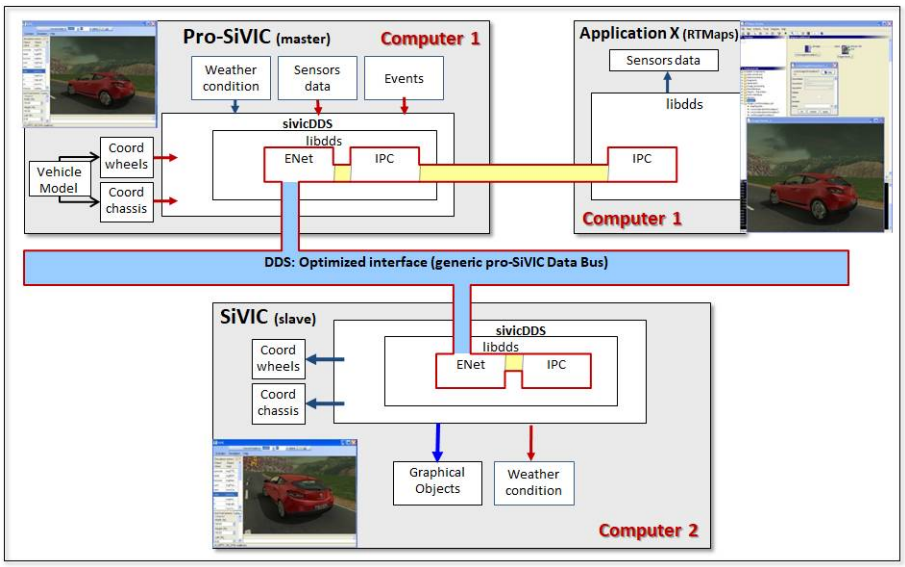


Figure 19: DDS and DDsL, a generic library for tools and softwares interfacing (Source UGE).

2.2.1.5 Leveraging distributed test system environment with U-TEST

Continuing the work conducted during the definition of a logical architecture for distributed test and simulation systems (see deliverable 2.4 PRELIMINARY DEFINITION OF INTERFACES AND SIMULATION ENVIRONMENT), we instantiated a platform with U-TEST and the CARLA simulator, as well as a gateway for interfacing Pro-SiVIC via the DDS standard.

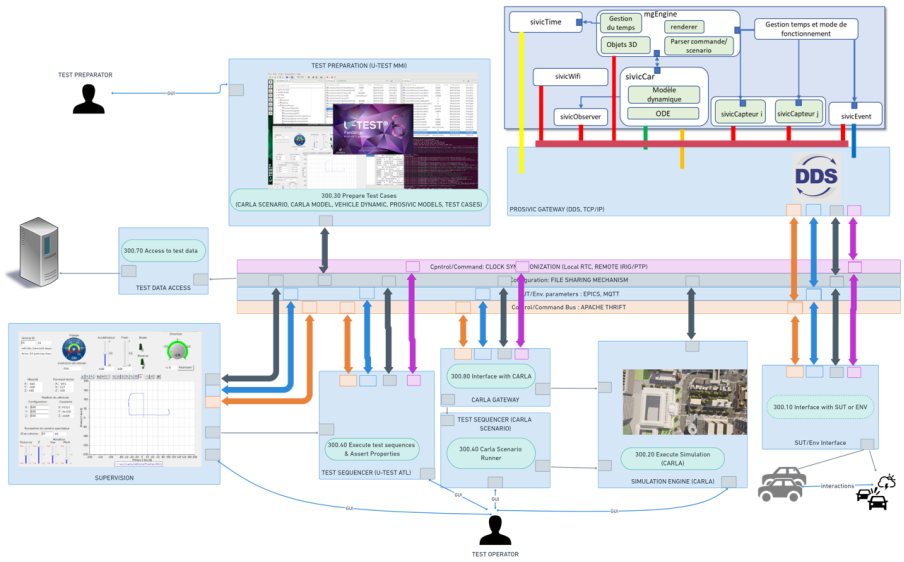


Figure 20: SPHEREA U-TEST integration with Carla and Pro-SiVIC through DDS

The two communication models of U-TEST VS, base on EPICS, and DDS differs in many aspects:

- **Data Model:**
  - **DDS:** Publisher/Subscriber model transferring complete data trees.
  - **EPICS (U-TEST VS):** Broadcast model focusing on scalar data.
- **Quality of Service:**
  - **DDS:** Offers multiple quality of service options to ensure data validity.
  - **EPICS:** Lacks these specific quality of service controls, focusing instead on real-time performance.
- **Integration Complexity:**
  - **DDS:** Requires case-by-case mapping for integration with EPICS-based systems.
  - **EPICS:** Typically simpler within its own ecosystem but more challenging when integrating with DDS.
- **Scalability and Flexibility:**
  - **DDS:** More scalable and flexible due to its robust data model and quality of service features.
  - **EPICS:** Generally less flexible outside its native environment, with scalability primarily focused on real-time constraints.
- **Use Case Suitability:**
  - **DDS:** Better suited for complex, heterogeneous environments requiring rigorous data integrity.
  - **EPICS:** Optimal for real-time applications within homogeneous systems where simplicity and speed are crucial.

The main challenges of this integration are related to the differences in the data models of U-TEST VS and Pro-Sivic DDS. DDS operates on a publisher/subscriber model that allows for the transfer of complete data trees, with various qualities of service ensuring the validity of the received data. In contrast, the U-TEST VS exchange bus is based on a broadcast model of scalar data, rather than a publish/subscribe model of data trees like DDS.

Mapping the two data models between DDS and U-TEST VS must be done on a case-by-case basis, depending on the DDS domain. This integration of the two functional domains is challenging and requires extensive testing with each model evolution. The work carried out under the PRISSMA project did not demonstrate the relevance of using a gateway between U-TEST VS and DDS. The benefits, compared to dedicated integration without using DDS, have not been clearly established. Significant modifications to the U-TEST VS stack would be necessary to enable proper integration of these data exchange buses.

2.2.2 Real-World Testing Conditions

In the framework of the development of the XiL platform (ViL, HiL, MiL, SiL) called ImPACT 3D, a real live analogous tests facility call ImPACT 3D AV has been developed. This facility consist to a Renault Zoé fully equipped with embedded hardware and software environment, an adapted power supply system, a low level architecture for actuators control, and a large set of embedded sensors. In 2024, an OBU (On Board Unit for communication) will be implemented in this vehicle. At the end, this ImPACT 3D AV platform will allow to implement L2, L3, and L4 services for automated driving and communication functions for cooperative application and CAV development.

2.2.2.1 Topology of sensors in the Renault Zoé

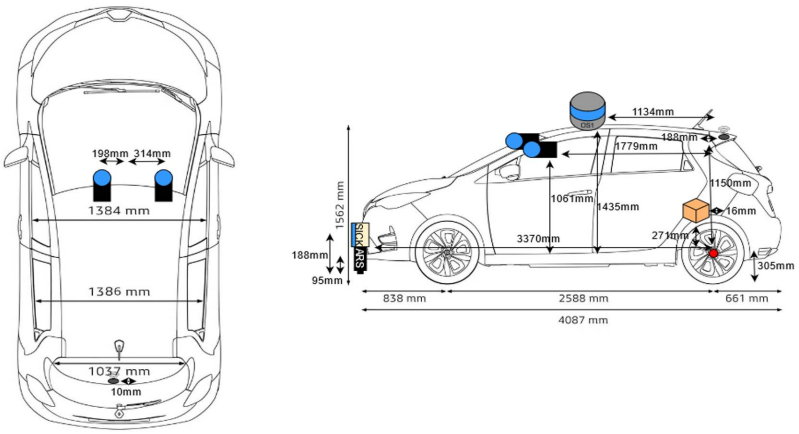


Figure 21: Sensor topology in both real and virtual Renault Zoé in ImPACT 3D (Source UGE).

Figure 22: Embedded architecture in the real Renault Zoé called ImPACT 3D VA (Source UGE).

The Full prototype workflows is made of different layers. The fist one concerns the sensors layer:

- stereovision PointGrey cameras,
- SICK LiDAR with 4 layers (SICK LD-MRS400001S01),
- A Continental RADAR ARS300,
- An INS sensor with accelerometers and gyrometers
- A natural GPS and a RTK GPS
- A neuromorphic camera (event camera)
- A LiDAR OS1 Ouster 64 layers Lidar.

These sensors are connected to the Windows PC through CAN, ethernet and serial interfaces.

The other layers are ADSF, software and interface:

- Car embedded computer (Windows 10): hosting ADSF under RTMaps software and communication platform with ROS computer. Under RTMaps, data from sensors, ADSF, and actuators are processing, recorded and synchronised.
- ROS Linux computer: This computer is mage for low level actuator control.

The last layer is dedicated to the actuators with acceleration and braking pedals, and steering wheel. Moreover, in order to provide information about road marking and to help the vehicle in the lane keeping function, the vehicle can use an HD Maps involving accurate attributes of the road markings and road geometry. This HD Map is obtain either with an OpenDrive file or a trk file. These files are used both in the simulation platform and in the actual prototype. Also, these files are useful for the path planning and trajectory following modules. The OpenDrive file is also used in Symuvia in order to generate different conditions of traffic.

2.2.2.2 Sensor strategy

Sensor deployment is application-specific, with different sensors required for tasks such as parking, braking, steering, lane change, takeover, active cruise control, city driving, and highway driving. There are four key areas to consider when defining a sensor strategy: cost, weight, performance, and size. These differ according to whether the vehicle operates at L1, L2 or L3. There are then also numerous technology considerations, including life-cycle of capability, maintenance, and installation of sensors. Another important factor is the technology maturity. Despite a growing number of applications for LiDAR sensors, the technology is still in its infancy, with a number of issues still to be resolved, including beam steering, laser technology, and receiver technology. Suppliers are reliant on OEMs to provide guidance on requirements, and these vary according to application, with differences between ADAS and AV requirements.

2.2.3 Choice of Artificial Intelligence Algorithm

The choice of the algorithms is subject to the Operation Definition Domain as it is described in the ODD subsection 2.1.2. The shuttle service must safely run in a two-lane road without overtaking and stopping when an obstacle is too close. The vehicle must also stop at bus stops to receive shuttle users. To achieve all this safely, the perception of the environment is essential. Thus, the capabilities of perception should be as follows:

- lane-marking detection in order the keep the lane when the shuttle is moving,
- dynamic and static obstacles detection (classification+ accurate positioning) in order to stop for the collision avoidance,
- and the bus-stops detection in order to engage the shuttle docking.

In the proposed autonomous shuttle system, the perception abilities are ensured by the LiDAR and camera sensors. And finally, the shuttle autonomous system calls the path planning algorithm to decide how to move in the perceived environment.

The object detection task mainly includes detecting whether there is an object of interest, classifying the object semantics, locating the object position, and determining the space range occupied by the object. Object detection is a classic task in the perception module of autonomous driving systems, where the common application is in two-dimensional RGB images. There are many kinds of automated vehicle sensors, and object detection based on LiDAR, millimeter-wave radar or ultrasonic radar is also essential. Object detection tasks can be divided into road participant detection (including vehicles, pedestrians, non-motor vehicles, etc, as in Figure 24(a)), traffic sign detection (including road boundary lines, lanes, traffic signs, cones, etc, as in Figure 24(b)), and general obstacle detection. Driving in public transportation spaces, autonomous vehicles can coexist harmoniously with other traffic participants, and friendly interaction is the premise and necessary condition for the realization of advanced autonomous vehicles. Therefore, it is particularly important to detect and identify objects such as pedestrians and vehicles in real-time during the driving process and even track and predict their moving trajectories, which contributes to understanding their driving intentions and making appropriate actions such as yielding, changing lanes, and overtaking.

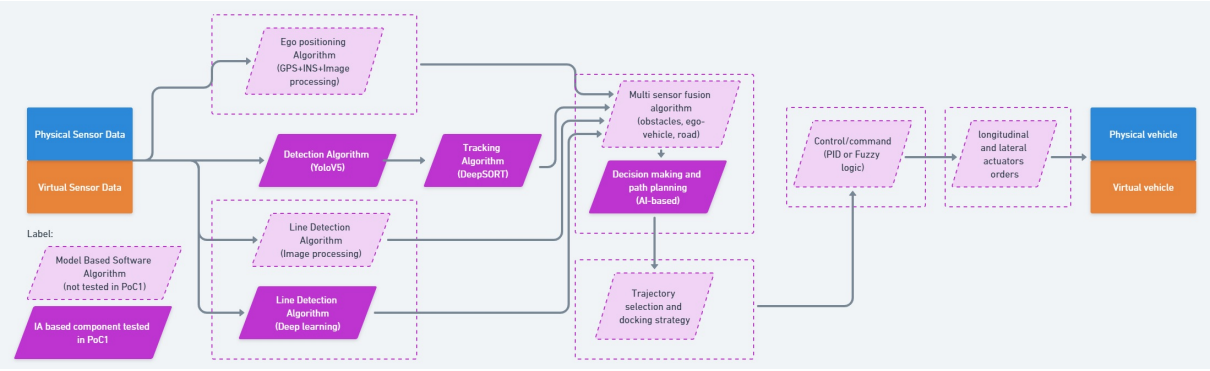


Figure 23: POC1, a simple overview of the functional architecture of the bus station autonomous desert service (Source UGE and ESI).

(a) Road participants (b) Lanes

Figure 24: Applications of object detection tasks

2.2.3.1 YOLOv5 in Object Detection

Object detection can be divided into two categories in general: one is the detection task of detecting the presence or absence of the target; another is the regression task of accurately returning the object detection frame. The classic deep learning models of object detection algorithms based on images are divided into two categories as shown by the object detection algorithms milestones [2] in Figure 25:

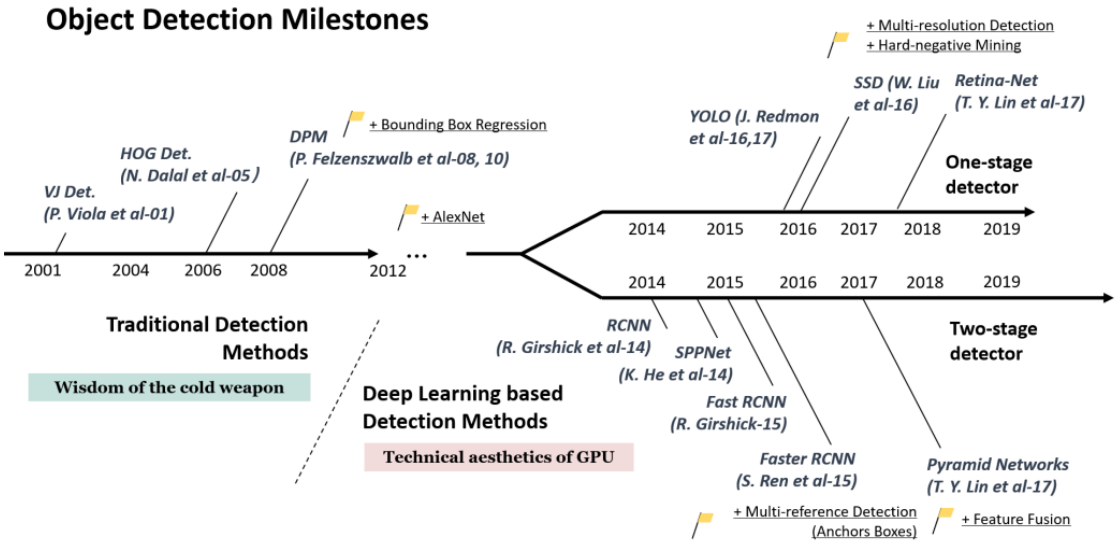


Figure 25: Milestones of Object Detection Algorithm [2]

- Two-stage model: firstly generate candidate regions, then classify these candidate boxes, and return the detection boxes. Generally, this kind of algorithm has a large amount of calculation but high accuracy, and the representative algorithms include sliding window detection [26], R-CNN [27], Fast R-CNN [28], and Faster R-CNN [29];
- One-stage model: evenly perform dense sampling in the image, and then directly perform classification and regression. This method is more efficient, and the representative

algorithms include YOLO[30] and SSD [31].

YOLO is the pioneer of the single-stage approach. It formulates the detection task as a unified, end-to-end regression problem, and is named after processing an image only once to obtain both location and classification. YOLO completes the prediction of bounding boxes and categories of all objects in the image in one network model, avoiding spending a lot of time generating candidate regions. Its strengths are detection speed and recognition, rather than perfectly locating objects.

Unlike object recognition algorithms, object detection algorithms not only need to predict the class label of the object but also need to provide the location of the detected object. The YOLO algorithm uses a fully convolutional neural network for the entire image, divides the image into multiple grid regions, and predicts the bounding box and probability of the target in each region, and the predicted probability of the target is then used to determine the accuracy of the bounding box. weighted to obtain accurate bounding box location and size.

There are a large number of real-life scenarios where autonomous vehicles must detect the location of all objects around them in real-time in order for the system to make correct decisions and controls. The YOLO algorithm can quickly locate and classify different objects and have a bounding box and corresponding classification label around each object.

YOLO model [30, 32, 33, 34, 4] has been updated with 5 versions, namely from V1 to V5. YOLOv5 [4] (Overall architecture is in Figure 26), a family of object detection architectures and models pre-trained on the COCO dataset, represents Ultralytics' open-source research into future vision AI approaches, incorporating lessons learned and best practices developed over thousands of hours of research and development practice. In the official version of YOLOv5, there are four object detection networks given, namely five models: YOLOv5n [4], YOLOv5s [4], YOLOv5m [4], YOLOv5l [4], and YOLOv5x [4].

The performance of these models is shown in Figure 27. From YOLOv5n to YOLOv5x, the detection accuracy of these models gradually increased, and the detection speed gradually decreased. There is a trade-off between the speed and accuracy of neural networks, and EfficientDet is a general term, which can be divided into EfficientDet D1 EfficientDet D7, the speed gradually slows down, but the accuracy also gradually improves. Compared to these terms, YOLOv5 not only achieves higher accuracy but also faster inference time. As a consequence, we deployed YOLOv5 into our real-time simulation as the functional module for object detection.

2.2.3.2 SORT and DeepSORT

The main task of Multiple Object Tracking (MOT) is to find the moving objects in a continuous image sequence, and identify the moving objects in different frames, that is, given an accurate id, of course, these objects can be Arbitrary, such as pedestrians, vehicles, various animals, etc. SORT (Simple Online and Real-time Tracking) algorithm [5] combines Kalman Filter and Hungarian algorithm, for the object motion state estimation and position matching respectively. As its name suggests, SORT is simple and can meet real-time requirements. On the one hand, its principle is simple and easy to implement; on the other hand, its speed is extremely fast, and it can reach 260HZ on the configuration of Intel i7 2.5GHZ/16GB (excluding the target detection time). At the same time, due to the introduction of object detection technology, the accuracy of tracking is also greatly improved, which can be said to be a good trade-off between accuracy and speed(as in Figure 28).

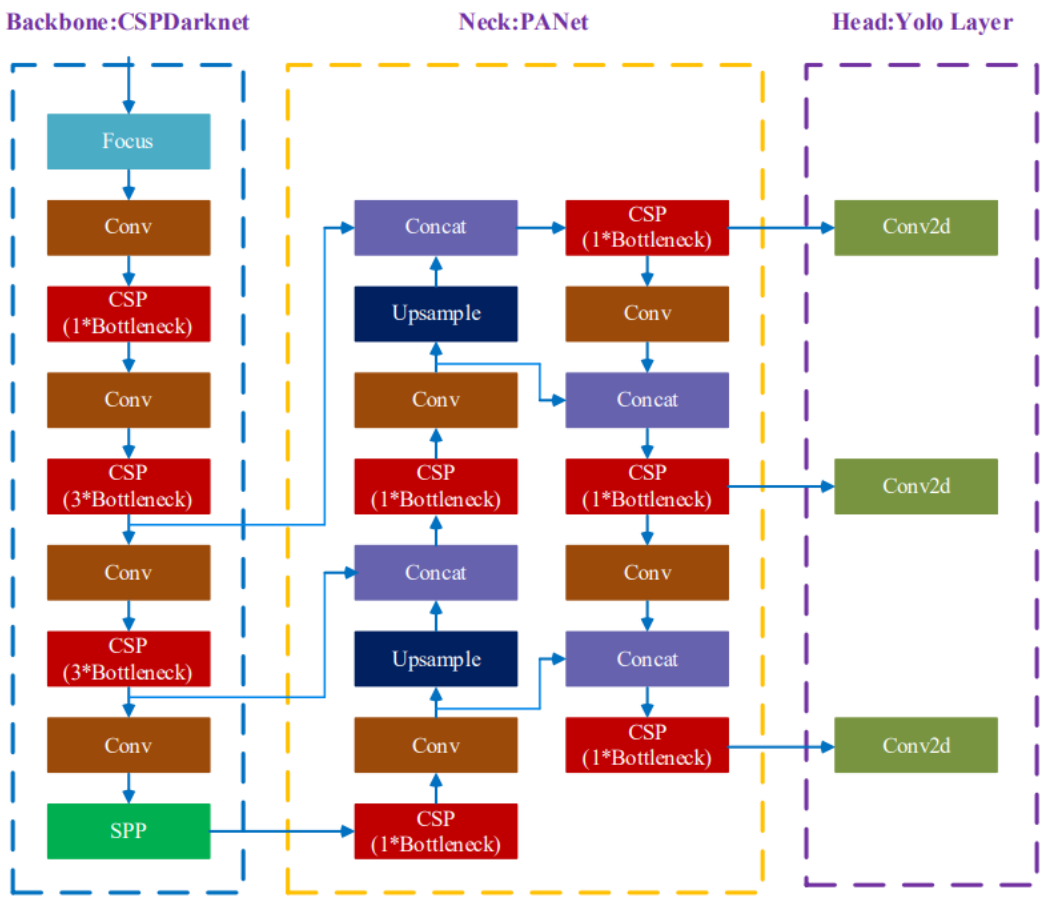


Figure 26: Overall architecture of YOLOv5 model [3]

The SORT algorithm uses the Kalman filter algorithm to predict the state of the detection frame in the next frame and matches the state with the detection result of the next frame to achieve vehicle tracking. However, once the object is occluded or not detected for other reasons, the state information predicted by the Kalman filter will not be able to match the detection result, and the tracking segment will end prematurely. After the occlusion is over, vehicle detection may continue to be performed, so SORT can only assign a new ID number to the object, representing the beginning of a new tracking segment. Therefore, the disadvantage of SORT is that it is greatly affected by occlusion and other conditions, and there will be a large number of ID switching. In order to overcome this problem, DeepSORT uses a simple (small amount of computation) CNN to extract the appearance features (low-dimensional vector representation) of the detected object (in the detection frame object), and after each (each frame) detection + tracking, the object is performed once extraction and preservation of appearance features. In each subsequent step, the similarity calculation between the appearance feature of the detected object in the current frame and the appearance feature stored before must be performed, and this similarity will be used as an important discriminant basis.

Due to the inherent strength (above-mentioned) of SORT and the improvement of the ID-switching problem, we employed DeepSORT [35] as our tracking algorithm, whose inputs are fed by the YOLOv5.

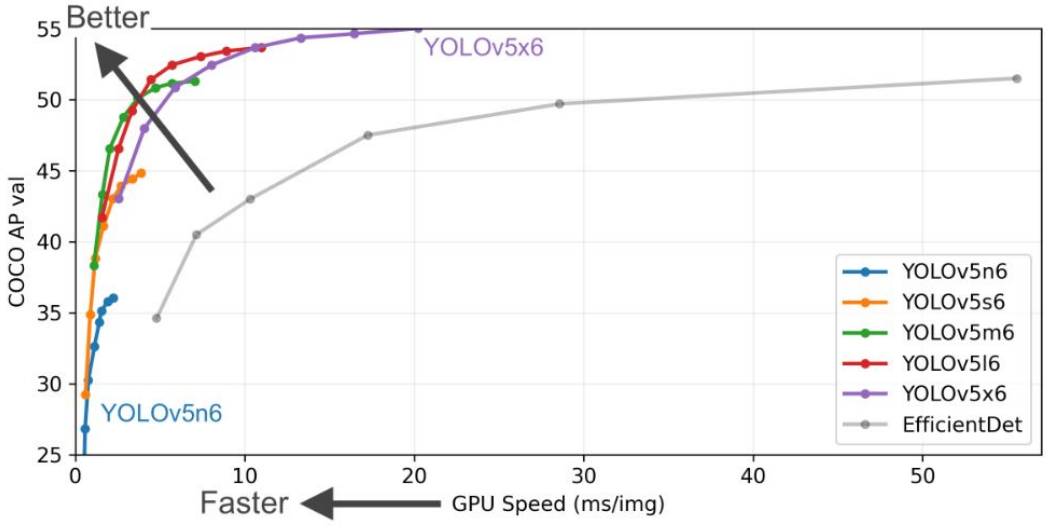


Figure 27: Performance of YOLOv5 on COCO dataset [4]

2.2.3.3 Ultra-Fast-Lane-Detection in Lane Detection

The lane detection algorithms can be divided into two categories: lane detection algorithms based on traditional image processing and lane detection algorithms based on deep learning processing.

In the first category, the lane area is always detected by edge detection, filtering, and other methods, and then combined with Hough transform [36], RANSAC [37], and other algorithms for lane detection, which suffer from the limitations:

- Limited application scenarios: Hough line detection algorithm is accurate but cannot perform curve detection, the fitting method can detect curves but is unstable, affine transformation can be used for multi-lane detection but serious interference in the case of occlusion, and so on.

- The perspective transformation operation will have some specific requirements on the camera. The image needs to be adjusted before transformation, and the installation of the camera and the inclination of the road itself will affect the transformation effect.

At present, the methods based on deep learning have become the current mainstream due to their high accuracy. As a fundamental component of autonomous driving, the lane detection algorithm is heavily executed. This requires an extremely low computational cost of lane detection. Besides, present autonomous driving solutions are commonly equipped with multiple camera inputs, which typically demand lower computational costs for every camera input. Another important problem occurs when the detection is based on scenarios with severe occlusion and extreme lighting conditions, which urgently needs a higher-level semantic analysis of lanes. Deep segmentation methods naturally have stronger semantic representation ability than conventional image processing methods and become mainstream.

As a consequence, we employed an Ultra-Fast-Lane-Detection detection algorithm [6] (overall architecture is shown in Figure 29), which is able to achieve SOTA-level performance while maintaining ultra-high detection speed. The lightweight version can achieve a detection speed of 300+FPS with close to SOTA performance. The model defined Lane detection as a set of

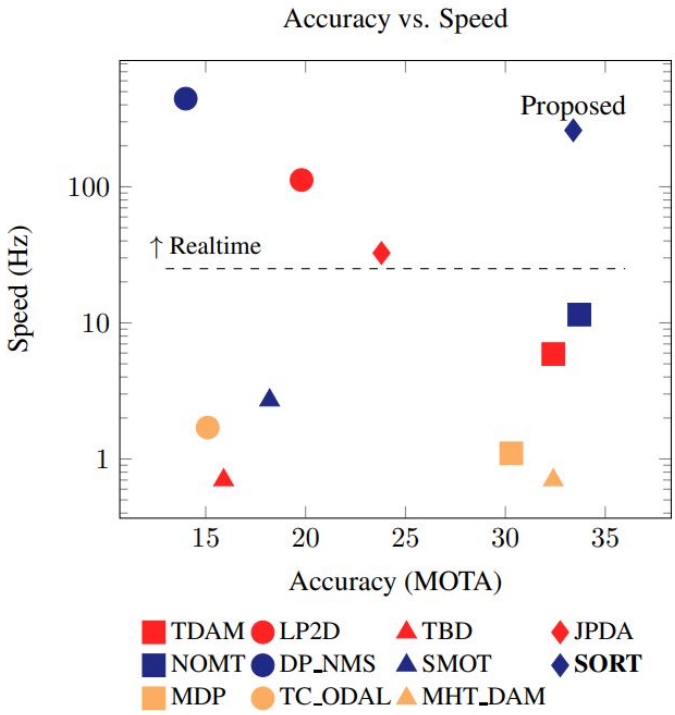


Figure 28: Benchmark performance of SORT [5]

finding the positions of lanes in certain rows in the image, that is, based on the position selection and classification in the row direction, as shown in Figure 30. Therefore, the method reduces the computational complexity to a very small range, solves the problem of slow segmentation, and greatly accelerates the process of lane detection. Besides, since the method is not a fully convolutional form of segmentation, but a general fully connected layer-based classification, the features it uses are global features. This directly solves the problem that comes from the difficulty in detecting complex lanes caused by a small local receptive field. For this method, when detecting the position of the lane line of a certain row, the receptive field is the size of the full image.

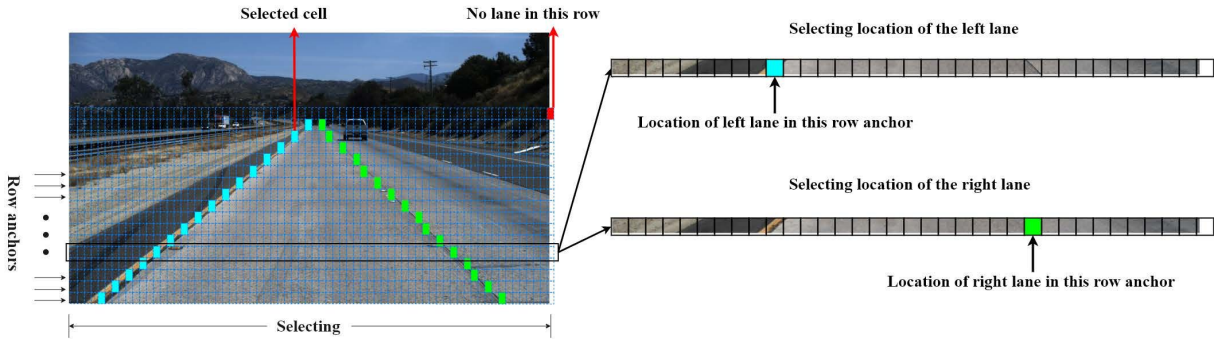


Figure 30: Definition of Lane Detection [6]

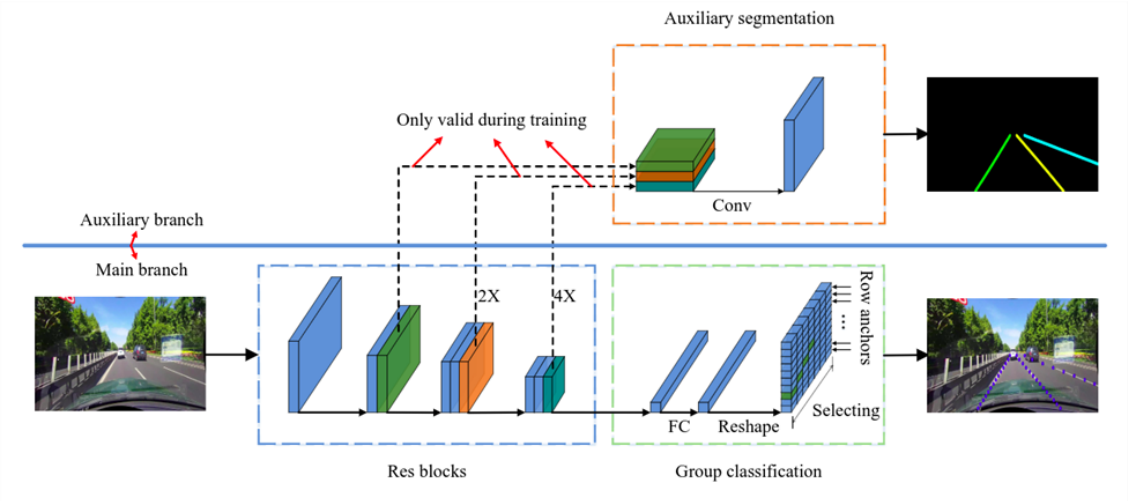


Figure 29: Overall architecture of Ultra-Fast-Lane-Detection model [6]

2.2.3.4 TensorRT for acceleration

NVIDIA TensorRT is an SDK for deep learning inference [7]. TensorRT provides APIs and parsers to import trained models from all major deep learning frameworks. Then generate optimized runtime engines that can be deployed in the data centers, automotive, and embedded environments as shown in Figure 31.

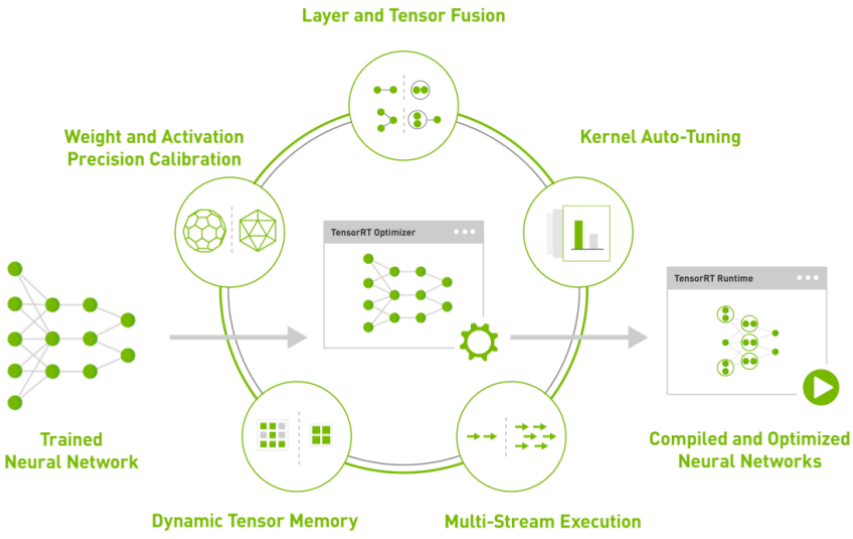


Figure 31: Deployment of TensorRT [7]

In the implementation part, we convert the PyTorch Trained model(YOLOv5, YOLOv5 with DeepSORT, and Ultra-Fast-Lane-Detection) to ONNX format as input and populated with a network object in TensorRT, which are used to generate the corresponding engines optimised for our platform.

2.2.3.5 Architecture

It is important to note that the virtual Co-Pilot (ADS) was prototyped and developed using the PROSIVIC platform (ESI group), coupled with the RTMaps platform (Intempora). Pro-SiVIC (presented in Figure 32) is a 3D simulation software designed for autonomous vehicle prototyping, test, evaluation, and validation: it features complex environments, realistic modelling of the vehicles, and of all the embedded sensors (cameras, radars, lidars). RTMaps is a data processing platform that controls the car and does all the trajectory computations, either in simulation (when connected to Pro-SiVIC) or in the field (when implemented on a real vehicle).

Figure 32: Pro-SiVIC, a generic and physically realistic simulation platform for sensors, vehicles, and the environment. Right: Co-Pilot application (Source UGE and ESI).

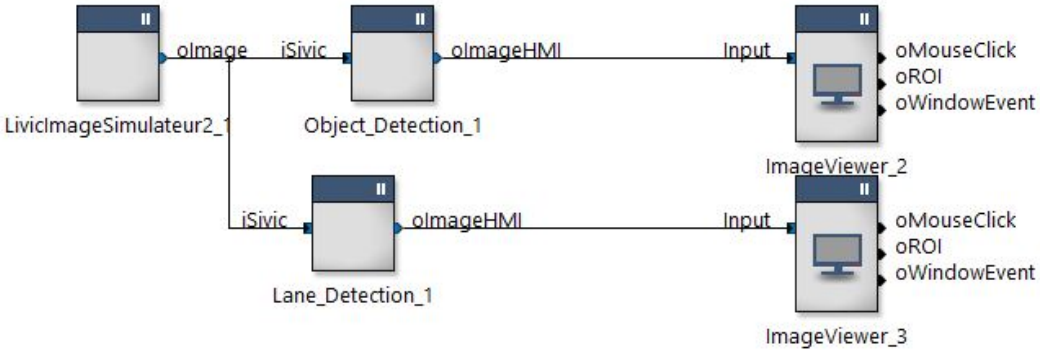


Figure 33: Architecture in RTMaps (Source UGE).

We implement the above-mentioned algorithms as the functional module inside the virtual Co-Pilot (ADS), as shown in Figure 33. YOLOv5 and DeepSort are coupled inside the Object Detection module, where YOLOv5 can work independently or combined with DeepSORT. In addition, Ultra-Fast-Lane-Detection is implemented in the Lane Detection module. Both modules take the image sequences (pixel size of 640x480) as the input during the simulation, and

output the visual and digital results, where the detected objects, Lane, and trackings are marked (as Figure 34 and 35), and the corresponding structured data is recorded.

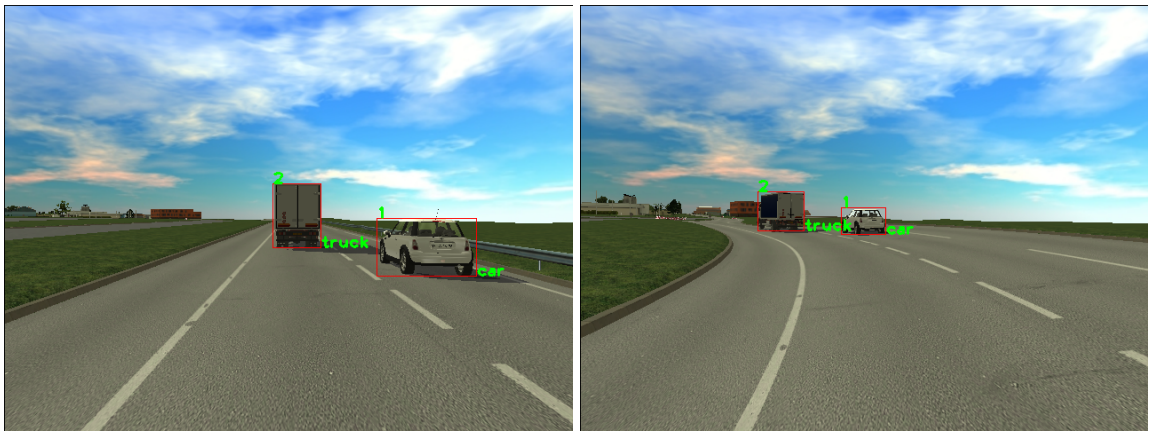


Figure 34: Object detection with tracking applied in virtual Co-Pilot (Source UGE).

Figure 35: Lane detection applied in virtual Co-Pilot (Source UGE).

2.3 Proposal of a generic multi-modal framework

In PRISSMA and with Pro-SIVIC, we have proposed, in the POC BuSAS, to develop a multi-modal generic framework based on 3 sub frameworks: the framework *Software*, the framework *Scenario*, and the framework *DataSet*. The framework *Software* provides the 4 main set of tools and models to implement for the development of a simulation architecture (see Figure 8. The second framework *Scenario* involves 3 layers: the definition and generation, the execution, and the analysis of scenario (See Figure 36). Finally, the last framework *DataSet* gives the way to generate the DataSet needed to obtain the data and ground truth which will be used by the evaluation process. This third framework involves the Dataset configuration, the Dataset generation, and the Dataset post-processing (see Figure 38). This global framework is called SiVIC-ADVeRSce. In both framework *Scenario*, and the framework *DataSet*, the ground truth identification, configuration, generation, and using are essential stages in both the DataSets production and the evaluation and validation stage. This generation of the ground truth will be addressed in the section 2.3.2. The following sections provide more details about these generic framework used in PRISSMA to implement the POC BuSAS.

2.3.1 General framework for scenario management

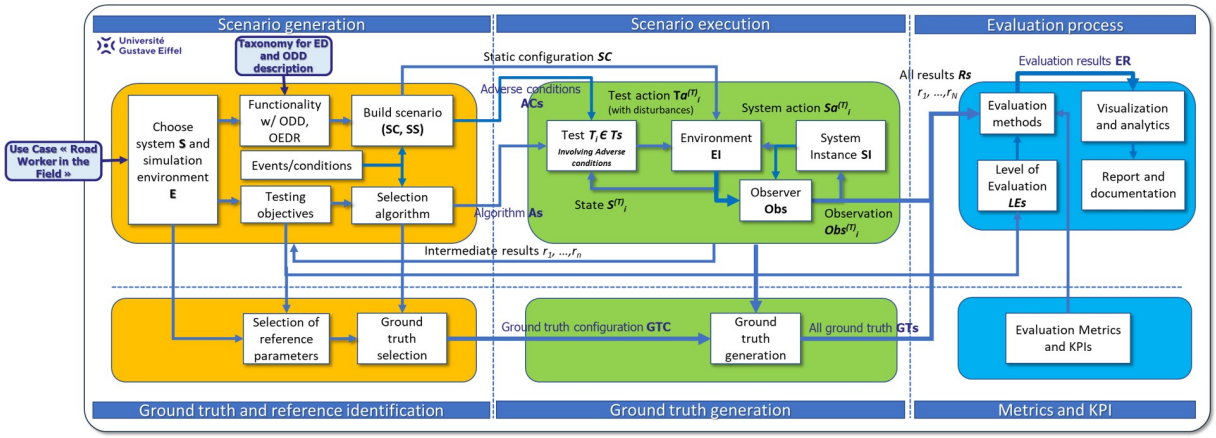


Figure 36: Generic Conceptual Framework of SiVIC-ADVeRSce: The scenario definition, management, execution (Source UGE)

2.3.1.1 The Scenario Generator

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 generating 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.

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

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:

- 1) At the system level, the overall performance and quality of the AI system are evaluated in simulated environments;
- 2) The components/functionalities level focuses on evaluating specific functions and algorithms necessary to meet the expected functionalities of the system;

- 3) 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 and Configuration

Scenario Definition

The scenario definition involves the conceptualisation and specification of the fundamental elements:

- **Scene** contains the overall environment where the scene takes place, including 1) Dynamic elements which are objects capable of movement or state changes, such as vehicles, pedestrians, or cyclists; 2) Static elements, which are stationary objects in the scene, such as road infrastructure or buildings; 3) 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

The scenario configuration 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 recognise 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.

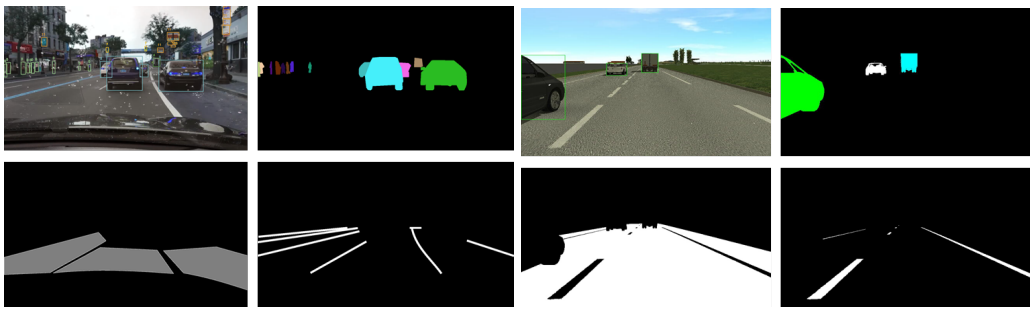
Algorithms selection

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 [38] has been widely used in visual perception research [39, 40] and provides ground truth labels for various tasks as shown in Fig. 37a. In the proposed framework, this type of ground truth is used for training and preliminary validation of the selected algorithm.



(a) BDD (b) Pro-SiVIC^{TM}

Figure 37: Ground truth of visual perception from the real world and simulation

In the simulation framework, the selection and configuration of the ground truth are based on the chosen algorithms and the characteristics of the environment. Ground truth data can be generated by using a physics engine to model the behaviour of the vehicle and its interaction with the environment, Fig. 37b shows the different ground truth for visual perception tasks in Pro-SiVIC^{TM}. The ground truth generated by the simulator will be collected by the executor and used for the final evaluation process.

2.3.1.2 The Scenario Executor

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. 8 illustrates a case of integration of two interconnected software, RTMaps$^{\text{TM}}$ and Pro-SiVIC$^{\text{TM}}$ in our framework. By utilising the capabilities of Pro-SiVIC$^{\text{TM}}$ to design realistic and complex virtual environments, developers can simulate various road, traffic, and weather conditions that their systems might encounter. Meanwhile, RTMaps$^{\text{TM}}$ 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. 8, it is worth mentioning that the Data Distribution Service (DDS) as the communication mechanism is integrated within the evaluation framework, which offers an effective and interoperable 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.

2.3.1.3 The Evaluator

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 LEs
  $Metrics \leftarrow Select(LEs)$ ▷ Select metrics for different evaluations
  $R_{metrics} \leftarrow Process(Rs\ ,\ GTs\ ,\ Metrics)$ ▷ Calculate the result of metrics $R_{metrics}$
  return $Visualize(R_{metrics})$, $Analyze(R_{metrics})$ ▷ Visualise and analyse the result of metrics $R_{metrics}$
end procedure

System evaluation

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. The 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.

Component evaluation

This level of evaluation focuses on the performance of individual algorithms or functions within the AI-powered system. The metrics are typically related to the functionalities of the system, as the metrics of the visual perception system shown in Table 1.

2.3.2 General framework for DataSets and Ground Truth generation

2.3.2.1 DataSet generation process and format

Conceptual Framework[8]

Table 1: Metrics cross different functionalities

Perception Function	Explanation	Metrics
Detection	Identifying and localising 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) [41], Multiple Object Tracking Precision (MOTP) [41], Higher Order Tracking Accuracy (HOTA) [11], etc.

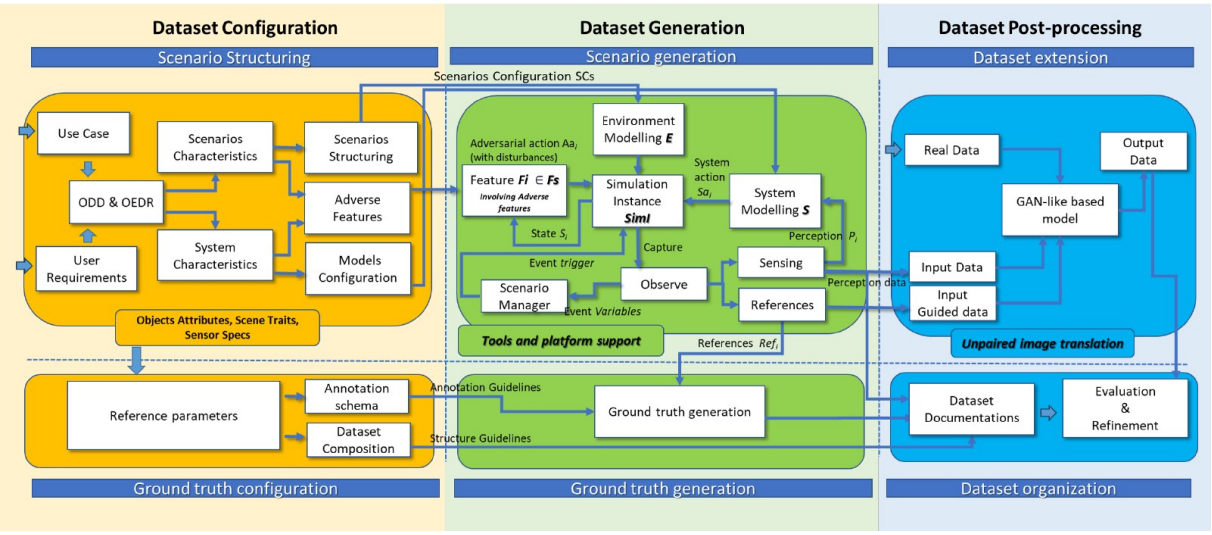


Figure 38: Generic Conceptual Framework of SiVIC-ADVeRSce: The Dataset definition, generation, and post processing (Source UGE)

The conceptual framework demonstrated in Figure 38 serves as the theoretical support for implementing the synthetic dataset generation framework for visual perception in adverse scenarios, which is inspired and extended by [42]. It provides a structured approach comprising three distinct layers: Dataset Configuration, Dataset Generation, and Data Post-Processing. Dividing the framework into sub-layers facilitates a more in-depth understanding of the whole process, and contributes to defining essential components and their corresponding interconnections. Moreover, we further parameterise each layer into an algorithmic presentation, outlining specific methodologies and a breakdown of the process, offering clarity on implementing its objectives and functionalities, to systematically and comprehensively guide the generation of synthetic datasets.

2.3.2.2 Upstream Layer: Dataset Configuration

Algorithm 4 DATASET CONFIGURATION

Input: Use Case *UC*, User Requirements *URe*
Output: Scenarios Configurations *SCs*, Model Configuration *MCs*, Adversarial Features *AFs*, Annotation schema *AS*, Dataset Composition *DC*
procedure SPECIFY(*UC* , *URe*)
    ODD , *OEDR* ← DEFINE(*UC* , *URe*) ▷ Define and specify ODD, OEDR based on *UC* and *URe*
    SysChs , *SceChs* ← CHARACTERIZE(*ODD* , *OEDR*) ▷ Extract the characteristics of scenarios and system *SysChs* and *SceChs* upon *ODD* and *OEDR*
    SCs ← STRUCT(*SceChs*) ▷ Struct (parameterize, specify, and configure) the key scenarios for dataset
    AFs ← BUILD(*SceChs* , *SysChs*) ▷ Build Adversarial Features *AFs*
    MCs ← BUILD(*SysChs*) ▷ Generate Model Configuration *MCs*
    OAs , *STs* , *SSps* ← GENERATE(*SceChs* , *SysChs*) ▷ Generate references parameters containing Objects Attributes *OAs*, Scene Traits *STs*, Sensor Specs *SSps*
    AS , *DC* ← FORM(*OAs* , *STs* , *SSps*) ▷ Generate the Annotation Schema *AS* and Dataset Composition *DC*
    return *SCs*, *MCs*, *AFs*, *AS*, *DC* ▷ Return for the next generation layer
end procedure

The Data Configuration layer is particularly crucial for establishing the foundational parameters and specifications necessary for the generation phase, mainly reflecting in configuring key scenarios, system models, ground truth, and adverse features. The main process of this layer, demonstrated in Algorithm 4, can be summarized into the following key steps:

- *Analyzing Use Cases and User Requirements*
  This initial step starts by thoroughly analyzing the intended use cases and users' requirements for the autonomous driving perception system. The former provides crucial insights into the operational contexts and environmental conditions that the system will encounter. Meanwhile, the latter forms the foundation for defining the system's specifications and capabilities. This analysis informs subsequent decisions throughout the dataset configuration process, ensuring the generated datasets are relevant and valuable for the intended applications.

- *Defining ODD and OEDR*
  Building upon the analysis gleaned from use case analysis and user requirements, the Operational Design Domain (ODD) defines the autonomous driving system's operational boundaries and environmental constraints. Incorporating ODD ensures that the synthetic datasets accurately represent the conditions the system is designed to encounter in real-world deployments. Meanwhile, the Object and Event Detection and Response (OEDR) framework describes the system's capabilities in detecting and responding to objects and events encountered within the defined ODD. This includes specifying object detection requirements, critical events recognition, and corresponding response strategies, which contribute to further identifying system and scenario characteristics.

- *Extracting Scenario and System Characteristics*
  Scenario and system characteristics are derived from the scope of ODD and OEDR. Scenario characteristics encompass a broad array of scenes, events, and criteria that are also fundamental elements of scenario definition. Meanwhile, system characteristics

refer to the configuration of different models, including sensor specifications, processing(perception) capabilities, and response (decision-making) mechanisms. These characteristics are the basis for parameterizing the configuration of datasets, so it is crucial to ensure their accurate representation of the diverse scenarios and systems specified by the ODD and OEDR.

- *Incorporating Adverse Features*
  Scenario characteristics provide environments where adverse conditions may arise, such as challenging weather, complex traffic patterns, or unexpected obstacles. Meanwhile, system characteristics contain inherent adverse features, including sensor limitations, potential model failures, processing constraints, and challenges related to decision-making. Integrating adverse features into dataset configurations ensures that datasets accurately represent the complexities of real-world driving scenarios.

- *Dataset Configuration*
  Finally, the dataset configuration is structured and organized to facilitate the dataset generation process. This involves configuring scenarios based on scenario characteristics, generating model configurations reflecting system characteristics, and incorporating adverse features. Additionally, object attributes, scene traits, and sensor specifications, derived from scenario and system characteristics play essential roles in defining the annotation schema and dataset composition. The annotation schema establishes guidelines and formats for labeling objects and events within the synthetic dataset, while dataset composition encompasses the structural organization, including the distribution of scenes, objects, and environmental conditions. This encompasses parameters such as scene complexity, object diversity, sensor configurations, and temporal dynamics.

2.3.2.3 Midstream Layer: Dataset Generation

In the dataset generation layer, the parameterized configurations defined in the upstream layer are translated into synthetic datasets through related platforms, tools, and techniques. The main process, outlined in Algorithm 5, involves several essential steps.

- *Environment and System Modelling*
  The environment and system are modelled aligning with upstream-defined configurations, which are the foundational building blocks for subsequent dataset generation processes.

- *Simulation Instantiation*
  Using the previously generated models, simulations are initiated and executed through suitable platforms and tools, facilitating the dynamic generation of diverse driving scenarios. These simulation instances incorporate different configured adverse features from the upstream layer, and the impact of system actions during the simulation process.

- *Observation from Simulation*
  During simulation execution, perception and reference information are systematically observed and recorded from the simulation instance using specialised tools. Perception data encompasses object and environmental observations captured by the autonomous driving system as sensor readings during simulation, which are then stored as raw data in the dataset. Reference data comprises ground truth information, which will be processed following the annotation schema configured in the upstream layer.

Algorithm 5 DATASET GENERATION

Input: Output from Framework Layer 4
Properties: Platform and Tools *P&T*
Output: Ground Truth Labels *GTLs*, Perception Data *PD*, References *REFs*
procedure GENERATE(*SCs*, *MCs*, *AFs*, *AS*)
    $SimI \leftarrow$ INSTANTIATE($SyS = P\&T$.MODELLING(MCs) , $E = P\&T$.MODELLING(SCs)) ▷ Instantiate the simulation *SimI* based on the modelling of environment *E* and the System *SyS*
    for $AF_i \in AFs$ **do** ▷ Add adverse features into simulation instances
        $Aa_i \leftarrow AF_i$.GENERATE(S_{i-1}) ▷ Generate corresponding Adversarial Actions Aa_i based on the last state S_{i-1}
        while ISRUNING(*SimI*.UPDATE(Aa_i) **do** ▷ Update the simulation instance with adversarial action and run
            Ref_i , $P_i \leftarrow$ CAPTURE(*SimI*) ▷ Observe and capture the sensor perception P_i and references Ref_i
            $Sa_i \leftarrow SyS$.GENERATE(P_i) ▷ Generate System Actions Sa_i based on the perception P_i
            SimI.INTERACT(Sa_i) ▷ Simulation instance interact with system actions
        end while
    end for
    $GTLs \leftarrow$ GENERATE(COLLECT(Ref_i) , *AS*) ▷ Generate ground truth labels based on annotation schema
    return *GTLs*, *PD* = COLLECT(P_i), *REFs* = COLLECT(Ref_i) ▷ Return for the post-processing layer
end procedure

Any issues or errors encountered during simulation execution must be addressed before passing the final results to the post-processing layer. For instance, if the intermediate results of the simulation instance exceed the defined ODD and OEDR, it may necessitate identifying and reconfiguring scenarios or systems. Upon completion of executions, the final results, comprising raw data, reference data, and labelled ground truth, are transmitted to the downstream layer for post-processing.

2.3.2.4 Downstream Layer: Dataset Post-processing

Within the Data Post-processing layer, two essential components are employed to enhance and prepare the synthetic datasets for downstream tasks: Dataset Enhancement and Dataset Organisation, as shown in Algorithm 6.

In recent years, various types of generative adversarial networks (GAN) have emerged with the aim to enhance computer-generated images [43]. Our approach draws inspiration from these techniques, allowing us to generate variations of synthetic images from the proposed dataset. Specifically, our method is an unpaired image translation built on a Cycle-GAN architecture [44]. Depending on the learning base used for model training, it facilitates post-processing of various types of images within the dataset, including in clear weather, rain, or fog.

- *Algorithm inputs*
  The algorithm receives as input computer-generated images and an input guided data associated with the ground truth. The guided data can be depth or segmentation maps and guides the model to focus on particular areas of the image.

- *Unpaired image translation*
  The image translation technique, based on a Generative model such as Cycle-GAN, involves learning a mapping function between an input image and a target image, in order to transferred to an another domain. Specifically, it enables the transformation of input data into real data in this scenario.

After collecting all perception data, as well as reference data and annotation labels, the Dataset Post-Processing layer proceeds to organize and structure the datasets. This involves generating comprehensive documentation to provide details about the dataset's composition, structure, and attributes.

Finally, the dataset is ready for evaluation, training, and additional refinement processes.

Algorithm 6 DATASET POST-PROCESSING

1: **Input:** Output from Framework Layer 5
2: **Output:** Dataset for ready *DSetR*
3: **procedure** POST-PROCESSING(*GTLs*, *PD*, *REFs*, *DC*)
4: *PD* ← ENHANCE/EXTENSION(*PD* , *REFs*) ▷ Enhance the dataset
5: *DSetR* ← GENERATE(*GTLs* , *PD* , *REFs* , *DC*) ▷ Generate the organized dataset
6: **return** EVALUATE&REFINE(*DSetR*) ▷ Evaluate and refine the datasets
7: **end procedure**

The dataset was built along the Digital Model (as shown in the figure in the section 2.5 about the final implementation) 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. In the scenario, 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.

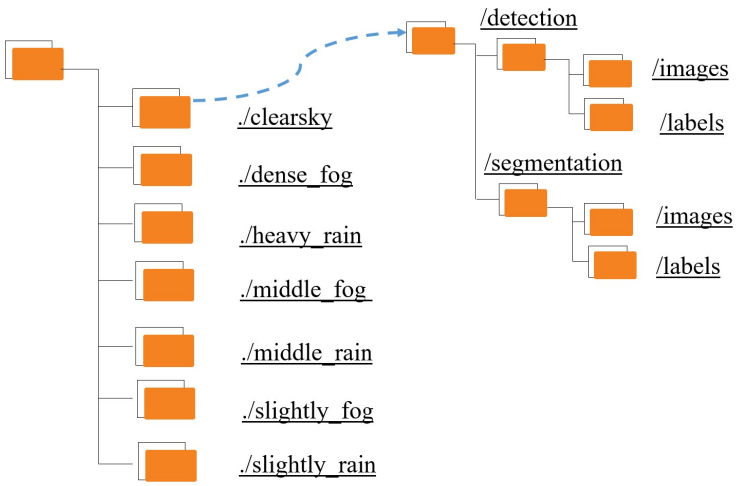


Figure 39: Structure of the dataset generated in BuSAS

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

2.3.2.5 Ground truth and reference generation

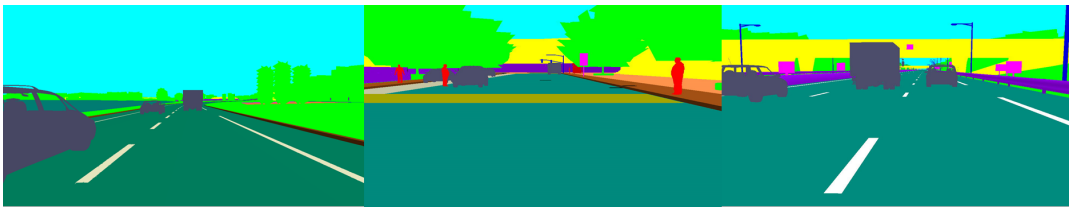
Automated vehicles rely heavily on perception systems to interpret their surroundings and generate decision and information for path planning module allowing the operating of automated driving. However, developing and validating these perception systems require extensive testing and evaluation in simulated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

During the simulation process of Pro-SiVIC, perception data and ground truth data are gathered using specific plug-in in Pro-SiVIC and some module in RTMaps through the data-sharing mechanism developed to apply an efficient interconnection between several applications either on the same computer or remote. For instance, as illustrated in Figure 40(b), simulated image frames produced in Pro-SiVIC are captured and stored by the sensor module defined in RTMaps to construct the dataset. Various mechanisms embedded within Pro-SiVIC are employed to generate the reference data. One method involves altering the rendering texture of objects and the environment (such as vehicles, pedestrians, lanes, roads, buildings, etc.), resulting in the creation of segmentation masks (depicted in Figure 40(d)) that are then collected as part of the reference data.

(a) Scenario from Test track, City Center, Highway

(b) Camera readings (simulated images)

(c) Depth information (visualize as depth map)



(d) Segmentation mask

Figure 40: Generic Conceptual Framework of SiVIC-ADVeRSce: Data collected from Pro-SiVIC involving Depth Map and segmentation ^{TM} (Source: UGE)[8]

In addition to visibility-based mechanisms, a specific mechanism known as the ”observer” in Pro-SiVIC facilitates the real-time generation of the state vector of different objects in the scene, including vehicles, pedestrians, static objects, and road configurations. Notably, the depth matrix (visualised in Figure 40(c)) of sensors can also be captured as reference data,

which can contribute to refining annotation and improving the dataset. The ”observers” are addressed in this deliverable in the next section.

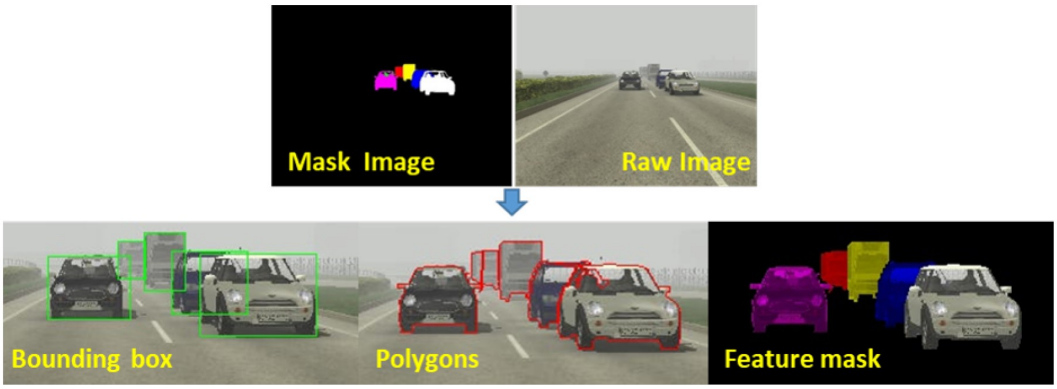


Figure 41: Generic Conceptual Framework of SiVIC-ADVeRSce: Generation of a set of annotation (Source UGE)

As defined in the conceptual framework, the annotation labels are generated based on the configured annotation schema from the upstream layer. In the implemented SiVIC-ADVeRSce framework, by leveraging various mechanisms for reference data generation, multiple annotation schema possibilities have been provided, each corresponding to different functional aspects of perception. These annotation schema are primarily categorised into object, semantic, and temporal domains, enabling comprehensive annotation of multi-data modalities. Within object annotations, the goal is to obtain precise annotation of objects using bounding boxes, polygons, and pixel-level masks. Figure 41 illustrates the different object annotations in the implemented framework. The second type of annotation implemented in SiVIC-ADVeRSce, namely semantic annotations, encompasses the entire perception data and allows for the extraction of coherent sub-segments or regions, assigning meaningful labels to each segment based on its semantic content. Furthermore, SiVIC-ADVeRSce extends its annotation schema to include the temporal aspect, enabling the annotation of timestamps, events, and temporal segments. This feature aligns with the virtual timestep in Pro-SiVIC, ensuring accuracy and consistency throughout the annotation process.

2.3.2.6 Data from ground-truth sensors

Multiple observers exist, based on the type of objects that we want to observe from the environment. These sensors give ground-truth data regarding the exact state of various objects in the simulation. This is useful when validating the sensor-based algorithms, as provided values are the ideal result of such algorithms. Conversely, this is also useful to test the control algorithms without having to worry about the correctness of input values.

- **Generic object observer** is a generic observer which produces the position and orientation values of the targeted object, which can be any positionable object. The values are exported in Table 2:
- **Car observer** produces information about the state of the vehicle. The values are exported in Table 3 :

# position of Objects	# angle of the objects
0: Object Coordinate X (m)	3: Angle X (rad)
1: Object Coordinate Y (m)	4: Angle Y (rad)
2: Object Coordinate Z (m)	5: Angle Z (rad)

Table 2: Description of the data frame provided by Generic object observer

# vehicle speeds	# forces applied to the tires
0: Speed X (m)	18: Tire Force X Front Left (Newton)
1: Speed Y (m)	19: Tire Force Y Front Left (Newton)
2: Speed Z (m)	20: Tire Force Z Front Left (Newton)
# vehicle angular velocities	21: Tire Force X Front Right (Newton)
3: Angle Speed X (rad/s)	22: Tire Force Y Front Right (Newton)
4: Angle Speed Y (rad/s)	23: Tire Force Z Front Left (Newton)
5: Angle Speed Z (rad/s)	24: Tire Force X Rear Left (Newton)
# rotational speeds of vehicle wheels	25: Tire Force Y Rear Left (Newton)
6: Wheel Speed FrontLeft (round/s)	26: Tire Force Z Rear Left (Newton)
7: Wheel Speed FrontRight (round/s)	27: Tire Force X Rear Right (Newton)
8: Wheel Speed RearLeft (round/s)	28: Tire Force Y Rear Right (Newton)
9: Wheel Speed RearRight(round/s)	29: Tire Force Z Rear Right (Newton)
# torque of the wheels	# acceleration vector
10: Torque FrontLeft (Newton.m)	30: Acceleration X (m/s^2)
11: Torque FrontRight (Newton.m)	31: Acceleration Y (m/s^2)
12: Torque RearLeft (Newton.m)	32: Acceleration Z (m/s^2)
13: Torque RearRight(Newton.m)	# radiuses of the wheels
# wheel heading	33: WheelRadius FL (m)
14: Wheel Angle(rad)	34: WheelRadius FR (m)
# position of vehicle	35: WheelRadius RL (m)
15: Vehicle Coordinate X (m)	36: WheelRadius RR (m)
16: Vehicle Coordinate Y (m)	# rotation vector
17: Vehicle Coordinate Z (m)	37: Angle X (rad)
	38: Angle Y (rad)
	39: Angle Z (rad)

Table 3: Description of the data frame provided by car observer

- **Road observer** produces information about the position of the vehicle relative to the road track. This sensor requires a reference road track definition file. The values are exported in Table 4:

- **Human observer** produces information about the state of the human character. The values are exported in Table 5:

2.3.2.7 Data from simulated sensors

Different artificial algorithms(Object Detection, Tracking, Lane Detection, etc) are employed as functional modules in our autonomous driving systems prototype. At the same time, a set of sensors corresponding to the real world are simulated in Pro-SiVIC, providing various types of data as input for the functional modules that have been developed and are possible in the future.

- **Camera**: The camera object holds sensor configuration and can be configured through different properties. Meanwhile, a list of filters can be added as camera functions, whichs aim to enhance the camera. The definition is detailed in guidance document of Pro-SiVIC.

  It is worth mentioning that a wide-angle camera (FOV $\geq 180°$) is able to be created by

# road profile	# position of the left side of the road
0: curvilinear abscissa (m)	9: X coordinate of the right road border (m)
1: Curvature	10: Y coordinate of the right road border (m)
2:Heading (rad)	11: Z coordinate of the right road border (m)
# road center position	# current vehicle parameters
3: X coordinate of the road center (m)	12: Ego vehicle X state (m)
4: Y coordinate of the road center (m)	13: Ego vehicle Y state (m)
5: Z coordinate of the road center (m)	14: Ego vehicle Z state (m)
# position of the left side of the road	15: Ego vehicle heading (rad)
6: X coordinate of the left road border (m)	# position of the vehicle relative to the road
7: Y coordinate of the left road border (m)	16: Lateral deviation between Vehicle and road center (m)
8: Z coordinate of the left road border (m)	17: Heading deviation between vehicle and road center (rad)

Table 4: Description of the data frame provided by road observer

# speed	# position of pedestrian
0: speed (m/s)	5: Pedestrian Coordinate X (m)
# torque applied to the pedestrian	6: Pedestrian Coordinate Y (m)
1: Torque (newton.m)	7: Pedestrian Coordinate Z (m)
# pedestrian angle	
2: Angle X (rad)	
3: Angle Y (rad)	
4: Angle Z (rad)	

Table 5: Description of the data frame provided by human observer

using a fish eye camera object. The entire fish eye field is captured by the sensor. This plug-in uses 6 intermediate cameras to compute the view over the whole field, then reconstruct the final image. Fish eye cameras work in a similar way to the standard cameras, thus accepting a very similar set of properties. However, there are some differences. Unlike the standard camera, fish eye cameras do not support filters yet. They do not support multisampling or high-precision colour formats either, corresponding properties are thus not available.

The simulated camera sensors export image data to separate image files for each period, with specified colour depth and file format. In each of its entries, the main output file contains a timestamp followed by the width and height of the image and then is passed to the next module as input or stored on the disk. In addition, the images inside the output can also be displayed in the main rendering window (camera resolution is independently configured at the sensor level).

- **LiDAR** : The simulator features two methods to simulate a LiDAR range finder, the first makes use of ray tracing to compute collision to the closest object, and the latter uses the image depth buffer projection distances.

  The simulated LiDAR sensors export frame data in the main output file. In each of its entries, this file contains a timestamp followed by the number of impacts (frame width), then a list of distances, then a list of luminances (one distance and one luminance per impact).

2.3.2.8 Data from functional modules

The data that we obtain from the different functional modules is represented in Table 6:

Functional model	Input Data frame	Output Data frame
Object Detection	Raw images from camera sensors Size of input images Classification Id (Class_ID) Corresponding name to Class_ID Threshold of confidence Max. Number of BBOX	BBOXs of detected objects Class_ID of BBOXs Confidence of BBOXs Result images with annotations Size of output images
Tracking	Raw images from camera sensors Size of input images Ouput BBOXs from Object Detection Corresponding name to Class_ID Threshold of confidence Threshold of NMS Dimension of features Max. Age (Threshold pf deleting a track) Nearest Neighbor Distance Metric parameters	Result images with annotations Size of output images Tracker List of matched detections List of unmatched detections
Lane Detection	Raw images from camera sensors Size of input images Number of row anchors Number of gridding cells Number of lanes	Result images with annotations Size of output images Probability of each location Location of lanes

Table 6: Description of the data frame provided by different functional modules

2.3.3 General framework and methodology for Digital Models generation

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

- **Define Objectives and Scope**: Clearly define the objectives of creating the digital twin and identify the scope of the physical system or asset to be modelled. In the case of PRISSMA, the objectives and scope are define by the different POCs but addresses systems of systems and AI-based systems evaluation and validation for automated mobility. Different environment have been identified (open road like Paris2Connect, and controlled environments like UTAC, Transpolis, and Satory test tracks).

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

benches, human expertise, observation facilities, and theoretical and physical knowledge are essential.

- **Modelling and Simulation**: Develop models that accurately represent the physical system or asset, including its structure, behaviour, and interactions with the environment. In automated mobility, these models are represented and presented in the four domains (User, Ego-Vehicle, Infrastructure, and Mobility) of the figure **??**. Some needed models to develop are provided for each domain. Use simulation techniques to validate and refine the models, ensuring they accurately capture the dynamics and the high fidelity behaviour of the real-world system.
  - **Digital Model Development**: From the high resolution 3D model, a set of 3D lighter (but representative) models are extracted or generated (meshes, material, textures, ...). This stage is provided in the figure 42 by the green and cyan boxes. The data of the Digital Shadow may also include in addition to the spatial information, sensor data, environmental conditions, and operational parameters. Cleanse, preprocess, and transform the collected 3D data to ensure consistency, accuracy, and compatibility with the digital twin environment and a real-time operating. This may involve data reduction, sharing, filtering, noise reduction, calibration, alignment, and normalisation.
  - **Digital Shadow Development**: A Digital Shadow relies on historical data from sensors, databases, and operational systems to understand past behaviours and trends. It utilises physics-based models or simulations to accurately replicate physical entity interactions and behaviours, enabling predictive analysis and optimisation. Like the Digital Twin, methods used to develop dynamic and physical models include data-driven approaches using AI, physics-based modelling with mathematical equations, and hybrid approaches integrating both techniques for digital twin development. Nevertheless, Digital Shadow stay a Model and software in the loop approach without real-time links with the real system.
  - **Digital Twin Development**: Implement the digital twin environment, including software platforms, databases, and communication infrastructure. Integrate the developed models into the digital twin framework, ensuring interoperability and scalability. This stage involves to develop mathematical models, algorithms, and simulations to represent the behaviour, dynamics, and interactions of the physical entities or systems. This includes creating 3D geometric models, physics-based models, control algorithms, and scenario simulations. Integrate the processed data and simulation models into a cohesive digital twin framework. Merge the diverse data streams and models to create a comprehensive representation of the physical entities or systems.
- **Deployment and Integration**: Deploy the digital twin into operational environments, ensuring seamless integration with existing systems and processes.This part, for a sub part of Digital Shadow, corresponds to the last stage (blue box) of the figure 42
- **Feedback and Continuous Improvement**: Establish feedback loops to capture insights from users and real-world observations, incorporating them into the digital twin to improve accuracy and reliability. Continuously update and refine the digital twin based on new data, changes in the physical system, and evolving requirements.

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

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

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

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

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

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

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

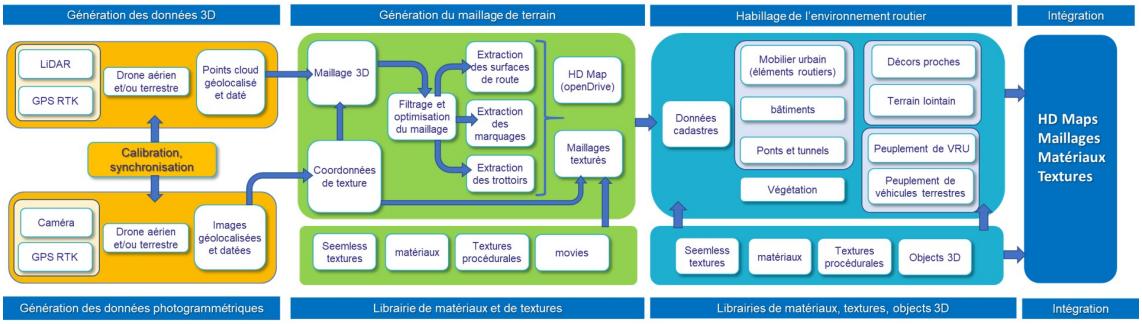


Figure 42: Process for the generation of Digital Model (source: UGE).

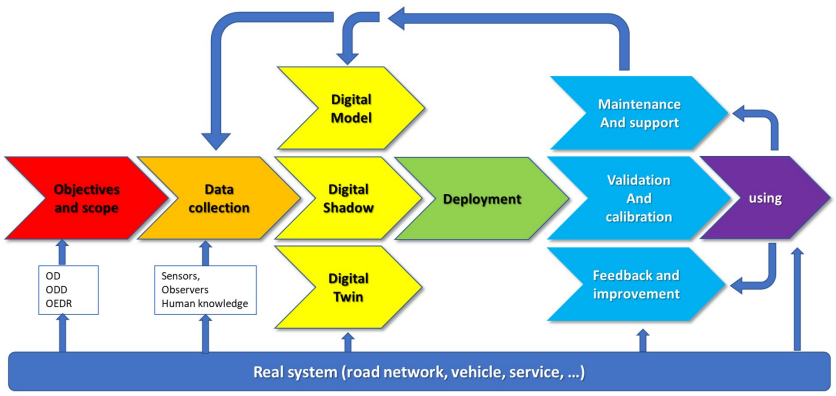


Figure 43: Digital Twin process of development (source: UGE).

2.3.4 Generic and interoperable simulation framework

2.3.4.1 Overview of the modules, tools, functions involved in a global architecture

In PRISSMA, from the implementation made for the POC 1, it appeared a clear generic framework involving the different tools, models, and platforms defined in the previous sections of this deliverable. It appeared too that propose a generic simulation framework is very complex and needs to share this problem in sub-problems. From the global framework (non exhaustive) presented in the figure 44 and proposed by UGE, we have an overview of this complexity. in order to be more understandable, a simpler generic framework has been proposed in the figure 45. A part of this framework has been used and implemented in the POC 1 (figure 46). It is interesting to emphasise that similar functions and tools with different complexities appear in this framework. It is the case for the traffic generation and management. With low density traffic situation, it is recommended to use the models (vehicles, 2 wheels, and pedestrians) proposed in the simulation platform (in our case Pro-SIVIC). In this condition, each vehicle (generally limited to 15 till 20 vehicles) can be controlled by a specific decision-making and path planning algorithm implemented as a package or a DLL in the application environment (RTMaps). In the case of a very dense traffic with a couple of hundred vehicles, it is recommended to use a dedicated simulator (Traffic generation tool in the generic framework). Of course, both can be used in same time. We favour the complex and dynamic model in Pro-SiVIC for the ego-vehicle, and the traffic simulator for populating the environment with simpler evolution models (i.e. IDM).

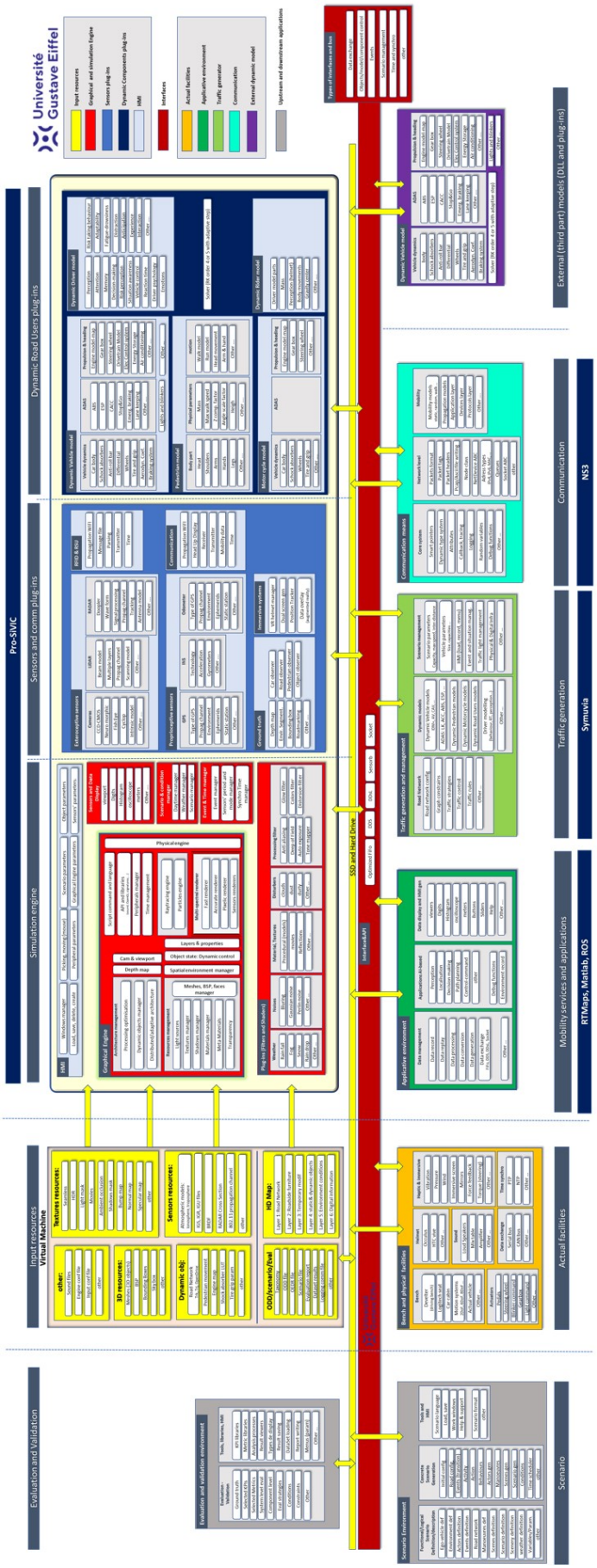


Figure 44: Final overview of the generic simulation framework proposed by UGE for the evaluation and validation of AV (Source: UGE)

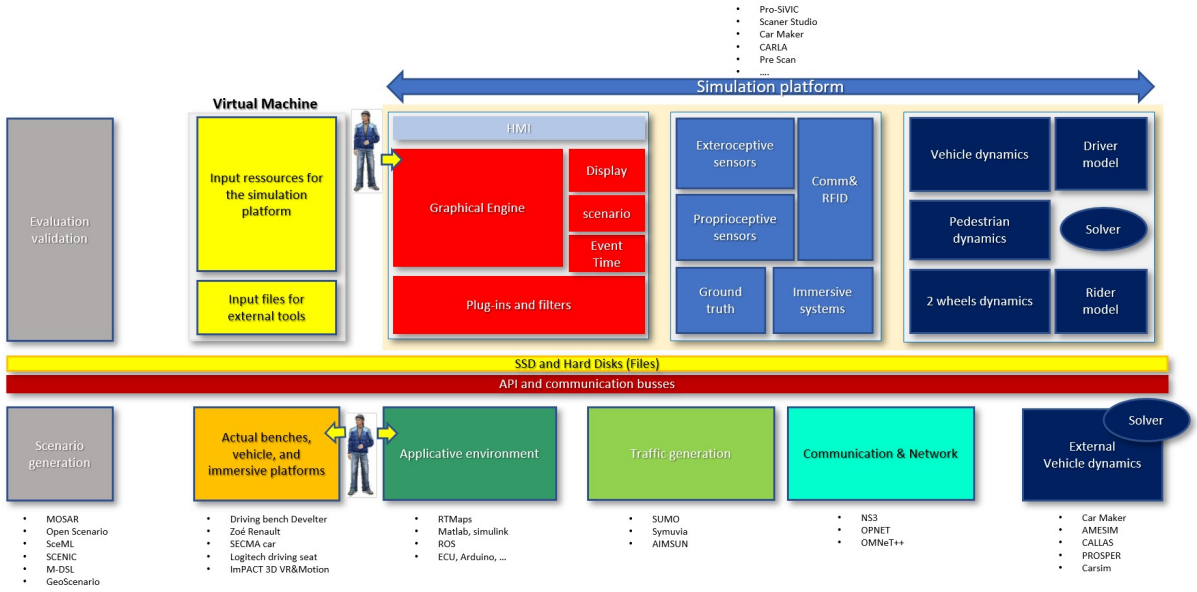


Figure 45: Simplified generic simulation framework proposed and developed by UGE (Source: UGE)

2.3.5 Simulation platforms derived from the generic framework

2.3.5.1 Overview of Driving Simulation environment

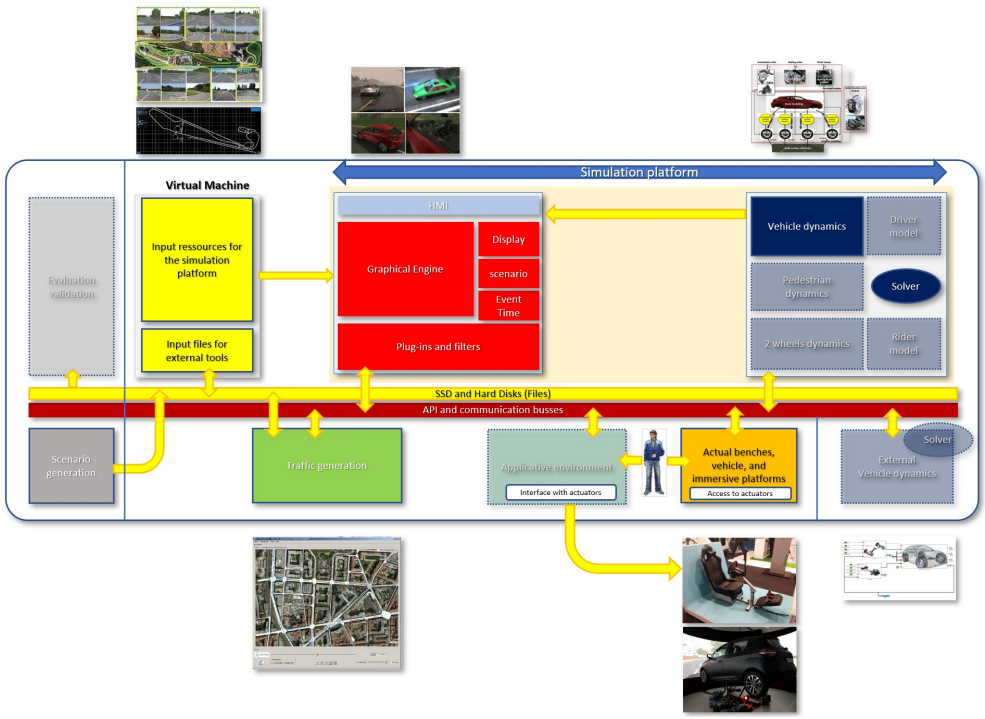


Figure 47: Proposal of a Driving Simulation architecture from the generic framework proposed by UGE (Source UGE)

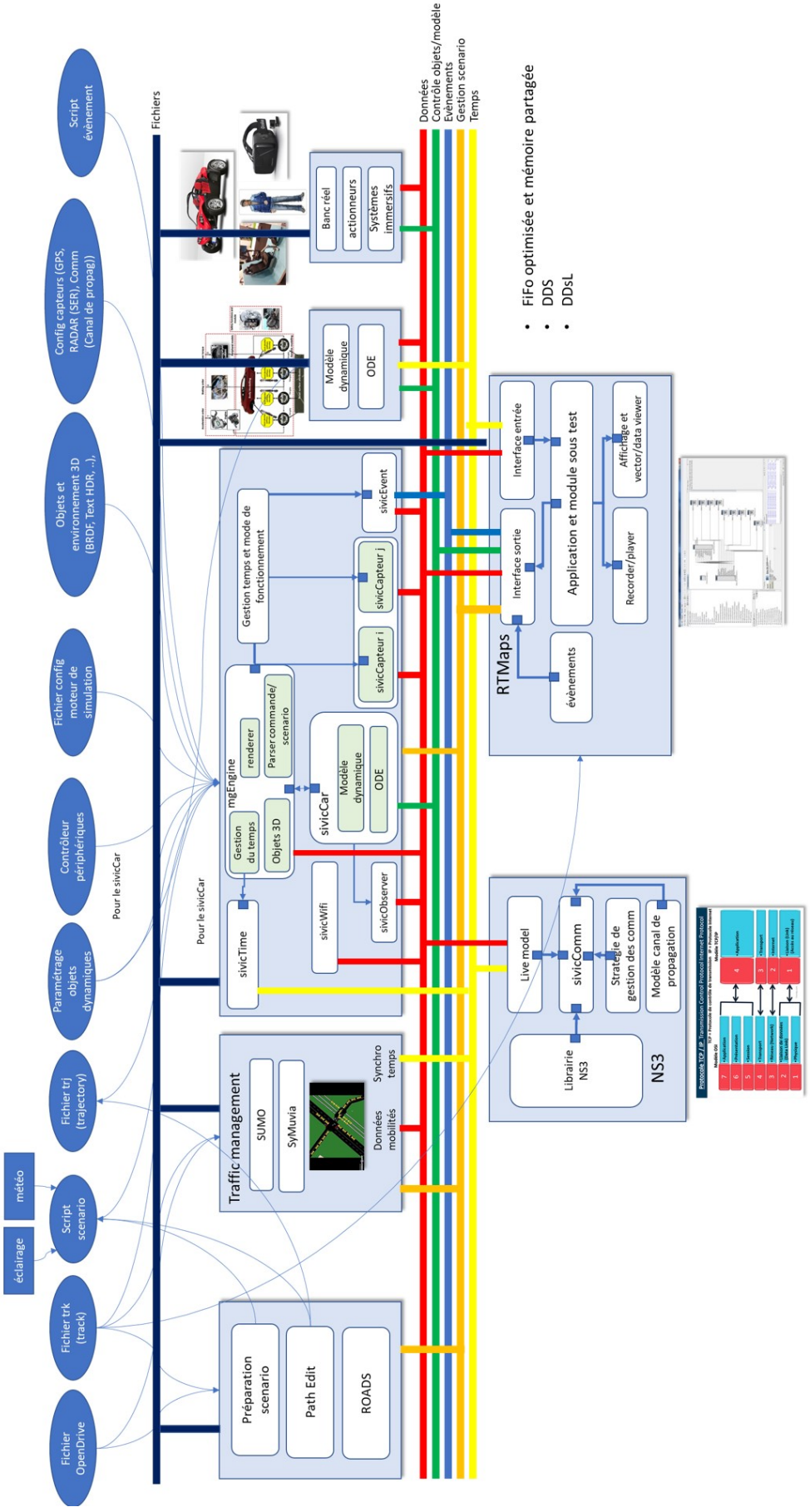


Figure 46: Implementation of the POC 1 simulation platform (Source UGE)

2.3.5.2 Overview of AV and CAV simulation environments

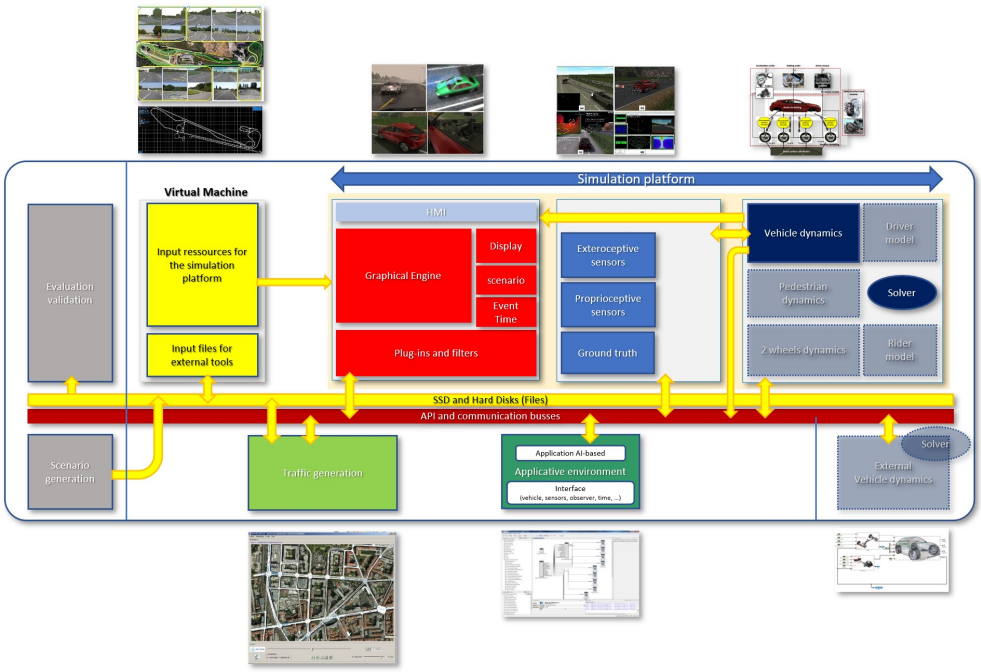


Figure 48: Proposal of an Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

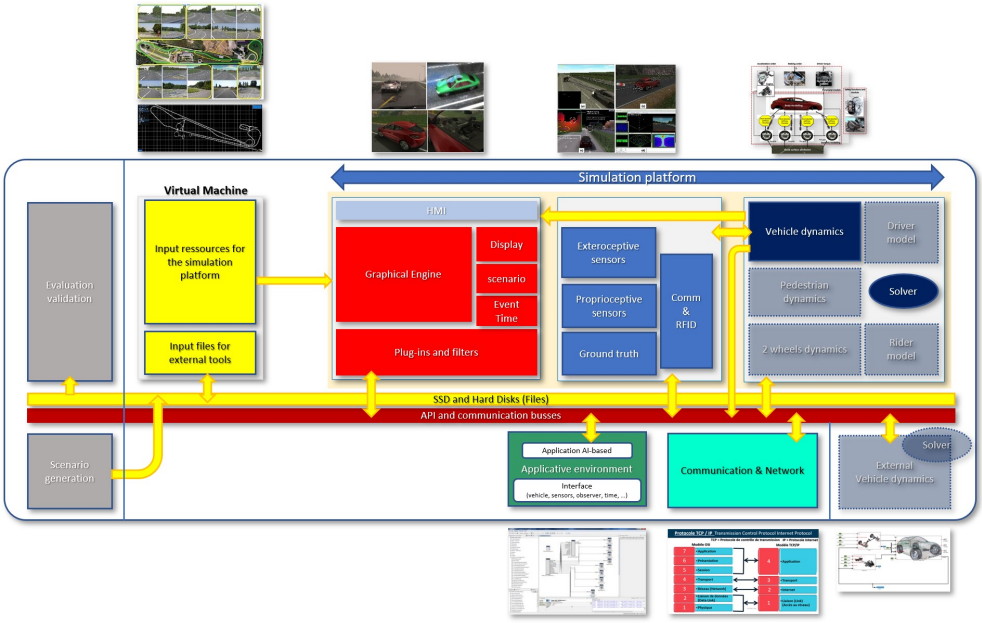


Figure 49: Proposal of a Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

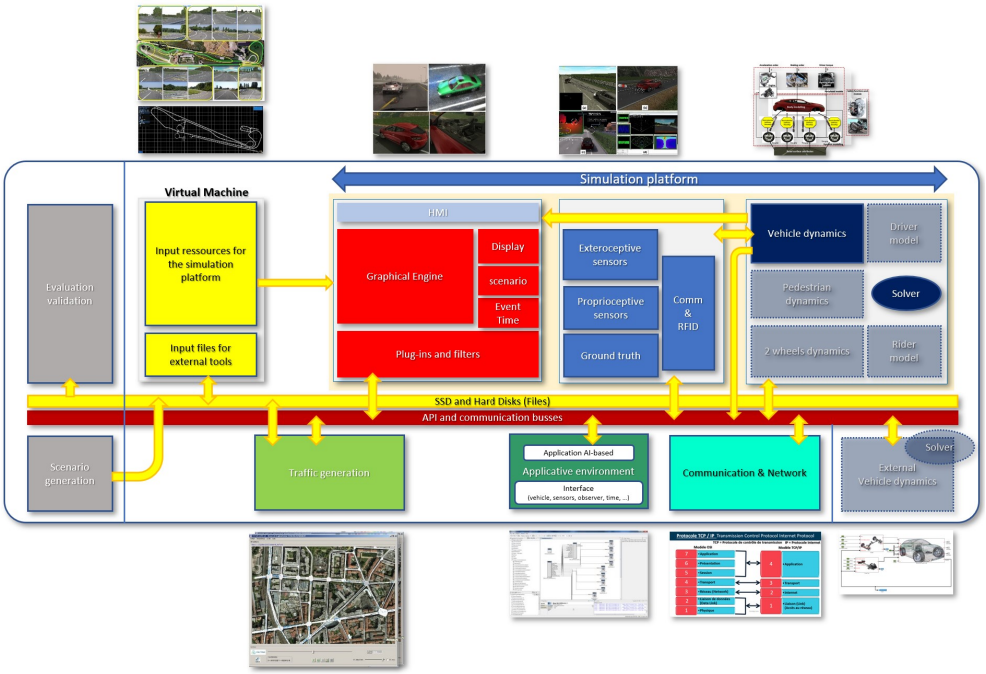


Figure 50: Proposal of a Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

2.3.5.3 Overview of the distributed simulation environment

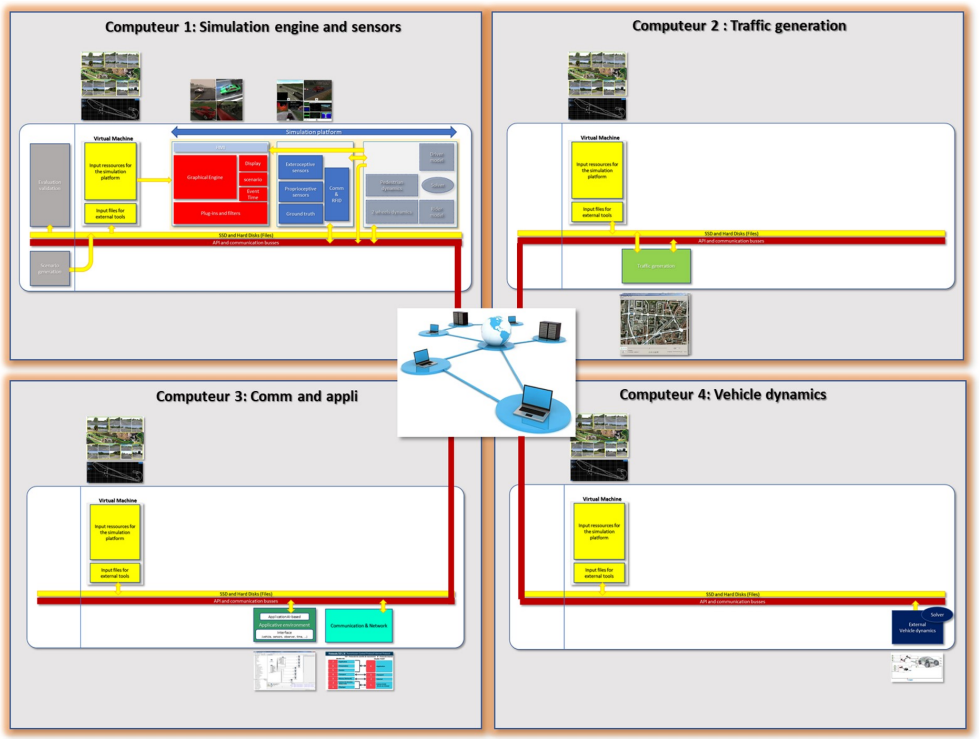


Figure 51: Proposal of a Distributed Connected and Automated Vehicle Simulation architecture from the generic framework proposed by UGE (Source UGE)

2.3.5.4 Overview of ViL simulation environments

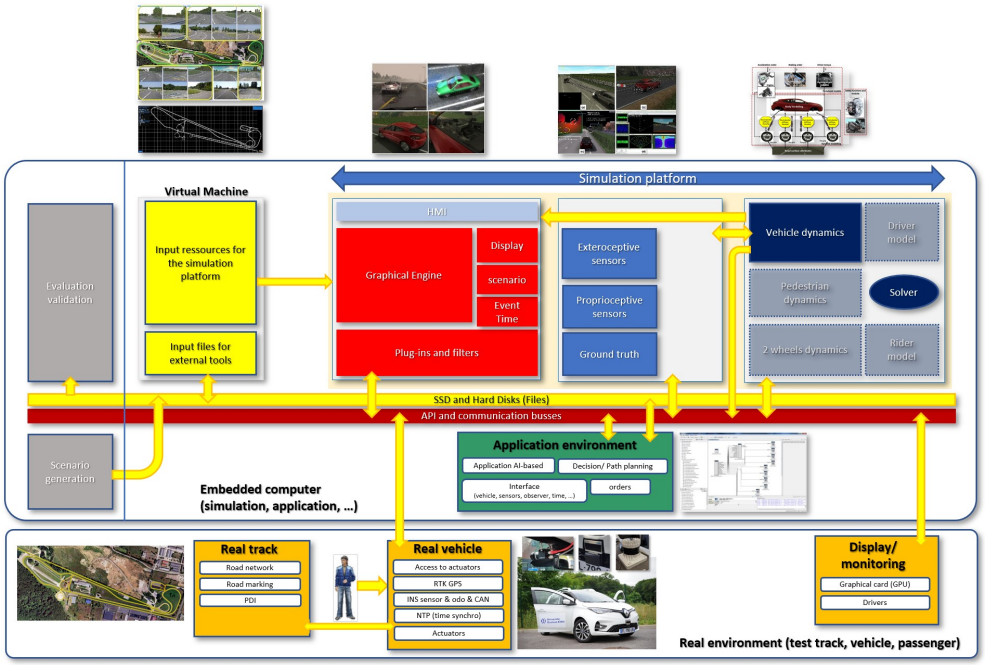


Figure 52: Proposal of a Vehicle in the Loop architecture involving AV simulator and application environment from the generic framework proposed by UGE (Source UGE)

The figure 53 presents a complex architecture merging a simulation framework, a dynamic and immersive platform (CKAS, Zoé cabin, dashboard, sound, steering wheel and force feedback, ...), and the real automated Zoé Renault. This platform is called ImPACT 3D and is developed by UGE under the responsibility of Dominique Gruyer. In this architecture, the first ego vehicle is controlled by using API and control law using the external vehicle dynamics implemented in an external tool or library (DLL). The control laws generate orders used by the CKAS motion system. The simulation engine provides the rendering of the road scene using the Digital Model of the Satory test tracks. The second ego-vehicle corresponds to the real automated Zoé Renault moving on the real Satory test tracks. The embedded RTK GPS and INS/Odo sensors provides the reference about the second ego-vehicle state vector in order to control the virtual second ego-vehicle (avatar) in the simulation environment. Concerning the first ego-vehicle, the state vector is sent to the real ego-vehicle in order to provide an augmented reality usable by the embedded application environment. In this real environment, the LiDAR data and the camera images are augmented and enriched with the projection of the virtual ego-vehicle. In both real vehicle and simulation platform, it is possible to put in the loop 2 real drivers. The synchronisation of the different part of this distributed platform is done by using of PTP and NTP. NTP used for synchronisation at the application level. NTP has a coarse-level granularity, and a lack of synchronisation guarantee requirement. But it is enough to synchronise the computers and applications in the real ego-vehicle. NTP is mainly a software synchronisation. PTP is a hardware synchronisation system used for accurate synchronisation. PTP is used for critical applications and deletes the network and equipment delay and jitter in the time accuracy.

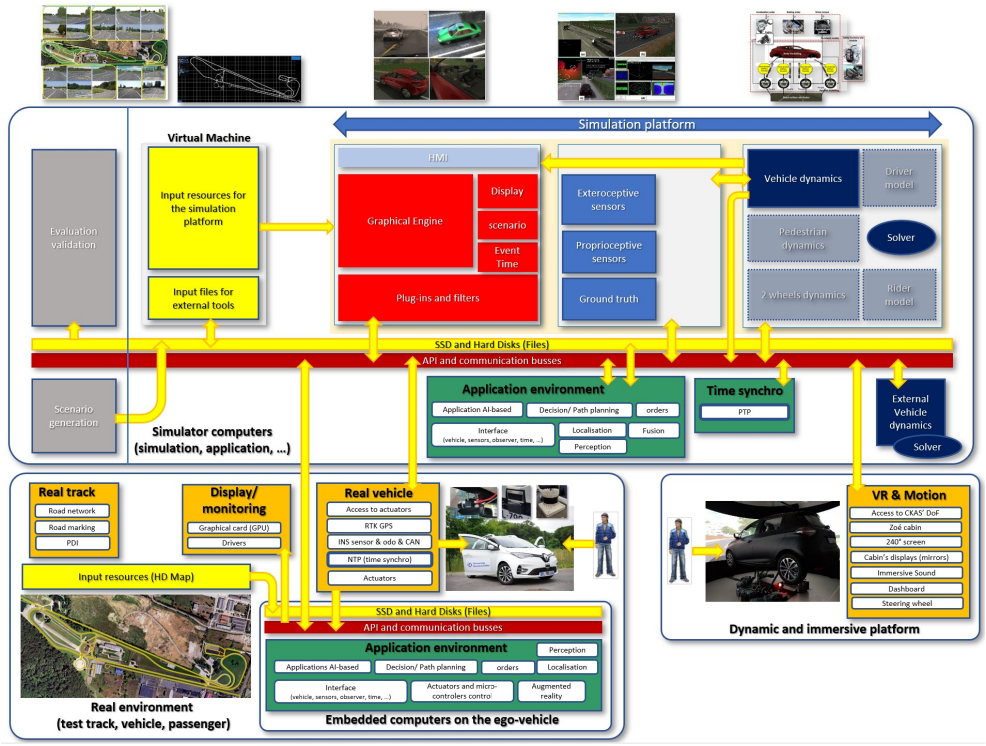


Figure 53: Proposal of an Interconnected platform as the concept of ImPACT 3D. ImPACT 3D is developed by UGE and will work in real time with a real proprtype on the test track and a dynamic and immersive simulation platform. This distributed, interconnected, and dynamic platform relies on the generic framework proposed by UGE (Source UGE)

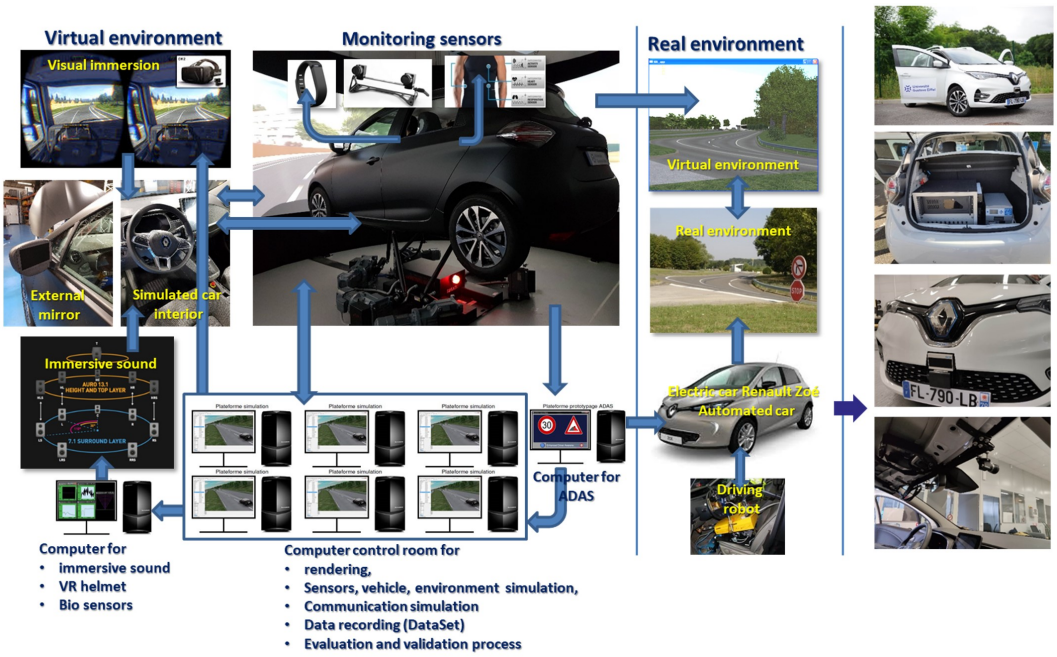


Figure 54: Impact 3D: an interconnected platform with dynamic and immersive platform and real automated vehicle (Source UGE)

2.4 Methods, procedures, and protocols for evaluation and validation

The evaluation process has to be done at 3 levels: system, component, and model/tool. In the POC 1 called BuSAS, even if we have developed the full service, we have focused our effort mainly on the second level (evaluation and validation of AI-based perception modules). The two other levels are addressed in other POCs.

2.4.1 Evaluation of 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 miss-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), miss-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 [9, 45, 46, 47]:

- **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 55). When the distribution of positive and negative samples in the test set changes, ROC curve can remain unchanged

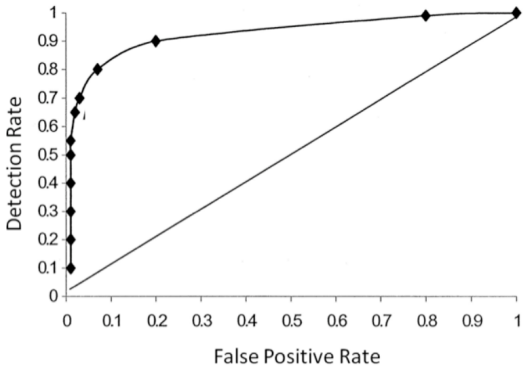


Figure 55: An example ROC curve (Detection Rate vs. False Positive Rate) [9]

- **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).

2.4.2 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[41, 9, 48] 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 56, the switch from red to blue is recorded as an IDSW, the sum of the number of mismatches in the entire simulation.

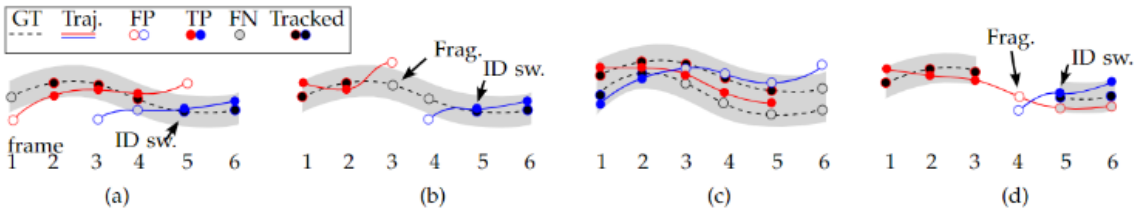


Figure 56: Prameters of MOT evaluation [10]

- **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.

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

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)** [11] 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 57).

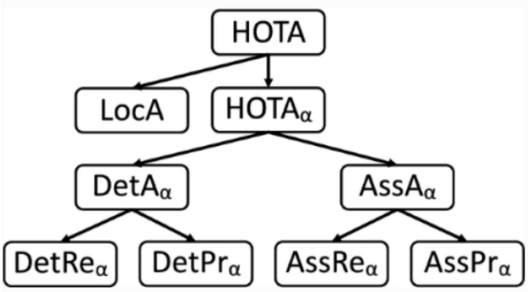


Figure 57: Sub-metrics of HOTA [11]

- **LocA (Localization Accuracy)**: Average localization similarity averaged over all matching detections and averaged over localization theresholds.
- **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 58) averaged over all matching detections and then averaged over localization thresholds.
- **AssPr(Association Precision)**: TPA/(TPA+FPA) (shown in Figure 58) averaged over all matching detections and then averaged over localization thresholds.

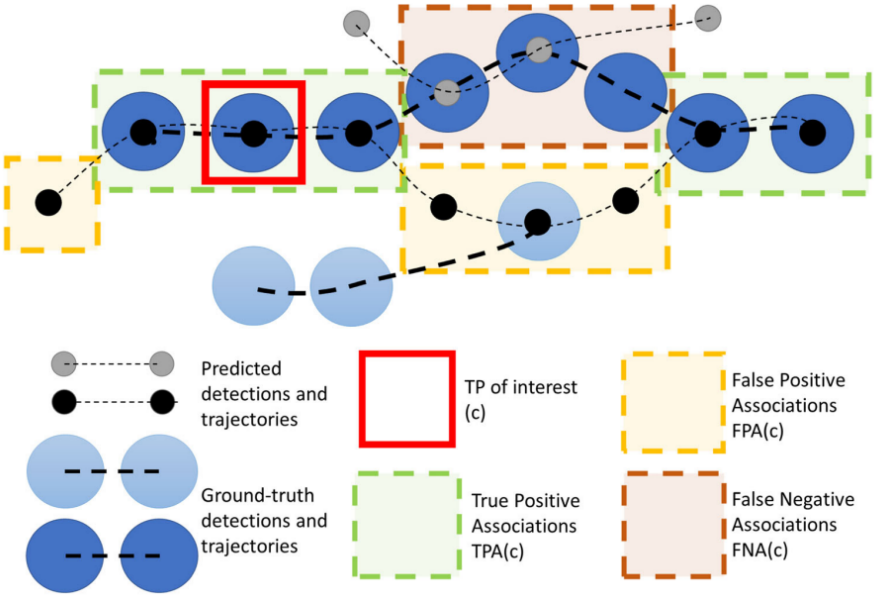


Figure 58: Parameters of ASSA [11]

2.4.3 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 [12]:

- **Accuracy of Lane Feature Extraction**: As shown in Figure 59, 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

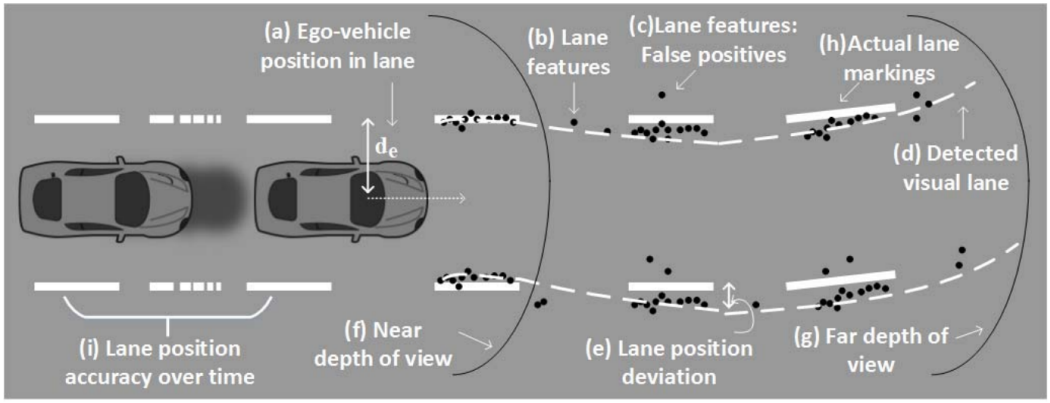
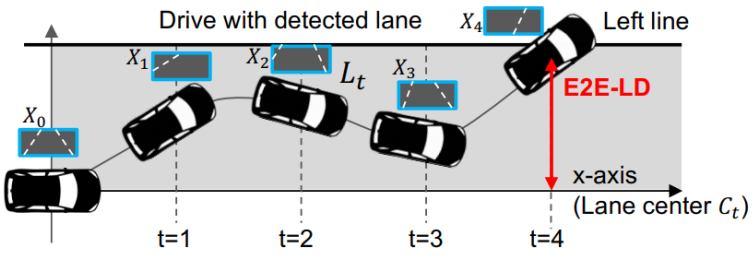


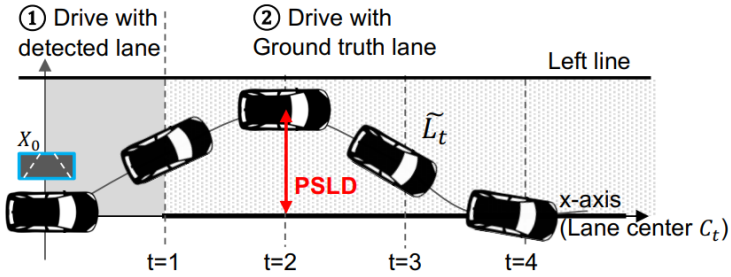
Figure 59: Illustration of the key parameters and performance metrics for evaluating lane analysis process [12]

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. 59) measures the deviation of detected lane (d) from the actual lane that is obtained by joining the actual lane markings ((h) in Fig. 59). 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 [49]



(b) Per-frame Simulated Lateral Deviation Metric [49]

Figure 60: E2E-LD and PSLD

  - **End-to-End Lateral Deviation Metric** (in Fig. 60(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. 60(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. 59). 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.

2.4.4 Verification and validation of sensor models

Sensor KPI:

- The most important KPIs for camera-based sensors are the following:
  - Usable field of view (FOV) in the horizontal (H) and vertical (V) directions, measured in degrees (°)
  - Range accuracy, measured in + cm and range precision, measured in cm

  - Range resolution
  - Maximum detection range
  - Confidence measure (per pixel), standard deviation or confidence interval
  - Compute requirements
  - Area coverage measured in points per second or m^2/s

- The most important KPIs for LiDAR sensors tend to be the following:
  - Usable field of view (FOV) in the horizontal (H) and vertical (V) directions, measured in degrees (°)
  - Angular resolution in H and V directions, measured in degrees (°) at the operating frame rate or refresh rate of the sensor (Hz)
  - Range accuracy, measured in + cm and range precision, measured in cm
  - Maximum detection range on a dark target (e.g., 10 % reflective target) usually measured as a Probability of Detection (PD) in %
  - Noise level in the lidar point-cloud, usually measured as a Probability of False Alarm (PFA) in %
  - Power consumption of the lidar, measured in W
  - Area coverage measured in m

- The most important KPIs for RADAR sensors tend to be the following:

2.4.5 Method of evaluation of the fidelity of synthetic data: correlation between physical test and simulation

In PRISSMA, we have developed a new innovative feature-based analysis framework for quantitatively assessing the fidelity of RGB computer-generated images from different synthetic datasets ([13] [50]). Our approach focuses on two key aspects of images: texture and high-frequency details. To address texture, the grey-level co-occurrence matrix (GLCM) method is applied. This method characterises image texture by analysing pixel co-occurrences within image regions. This approach also allows to extract Haralick metrics, providing statistical measures of texture features. For high-frequency details, discrete wavelet transforms are used. Wavelet transforms allow to extract information across different scales and frequencies, enhancing our capability to analyse and evaluate image fidelity. By combining GLCM and wavelet transforms, we aim to emphasise frequency features and capture intricate details in the images.

Our first approach, presented in figure 61, involves using the extracted features from GLCM and wavelet transforms as inputs to a Convolutional Neural Network (CNN) to quantify the degree of fidelity. This allows us to leverage the power of deep learning to assess the similarity between synthetic and real images. The second approach is more direct, where we analyse the Haralick metrics themselves to assess the degree of fidelity by comparing the metric values between synthetic and real images.

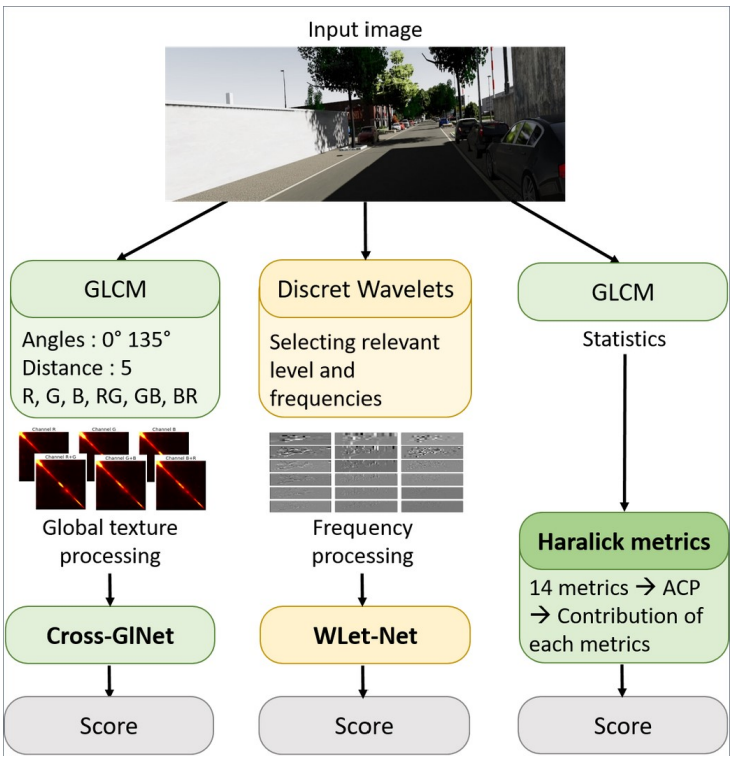


Figure 61: Diagram of the proposed method for scores generation about synthetic image fidelity (Source: UGE).

2.4.5.1 Experiments

The purpose of the experiments carried out in PRISSMA is to provide a fidelity score of synthetic images (from both learning and statistic based methods). The scores computation is also conducted on real datasets to provide a more concrete indicator of the fidelity level in synthetic images.

2.4.5.2 Learning-based method

Two sub-networks, namely Cross-GlNet and WLet-Net, are trained separately in a supervised manner using virtual and real images from a custom dataset. This dataset contains virtual and real images provided by the datasets mentioned in this paper. There are 20572 images in the training set, 6755 in the validation set and 1000 in the test sets. Each image in the dataset is assigned a label, where the label 0 represents virtual images and the label 1 represents real images. The outcomes produced by these networks represent the probability that the images are faithful to reality. Consequently, results approaching 0 indicate a higher likelihood of the images being synthetic, while probabilities closer to 1 suggest that the images are more likely to be close to realistic.

Inspired by the CoNet and Cross-CoNet [51], [52], Cross-GlNet model use the GLCM of cross-band channels (R+G, G+B, B+R) RGB images as input and takes the computation in two directions (horizontal and diagonal) and a distance of 5 pixels in the images. These GLCMs are then stacked together. The two models share a common architecture with a convolutional layer (CL), a batch normalisation (BN), and a ReLu function following by a max-pooling.

- CB 1: A CL with 32 filters of size 3×3, a BN and ReLu followed by a max-pooling layer

- CB 2: A CL with 64 filters of size 3×3, a BN and ReLu followed by a max-pooling layer
- CB 3: A CL with 128 filters of size 3×3, a BN and ReLu followed by a max-pooling layer
- A dense layer with 256 nodes followed by a ReLu layer
- A dense layer with 1 node followed by a Sigmoid layer

The key distinction between WLet-Net and Cross-GlNet lies in their handling of multi-scale inputs. Levels 5 and 6, with lower resolutions and finer frequencies, are incorporated at a later stage in the network, fused into CNN layers with 32 and 64 filters of size 1x1, followed by a concatenation layer. This coarse-to-fine architecture, inspired by [53], effectively restores high-frequency information as the network progresses. For model implementation, we use the Keras/TensorFlow framework, employing the Stochastic Gradient Descent (SGD) optimiser with a learning rate of 0.0001 and binary cross-entropy as the loss function. The batch size is set to 32, and training starts with 40 epochs, incorporating early stopping to prevent over-fitting. Figure 62 provides a visual representation of the proposed network architectures.

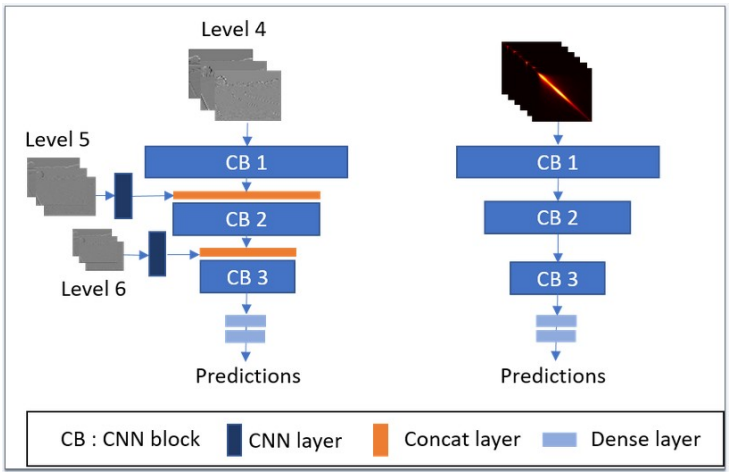


Figure 62: AI-based networks for the computation of fidelity scores ([13]) (Source: UGE)

Table 7: Accuracy (%) of Cross-GlNet and WLet-Net on several test sets (1000 images).

Test sets	Cross-GlNet	WLet-Net
GTA (virtual)	91.90	97.16
vKitti (virtual)	100	47.97
Synthia (virtual)	99.90	98.94
Cityscapes (real)	99.59	99.86
Kitti (real)	100	98.49

Table 7 shows the accuracy performance of Cross-GlNet and WLet-Net. The accuracy values (in percentage for more clarity) were computed using the TensorFlow model evaluation method. This approach, which involves evaluating the performance of the models, gives us confidence with the predicted results presented in Table 8. These predicted results, estimated with the same framework as accuracy values, correspond to the fidelity scores. Moreover, in both tables, both

virtual and real datasets are evaluated. The scores provided by the real datasets are only used to compare fidelity scores and provide an indicator of the level of fidelity of the synthetic datasets.

In Table 7, Cross-GlNet shows satisfactory performances, particularly on the GTA V and the Cityscapes datasets. The WLet-Net model achieved satisfactory accuracy across the dataset, except for vKitti. The overall results suggest that using frequency decomposition in a multi-scale manner allows to capture relevant information about texture, edges, and other details that can be useful in differentiating virtual images from real images.

Table 8: Fidelity scores (predictions in %) of Cross-GlNet and WLet-Net on several test sets.

Test sets	Cross-GlNet	WLet-Net
GTA (virtual)	12.03	4.30
vKitti (virtual)	0.05	51.21
Synthia (virtual)	10.42	3.60
Cityscapes (real)	96.67	98.05
Kitti (real)	99.81	96.85

Table 8 presents the predicted fidelity scores computed by different models on several test sets. The high accuracy achieved by the models presented in Table 7 provides with confidence in the accuracy of the score results. As expected, Kitti has a high score of fidelity, with 98.34%, while vKitti, as a synthetic dataset, has a low score (25.63%). This table allows us to be aware of the significant difference in fidelity scores between the synthetic datasets and the real datasets. As reminder, evaluating the level of fidelity of synthetic images enables to determine whether a synthetic image is realistic enough to be used in learning-based methods or in a process of evaluation and validation of a perception system.

The initial approach yields promising results, enabling us to quantitatively assess the level of fidelity of synthetic datasets. Nonetheless, its implementation can be cumbersome, particularly when it comes to dataset setup and subsequent learning processes. As a more time-saving alternative, we investigate a different approach in the following subsection.

2.4.5.3 Statistic-based method

In a second time, we have compared four synthetic and two real datasets using Haralick metrics, which are computed on 100,000 image patches with a resolution of 64x64. The hue channel of the HSV color space is used for analysis as it offers better discrimination between image types. Min/max normalisation is applied to ensure metric values range from 0 to 1. After computing Haralick metrics, Principal Component Analysis (PCA) is applied to reduce dimensionality and interpret the data effectively. PCA helps in understanding the contribution and relationships of each metric to the overall information. Focusing on the first two principal components, which contain over 50% of the data's information, we analyse the datasets. This step ensures a comprehensive comparison, allowing us to interpret the individual contribution of each metric and their correlation with the characteristics of real or synthetic datasets. Then, we compute the contribution of each metric to each PC, PC_1 and PC_2, with:

$$K_{i,k} = \frac{c_{i,k}^2}{\lambda_k} \tag{2}$$

where k is the PC index, λ_k is the eigenvalue associated to the PC_k and $c_{i,k} =$ is the component of the vector $\sqrt{\lambda_k}\mathbf{u_k}$ and $\mathbf{u_k}$ is the k^{th} eigen vector.

Table 9: Contribution of each metric to PC_1. The best contributions are in bold.

Metrics	Kitti	City	Once	NuScenes	vKitti	GTAV	Kitti-C	Synthia
ASM	0.11	0.032	0.089	0.10	0.098	0.083	0.094	0.078
Contrast	0.069	0.036	0.063	0.069	0.072	0.070	0.036	0.075
Corr	0.064	0.061	0.012	0.031	0.014	0.0002	0.068	0.027
Var	0.061	0.058	0.065	0.059	0.075	0.073	0.051	0.067
IDM	0.11	0.082	0.12	0.12	0.12	0.12	0.073	0.082
SA	0.052	0.011	0.021	9e-6	0.002	0.004	5e-5	0.010
SVar	0.048	0.058	0.064	0.055	0.071	0.068	0.052	0.063
SE	**0.12**	**0.14**	**0.13**	**0.13**	**0.12**	**0.15**	**0.13**	**0.14**
E	**0.12**	**0.13**	**0.12**	**0.13**	**0.12**	**0.14**	**0.12**	**0.13**
DVar	0.032	0.088	0.12	0.075	0.065	0.075	0.074	0.10
DE	**0.12**	**0.12**	**0.13**	**0.13**	**0.12**	**0.14**	**0.11**	**0.12**
IMC1	0.0004	0.065	0.028	0.060	0.068	0.0002	0.075	0.025
IMC2	0.08	0.11	0.032	0.039	0.041	0.056	0.10	0.068

Table 10: Contribution of each metric to PC_2. The best contributions among the synthetic datasets are in bold. The best contributions among the real datasets are underlined.

Metrics	Kitti	City	Once	NuScenes	vKitti	GTAV	Kitti-C	Synthia
ASM	0.015	0.15	0.095	0.032	0.027	0.040	0.015	0.091
Contrast	0.064	0.10	0.19	0.16	0.009	0.029	0.071	0.020
Corr	0.055	0.016	0.030	0.006	**0.18**	**0.15**	**0.14**	**0.10**
Var	0.18	0.19	0.21	0.27	0.008	0.11	0.036	0.079
IDM	0.033	0.12	0.012	0.024	0.006	0.074	0.12	0.13
SA	0.018	0.005	0.079	0.084	**0.34**	0.009	**0.25**	**0.16**
SVar	0.22	0.19	0.21	0.28	0.013	0.12	0.028	0.084
SE	0.002	0.011	0.025	0.019	0.0005	0.001	2e-5	0.009
E	0.014	0.031	0.026	0.026	4e-5	0.019	0.003	0.036
DVar	0.077	0.088	0.12	0.075	0.056	0.008	0.081	0.005
DE	0.12	0.12	0.13	0.13	0.003	0.020	0.049	0.053
IMC1	0.28	0.10	0.005	0.06	**0.17**	**0.30**	**0.13**	**0.18**
IMC2	0.037	0.002	0.084	0.014	0.18	0.11	0.076	0.042

The contributions to the PC_1 and PC_2 (Equation 2) seems to be interesting criteria to discriminate real image data or CGI data.

Tables 9 and 10 present the computed contribution of each metric to PC_1 and PC_2 for different datasets. In Table 9, sum entropy (SE), entropy (E) and difference entropy (DE) metrics contribute equally to PC_1 for both synthetic and real datasets. These results do not allow to draw conclusions about the representativeness of the metrics based on the datasets. However, in Table 10, we can observe that some metrics are more significant for the synthetic datasets (highlighted in bold, including correlation, sum average, and IMC1), while others (underlined, such as var and svar) are more indicative for the real datasets. Based the metrics highlighted in bold, some metrics are selected to create a score. The contributions to PC_2 of correlation, sum average and IMC1 metrics, which are representative of synthetic datasets, will be included in the fidelity score as penalties, using the associated correlation contributions. This will place greater emphasis on the metrics that are representative of real datasets.

2.4.5.4 Results with the different scores

Several indices need to be considered when quantifying the fidelity of images due to the complexity of real scenes. Therefore, we propose a set of fidelity scores with respect to the

different features (Textural features and Wavelet based feature) including models described in section 2.4.5.2 and the contribution of selected Haralick metrics. This set provides a more comprehensive assessment of fidelity. The sub-score sH based on the Haralick metrick contribution (2) is defined by the following equation :

$$sH = \frac{1}{5}(\lambda_2 K_{Var,2} + \lambda_2 K_{Svar,2} + (1 - \lambda_2 K_{Corr,2}) + (1 - \lambda_2 K_{SA,2}) + (1 - \lambda_2 K_{IMC1,2})) \quad (3)$$

where $K_{Var,2}$, $K_{Svar,2}$, $K_{Corr,2}$, $K_{SA,2}$ and $K_{IMC1,2}$ are respectively the contributions to PC_2 of Var, Svar, Correlation, SA and IMC1 metrics. λ_2 is the eigenvalue associated to the PC_2. Equation 3 is using arithmetic average of the correlation contributions $\lambda_2 K_{i,2}$ of selected metrics. It take into account all the best contribution to PC_2 for both synthetic and real datasets.

Table 11: Fidelity scores (%) computed from the synthetic and real datasets.

Datasets	Synthia	GTA	vKitti	Kitti	Cityscapes
Cross-GlNet	10.42	12.03	0.05	99.81	96.67
WLet-Net	3.60	4.30	51.21	96.85	98.05
sH	41.80	48.14	34.10	62.42	72.76

Table 11 presents the fidelity scores of synthetic and real datasets using the proposed methods, including model-based and Haralick metrics. While the scores from real datasets provide valuable context, the primary focus is on assessing the fidelity of synthetic data. The results reveal a significant gap between synthetic and real data, with synthetic datasets exhibiting relatively low fidelity in terms of texture and frequency aspects. To further validate the effectiveness of our method, we propose comparing the GTA dataset with enhanced versions where photo-realism has been improved using a GAN-based image translation method [54]. The goal is to enhance the GTA dataset's photo-realism by adopting styles from real datasets like Cityscapes and Mapillary Vistas. Figure 63 presents some images from these datasets. Table 12 presents the fidelity scores of these enhanced datasets, showing a significant improvement in fidelity across models and sH. Our proposed method provides a quantitative measure of the level of fidelity in synthetic datasets, offering valuable insights into their realism and similarity to real-world data. The improved scores of the enhanced GTA datasets validate our hypothesis, showcasing the effectiveness of GAN-based image translation methods in enhancing photo-realism and, consequently, fidelity.

2.4.5.5 Results with the merging of scores with multi-criteria combination rule ([50]

In a second stage, we have proposed to build a multi-criteria combination rule in order to merge the different scores generated by the first processing stage. Hence, it is pertinent to examine the impact of the results using the multi-criteria approach. The Cityscapes dataset serves here as a reference. The four scores calculated for each dataset, corresponding to the four criteria, as shown in Fig. 64.

Fig. 65 illustrates the graphs obtained from the multi-criteria combination and the generation of BBA with BBF (see the explanation in [50]. They will help to establish a level of fidelity (H) or non fidelity ($notH$), the level of uncertainty ($Omega$), and the detection of conflict ($Empty$) between scores. Table 13 details the parameters used to produce these graphs. Each criterion has been assigned reliabilities α and τ values. The τ values are set here to be pessimistic with

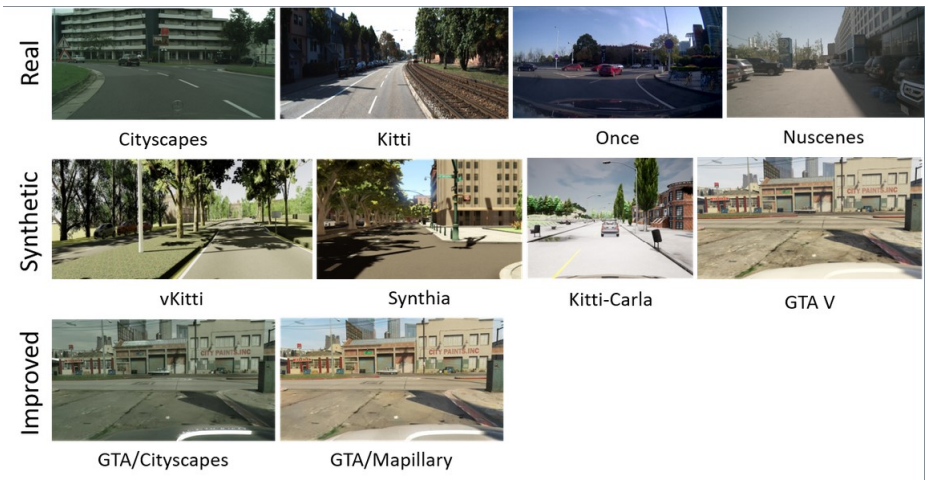


Figure 63: Images from GTA V, GTA/Cityscapes and GTA/Mapillary datasets.

Table 12: Final scores obtained with the enhanced synthetic datasets(%) GTA V to Cityscapes (GTAV/City) and GTA V to Mapillary (GTAV/Map) compared to the original GTAV dataset.

Datasets	GTAV	GTAV/Map	GTAV/City
Cross-GlNet	12.03	21.04	42.32
WLet-Net	4.30	29.92	59.63
sH	48.14	81.89	82.54

τ = 0.6 (model tends to allocate mass on $\overline{H}$). For the time being, τ is fixed, but it will be optimised as part of a future work. The reliabilities associated with the criteria, obtained from the learning-based models, correspond to models' accuracy. These accuracies are computed using functions from the the Keras/TensorFlow framework. The reliability assigned to the S_H criterion is set to 0.5 as it is impossible to obtained a similar accuracy to the learning-based methods.

Table 13: Reliability α and τ associated to each criterion Sc with a certain value for three datasets.

Criteria	**GTA V**			**GTA/Map**			**Cityscapes**		
	value	α	τ	value	α	τ	value	α	τ
Sc_1	0.12	0.92	0.6	0.21	0.64	0.6	0.97	0.99	0.6
Sc_2	0.04	0.97	0.6	0.30	0.73	0.6	0.98	0.99	0.6
Sc_3	0.11	0.90	0.6	0.37	0.70	0.6	0.99	0.99	0.6
Sc_4	0.48	0.50	0.6	0.82	0.50	0.6	0.72	0.50	0.6

The graph on the left in Fig. 65, for the GTA dataset, indicates that it has a strong tendency to $\overline{H}$ with 97%, as well as 2% of conflicts. These results suggest that the GTA V dataset is not faithful to reality. The graph on the right, resulting from the generation of BBA, present the outcomes concerning the triplet of hypotheses $H, \overline{H}, \Omega$ for each criterion. This allows us to obtain detailed results for each criterion with $m(H)$ = 0%, 0%, 0%, 2%, $m(\overline{H})$ = 73%,95%, 74%, 0%, $m(\Omega)$ = 27%, 5%, 26%, 98%. We can see that criterion 4 shows a tendency of 2% to H and an uncertainty of 98%.

Concerning the GTA/Map dataset, it indicates a weaker tendency to $\overline{H}$ than the GTA V dataset with 23%. It also includes 25% of H, 38% of uncertainty and 15% of conflicts. Concerning the triplet of hypotheses : $m(H)$ = 0%, 0%, 0%, 40%, $m(\overline{H})$ = 29%, 11%, 1%, 0%,

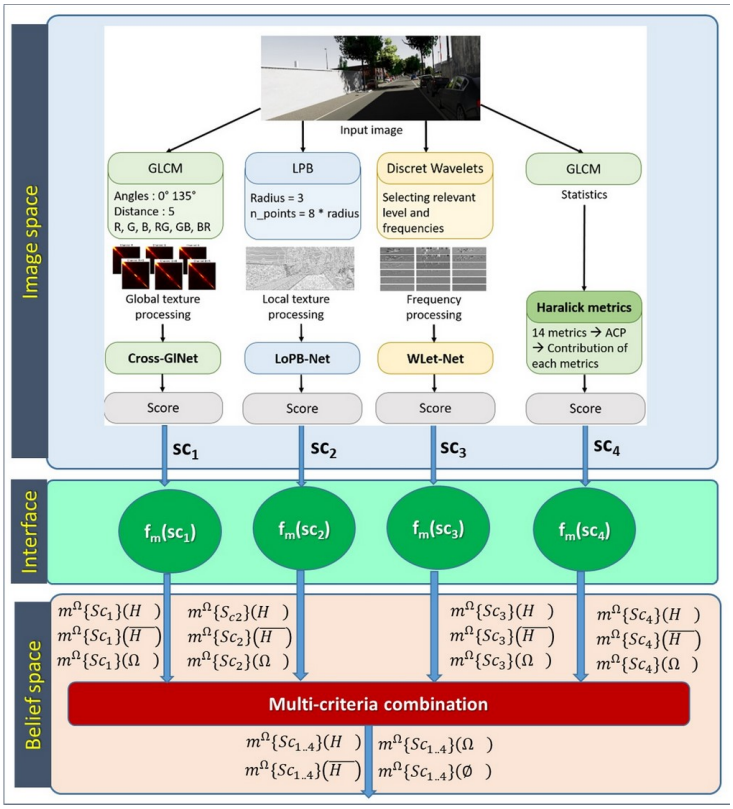


Figure 64: Overview of the multi-criteria combination method for the assessment of a global fidelity score involving uncertainty and potential conflict detection. (Source: UGE)

$m(\Omega)$ = 70%, 89%, 99%, 60%. While the level of uncertainty is very high for all criteria, the first shows a significant tendency to $\overline{H}$ and the fourth an increasing tendency to H with 40% and an uncertainty of 60 %. This shift of $\overline{H}$, which correspond to a level shift of non fidelity from 97% to 24%, suggests an improved fidelity of the enhanced GTA dataset compared to the GTA V dataset. The third row of the figure shows the graphs for the real Cityscapes dataset. These results serve as an ideal basis for datasets requiring to be faithful to real-world scenarios. The presented study is done on the full datasets but could be relevant on specific synthetic images in order to have the capability to explain specific scores.

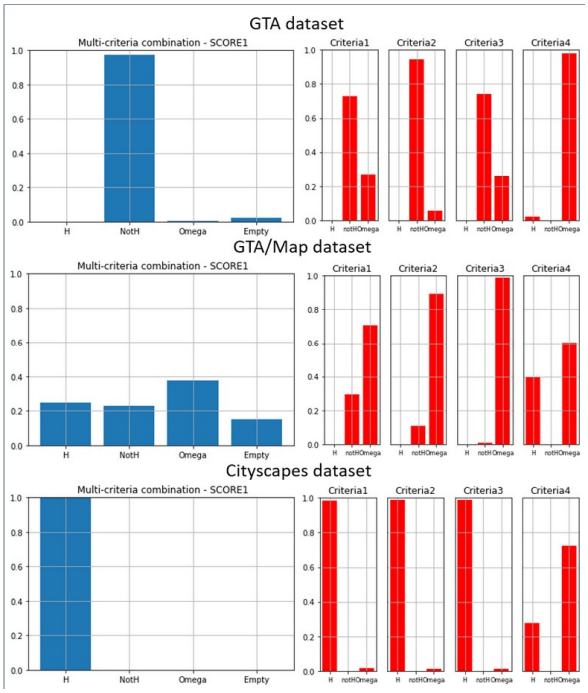


Figure 65: Graphs resulting from the multi-criteria combination (left) and the generation of BBA with BBF (right).(Source: UGE)

2.5 Final Implementation

2.5.1 Implementation basis

This section explores the essential aspects of implementing the synthetic dataset generation framework for visual perception in adverse scenarios and autonomous driving, focusing on leveraging specific platforms and tools. As shown in the Figure 46, the implemented SiVIC-ADVeRSce framework deploys an instance of integration between two interconnected software, RTMaps^{TM} and Pro-SiVIC^{TM}. Data Distribution Service (DDS) as the communication mechanism is integrated within the framework, which offers an effective and interoperable Application Programming Interface (API) for seamless data sharing and communication among the various components.

2.5.2 Environment and System Modelling

The simulated environment models aim to mimic the complexity and diversity of real-world driving situations, allowing the generation layer to be controlled and repeatable. In the framework, the capabilities of Pro-SiVIC^{TM} are used to craft realistic and intricate virtual environments, encompassing road networks, traffic conditions, weather phenomena, pedestrians, vehicles, and various objects. The environment model tested within the framework comprises a digital twin derived from the digital twin constructed at the Satory test track, as illustrated by Gruyer et al [55], and also two simulated scenes (Highway and City center) as depicted in Figure 40.

Autonomous driving systems perform various critical functions within their operating environment, encompassing perception, localization, decision-making, path planning, control, and more. Constructing a realistic and complex dataset for perception functionalities necessitates the utilization of a comprehensive simulated sensor suite, enabling the vehicle to perceive its

surroundings through cameras, LiDAR, radar, and other sensors. In SiVIC-ADVeRSce, the visual perception system is equipped with multi-sensors derived from Pro-SiVIC™, previously validated within the digital twin of the Satory test track [56]. Moreover, the virtual vehicle model is integrated into Pro-SIVIC^{TM}, featuring dynamic modeling containing the car body, shock absorber, wheels and tires, powertrain, and steering wheel. The interaction of this complex model with the ground and other objects is facilitated by a raytracing engine embedded within the simulation engine. For controlling the vehicle and pedestrian models, various modes are available, including "human control," "trajectory following," "control/command," and "control from RTMaps^{TM} (the control system deployed)."

2.5.3 Scenarios Management

The scenario management operates on an event-based paradigm. Initially, the environment elements are rendered within Pro-SiVIC^{TM} according to the specified scenario configuration, instantiation as a Script file, which also initialises various events and their corresponding variables. Subsequently, a scenarios manager is integrated as a module within RTMaps^{TM}. Within each frame, information sources from Pro-SiVIC^{TM} are forwarded to this module as either observation or sensor data. Leveraging this observation, the event variables are computed and updated accordingly. Upon the occurrence of an event, such as a vehicle deviating from the road or a collision appearance, the event command is communicated to Pro-SiVIC^{TM}via the aforementioned DDS mechanism. Within Pro-SiVIC^{TM}, diverse actions are performed based on the command and parameters shared by the manager, including the modification in dynamic and static aspects. This mechanism allows for generating a loop of scenarios with interval values for variables and parameters under test, i.e., for vehicle, environment, weather conditions, etc. In order to have a clearer overview of what is needed in order to define, execute, and analyse scenarios, the reader can use the generic framework presented in figure 36 of the section 2.3.1. This framework in share into 3 main parts or layers: The scenario definition, the scenario execution, and the results analysing.

2.5.4 Digital Models developed in the framework of PRISSMA or associated projects

2.5.4.1 Satory test tracks

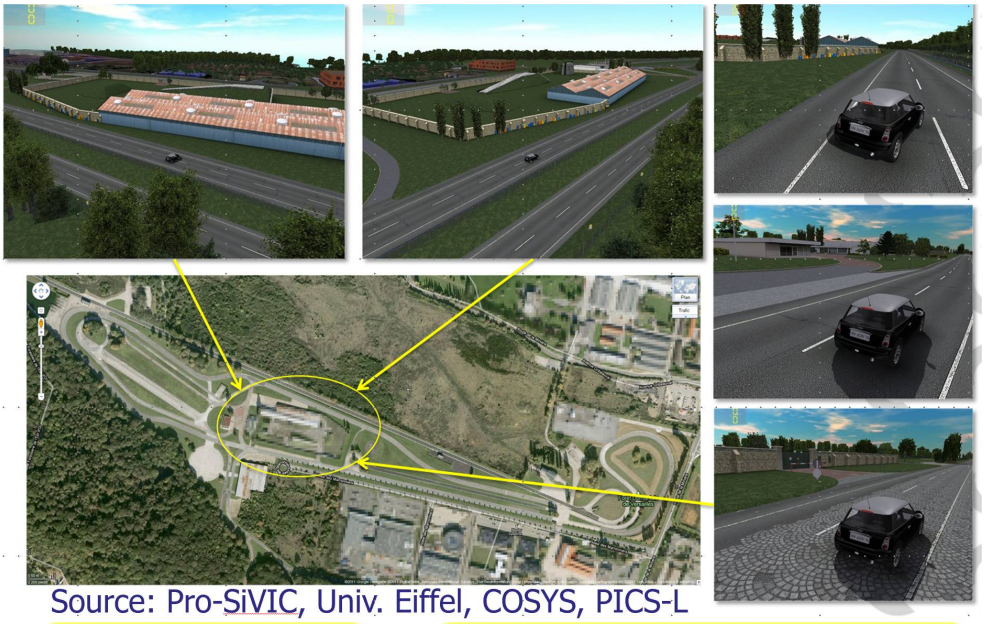


Figure 66: Digital Model of the Satory's test track (Source UGE).

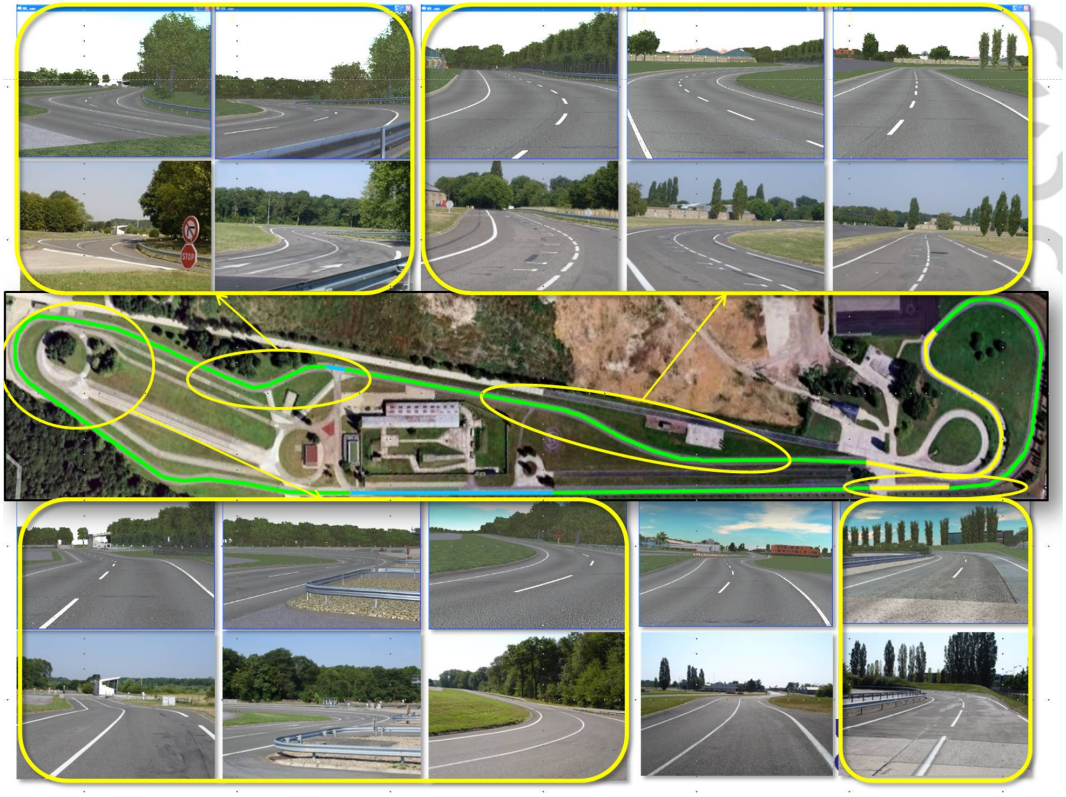


Figure 67: Digital Model of the Satory's test track in comparison with real test track (Source UGE).

2.5.4.2 Transpolis test tracks

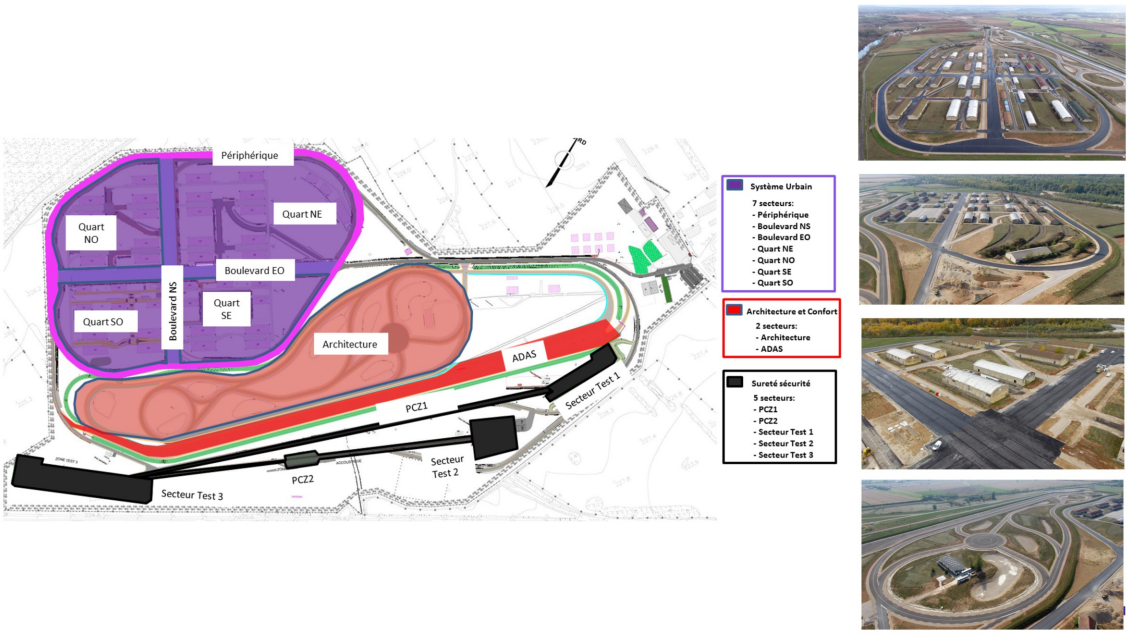


Figure 68: View of the Transpolis test tracks.

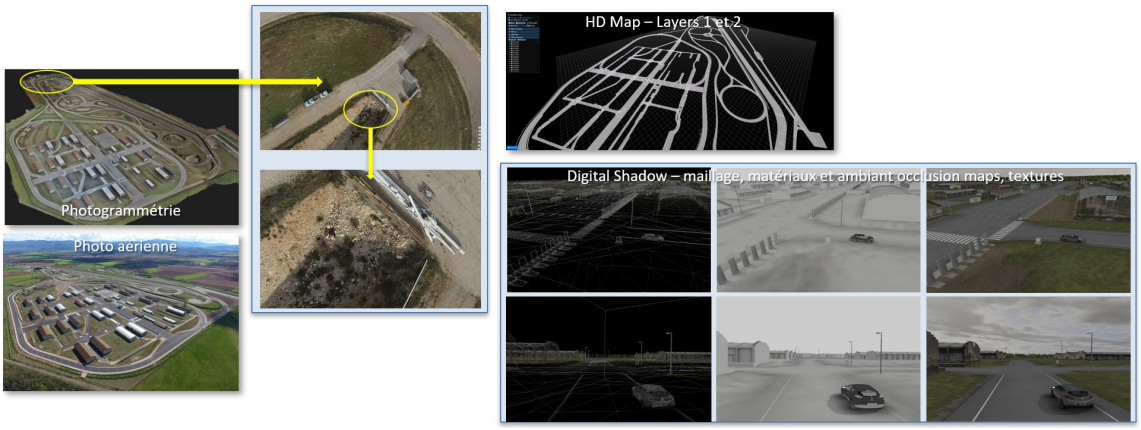


Figure 70: Digital Model and HD Maps (Transpolis), a long and resource consuming procedure (source: UGE).

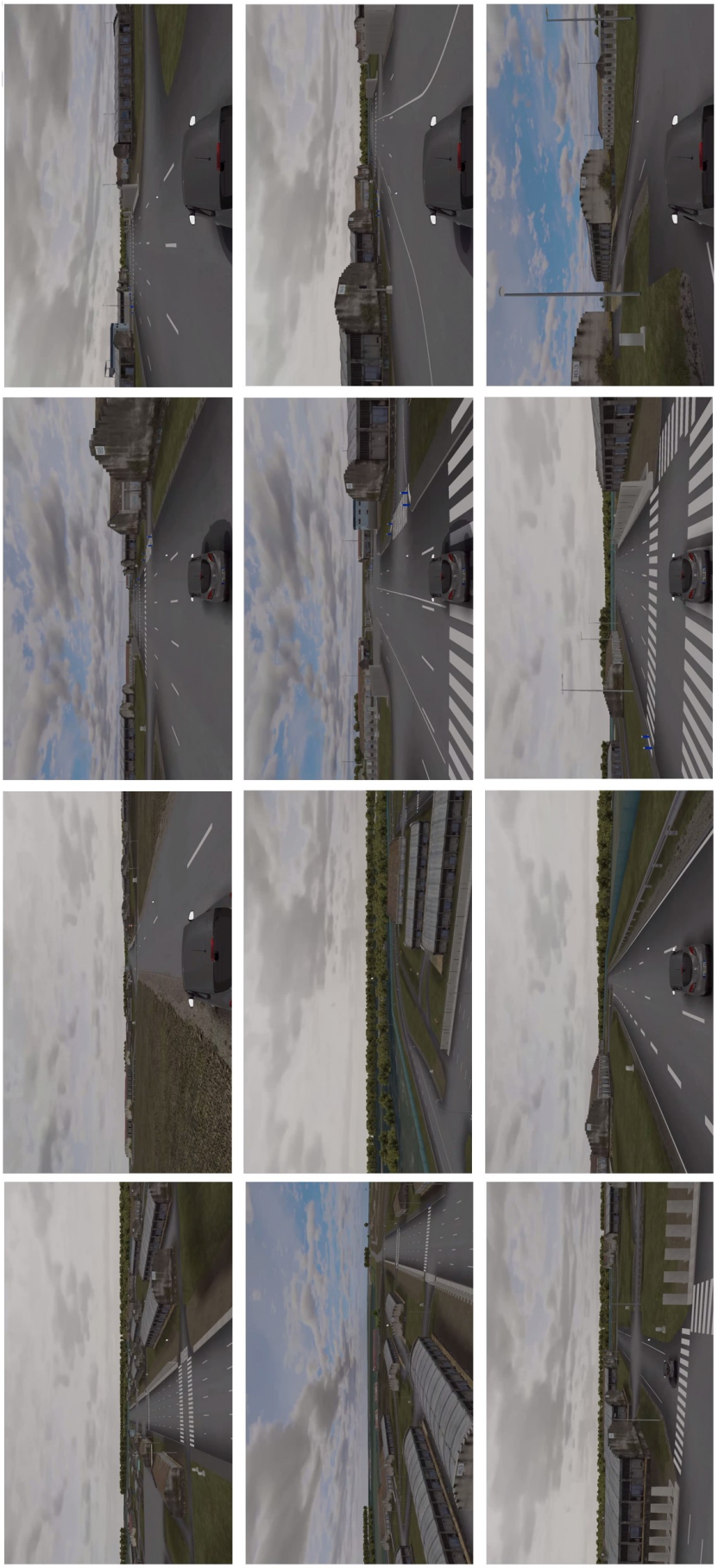


Figure 69: Digital Model developed for Transpolis test track (source: UGE).

2.5.4.3 Paris2Connect open area

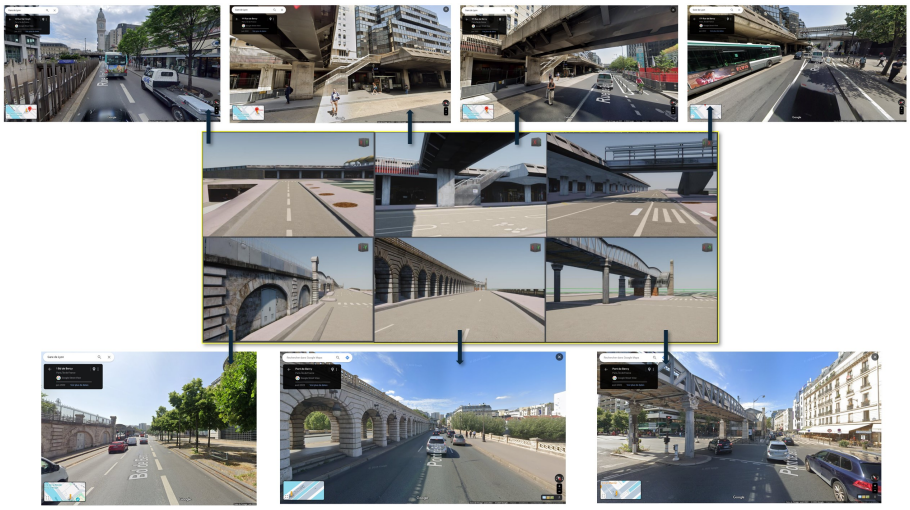


Figure 71: Digital Model in progress for the Paris2Connect Use case in PRISSMA (source: UGE and VALEO).

Figure 72: Digital Model and Ambient Occlusion Map, a mandatory rendering mechanism in order to improve significantly the image fidelity (source: UGE and VALEO).

2.5.5 Adverse Features and simulation under complex scenarios

We've employed a mechanism of adapted rendering integrated into the Pro-SiVIC^{TM} engine to enhance the implementation of adverse features within the implemented framework. This mechanism, presented in deliverable 2.5 of the WP2 of PRISSMA, enables the generation of disturbances affecting both sensors and vehicle behaviour. It benefits from the definition and utilisation of various rendering plug-ins, such as HDR textures, shadows, filters, and tone mapping, tailored to specific requirements.

Through the employment of a multiple-filters mechanism, diverse adverse conditions can be defined. These include light filters (headlight with light map), weather condition filters like rain (comprising rainfall and raindrops), fog, and snow(snowfall), as well as sensor degradation effects such as noise, blur, depth of field, optical deformation, self-exposure, and auto-focus.

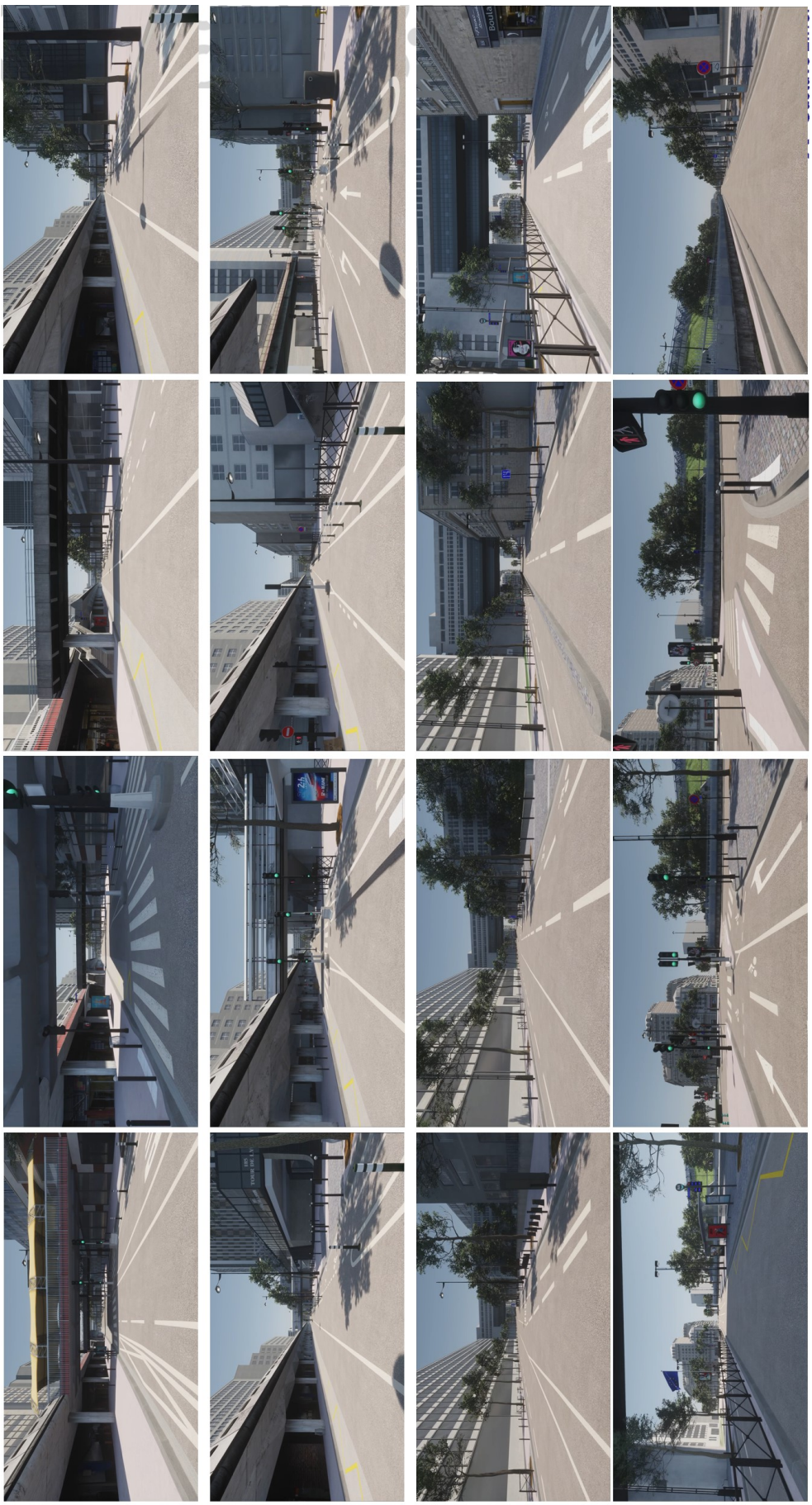


Figure 73: Screenshot of the Digital Model usable in the Digital Shadows Paris2Connect (source: UGE and VALEO).

Furthermore, the multiple reflection mechanism enables close-realistic environmental reflections on the car body, windows, wet roads, and other surfaces, as presented in deliverable 2.5 of PRISSMA dedicated to the definition of interfaces and simulation environment for evaluation and validation procedures.

By adjusting the parameters of these mechanisms, a spectrum of adverse conditions with different degrees can be obtained, allowing for generating a comprehensive dataset.

This complex and adverse scenarios generation cabability is essential in order to test the performance of an AI-powered system under various challenging conditions covering the larger possible operating space. In addition to adverse weather or low lighting as mentioned previously, this level of simulation allows to generate 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, analysed, and modified to evaluate, analyse, 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. 74.

Figure 74: Adverse scenarios from Pro-SiVIC^{TM} (Source: UGE)

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

- **Robustness**: This metric evaluates the ability of the system to perform consistently and accurately in various challenging and unforeseen situations, and 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.

- **Reliability**: This metric evaluates the system's ability to make reliable decisions in emergency situations or other adverse situations.

2.5.6 Datasets Collection and Annotation

During the simulation instance, perception data and reference data are collected via the module in RTMaps^{TM} by the data-sharing mechanism mentioned above. For example, as depicted in Figure 40(b), simulated image frames generated in Pro-SiVIC^{TM} are captured and recorded by the sensor module defined in RTMaps^{TM} to build the dataset.

Several mechanisms integrated into the Pro-SiVIC^{TM} have been employed to generate the reference data. One approach involves modifying the rendering texture of objects and the environment (such as vehicles, pedestrians, lanes, roads, buildings, etc.), which results in segmentation masks (shown in Figure 40(d)) being generated and collected as a part of reference data. In addition to visibility-based mechanisms, a specific mechanism known as the "observer" in Pro-SiVIC^{TM} enables the real-time generation of the state vector of different objects in the scene (including vehicles, pedestrians, static objects, and road configurations). Notably, the depth matrix (visualized in Figure 40(c)) of sensors can also be collected as reference data, which may contribute to the refinement of annotation and enhancement of the dataset.

As defined in the conceptual framework, the annotation labels are generated according to the configured annotation schema from the upstream layer. In the implemented SiVIC-ADVeRSce framework, by leveraging various mechanisms for reference data generation, we have provided several annotation schema possibilities, each corresponding to different functional aspects of perception. Primarily categorised into object, semantic, and temporal domains, these annotation schemas enable comprehensive annotation of multi data modalities.

Within object annotations, we aim to obtain the precise annotation of objects through bounding boxes, polygons, and pixel-level masks. Figure 41 shows the different object annotations in the implemented framework. The second type of annotation implemented in SiVIC-ADVeRSce, namely semantic annotations, targets the whole perception data and enables extracting coherent sub-segments or regions and assigning meaningful labels to each segment based on its semantic content. Furthermore, SiVIC-ADVeRSce extends its annotation schema to encompass the temporal aspect, allowing the annotation of timestamps, events, and temporal segments. This feature aligns with the virtual timestep in Pro-SiVIC^{TM}, ensuring accuracy and consistency.

The proposed Cycle-GAN based model consists of two generators and one discriminator. The process involves taking input data from the synthetic domain and passing it through the generator 1 to transform it into the real domain. Subsequently, the transformed image is passed back to domain synthetic through the generator 2 in order to keep the initial image content (cycle consistency). The generator 1 generates a prediction (enhanced output data) that is then passed to the discriminator along with real samples. The discriminator classifies them, through back-propagation, as real or fake, providing a signal that enables the generator to update its weights. Unlike Cycle-GAN, the proposed method uses a single cycle with a single discriminator in order to reduce the time-consuming process. The two generators are identical, based on Unet architecture [57]. They consist of two encoders based on pre-trained ResNet to extract features from the fog image and depth maps. The weights of the encoders' selected layers are then fused with a spatial feature transform layer that generates affine transformation parameters for spatial-wise feature modulation. The resulting outputs passed through Attention layers before being fed into the decoder. The discriminator consists of a classical Patch-Gan [58] architecture.

The experiments were carried out using the Keras/Tensorflow frameworks. The ADAM optimizer is used for both generators and discriminators with a learning rate of 0.0002. The model is formed on foggy data, with around 300000 iterations and a batch-size of 1, due to the limited memory capacity. The synthetic domain contains 950 images of the foggy virtual Kitti dataset [59] and the real domain contains 1003 images from the RTTS subset of the RESIDE dataset [60].

2.6 BuSAS DataSets generation and analysis

To assess the applicability and effectiveness of the proposed SiVIC-ADVeRSce framework, we employed BuSAS application of PRISSMA PoC, to generate a dataset containing different

adverse scenarios for evaluating the visual perception system.

2.6.1 DataSet generation

According to the use case and requirements of the project, ODD and OEDR can be specified follows:

Category	Details
Objects	Vehicles, immobile structures (e.g., bus stations)
Event	Vehicle dynamics (deceleration, acceleration, emergency stop), ego vehicle nearing bus station, expired waiting duration
Detection	Vehicle identification (classification, localisation, tracking), bus station localisation, self-perception
Response	Obeying speed limits, following vehicles, emergency braking, docking at the bus station, returning to ego lane
Scenery	Urban area, bus stations
Infrastructure	Configuration of roadway, road markings
Environment	Weather conditions (clear sky, rain, fog) and light conditions (daylight)
Vehicle Capabilities	Speed up to 20 kph, docking / stopping / restarting at bus stations, lane centering and keeping
Traffic Condition	Fluid and congested traffic conditions, presence of other road users (vehicles and pedestrians)
Key Scenes	6 scenarios of use

Based on the ODD and OEDR, our perception system incorporates a monocular camera designed in Pro-SiVIC^{TM}, complemented by a module in RTMaps^{TM} for sensor data acquisition. For the scenarios, we utilize the digital twin of the Satory test track as the primary setting, supplemented by virtual city center and highway scenarios rendering in Pro-SiVIC^{TM}.

To simulate adverse weather conditions, we equipped the camera with seven weather filters. The baseline filter represents clear skies, serving as a reference. The remaining six filters are tailored to replicate various adverse conditions: 1) Three fog filters, following to the Koschmieder law, provide a graduated representation of homogeneous fog densities. 2) Rain filters offer visual effects by combining two main factors: one simulates the visual rain effect in the skies, and the other mimics raindrops on the camera lens or windshield, both with three intensity levels to reflect different rainfall severities. Figure 75 details the parameters for these filters.

Following pre-defined weather conditions, seven distinct groups of datasets are generated from SiVIC-ADVeRSce, each consisting of approximately 10,000 images (± 300). These datasets included various simulated vehicles, pedestrians, and multiple driving scenes. Different camera perspectives were integrated, including frontal and rear views from the collect vehicle, alongside proximal and distal viewpoints. Annotations are generated in compliance with the specifications outlined in ODD and OEDR, with the objective of promoting the effective utilization of datasets for by the functions of the visual perception system.

In order to demonstrate the evaluation of visual multitasks using the generated datasets, we have chosen two CNN-based algorithms: YOLOv5 [61] and YOLOv8 [62] for detection and segmentation, in cooperation with DeepSORT [35] for tracking. The evaluation values and relevant metrics for AI evaluation are presented in Table 15. The results demonstrate a distinct correlation between weather conditions defined in our synthetic BuSAS datasets and performance metrics. For instance, adverse rain conditions notably degrade most metrics. On

	l_{sky} fog luminosity	k maximum fog density	d distance of the centroid from the camera
Dense Fog	0.6	0.08	We don't set the centroid, the density is the same everywhere in the scene.
Middle Fog	0.7	0.04	
Slightly Fog	0.8	0.02	

(a) Related parameters of fog situations

	Paticle Count Number of particles.	*Speed* Speed vector of the particles (m/s).	*Color* RGB color of the particles.	*Width* Particle width in pixels.	*Area* Box in which particles will spawn (expressed in meters in the coordinate system of the object)	*Opacity* Particle opacity per meter.	*MaxTTL* Maximal lifetime of a particle.
Heavy Rain	100000	[0 , 5, -30]	[0.9 0.9 0.9]	2	Following the scenarios definition	0.4	1.0
Middle Rain	60000	[0 , 5, -20]	[0.9 0.9 0.9]	1	Following the scenarios definition	0.3	1.0
Slightly Rain	30000	[0 , 5, -10]	[0.9 0.9 0.9]	0.5	Following the scenarios definition	0.2	1.0

(b) Related parameters of raindrop in the sky

	Texture Texture for the water film thickness				*Dispersion* Deformation factor	*Thickness* Maximum thickness of water film in pixels	*NRefract* Refractive index of the water film
	Max drop size	Min drop size	Drop per second	Fade			
Slightly Rain	3.0	1.0	100	1.0	2	0.6	2
Middle Rain	6.0	4.0	150	2.5	2	0.8	2
Heavy Rain	10.0	5.0	200	5.0	2	1.0	2

(c) Related parameters of raindrop on the windsheid

Figure 75: Parameters of different weather filters defined in Pro-SiVIC^{TM}

the contrary, while fog levels affect metrics to a lesser extent, the impacts are still captured, especially in improving precision accompanying the reduction of visibility. This indicates the effectiveness of the dataset in representing adverse weather conditions, thereby proving to a certain extent the applicability and practicality of the SiVIC-ADVeRSce framework.

2.6.2 DataSet extension

The use of the image translation method presented in the downstream layer of the Figure 76, allows to provide some image variations of the foggy synthetic images.

Figure 76 presents some post-processed foggy images taken from the BuSAS dataset. The inputs of the method are the synthetic foggy images and the depth information, illustrated in Figure 40(c). The generated images introduce some variations to the existing foggy images in order to expand the BuSAS dataset. Notably, there are noticeable variations in the hue and luminance of the fog within the scenes. Moreover, in the last month, some AI-based generative methods have been applied on synthetic images coming from Pro-SiVIC in several environment like Satory test tracks and Transpolis test tracks. The results are presented in figures 77, 81, 82, 78, 79, and 87. Some work need to be done in order to manage and to guarantee the spatial and temporal consistency of the improved images. Moreover, improvements of the methods need to be done in order to reduce significantly the processing time.

(a) City Centre scenarios: synthetic images and corresponding variations

(b) Highway scenarios: synthetic images and corresponding variations

(c) Test track scenarios: synthetic images and corresponding variations

Figure 76: Example of some post-processed foggy images (slight and dense fog) from the BuSAS dataset generated with the unpaired image translation method. The generated images have been resized.

2.6.2.1 Satory's test track synthetic images improvement

Figure 77: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Satory's test tracks with foggy and rainy conditions. The first image on the top left is the initial generated image from Pro-SIVIC. The height other images are generated from AI-based methods with different parameters allowing to fit with the initial image. (Source: UGE).

Figure 78: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Satory's test tracks with clear weather conditions. The first image on the top left is the initial generated image from Pro-SIVIC. The 5 other images are generated from AI-based methods with different parameters and environment variations allowing to fit with the initial image. (Source: UGE).

Figure 79: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Satory's test tracks. The top left image is the initial image generated from Pro-SiVIC. The other ones are generated from AI-based methods. In this AI-based generation, the rain drops are removed and the fog is kept with some effect of smoke cloud (Source: UGE).

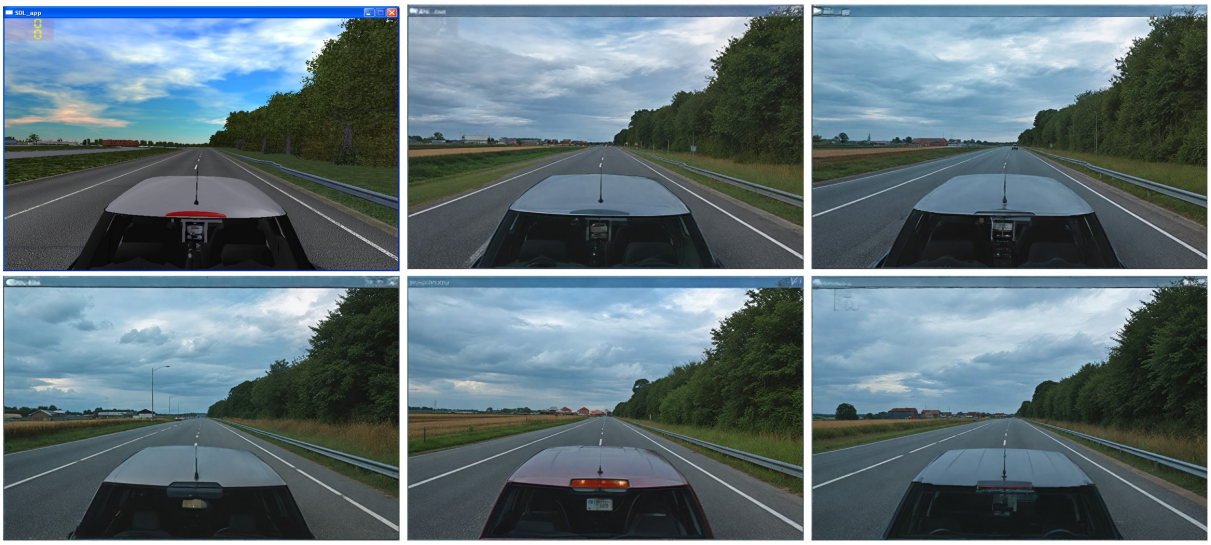


Figure 80: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Satory's test tracks. The top left image is the initial image generated from Pro-SiVIC. The other ones are generated from AI-based methods. (Source: UGE).

2.6.2.2 Transpolis' test track synthetic images improvement

Figure 81: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks. The top left and bottom left images are the initial images generated from Pro-SiVIC. The other ones are generated from 2 AI-based methods (Source: UGE).

Figure 82: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks. The top left image is the initial image generated from Pro-SIVIC. The other images are generated from AI-based methods with a variation of some parameters (Source: UGE).

Figure 83: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks. The top left image is the initial image generated from Pro-SiVIC. The other ones are generated from AI-based methods. In this AI-based generation, it is possible to see the different variations applied to the road in an intersection area. (Source: UGE).

Figure 84: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks, in the countryside part. The top left image is the initial image generated from Pro-SIVIC. The other images are generated from AI-based methods with a variation of some parameters (Source: UGE).

Figure 85: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on the Transpolis' test tracks, in the countryside part. The top left image is the initial image generated from Pro-SiVIC. The other ones are generated from AI-based methods. In this AI-based generation, it is possible to see the different variations applied to the road in an intersection area. (Source: UGE).

2.6.2.3 Motorway synthetic images improvement

Figure 86: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on a motorway with fog and wet road surface. The left image is the initial image generated from Pro-SIVIC. The other images are generated from AI-based methods with a variation of some parameters. Some corrections and variations are given on the colour and the general colour of the scene (Source: UGE).

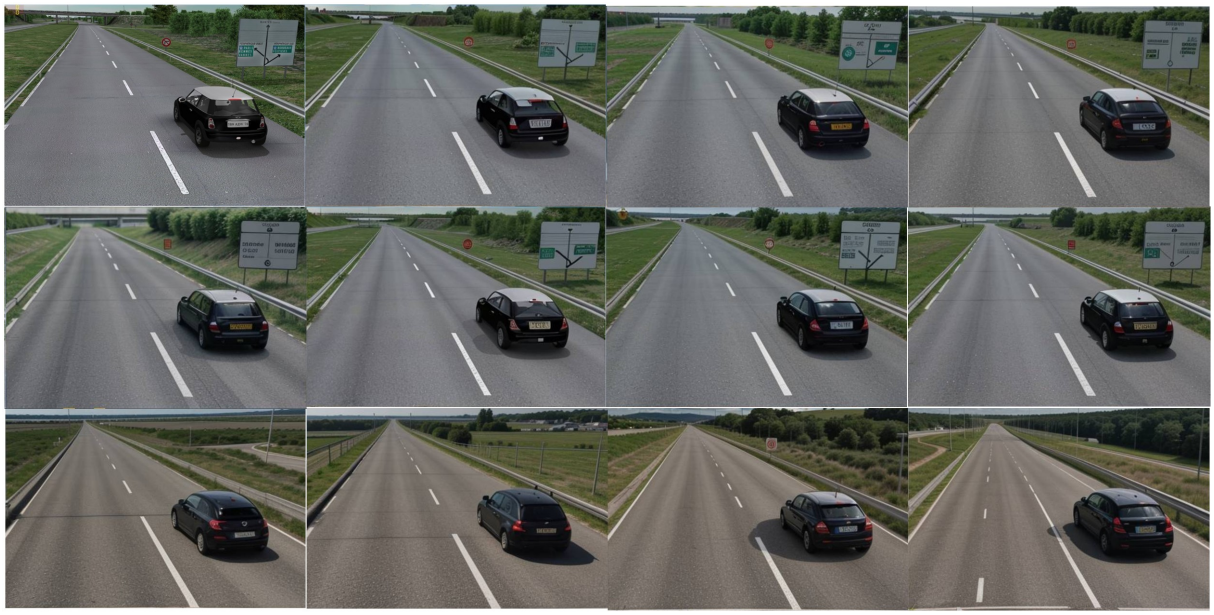


Figure 87: Improvement of the quality of the rendering for a synthetic image generated from Pro-SiVIC on a motorway with clear weather conditions. The top left image is the initial image generated from Pro-SIVIC and used as a seed. The other images are generated from AI-based methods with a variation of some parameters. On the images on the bottom, it is possible to appreciate the capacity of extrapolation of the AI-based generative method. (Source: UGE).

2.6.3 Evaluation and validation results

2.6.3.1 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 [63]. The camera-based perception system is specifically designed to provide essential functionalities such as object identification, localisation, 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 utilise 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.

2.6.3.2 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. 88, 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 utilises dedicated longitudinal and lateral profiles to return to the centre of the right lane and resume driving.

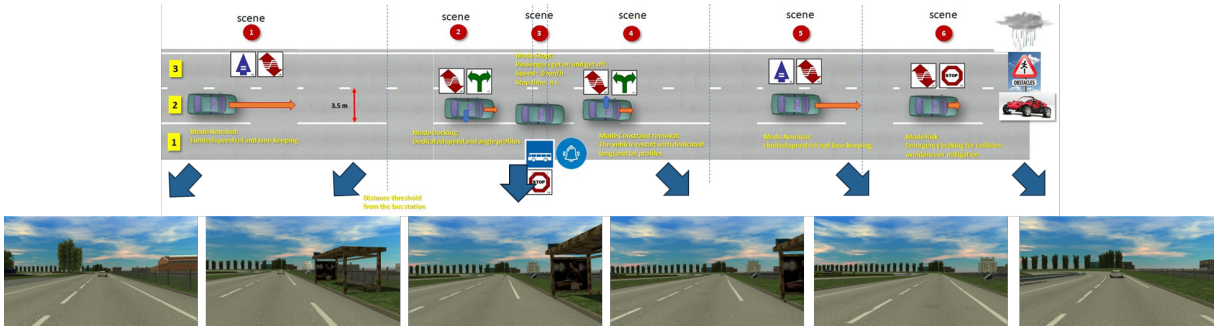


Figure 88: 6 scenes for the docking at a bus station

2.6.3.3 Algorithms and models

To achieve the system functionalities, we have selected two renowned visual perception algorithms: YOLOv5 [61] and YOLOv8 [62], 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 [64], the training phase is equipped with a 16-core Intel i9-12950HX 2.30 GHz CPU and NVIDIA-GTX A4500 GPU core graphics card.

2.6.3.4 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).

2.6.3.5 Evaluation and result

We proposed 3 levels of evaluation for this visual perception system equipped with different YOLO models.

Scenarios level

We have defined seven weather filters on the camera as shown in Fig. 89. The first scenario representing a clear sky is the reference. The remaining scenarios focus on simulating various weather conditions. We have categorized homogeneous fog 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, organised 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. 89.

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 performances 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. 14.

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. 14 also presents the evaluation results obtained using collision probability and risk of collision based on TTC. 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.

Table 14: First round: Evaluation value of system metrics for object detection

		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	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✔	✔

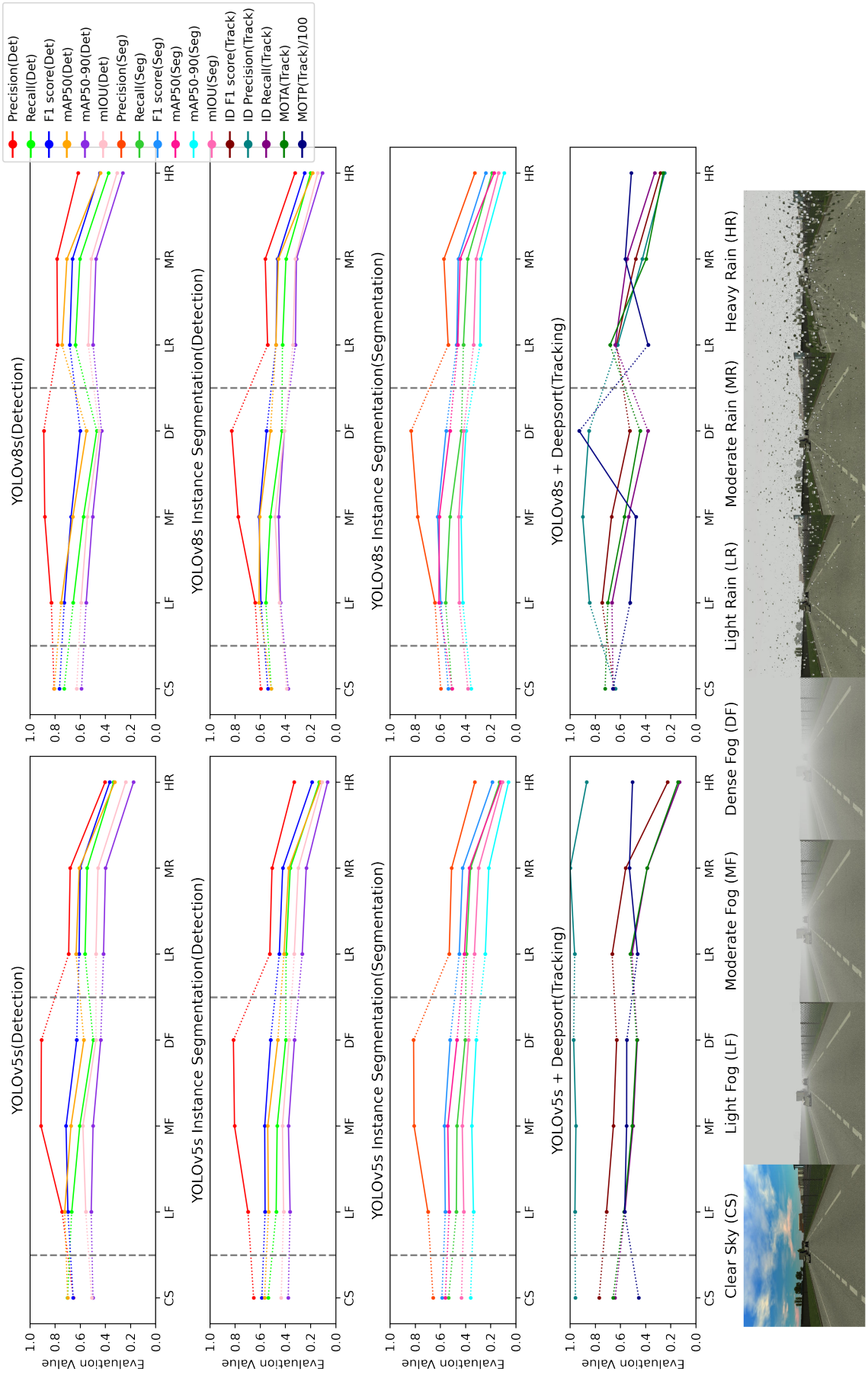


Figure 89: First round: Result of AI-based obstacles detection on different weather conditions at component level

Table 15: Second round: Evaluation value of detection, instance segmentation and tracking

		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
Precision	Detection	0.653	0.807	0.743	0.829	0.911	0.881	0.908	0.889	0.691	0.779	0.679	0.784	0.402	0.616
	Segmentation	0.657	0.596	0.698	0.642	0.808	0.779	0.812	0.832	0.529	0.537	0.51	0.571	0.325	0.325
Recall	Detection	0.7	0.727	0.667	0.655	0.603	0.574	0.496	0.468	0.561	0.638	0.544	0.602	0.334	0.374
	Segmentation	0.533	0.507	0.472	0.557	0.467	0.523	0.402	0.431	0.393	0.416	0.359	0.382	0.129	0.189
F1 Score	Detection	0.655	0.764	0.697	0.726	0.712	0.672	0.627	0.599	0.608	0.683	0.6	0.661	0.365	0.444
	Segmentation	0.587	0.536	0.56	0.596	0.566	0.621	0.521	0.554	0.449	0.469	0.422	0.457	0.185	0.239
mAP50	Detection	0.699	0.806	0.723	0.751	0.673	0.658	0.57	0.549	0.632	0.744	0.607	0.705	0.326	0.438
	Segmentation	0.56	0.506	0.529	0.61	0.54	0.608	0.467	0.523	0.408	0.461	0.369	0.442	0.119	0.171
mAP50-90	Detection	0.498	0.589	0.512	0.552	0.497	0.501	0.436	0.43	0.414	0.497	0.398	0.474	0.176	0.26
	Segmentation	0.359	0.354	0.336	0.421	0.349	0.434	0.314	0.395	0.242	0.284	0.214	0.279	0.0576	0.0926
mIOU	Detection	0.508	0.627	0.555	0.593	0.578	0.542	0.474	0.447	0.474	0.535	0.458	0.511	0.238	0.307
	Segmentation	0.431	0.38	0.413	0.449	0.425	0.453	0.375	0.413	0.329	0.335	0.295	0.315	0.105	0.137
IDF1	Tracking	0.769	0.648	0.71	0.745	0.655	0.67	0.629	0.524	0.665	0.631	0.557	0.478	0.224	0.281
ID Precision	Tracking	0.957	0.639	0.961	0.845	0.953	0.899	0.971	0.849	0.963	0.621	0.999	0.425	0.867	0.247
ID Recall	Tracking	0.642	0.658	0.564	0.666	0.499	0.534	0.465	0.38	0.508	0.641	0.386	0.546	0.129	0.327
MOTA	Tracking	0.658	0.721	0.57	0.699	0.506	0.567	0.469	0.443	0.519	0.682	0.387	0.395	0.142	0.259
MOTP	Tracking	0.453	0.659	0.565	0.523	0.55	0.476	0.549	0.927	0.463	0.379	0.528	0.561	0.503	0.512
FPS	Tracking	16	11	17	13	17	14	18	16	16	11	17	8	18	8

*v5s and v8s are small-scale model, the corresponding segmentation models (v5s-seg and v8s-seg) are also employed for instance segmentation task.
*All the model are re-trained and fine-tuned based BDD100K, experiments are based on 16-core Intel i9-12950HX 2.30 GHz CPU and NVIDIA-GTX A4500 GPU core graphics card.

2.7 Discussion and Recommendations for future developments and improvements

The main ideas guiding the future works towards a more efficient evaluation and validation framework consists to spend effort on the following actions:

- Define the scenario space with identification of influential parameters and variables containing elements and adverse/degraded conditions impacting the AI-based system behaviour and operating. This task also involved to take into account intentional and non intentional disturbances and interferences.
- At this moment, the evaluation and validation process has been applied on AI-based detection, identification, and tracking only on the Satory test track. It will be relevant to apply the full architecture of POC 1 on the other environments (Transpolis, Paris2Connect) in order to validate the full methodology presented in the figure 3 and figure 4. As presented in the section dedicated to Digital Models, we have developed both the Digital Model of Transpolis (already working in Pro-SIVIC) and the Digital Model of Paris2Connect (3D model in obj format and needs to be implemented in Pro-SiVIC. Work expected in July and August).
- Apply the interconnection of Pro-SIVIC to U-test from SPHEREA. The goal will be to take into account the implementation made between CARLA and U-Test, and to take in account this sub POC in the POC 1.
- Use the ANSYS' softwares chain. ANSYS decided to put their efforts on POC 2 with VALEO.
- Implement a set of cyber attacks on both the communication layer and the perception layer. This work could be done with the WP5 outputs.

- Extend the current work on scores and fusion of score for the evaluation of the synthetic data level of fidelity. A first method has been developed and provides interesting and encouraging results
- Extend the current work on Generative AI in order to improve the quality of the synthetic data in term of fidelity. [13] and [50]
- Development of the ImPACT 3D platform with interconnection between real and virtual environment as presented in the figure 53 and figure 52

2.7.1 Future Developments on ImPACT 3D

ImPACT 3D (see figure 90) will be the final platform implemented in Satory for the generic ViL evaluation and validation platform for CAV. This platform will consist of 2 parts. The first part is called **ImPACT 3D VA** and concerns the real prototype with the Robotized Zoé. The second part is **ImPACT 3D VR&Motion** which includes the 6DoF immersive platform, Zoé's cabin, the displays, and the computer control room (2 cabinets with 10 to 15 PCs to manage the displays, the mirrors, the dynamic model controlling the 6DoF motion , applications (ADAS, C-ITS, and automated and connected driving system), sensor simulations, simulation of means of communication, and data collection/analysis). In addition ImPACT 3D will be a ViL, HiL, MiL, and SiL platform. It is in this platform that the PROSIVIC, RTMaps, NS3, U-Test, MOSAR, ANSYS suite, and potentially Scaner softwares will be interconnected and interfaced. At the end of PRISSMA, it was expected to obtain a functional ImPACT 3D platform but it was not the case. Some development are yet needed. At the end, this platform should be a generic, interoperable, and adaptive real/virtual platform. This work is in progress in the TEF CitCom.AI and the HE Augmented_CCAM project.

2.7.1.1 ImPACT 3D VA

The first one concerns the real prototype (Renault Zoé) with hardware, software architectures, with embedded sensors, and with actuators (acceleration, braking, steering wheel). In this prototype, a set of algorithms has been implemented. This algorithms concern mainly the positioning with GPS and RTK GPS, and the virtual copilot ([65], [**?**]) allowing the L3 and L4 automation. The next step will be to implement the perception module (based on AI methods) already developed in RTMaps (using PROSIVIC data and environment). The following figures show the scenario implemented in the UGE's Renault Zoé in the framework of the H2020 TRUSTONOMY (European project). The same scenario was run both in real and virtual conditions. In this scenario, the study was to assess the trust of the driver toward the virtual copilot. In this scenario, obstacle detection and avoidance, lane change, and request to intervene were implemented. The use case implemented in PRISSMA is a light version of this use case but with new perception modules involving AI-based methods (obstacle detection/identification/-tracking, and road marking detection).

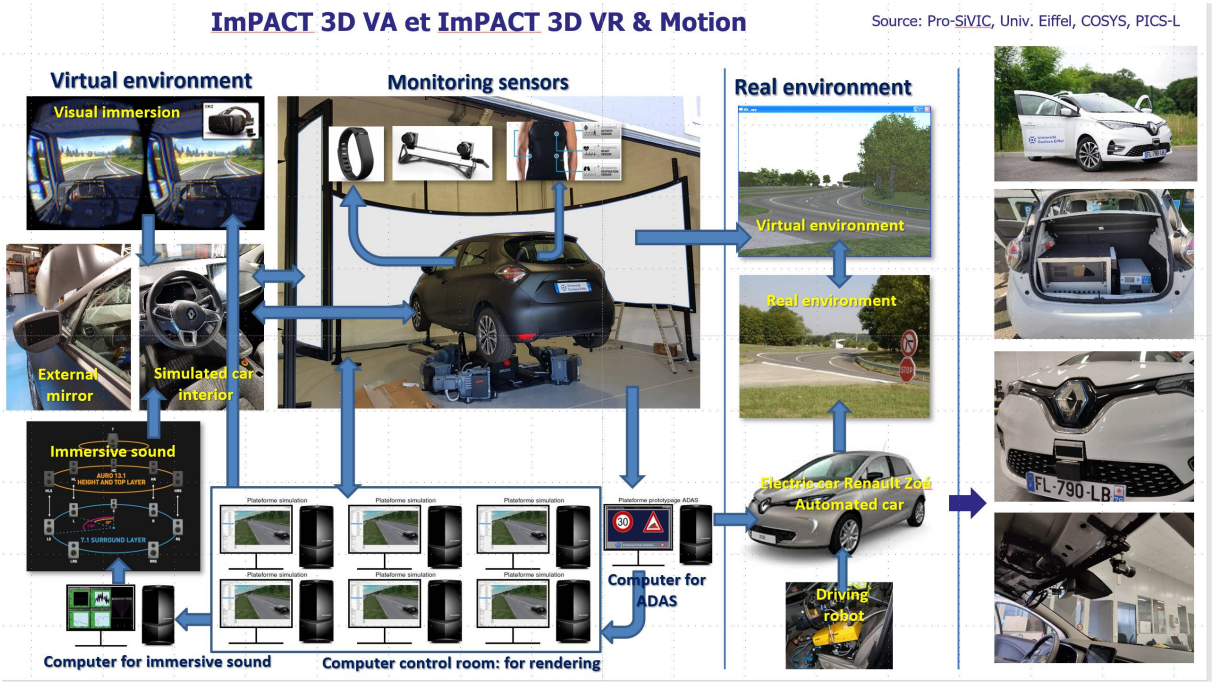


Figure 90: ImPACT 3D architecture with real and virtual test facilities: ImPACT 3D VA and VR&Motion (Source UGE).

Figure 91: TRUSTONOMY project: Virtual copilote implementation with L3 and L4 of automation in UGE's Renault Zoé (Source UGE).

Figure 92: TRUSTONOMY project: Virtual copilote implementation with L3 and L4 of automation in PROSIVIC platform (Source UGE).

2.7.1.2 ImPACT 3D VR&Motion

The second do not yet use the 6DoF motion platform (work in progress) but use the interconnection of the following set of tools:

- **NS3** in a virtual machine (Linux) and interconnected with PROSIVIC and RTMaps with the library DDsL.
- **RTMaps** as the application environment allowing to implement the full bus station automated desert service. Specific packages and modules have been developed in order to have an efficient interface between RTMaps and both PROSIVIC and NS3. This interconnection allows to have a full loop and allows to control the actuators in the PROSIVIC's car model.
- **Symuvia** as traffic generator. The interconnection between Symuvia and PROSIVIC has been done with the use and implementation of DDsL.
- **PROSIVIC** as the core of the platform with the simulation of the environment, the vehicles and dynamic actors (pedestrians and other object), the sensors, and the propagation channel for RADAR and communication simulation.

The figure 93 shows the existing implementation of the virtual copilot in RTMaps controlling 5 PROSIVIC's vehicles with different modes of driving (Conscientious, normal, sportive). This copilot also is working in the real Zoé prototype. The figure 94 presents the SiVIC MobiCoop platform ([66]) with PROSIVIC interconnected to RTMaps and NS3. The interface with NS3 is made with the using of a Virtual Machine and DDsL (interface between Windows and Linux OS). The figure shows the antennas and the link between the different transmitters/receivers.

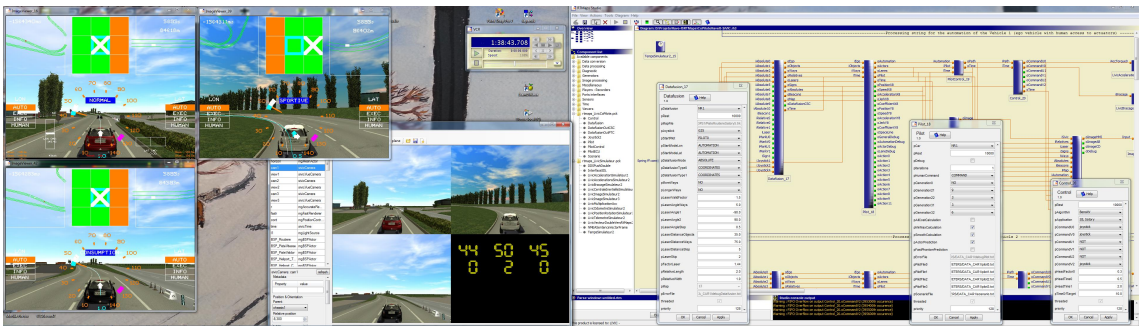


Figure 93: PROSIVIC and RTMaps: An efficient platform for virtual copilote implementation with L3 and L4 of automation (Source UGE).

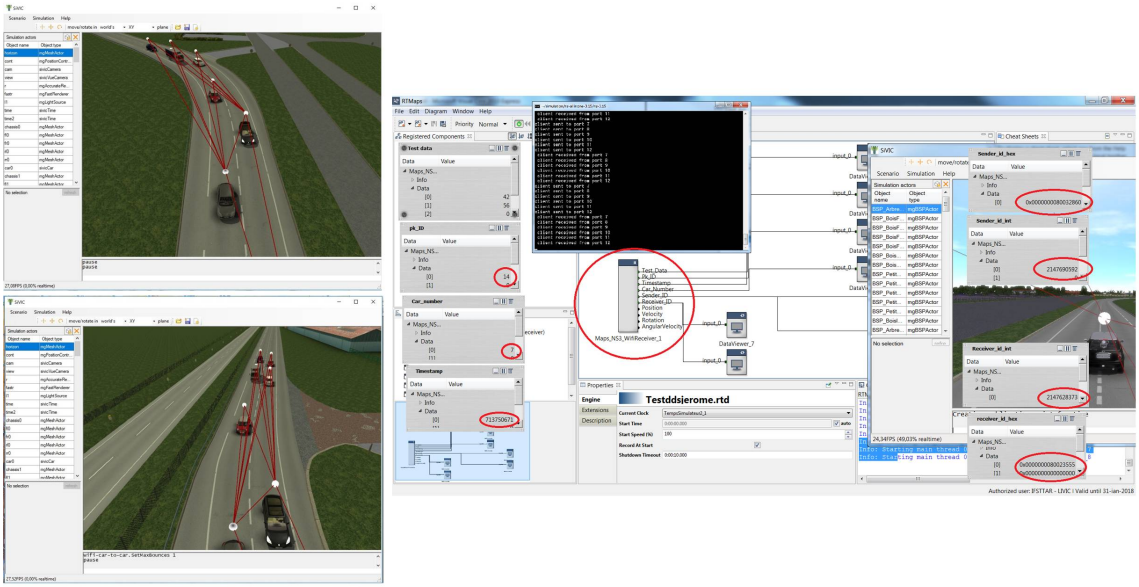


Figure 94: PROSIVIC, RTMaps, and NS3: An efficient platform for C-ITS prototyping with physico realistic communication means (Source UGE).

Moreover the scenario preparation stage is done by using of the set of following softwares:

- **ROADS**: road network generation and 3D mesh generation
- **PathEdit**: Trajectory generator and script building
- **GRoTex**: Road texture generator

2.7.2 CARLA and U-Test

2.7.2.1 Objectives

This sub project consists in integrating CARLA simulator and U-Test platform. The objective is to test and validate the ability to connect different autonomous driving system simulators and hardware in the same co-simulation and in real time in order to achieve the second objective of the POC 1 : propose a complete and generic simulation architecture. By adding a new simulator and connectable with any other equipment / simulator / system U-Test demonstrates its capability to act as a real-time interconnection layer in a generic test system architecture.

2.7.2.2 Carla

CARLA is an open-source simulator that has been developed from the ground up to support development, training, and validation of autonomous driving systems. It supports multiples specifications of sensors, environmental conditions, maps generation; provide many realistic vehicles with their real physicals characteristics, pedestrian, maps, a basic traffic manager, weather simulation; and can be connected with other third-party tools for car physics simulation, driving models, maps import and co-simulation...

Figure 95: Carla

2.7.2.3 U-Test

U-Test is a software solution developed by Spherea. U-TEST is architectured into three main items composing an optimized and efficient core:

- **Real-Time Core (RTC)** : a deterministic and reliable real-time scheduler;
- **U-TEST MMI** : this graphical user interface allows to access the whole of U-TEST functionalities;
- **Dataspace or Variable Service (VS)** : a very powerful and fast data sharing service, even between remote computers;

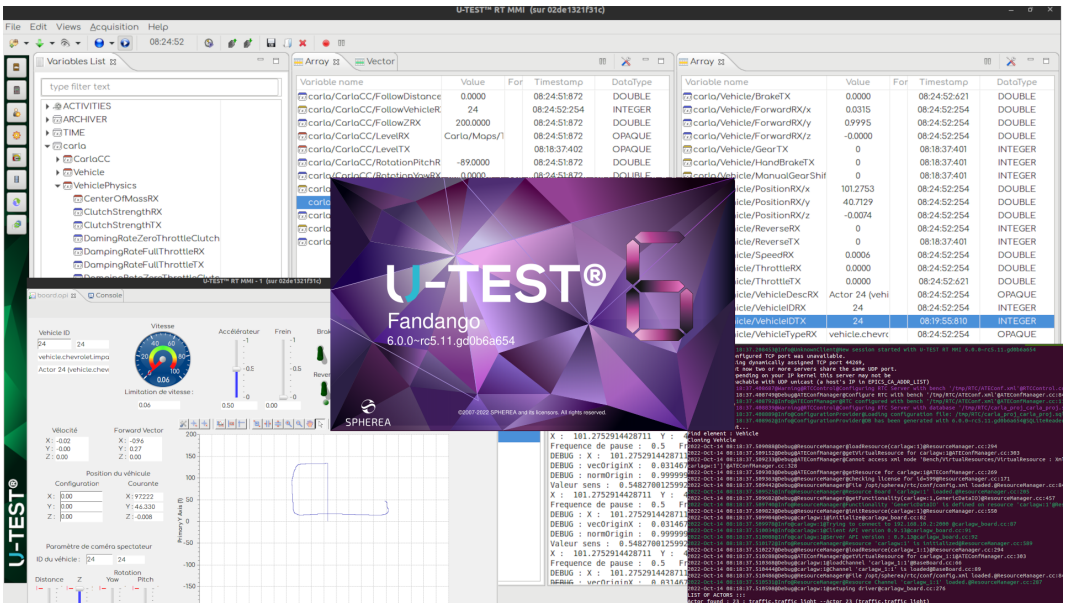


Figure 96: U-TEST Solution (Source SPHEREA).

U-TEST offers a full set of features to spy, monitor, record the signals coming from a system, or to generate the input signals toward this system. It was originally designed to conduct tests on a system during several phases of its life cycle :

- During design phase, to make tests on simulated systems ;
- During integration phase, to carry out tests on under-components of the system, by mixing or not the real elements and the simulated elements ;
- During validation/qualification phase of the system ;
- During production phase, for investigation, evolutions, and no-regression tests.

2.7.2.4 Architecture and state of project

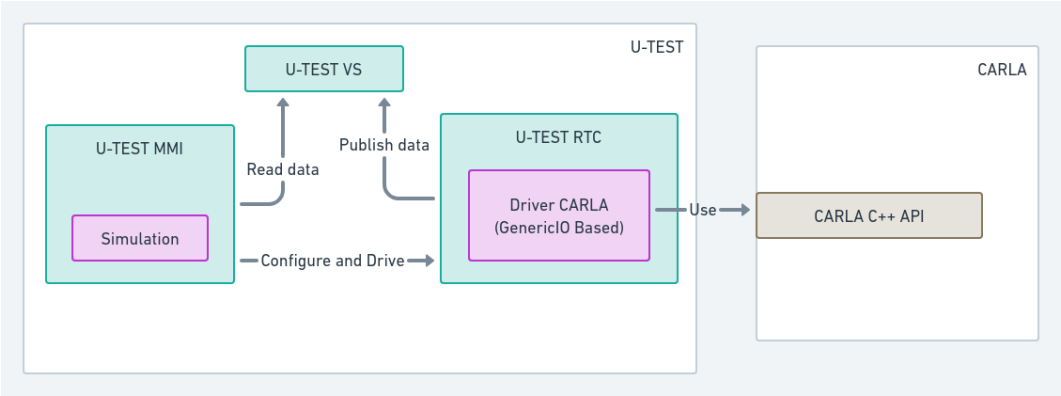


Figure 97: Carla - U-TEST architecture (Source SPHEREA)

A driver was developed in U-Test to control Carla. It is based on a generic driver interface offer by U-Test. The driver can drive a Car by an interface that maps Carla C++ API to the U-Test driver. It is possible to move the car, steer, and control some physics aspect like mass, center of mass, wheel max steer , . . .

There are not sensor interface yet. This sub project is not intended to implement a full interface of Carla. However, a documentation explain how to add new implementations and API connections.

A simple python script simulate a vehicle trajectory on a specific map to show a little demonstration. It's possible to change this script by a more complex models that can interact with U-TEST interface to drive the ego vehicle with a better precision and intelligence.

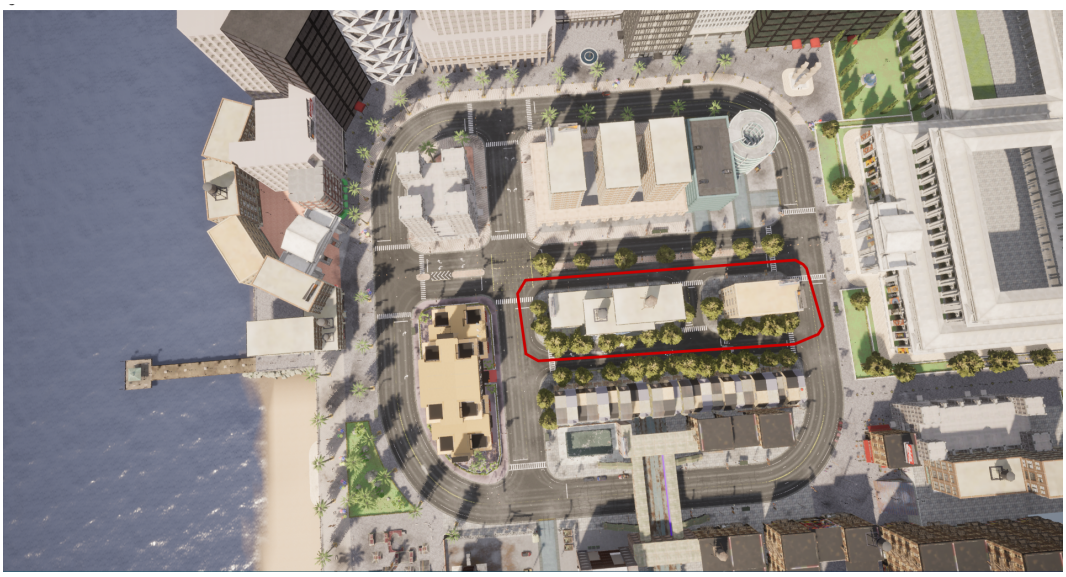


Figure 98: Carla demo Map (Town10HD) with circuit (in red)

The demonstration is performed with a dedicated Carla Computer. It works with a frequency of 60 frame per second and U-Test still perform command successfully at this speed.

Figure 99: POC Running : Carla Vehicle driven by U-TEST (Source SPHEREA).

3 POC 2: Valeo Urban Driving (VALEO)

3.1 Presentation

In the scope of PRISSMA Project, Valeo Driving Assistance Research is developing an AD Simulation Platform that aims at being a Digital Twin of the real Autonomous Driving System used in Open Road and Proving Ground testing. Digital Twin as a global concept can be divided in several activities in Virtual Testing activities. As described in the scheme below, a typical AD system evolves in closed loop interaction with its environments as following:

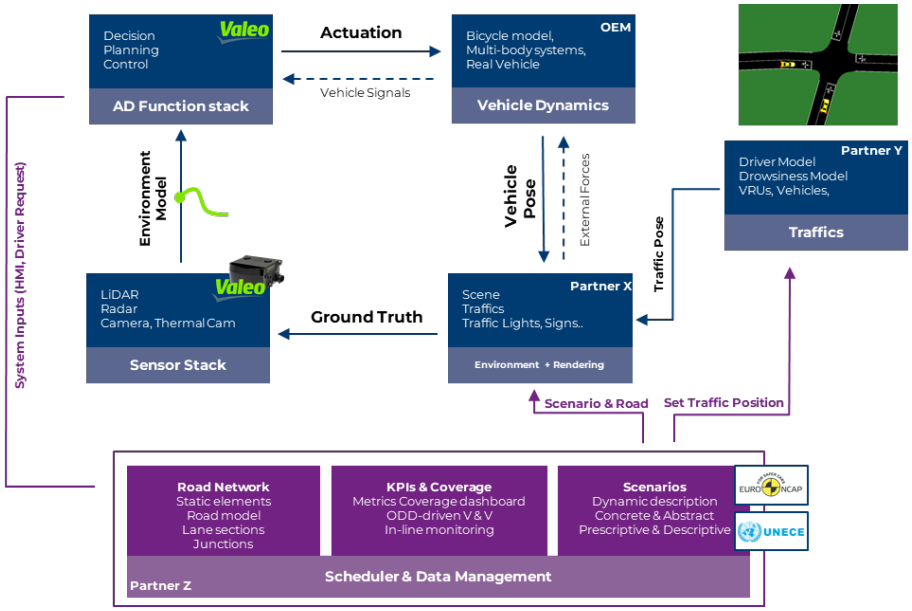


Figure 100: Simulation architecture for the Valeo Urban Driving PoC 2

- The **AD System** contains the typical Plan-Act the decide on Strategic, Operational and Tactical level the actions to be triggered by the Ego-vehicle
- The **Sensor Stack** gives to the AD System an Environmental model as sensed by sensors.
- The **Vehicle** receives actuation commands from the AD System which induce an evolution of its position on the environment
- **Environment** is to be understood as everything that is not the EGO-vehicle. (i.e. traffics and how they evolve on the road network, the road network, infrastructures...)

Based on this observation, we can then assert that each functional bloc stated above can be replaced its digital counterpart or "Digital Twin"
As described below, it gives, as a result, the well known taxonomy mapping of X-in-the-loop (Model/Software/Hardware/Vehicle/Driver).

Figure 101: Matrix of test possibilities for different component levels

Inside PRISSMA, the focus of the AD Digital Twin Simulation Platform is to develop a Digital Twin in a sense of **Software-In-The-Loop V&V toolchain**. Consequently:

- **AD Software** is the real system, embedded in the middleware RTMaps® developed by Intempora
- The **Vehicle** is digital, based on Vehicle Dynamics general model provided by IPG CarMaker®
- The **Environment** is digital, described in IPG Road 5, which is the proprietary format of IPG to describe the road network and the 3D layout (infrastructure etc...).
- The **Sensor Stack** is digital in a sense that there would not be Object Tracking or Computer Vision integrated in the loop. We would rather already semantic information the AD System: the Traffic Sign, Traffic Light, GNSS sensor, Bounding-Box object detection as provided in CarMaker

As all components are software based, there would not be any hardware in the loop, this SiL method then will be launched faster than real time, which allow to cover more corner cases in a fewer time.
On top of this closed-loop simulation, OptiSLang®, an Ansys software, is also used to monitor parameters variations by overwriting the Test Run file description, which is an input for CarMaker simulation. The goal is to perform variations of the functional left-turn scenario to detect concrete scenarios where the ego vehicle would not meet the safety criteria. Through the variations, a virtual assessment of the L4 function against various KPIs is performed.

3.2 Perimeter Definition

3.2.1 Description of the System Under Test

This chapter will focus on describing the system that is being tested in this simulation framework, i.e. the **AD Function Stack**.
Replaced in the context of the whole PRISSMA project, it is to be noted that the function that is

tested here is completely consistent with the other WPs, especially, WP3 and WP4. Indeed, the Software stack that is tested in WP2 will be the same as the one that will be tested in Proving Ground (WP3) and in Open Road (WP4).
The system that will be demonstrated is the Fullway Pilot as described in the scheme above:

- **Prediction** : Functional bloc that, based on the perception of the scene model (Map, Objects, Lines, Traffic Signs etc. . .), predicts the future behavior of the traffic participants on the road.
- **Planning**: Functional bloc that, based on the predicted behavior of traffic participants and scene model, will select the adequate maneuver and trajectory to be followed by the EGO-vehicle.
- **Control**: Functional bloc that, knowing the maneuver and trajectory to be followed will translate it into actuable actions more particularly steering wheel angle command [rad] and Acceleration request [m/s²]

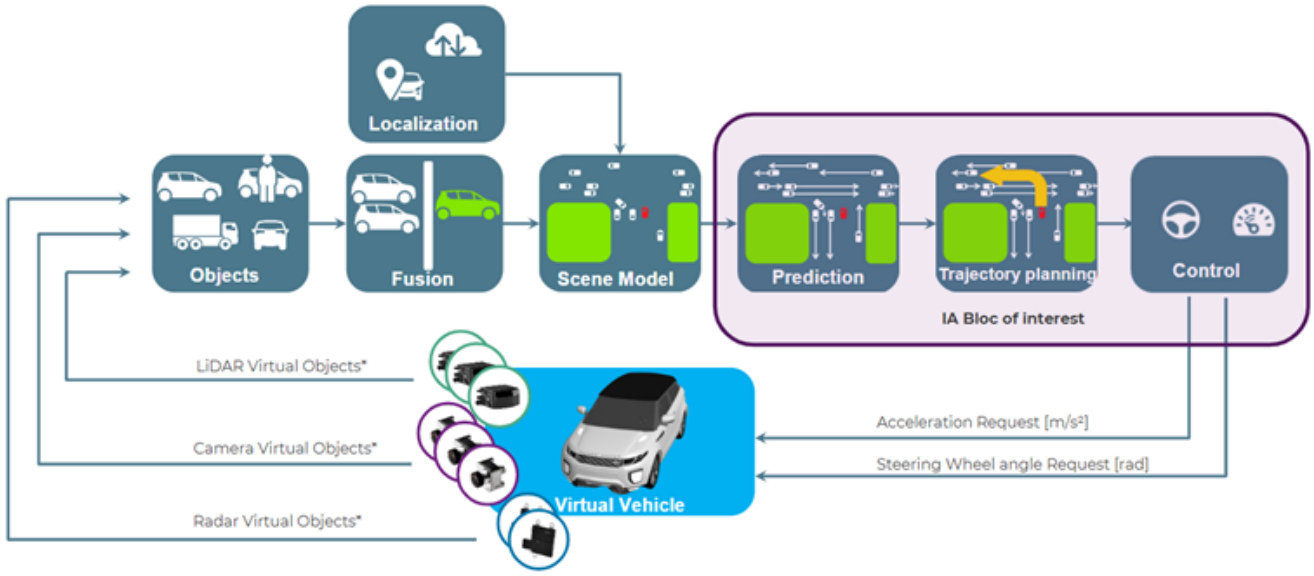


Figure 102: Overview of the IA System Under Test

The illustration shown before is a functional view of the architecture of the System Under Test. On the other hand, the algorithm is a C/C++ based one which is embedded in the middleware ad-hoc system, ie RTMaps©. Here after, as an illustration, there is a screenshot of the RTMaps diagram that embed the functional bloc described above.

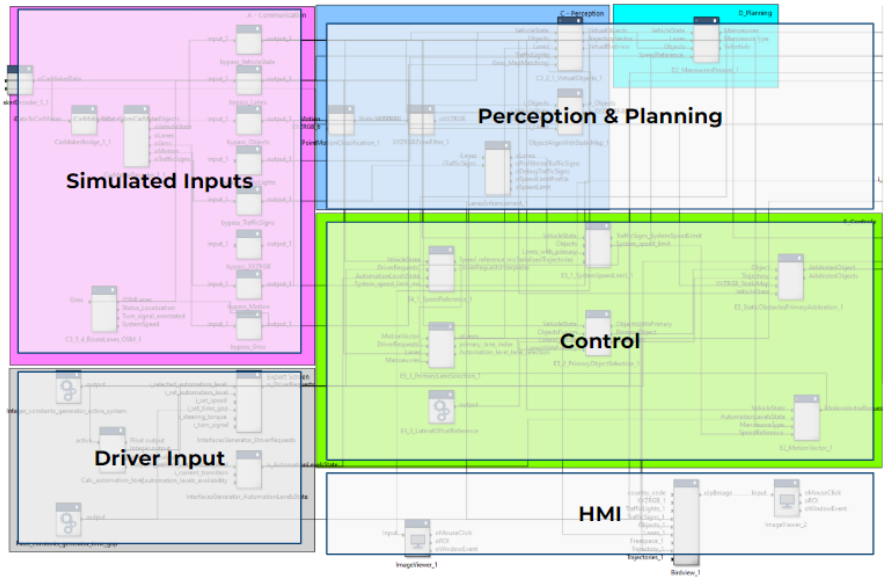


Figure 103: Screenshot of the RTMaps diagram (.rtd)

3.2.2 Operational Design Domain

The first activity to effectively create a digital twin was to replicate the road network of the geographical area where the system is evolving. To do so, we have taken a tile from the Open-StreetMap open-source data. From this (.osm) file we have used a function that is provided by the SUMO communité (netconvert) to transform it into OpenDRIVE 1.5 (.xodr). This .xodr can then be taken as input from IPG CarMaker to be transformed in (.road5).

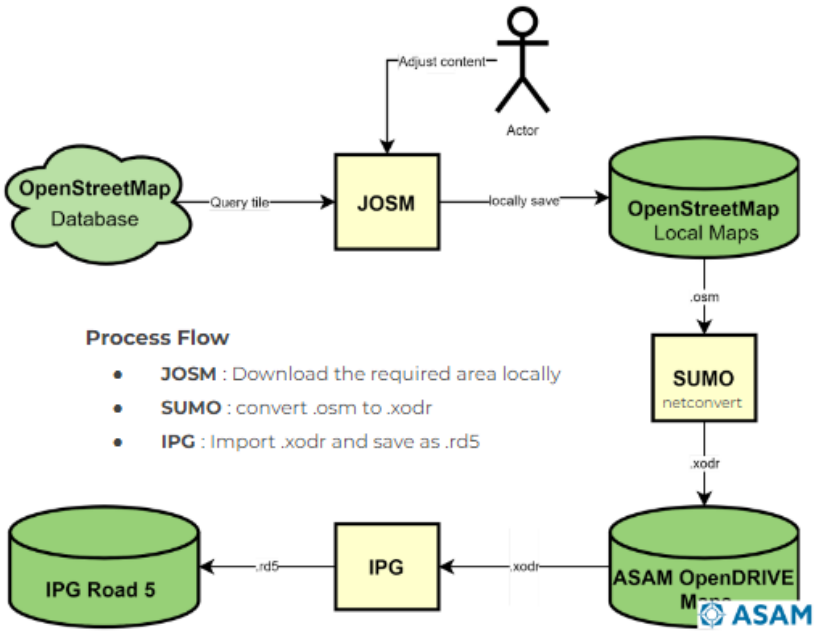


Figure 104: Globalview of the processflow to create road network

It is to be noted that even if this import method works well, it is still imperfectible because for example, the number of lane is not consistent with reality, trafic sign is not taken into account. To improve the road network, it is then advised to add manual work. Here is the final top-view result of the IPG Road 5 file.

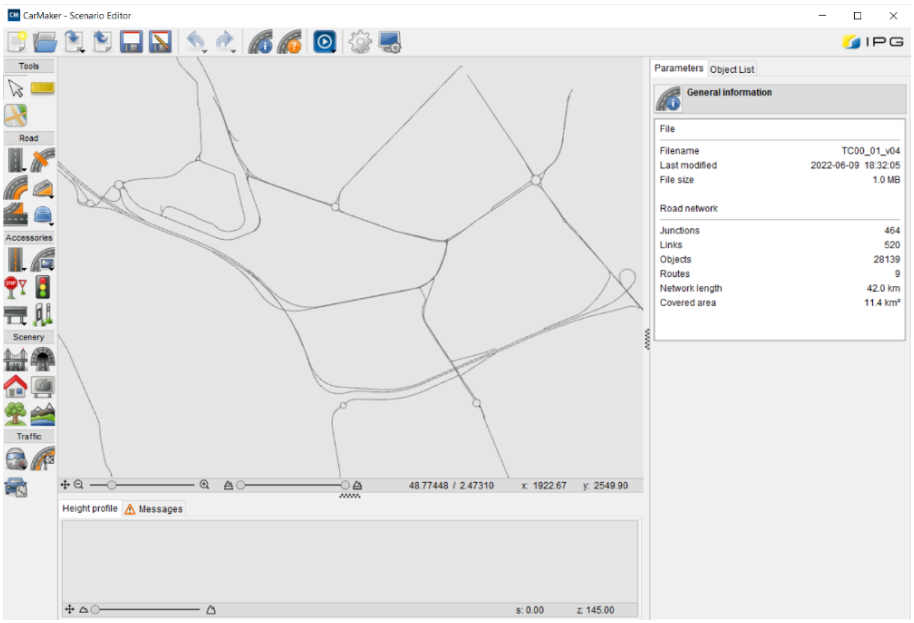


Figure 105: Topview of the roadnetwork imported from OpenStreetMap

3.2.3 Sensor Model

Let's first remind the different typologies of sensor models that are used in the scope of Automated Driving System Virtual Validation.

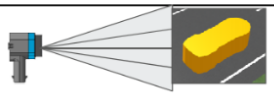
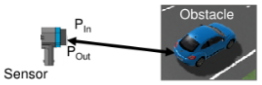
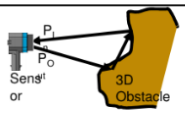
	L1 Functional	L2 Sensor Specific	L3 Semi Physical
			
Sensor	• FOV (Horz.\Vert. Angle\Range) • Number of Segments • Rotation	• FOV (Horz.\Vert. Angle\Range) • Number of Segments • Rotation • 3D-Directivity	• FOV (Horz.\Vert. Angle\Range) • Number of Segments • Rotation • 3D-Directivity
Propagation	• no	• Single Ray for one Reflection Point • Probabilistic	• Raytracing
Environment	• 2.5D Obstacle (Height → extrusion) • only one reflexion • no interferences	• 3D Obstacle • only one reflection • Damping • no interferences	• 3D Obstacles • multiple reflection • interferences • weather

Figure 106: Global Typology of sensor models

In the first deployment of the Valeo PoC, the sensor model used in **L1 Sensor Model** . Sensor Model L1 are functional sensor model where no physical properties intervene in the detection. Therefore, the L1 sensor model will output track that will not include physical specfic behavior (interference, reflexion, ray). As an example, for Object Detection, the L1 Sensor model will output bounding-box that tracks nominally the geometry of the deteted object. For Line Detection, the L1 Sensor Model will output waypoints (for e.g in x,y coordinates) of the detected road marking.

3.2.4 Overview of the Intermediate Results

At this stage, we have developed the toolchain, functional based on a computer at Valeo premises.

The running of a single test run is illustrated in the following scheme.

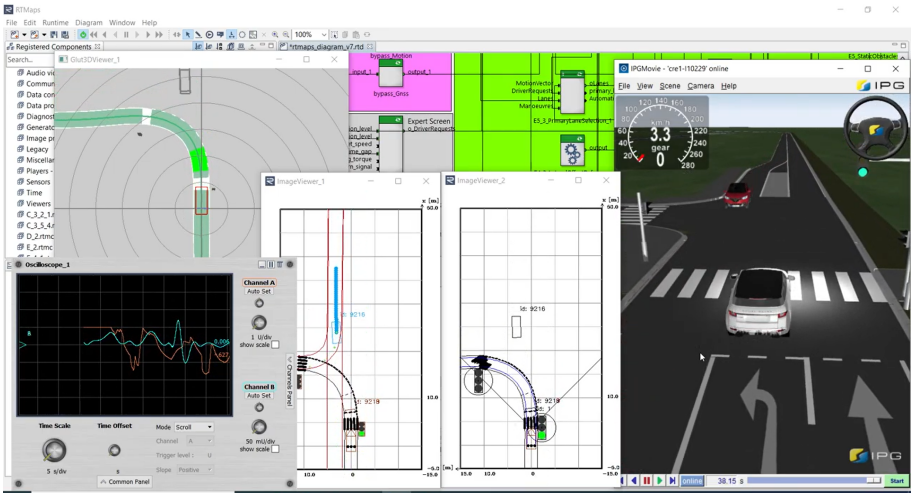


Figure 107: View of the co-simulation running on Valeo premises

On the left of the screeshot, we can find mainly all the monitoring scope of the testrun as for example :

- an oscilloscope to monitor the acceleration request and SW request from the AD System
- an Expert Birdview to see the detected object in the surrounding of the EGO
- an 3D Birdview to have a better understanding of the size of each object etc

On the right, the view is focus on the IPG Movie which helps the monitoring engineer to see the global environment (as it is) in a third person view.

3.3 Simulation platform enhancement

In continuation of the efforts deployed for POC Number 2, a second part of the activity involved the creation of a Python script designed to automate the orchestration and parameter selection processes. This development has been important for ensuring seamless synchronization between RTMaps and IPG CarMaker. The script facilitates efficient management of the interaction between these systems, enabling real-time data exchange and adjustments. This automation not only enhances the reliability and efficiency of the simulations but also significantly reduces the manual workload, paving the way for more advanced testing and validation scenarios in autonomous vehicle development. Here are the main features of the co-simulation (enhanced by the Python script orchestration) :

1. **Test Run Creation:**
    - The script initiates the test run setup, defining and loading the test cases.
    - Each test case is associated with a specific configuration and scenario, which includes the vehicle model and its sensor setups.
2. **System Under Test Co-simulation with RTMaps:**
    - The script integrates the environment model with real-time motion information, leveraging IPG CarMaker as the vehicle simulation model.

- It captures and transmits sensor data from GNSS, Object Detection (as environment model), and other sensors to RTMaps, ensuring dynamic interaction during the simulation.
- The script also facilitates the synchronization of data between IPG CarMaker and RTMaps, providing a cohesive simulation experience.

3. **Test Report Generation:**

- Post-simulation, the script processes the data to compute various KPIs.
- It then generates detailed test reports that include KPI calculations and their implications for the tested features.
- The reports are designed to assess the performance and accuracy of the system under test against predefined benchmarks.

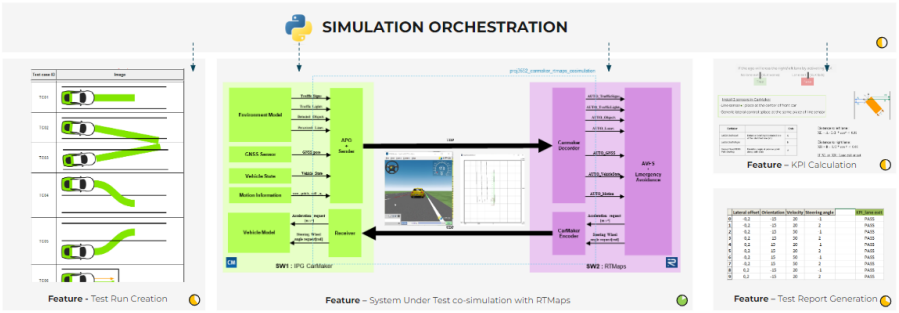


Figure 108: Global view of the Simulation Framework orchestration

3.3.1 Test run creation

The process of test run creation, as previously described in the report, requires a detailed understanding of the scene, which is achieved by interpreting pictograms and through the expertise of using the Simulation Software IPG CarMaker. Designing the scene involves utilizing IPG Road 5 to accurately create and position traffic elements within the simulation environment. This initial setup is useful for ensuring that the subsequent simulations.

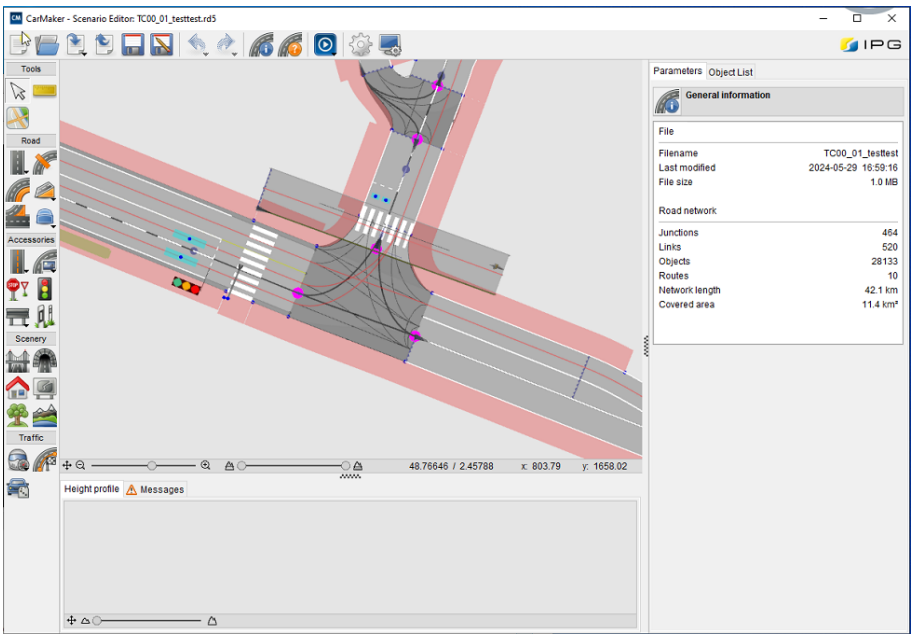


Figure 109: View of the IPG Road 5 Scenario Creation

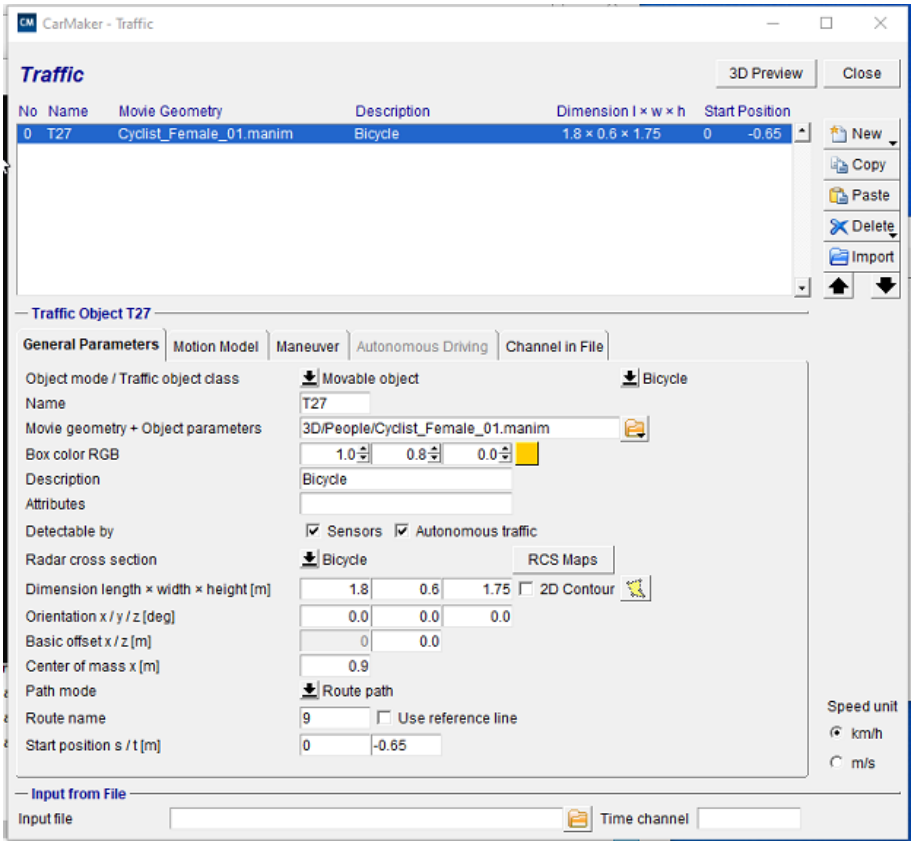


Figure 110: Adding the traffic elements

As a concrete example to illustrate the capabilities of our simulation framework, we can explore a Car-Bycicle Turning Accident (CBTA) scenario. This type of accidentology scenario

is critical for testing the interaction between vehicles and vulnerable road users (VRUs), such as pedestrians.

In the CBTA simulation, a vehicle is set to make a turn at an intersection where a bycicle is crossing. The scenario layout involves the vehicle (VUT for Vehicle Under Test). It has to be noted that this test differ from the one in Euro NCAP as the system is a L4 system, therefore the EGO-speed can not be imposed by the test because it's the system itself that will determine the speed profile according.

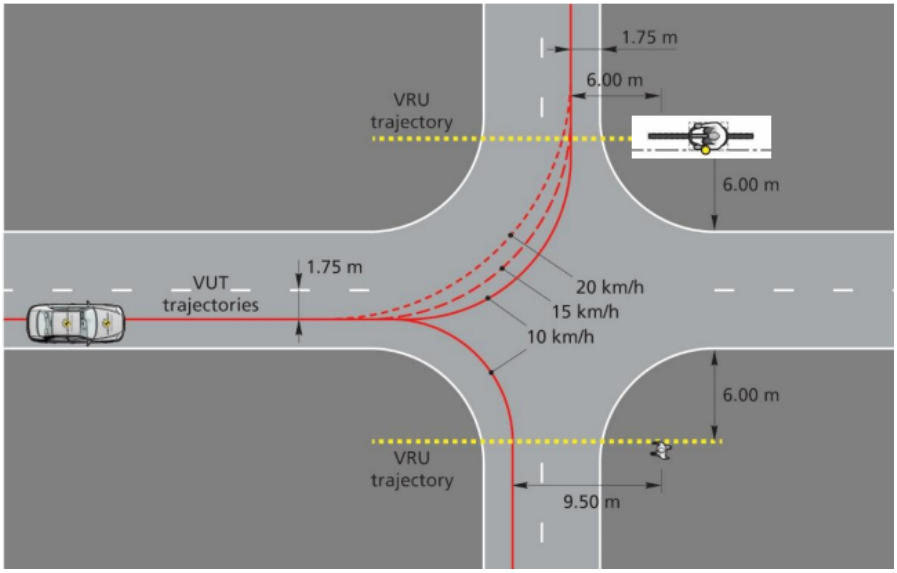


Figure 111: Adding the traffic elements

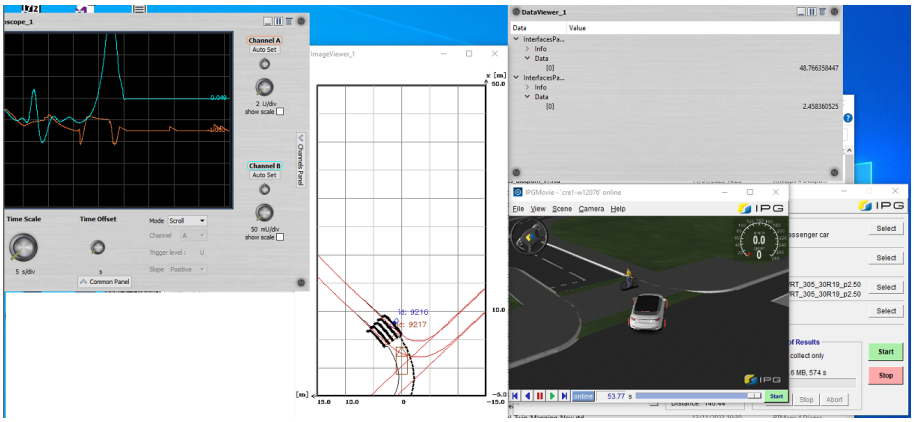


Figure 112: Rendering of the simulation on IPGMovie and RTMaps birview

3.3.2 Simulation orchestration

A Python library has been developed and is maintained to orchestrate and control simulations within the CarMaker and RTMaps environments, as well as to automate parameter settings. This setup utilizes socket messaging to facilitate remote control capabilities, allowing for dynamic adjustments and real-time simulation feedback.

The architecture is designed as follows:

RTMaps Control: A dedicated Python module named `'carmaker_control'` interfaces directly with RTMaps via socket communication. This module sends commands and receives responses, enabling the script to control simulation parameters and execution flow in RTMaps.

CarMaker Control: Similarly, the `'rtmaps_control'` module manages interactions with CarMaker through its own socket connection. This allows for adjustments to vehicle dynamics, environmental settings, and other simulation variables in CarMaker.

These scripting, as a generic and interoperable methods tools, provide a framework for automating scenarios and testing protocols. By integrating Python scripting directly with these tools, developers can create, modify, and run customized test scenarios efficiently. This capability is a step for validating autonomous driving algorithms under varied and controlled conditions, thereby enhancing the development cycle of safety-critical automotive applications. For the example of POC2 of PRISSMA, the goal here is to assess the ability of the Decision/

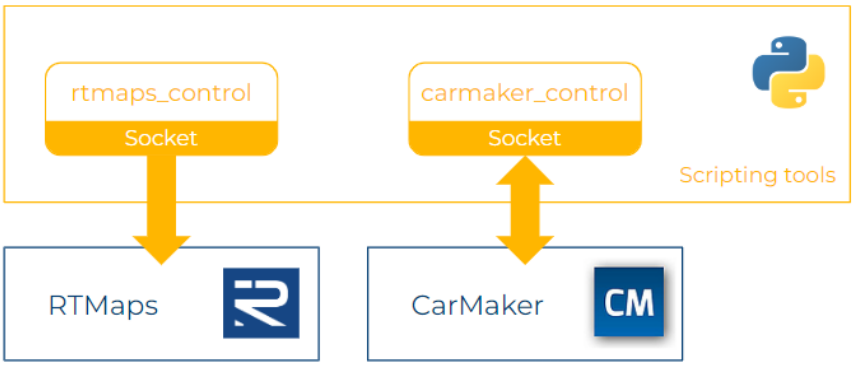


Figure 113: Architecture d'orchestration à travers les socktets

3.3.3 Parameter distribution

To showcase the flexibility and scalability of our toolchain, we will demonstrate its capability for conducting parameter variations, specifically focusing on the dynamics of a cyclist in a traffic simulation scenario.

The first set of parameter variations we will explore involves the velocity of the cyclist, denoted as v_{cyc}, and their curvilinear distance from a critical interaction point, referred to as *dist*.

- **Velocity of the Cyclist** (v_{cyc}): The speeds at which the cyclist will travel are set at various increments to understand different interaction outcomes. The velocities are defined as follows:

$$v_{\text{cyc}} = \{2, 5, 7, 10, 15, 20\} \text{ km/h}$$

- **Curvilinear Distance** (*dist*): This represents the path distance from the starting point of the cyclist to the meeting point with the ego vehicle. The distances are set as:

$$\text{dist} = \{8.5, 10.5, 12.5, 14.5, 16.5\} \text{ meters}$$

These parameters are illustrated in the diagram, where the curvilinear distance ('dist') represents the path of the cyclist from their starting position to the meeting point — marked in green on the diagram — at the moment the ego vehicle arrives at this point. Notably, a distance of 12.5 meters corresponds to a scenario where both the ego vehicle and the cyclist arrive simultaneously at the meeting point.

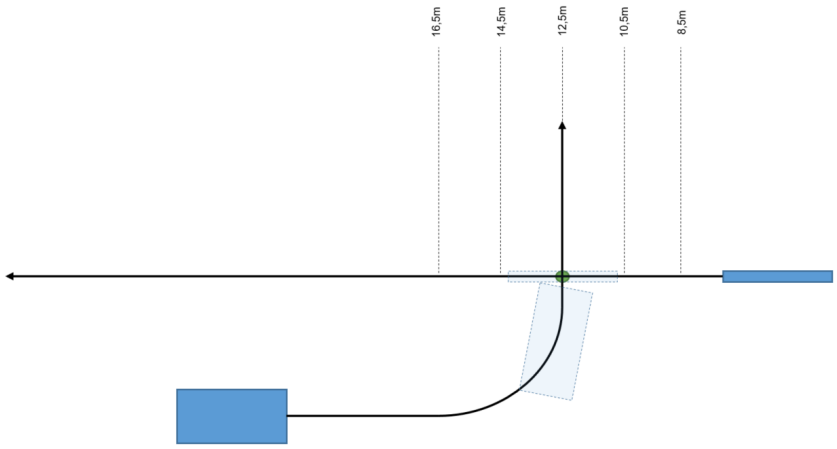


Figure 114: Vue schématique du paramètre 'dist'

This test setup allows us to simulate and analyze the interactions between the cyclist and the ego vehicle under varying conditions. By adjusting the cyclist's speed and the relative distance to the meeting point, we can explore different interaction dynamics and outcomes. This setup is crucial for developing and testing safety algorithms that can adapt to dynamic urban traffic environments where bicycles and vehicles share close proximity. To thoroughly investigate the interaction dynamics between the cyclist's speed and curvilinear distance from the meeting point, we implement a **Full Factorial Design** in our simulations. This approach involves systematically varying each factor across all possible levels and observing the resultant effects on the meeting dynamics. Given the defined levels for velocity v_{cyc} and distance dist, the total number of experimental scenarios can be calculated as the product of the number of levels for each factor. Below, the table summarizes the results obtained from simulations involving different VRU (Vulnerable Road User) speeds and starting distances, aimed at assessing the impact of these parameters on the occurrence of collisions. This analysis highlights critical configurations where interactions between vehicles and vulnerable road users can lead to incidents, thus providing valuable insights into the safety of simulated traffic scenarios.

3.3.4 Conclusion and perspectives

In conclusion, PRISSMA project has been the project that allow us to successfully develop a platform that seamlessly integrates Valeo's L4 Autonomous Driving software with the simulated environment of IPG CarMaker, thereby emulating complex driving scenarios. The integration of the system under test was achieved through a interface with the RTMaps middleware, which has proven instrumental in our testing procedures.

On-premises, we have been able to execute a variety of test runs using classical Design of Experiments techniques, such as Full Factorial, to instantiate concrete test cases. This proof of value have demonstrated that simulation data can be effectively used to functionally test decision and control systems within a function-agnostic environment.

Additionally, this project has helped us step into the world of digital twins, seeing them as valuable assets per se. This led to the development of a 3D digital model of the Paris2Connect area, as previously mentioned. This model provides a precise simulation environment that is compatible with IPG CarMaker but not only. Therefore, when integrated inside the Simulation Platform, the 3D Digital Environment has been a way to have a more representative model of the real world, increasing then the realism of the simulation platform.

Looking forward, the inherent challenge remains our reliance on RTMaps, a tool primarily designed for rapid prototyping that operates in real-time, thus limiting temporal scalability. The future lies in devising a strategy to integrate the System Under Test (SUT) within a dedicated environment that can operate faster than real-time, allowing for scalability in an infrastructure not confined to on-premises limitations. This shift will enable more extensive and efficient testing capabilities, propelling forward our abilities to innovate and validate enhance autonomous driving technologies.

VRU Speed (km/h)	Start Distance (m)	No Collision
2	8.5	VRAI
2	10.5	VRAI
2	12.5	FAUX
2	14.5	FAUX
2	16.5	VRAI
5	8.5	VRAI
5	10.5	VRAI
5	12.5	VRAI
5	14.5	VRAI
5	16.5	VRAI
7	8.5	FAUX
7	10.5	VRAI
7	12.5	VRAI
7	14.5	VRAI
7	16.5	VRAI
10	8.5	FAUX
10	10.5	VRAI
10	12.5	VRAI
10	14.5	VRAI
10	16.5	VRAI
15	8.5	FAUX
15	10.5	VRAI
15	12.5	VRAI
15	14.5	VRAI
15	16.5	VRAI
20	8.5	FAUX
20	10.5	VRAI
20	12.5	VRAI
20	14.5	VRAI
20	16.5	VRAI

Table 16: Results of VRU speed and start distance impact on collision occurrence.

4 POC 3: Vehicle-In-The-Loop (VIL) real vs. simulation (UTAC, AVS)

4.1 Presentation & Overall Goals

With regards to demonstrating the use of tests in simulation for both certification and homologation of AI-based autonomous driving systems, AVSimulation and UTAC gathered to put together a simulation software and a ADAS/AD system.

The AI-based ADAS/AD system to be included in the test loop for this POC will be openpilot [14], which is a open-source level-2 ADAS developed by the company Comma.ai, currently based in the San Diego, CA.

Initially, the goal is to demonstrate a real-time interface between SCANeR studio and the AI-based AD/ADAS systems of our choosing. Synthetic data will be fed into the system and the vehicle control signals (lateral and longitudinal) will be observed, analysed and put under evaluation.

4.1.1 Operational Design Domain

According to the engineers in charge of openpilot, it is ”an Adaptive Cruise Control (ACC) and Automated Lane Centering (ALC) system. Like other ACC and ALC systems, openpilot is a failsafe passive system and it requires the driver to be alert and to pay attention at all times. In order to enforce driver alertness, openpilot includes a driver monitoring feature that alerts the driver when distracted.” [67].

The engineering team in charge of R&D affirms that openpilot is developed to be compliant with FMVSS requirements and to follow industry standards for L2 ADAS systems. Functional safety norms issued from the ISO26262 standard guidelines are observed, including those from ”Functional Safety Assessment of an Automated Lane Centering System” [68] released by the National Highway Traffic Safety Administration (NHTSA).

Concerning the limitations of openpilot’s operational domain, many factors can impact the performance of ALC and LDW capabilities, causing these systems to be unable do function as intended. A non-exhaustive list of these is:

- Adverse weather conditions (fog, snow, rain etc.) causing poor visibility and interfering with sensor operation.
- Cameras lens are obstructed by mud, snow, ice etc. or damaged.
- Steering torque is limited in sharp curves, like some intersections.
- Existence of restricted lanes or construction zones.
- Driving in presence of strong cross-wind.
- Bright light (due to direct sunlight into the camera sensor, oncoming headlights etc.)

Likewise, longitudinal control related ADAS included in openpilot, namely ACC and FCW, may be impacted by many factors causing them to be unable to function as intended, thus requiring the driver to pay close attention to the vehicle’s surrounding and be ready to take control of both brake and gas pedals. The additional factors from the list above include, but are not limited to:

- Approaching a toll booth, a bridge of a large metal plate.

- Driving in roads with VRU (Vulnerable Road Users).
- In the presence of stationary vehicles in the same lane.

Moreover, openpilot is designed to limit the amount of deceleration and acceleration that its controller outputs to the car actuators, which may implicate in unsafe scenarios when abrupt braking manoeuvres are required. Besides that, openpilot does not detect speed limits, and the system may malfunction when the posted speed limit is below the speed set by driver.

Finally, regarding the Driver Monitoring System (DMS), openpilot may be unable to function as intended if (but not only) low light conditions exist (night drive, tunnels etc.), bright light conditions exist, the driver's face is partially outside the field-of-view of the driver-facing camera or if this camera is obstructed. [69]

4.1.2 Tests in Simulation and Expected Results

Our POC is based on digital twin project, that is to say, one part of the project is virtually realized in simulation in our case and the other part is physically realised with Vehicle-In-the-Loop.

Firstly, we will talk about the different tests we will perform in simulation and what we expect from this study.

Our AD system has several identifiable functions : ACC, ALC, FCW, and LDW. The main objective is to evaluate each function performance individually to make sure our system is reliable to be driven on real world/open roads.

For FCW function, we will use selected tests and scenarios from AEB VRU and C2C tests protocol. We have selected 2 scenarios to evaluate system performance : one scenario with a bicyclist as target (CBLA) and one scenario with pedestrian as target (CPLA).

7.3.4 Car-to-Bicyclist Longitudinal Adult

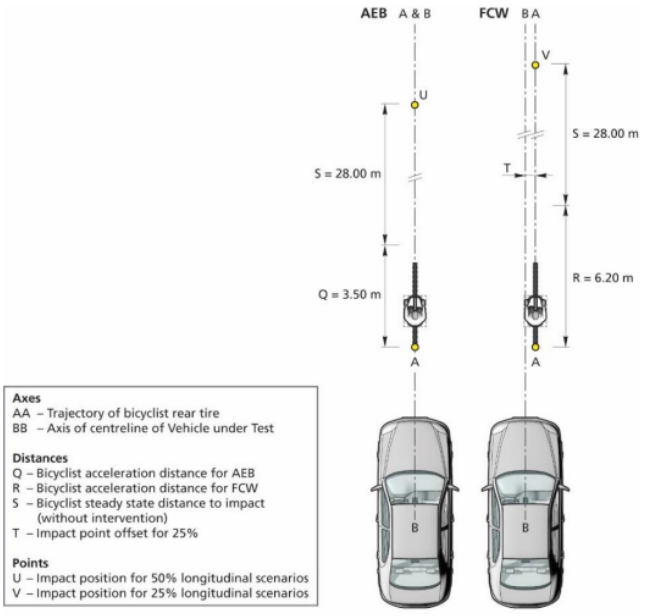


Figure 7-13: CBLA scenarios, Longitudinal Bicyclist (AEB left & FCW right)

Figure 115: Figure extracted from Euro NCAP AEB LSS VRU Test Protocol - v4.1

The Euro NCAP specification concerning the VRU FCW is when the FCW signal is triggered before 1.7s Time-To-Collision (TTC). Then , we have to be sure the OpenPilot system fits this requirement.

For LDW, Euro NCAP protocol gives some indication about the warning must award. We talk about Distance To Line Edge (DTLE), it means the remaining lateral distance (perpendicular to the Lane Edge) between the Lane Edge and most outer edge of the tyre. In order to get all the points given for this function, the system needs to award the warning before **DTLE $<$ -0.2m.**

4.2 Perimeter Definition

4.2.1 Simulation Environment

The Figure 116 shows a AVSimulation's software suite logical architecture, in which functions are allocated to software components. Reading the schematics from left to right, a typical user workflow can be found:

- Accessing data relevant to the test (a virtual model of the test terrain, vehicles, vulnerable road users, static road elements, scenarios etc.).
- Configuring a simulation environment through the GUI (modules to run, desired frequency etc.).
- Preparing a test by choosing a scenario to run and combining simulation elements (e.g. the ones listed above).
- Preparing a list of tests to run, through the definition of variables of interest/parameters and the generation of several combinations of them.
- Executing the simulation models.
- Recording data for further analysis.

The "simulation execution" component here presented corresponds to one possible way of exploiting the partner software simulation suite. It considers, for example, that any sensor model employed in the simulation setup is the one intrinsically embedded to SCANeR studio. External models can be interfaced, the guiding principles are found in the "interfaces specifications" section below.

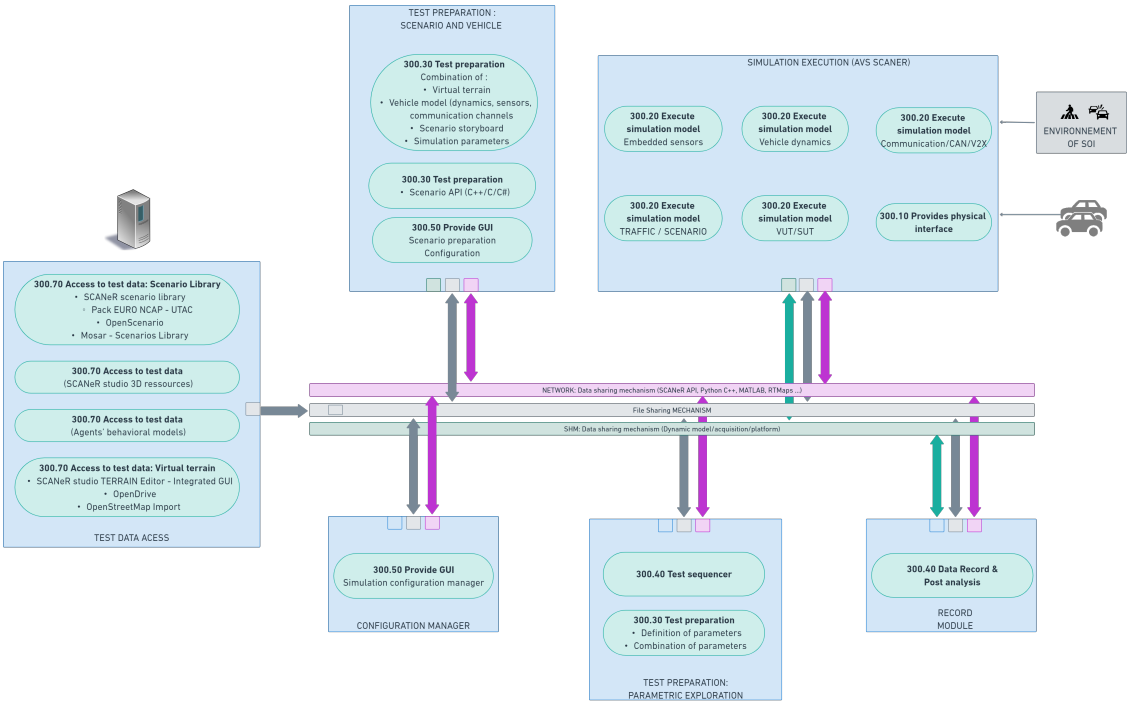


Figure 116: Logical architecture of the simulation environment involving SCANeR studio in PRISSMA. Source: [**?**].

Among the existing realistic virtual 3D environment in SCANeR studio, the Figure 117 shows a view the TEQMO test circuit, which replicates with fidelity the real terrain equivalent geometry, road elements, physical properties of materials etc.

Figure 117: 3D modelling of the TEQMO terrain in SCANeR studio.

To provide a visual example of a simulation scenario taking place at this terrain, the Figure 118 shows a virtual pedestrian crossing scenario, on the TEQMO virtual environment.

Figure 118: View of a virtual pedestrian crossing scenario test.

One important aspect of the simulation environment is the sensors models, parameterized based on their real counter-parts we want to model. Comma.ai sells the hardware on which openpilot is embedded and can run optimally. The most recent version of their hardware is called Comma 3 (Figure 119), on which three cameras are present. Two cameras are oriented towards the exterior of the vehicle (main road camera with narrow field of view and wide angle camera with 185° fish eye lenses), and the third one captures the cockpit, providing images to the Driver Monitoring System.

Figure 119: Comma 3 device and its three cameras.

On SCANeR studio, we parameterized our camera model to synthesize images similar to the ones the real camera sensor captures, taking into account the technical specifications of the image sensor, lens etc. The Figure 120 shows the GUI provided by SCANeR studio to define a sensor configuration on which the two road facing cameras are listed. For the moment, the DMS will not be included in the test loop as we are not interested in simulating the cockpit camera.

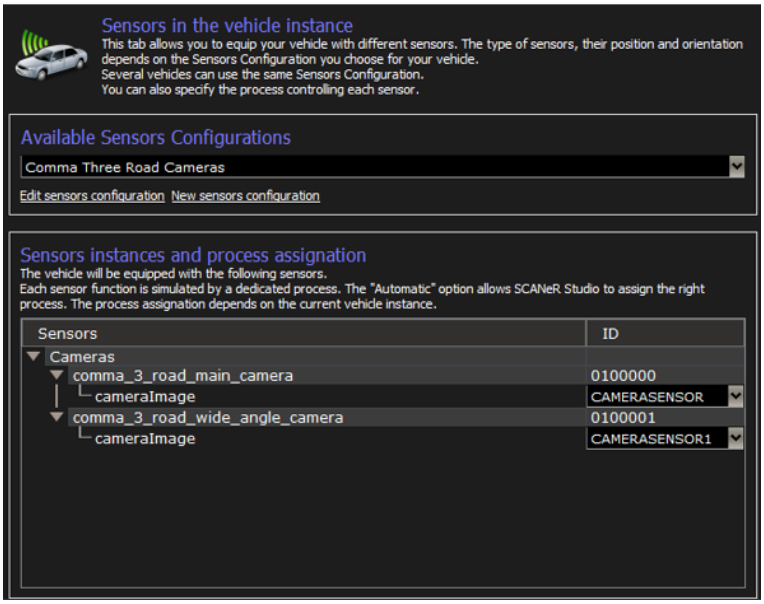


Figure 120: Screenshot of SCANeR studio GUI displaying the sensor configuration containing the camera models.

Figure 121: Sample of Comma Three Images.

Besides taking RGB images as input, openpilot also uses signal from the RADAR (when one is mounted on the vehicle), the GPS receiver (Ublox) and IMU. Models of these components shall also be part of the simulation environment.

4.2.2 Real-world Testing Conditions

UTAC, a leading group in the field of development and validation testing, automotive homologation and new technologies related to the autonomous, connected and electric vehicle. UTAC has several test sites around the world : US, Finland, Marocco, UK and France. In 2018, TEQMO test tracks were built in order to provide testing and validation services. It includes all the roads we can meet on European and french infrastructure : highways, city and rural roads. Related to PRISSMA use case, the city part will be used to realise test in real conditions.

In order to reproduce open-world conditions, UTAC has many types of targets : pedestrian, bicyclist, motorcyclist and car. All of these vehicles can be impacted up to 80km/h.

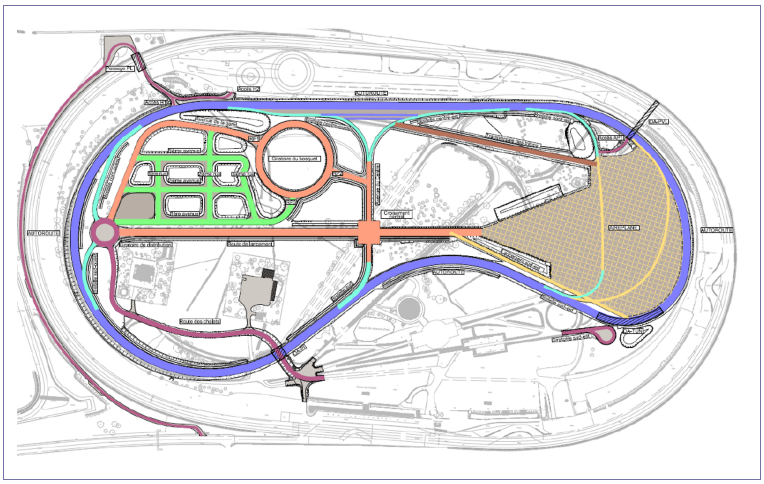


Figure 122: TEQMO map showing the different test tracks

Figure 5-1: Euro NCAP VRU Targets (EPTa, EPTc, EBT and EMT)

Figure 123: Euro NCAP targets used by UTAC

4.2.3 Choice of Artificial Intelligence Algorithm

The AI-based system to be included in the test loop for this POC will be openpilot [14], which is a open-source level-2 ADAS system developed by the company Comma.ai. Openpilot is based on a end-to-end machine learning approach, as the Figure 124 illustrates. The AI algorithm takes as input raw sensor data and outputs a desired trajectory for the vehicle. Looking at the simplified representation of the ML model, two main stages can be identified: vision and policy. The first one, vision, is in charge of reading raw image data from cameras and condensing this data into a lower-dimension, encoded variable. Also, it is based on the EfficientNet B2 convolutional neural network, introduced in [70]. The second stage of the NN architecture is in charge of policy learning, i.e., outputting the desired driving path.

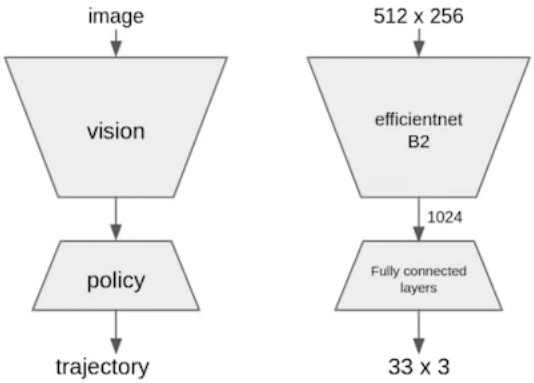


Figure 124: Simplified representation of openpilot end-to-end neural-network architecture.

The main advantage behind the end-to-end approach is that data collected while a human is driving a car can be later used to improve the algorithm. In other words, human drivers are basically annotating data, that are in the sequence logged and uploaded over-the-air into the company database, from where data can be analyzed and used to further train the neural network model. The Figure 125 draws a flow diagram of driving data, both online and offline. On the other hand, by choosing an end-to-end system, access to intermediate properties is hampered, for instance object localization and tracking.

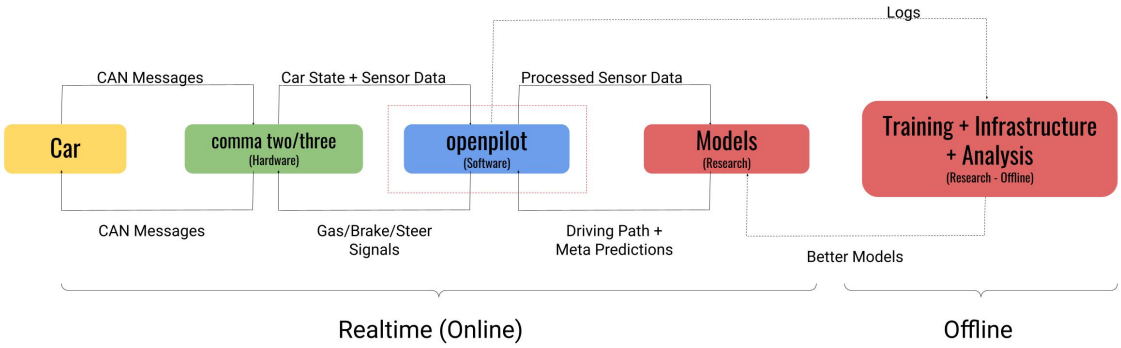


Figure 125: Data flow diagram around openpilot. Source: [14]

.

The AI model inside openpilot is known as ”Supercombo” and it is responsible for perception, planning, localization and control. On the preprocessing stage, consecutive RGB images are concatenated together to serve as model input. The main network is based on Google Efficientnet-B2 [70] as the model backbone, connected to gated recurring unit (GRU) to capture temporal information. Finally, the prediction head is composed of fully connected layers that output a few possible trajectories, among which the one with the highest confidence is chosen as the planned path. Besides that, Supercombo also predicts lane lines, road edges, the position and velocity of leading objects.

4.2.4 Data to be Extracted

Chen et. al. presented in their paper [15] qualitative and quantitative results of a real-world analysis of openpilot. The experiments show that the Automatic Cuise Control (ACC), Automatic Lane Centering (ALC) and Stop-and-Go ADAS behaved correctly in situations with and without visible lines, on a straight road with heavy traffic, in cut-in/cut-out and breaking scenarios. The same authors propose some metric to evaluate openpilot, such as the distance to lane lines, the distance to the leading car and the speed of the leading car. Finally, a list of typical failure cases is presented, including: conical barrels are not detected, other vehicles are not detected at night, ego car speed is too fast for the road curvature, others vehicles cut-in too closely.

Chen et. al. propose in their article [15] two kinds of metrics to evaluate the ADAS system: the imitation and the comfort metrics. Imitation metrics aim at showing the ability of an AI based model to learn from human drivers. These include: the average euclidean distance error and the average precision, both calculated using the predicted trajectory and ground truth points, i.e. the trajectory a human would take. The second kind of metrics correspond to comfort metrics, that aim at quantifying a trajectory in terms of jerk and lateral acceleration to provide insights on how comfortable the onboard passengers feel.

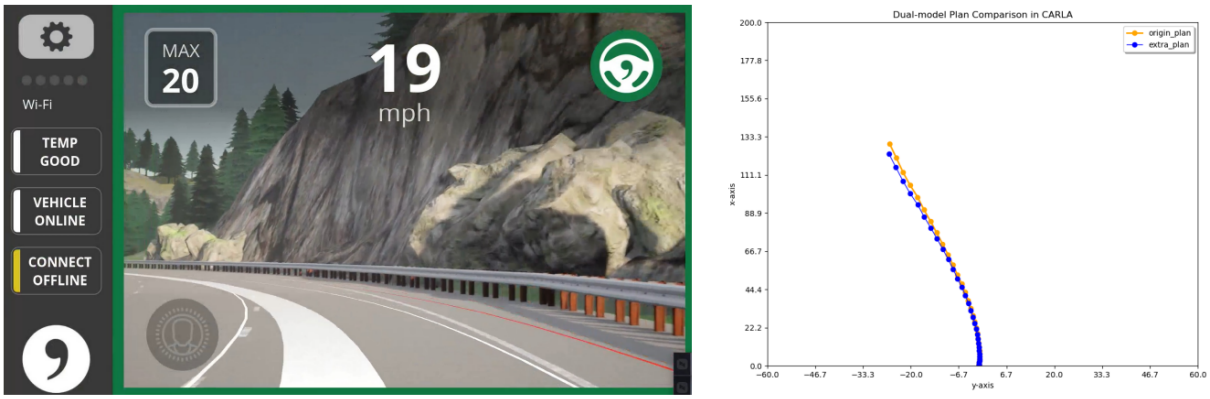


Figure 126: Data flow diagram around openpilot. Source: [15]

.

The authors of the same paper [15] present the results of some tests in a simulation environment, whose main purpose (in their approach) is to verify whether the model is logically correct or not. On the right side of the Figure 126, a BEV (Bird-Eye-View) visualisation of the predicted trajectory of the model, given a curved road, which may also be used for computing metrics (e.g. with respect to the expected trajectory an ideal driver would take).

4.2.5 Methods, Procedures and Protocols for Evaluation

To provide evidence on topics such as representativity of sensors models embedded in simulation, AVS has worked on other R&D collaborative projects, such as the 3SA Project [71]. Among many other activities, the modelling of camera sensors is of great interest in this project, towards the goal of replicating in simulation the behaviour of a physical sensor, specially in non-ideal conditions such as adverse weather.

The methodology developed within the project required collecting real data of driving scenarios on a test track and then re-projecting those scenarios on a camera bench. To do so, such scenarios were reconstructed in simulation, both in terms of positioning of the main actors (vehicles, pedestrians, relevant static elements of the scene) and sensor data. The figure 127 shows vehicles used during the phase of data collection from physical tests.

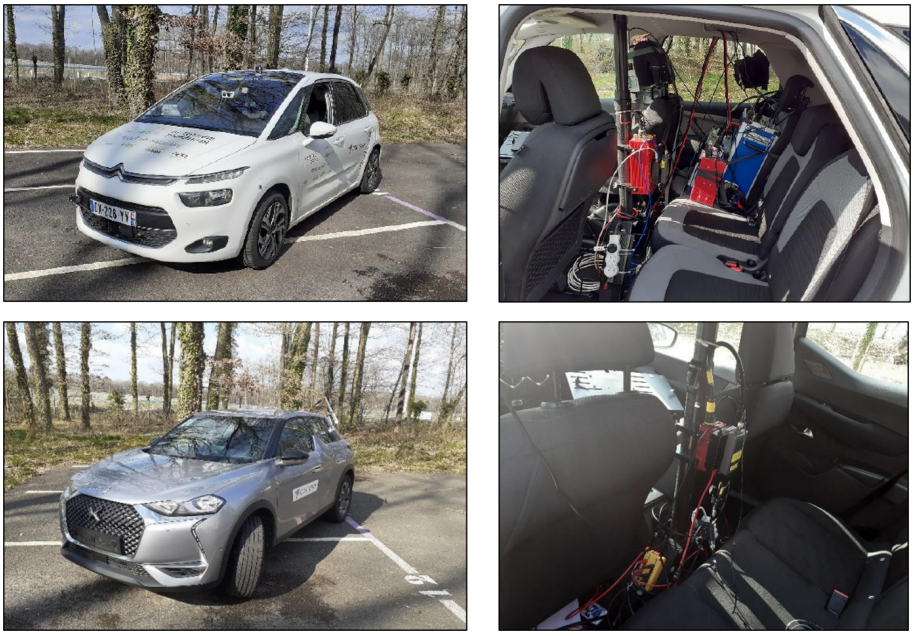


Figure 127: Vehicles under test used to collect multi-sensors data during driving scenarios

The next step following the data collection on UTAC test tracks was to replay in simulation such scenarios. The figure 128 corresponds to a snapshot of one of those scenarios, showing a comparison between a synthetic image generated in simulation and its real counterpart. Elements such as the road markings, the shape of the target vehicle shadow and the roll angle of the camera can be analyzed to determine representativity of the simulation in a qualitative manner.

Figure 128: Comparison between a synthetic image and its real counterpart.

The Figure 129 offers a quantitative indicator of the representativity of the replayed scenario in simulation, in terms of the output of the smart-camera. The example shows the longitudinal distance between the ego-car and its target vehicle. The black line corresponds to the value estimated on the test track with the smart-camera embedded in the ego-car. The red line corresponds to the value estimated by the smart-camera mounted in the camera bench when a real video recording is replayed. Finally, the blue line corresponds to the value estimated by the

smart-camera mounted in the camera bench when the synthetic image is displayed in the projection screen. The agreement of the the 3 lines evidences the capability of the simulator to replay the driving scenarios and produce synthetic images really similar to real ones.

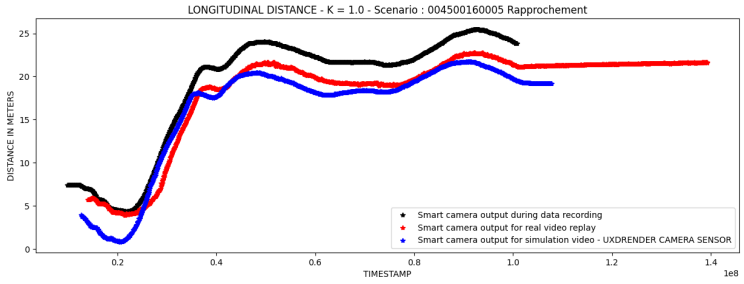


Figure 129: Longitudinal distance between ego car and its target estimated by a smart-camera.

4.3 Implementation

4.3.1 Scenario Management (MOSAR)

It is planned at this stage of the project that instantiation of this POC will be addressed in MOSAR in order to manage scenarios.

This section of the deliverable will be enriched in the following versions of this document in order to present the progress regarding the concrete use of MOSAR to manage scenarios that will be addressed by the POC.

At this point of the project iterations are planed to begin for this POC starting November 2022.

4.3.2 Scenario Management (Pack UTAC)

Since 2020, UTAC and AVSimulation has a strong relationship thanks to a partnership. UTAC and AVSimulation propose scenarios pack as an Add-On in SCANeR studio. These scenarios are directly executable in SCANeR studio and the content is based on NCAP protocols from different areas and based on EU regulatories.

For the tests in simulation, we will use some scenarios from UTAC European Pack as we in the the section 4.1.2.

4.3.3 Ongoing Development

The Figure 130 displays a view of the software solution we aim at building. Blue boxes represent sensors, whose generated data are fed into the ADAS/AD. The yellow box represents the end-to-end neural network (ie Supercombo), which is the main AI component in the test loop. The goal is to synthesize sensor data (camera, RADAR, IMU and GPS) in a similar manner to reality and observe the control actions produced by openpilot, laterally and longitudinally. To do so, a SCANeR module is under development, based on the SCANeR API (python).

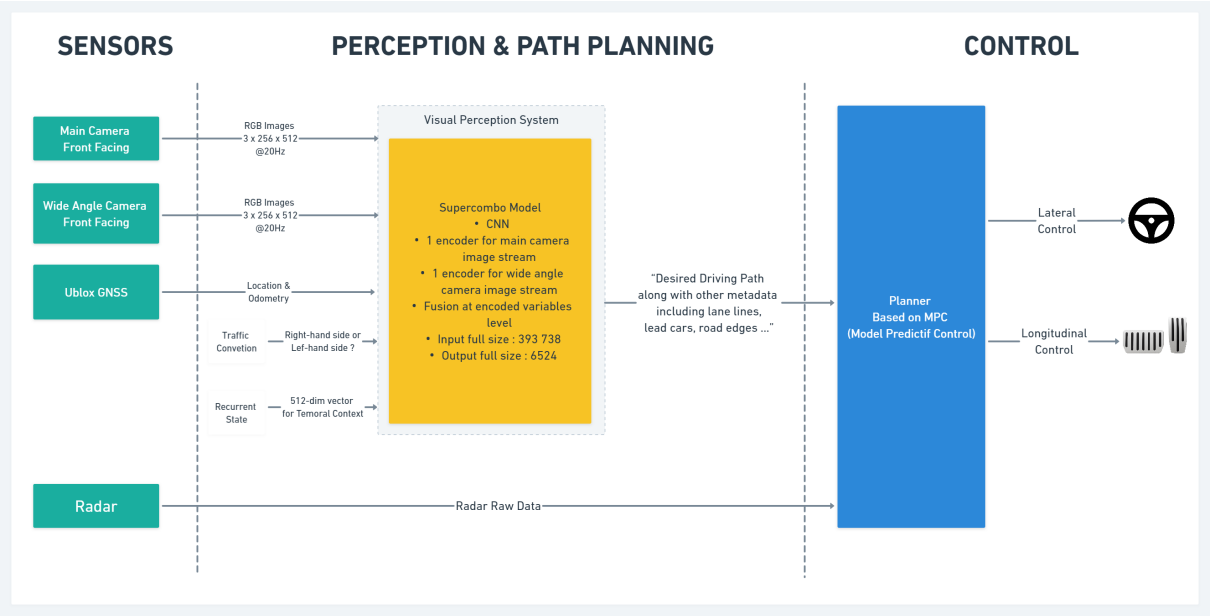


Figure 130: openpilot autonomous driving software stack.

5 POC 4: Vehicle-In-The-Loop (VIL) real vs. simulation (TRANSPOLIS, INRIA)

5.1 Presentation & Overall Goals

In order to test and validate Inria perception software [72] embedded on the experimental platform, shown in Figure 131, Inria and Transpolis propose to demonstrate a set of selected scenarios, using a novel Augmented Reality (AR) method able to merge in real time virtual and real sensor data [73]. By adopting this AR framework, simulated environments can be used to enrich real experiments and generate more complex, critical, and otherwise dangerous test scenarios.

Figure 131: Inria experimental platform: Renault Zoe car equipped with Velodyne HDL64, 4 Ibeo Lux LiDARs, Xsens GPS and IMU and cameras.

5.1.1 Design Domain

This proof of concept proposes to demonstrate representative scenarios [74], which have been proven to be at a high risk of collision, in an urban environment, at relatively low speed. The benefit of the proposed method is the ability to safely generate near misses and actual collisions at a Vehicle in the loop level, where the environment is a hybrid of the real test site, and both simulated and real road actors, using the augmented reality framework.

5.1.2 Tests in Simulation and Expected Results

This POC proposes a hybrid method between virtual and real testing. It takes advantage of simulation on the following points:

- Implement a set of scenarios for augmented reality tests. Scenarios are designed and tested first in simulation using Zoé digital twin then executed with real-world Zoé on a test track.
- Analyse each scenario with a set of metrics designed to evaluate occupancy grid-based perception.
- Generate a ground truth for occupancy grids using the georeferenced digital twin of the environment and the virtual actor data from the simulator.

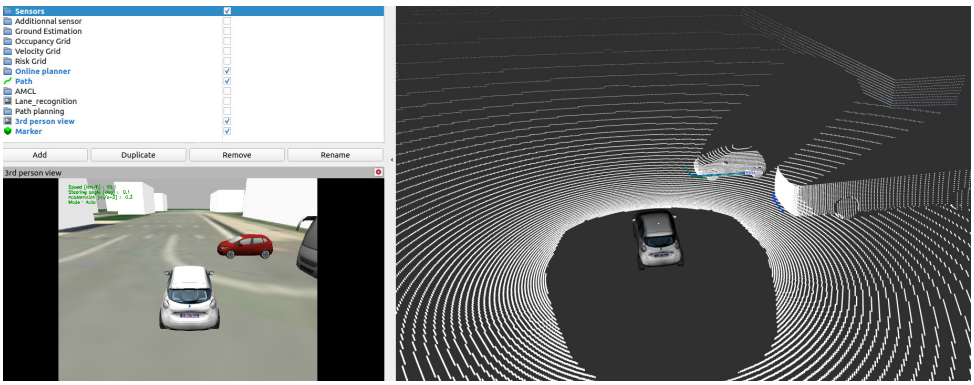


Figure 132: Experimental Platform: digital twin

5.2 Perimeter Definition

5.2.1 Simulation Environment

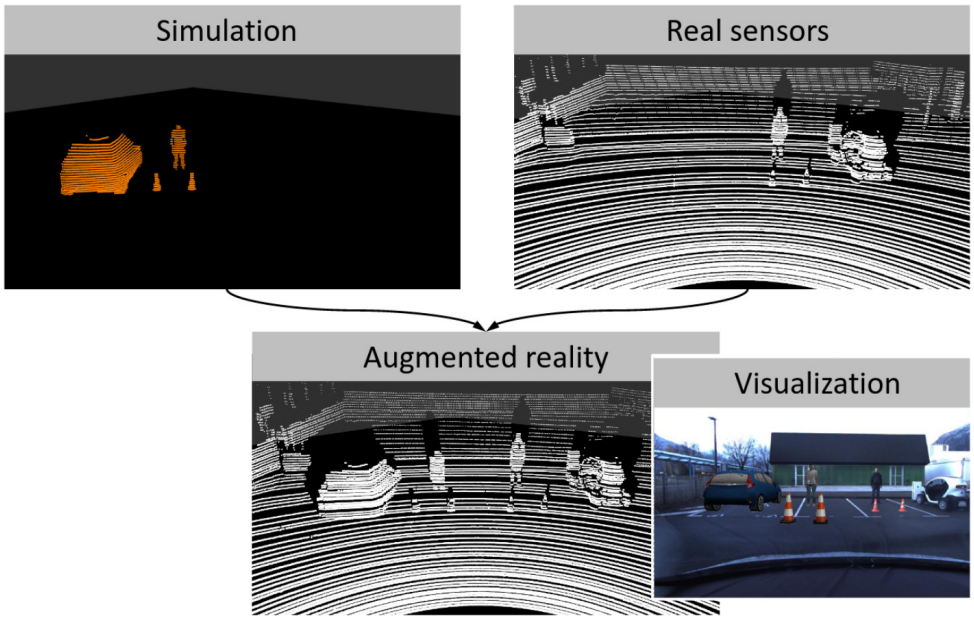


Figure 133: Augmented Reality framework

The INRIA CHROMA group has developed a virtual twin of its Renault Zoe experimental autonomous vehicle on the Gazebo simulator. This virtual twin generates the same outputs (sensors messages, localization) that the actual vehicle does, reacts to the same commands, and has a realistic kinematic and dynamic behavior. This allows to test software in Software-in-the-Loop and Hardware-in-the-Loop. CHROMA has also developed an Augmented Reality framework [73] for testing and validation of the perception software on the Renault Zoe experimental vehicle. This framework provides a flexible way to introduce any virtual element in real time in the data of the LiDAR sensors of the vehicle. Our Augmented Reality accurately handles all possible occlusions between real and virtual elements. The representability of test scenes generated by the augmented reality framework has been experimentally proven. It is then possible to easily and safely place the whole vehicle and all its software, spanning from perception to

control, in hybrid but realistic test scenes. This new testing methodology is intended to create a new bridge between Vehicle-in-the-Loop and real-world testing.

5.2.2 Real-World Testing Conditions

As seen in figure 134, a geo-referenced test site digital twin has been created, so as to test in simulation the validation scenarios, which were used in the augmented reality context, on the real site.

This digital twin was also used for the generation of the perception ground truth. We used Zoé localization and the digital twin geo-referencing to localize the real-world Zoé inside the digital twin with the virtual actors of the scenario. The result is an approximated ground truth, as accurate as the digital twin and the localization are.

Figure 134: Transpolis Fromentaux proving ground digital twin

5.2.3 Choice of Artificial Intelligence Algorithm

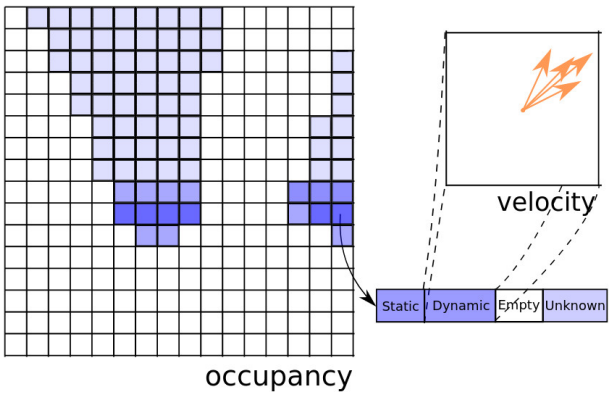


Figure 136: Inria CMCDOT framework

The CMCDOT framework [75] is a broad perception system, based on Bayesian filtering of dynamic occupancy grids (CMCDOT), allowing parallel estimation of occupancy probabilities

Figure 135: Transpolis site

for each cell of a grid, inference of velocities, collision risk prediction and dynamic object segmentation. From various heterogeneous sensor data, ground form is estimated, instantaneous occupancy grids are generated and filtered using hybrid sampling methods (classic occupancy grids for static parts, particle sets for parts dynamics), into a Bayesian unified programming formalism. Based on this perception framework, navigation systems have been developed and integrated, allowing path finding-and-following, dynamic obstacle avoidance, localization, and thus the automation of various mobile robots. Communication tools are also included, allowing data fusion from infrastructure systems. The software is composed of ROS packages, which encapsulate the optimized core system on GPU Nvidia (Cuda), allowing real-time application on embedded boards (Tegra X2). First developed in an automotive setting, it is now exploited in other areas of mobile robotics, and is particularly suited to highly dynamic and uncertain environment management. Thanks to an important engineering support over the years (notably thanks to IRT Nanoelec), this software has grown to be a core research and development tool of the team, an important technology demonstration and transfer vector, through maintained experimental platforms (most notably automated Zoe) and associated research contracts and software licensing with industrial partners.

5.2.4 Data to be Extracted

Table 17 lists the data that have been extracted for this POC and that are necessary for the analysis of the results. The Zoé software is based on ROS (Robot Operating System), therefore recorded data are composed of ROS messages and the recording is done using the tool ROSbag. ROSbag records messages without differentiating between the simulation or real-world sensors and Zoé. Simulation data are then used to generate the ground truth. More details on this subject can also be found in PRISSMA deliverable 3.3.

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 17: Topics recorded during the experiments using the tool ROSbag.

5.2.5 Methods, Procedures and Protocols for Evaluation

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 generated a corresponding ground truth. Subsequently, we used the metric recently presented in [76] to evaluate the similarity between those grids. By performing this procedure on a significant number of scenarios we can statistically evaluate the perception performance within the scenario context (i.e. crossing of an intersection in an urban area). The adopted metric is suitable for a validation process: it assesses the similarity by considering the behaviour of the AV navigating through the grids, effectively evaluating how closely driving using the perception grid aligns with driving using the ground truth.

AR replaces real actors of a scenario with virtual counterparts with the risk that these simulated actors are less realistic. We propose to evaluate the similarity between scenarios executed using augmented reality and real actors by evaluating the similarity of perception behavior. Scenarios with real obstacles can be replayed with only virtual actors by using the real actors localization from the ground-truth data, therefore, each scenario execution has its augmented reality counterpart to be compared with. This way, the scenario executions are synchronized enough to pair at each time step CMCODT occupancy grids from the scenario executions (an occupancy with real actors and an occupancy grid with virtual actors). The grids similarity is measured using the metric as CMCDOT validation. Occupancy grids from the real actors scenario can be seen as ground truth for the inferences from the scenario with virtual actors. This method evaluates the impact of using augmented reality on the validation process itself. The results should be used to assess how augmented reality can be integrated into a more general validation framework.

5.3 Implementation

5.3.1 Scenario Management

Scenario descriptions are stored in JSON files. As an example, Figure 137 shows the configuration file of an overtaking scenario. From the first to the last line, it describes: the scenario name, the reference frame used, actor names, actor models, actor initial poses, the ego vehicle path, the waypoints that trigger the start and end of the scenario using a tracked frame, and the actor trajectories. Other parameters such as weather conditions, traffic signs, and road markings are not studied in this POC, therefore they are omitted in the scenario descriptions. Also, Zoé ADS and the digital twins configurations are fixed for all the experiments and stored apart from scenario descriptions. The execution of a scenario is done by a ROS node run on Zoé onboard computer. It manages the life-cycle and controls the trajectories of the virtual actors of the scenario (i.e. it spawns and deletes the actor models in the simulator and controls their position and orientation to follow their trajectories)

5.3.2 Dataset generation and analysis

The experiments at Transpolis were concluded with the recording of 120 scenario executions. 20 executions were performed and recorded per scenario (scenarios 1 to 5 and scenario 5 with pedestrian target) totaling 850GB of data and about 1 hour of driving.

Following the methodology described in PRISSMA deliverable 3.3, we generated a dataset by extracting Zoé occupancy grids from the recordings and generating corresponding ground truth for each grid. Using metrics for occupancy grid similarity evaluation, Zoé's perception performance was evaluated for each scenario by assessing the similarity of its perception to the ground truth. More details and results on the quantitative and qualitative analyses of this POC can be found in the PRISSMA deliverable 3.6.

```
{
  "name": "overtaking",
  "ref_frame": "map",
  "actors": {
    "zoe": ["zoe/urdf/zoe_gpu.urdf", -2.5, 0, 0.35, -1.5708],
    "car1": ["hatchback_red_gps/model.sdf", 2.5, -85, 0, 1.5708
      ],
    "car2": ["hatchback_blue_gps/model.sdf", 2.5, -75, 0, 1.570
      8],
    "bus1": ["bus_gps/model.sdf", -10, -40, 0, -1.5708],
    "person1": ["person_walking_gps/model.sdf", -20, -20, 0, -1
      .5708],
    "person2": ["person_walking_gps/model.sdf", 20, -45, 0, 1.5
      708],
    "person3": ["person_walking_gps/model.sdf", 12.5, -35, 0, -
      1.5708]
  },
  "path": [[-2.5, -5, 0.35, -1.5708], [-2.5, -70, 0.35, -1.5708
    ]],
  "start_waypoint": [-2.5, -10, 0.35, 2, "zoe/base_link"],
  "trajectories": {
    "car1": [[2.5, -85, 0, 1.5708, 8.333],
             [2.5, -70, 0, 1.5708, 13.888],
             [-2.5, -55, 0, 1.5708, 13.888],
             [-2.5, -40, 0, 1.5708, 13.888],
             [2.5, -25, 0, 1.5708, 8.333],
             [2.5, -10, 0, 1.5708, 0]],
    "car2": [[2.5, -75, 0, 1.5708, 8.333],
             [2.5, -25, 0, 1.5708, 0]],
    "person1": [[-20, -20, 0, -1.5708, 2.777],
                [-17.5, -40, 0, -1.5708, 2.777],
                [-10, -60, 0, -1.5708, 0]],
    "person2": [[20, -45, 0, 1.5708, 1.333],
                [10, -25, 0, 3.1416, 0]],
    "person3": [[12.5, -35, 0, -1.5708, 1.333],
                [10, -52.5, 0, 3.1416, 0]]
  },
  "end_waypoint": [-2.5, -65, 0.35, 1, "zoe/base_link"]
}
```

Figure 137: JSON file describing an overtaking scenario

6 POC 5 : complementarities between PAVIN and simulation (CEREMA, LNE)

Testing on-board AI on vehicles requires carrying out tests. These tests can be carried out on physical, semi-simulated or fully simulated test means. In particular, the objective of the testing resources is now to be able to produce risky scenarios (vulnerable users, critical scenarios, degraded weather conditions, etc.).

The objective of the POC 5 consists to approach the simulation of fog in a real controlled environment and in a virtual environment. The system under test is a YOLO type perception system for the detection and identification of pedestrians. More specifically, this POC will have 2 sub-tasks:

- Qualify the physical testing means (PAVIN Platform, Leia) (repeatability, uncertainty, etc.)
- Compare the results obtained between the different types of test means (physical, semi-simulated, fully simulated).

This POC is cross WPs:

- WP1 Concrete applications to define an SDG, requirements and covering validation plans
- WP2 In simulation
- WP3 In a controlled environment

The PAVIN platform was used as part of a POC between LNE and CEREMA to compare the results of this platform with SIL or HIL simulation applications, and to see the complementarities between simulation and bench testing.

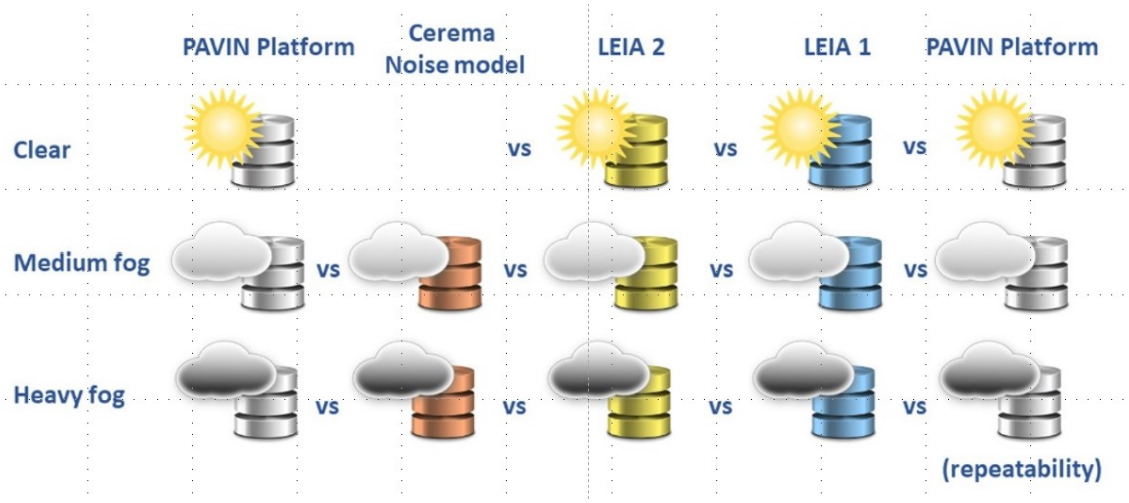


Figure 138: Set of scenarios applied and generated in the POC 5 dedicated to the pedestrian detection with YOLO (Source CEREMA and LNE).

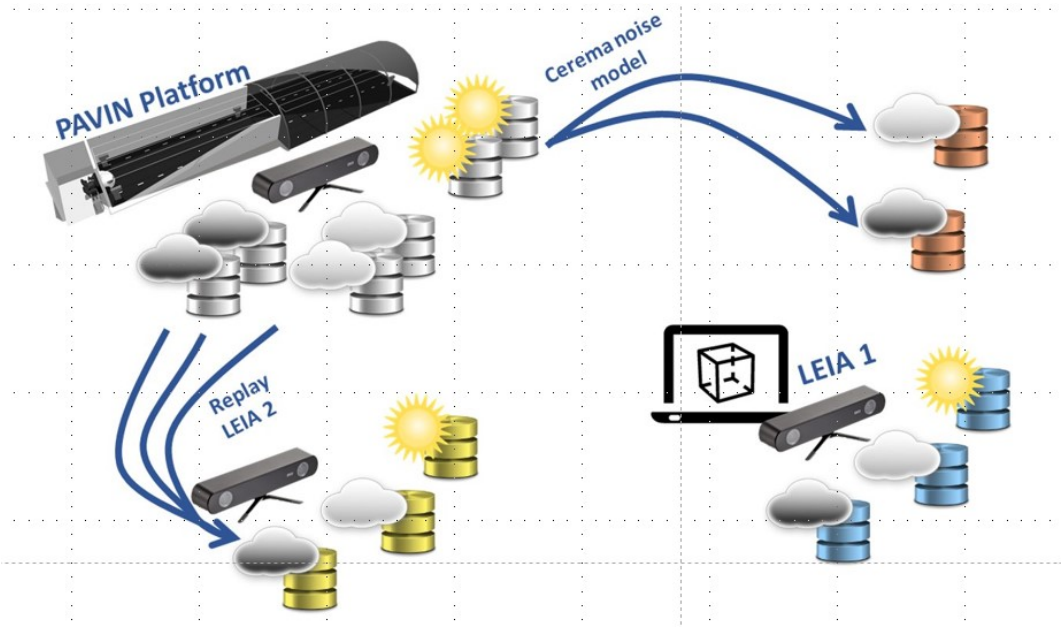


Figure 139: Experimental framework implemented in POC 5 with PAVIN facilities and LEIA simulation platform from LNE (Source CEREMA and LNE).

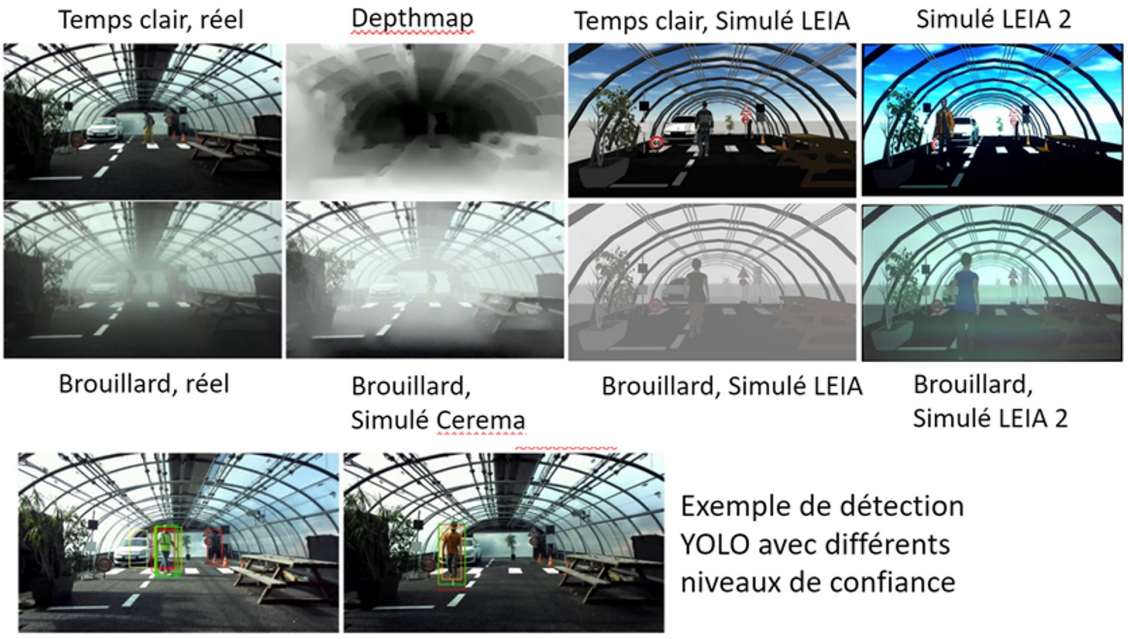


Figure 140: AI-based detection of pedestrian with with foggy conditions both in controlled environment and in simulation with a Digital Model of the PAVIN test site (Source CEREMA and LNE).

This POC was carried out as part of WP3, but straddles the line between this work package and WP2. You can find the detailed description of this POC in deliverable 3.3 and the results of the POC in PRISSMA deliverable 3.6.

Figure 141: Panel of pedestrian used in POC 5 for the AI-based detection in adverse and degraded weather conditions (foggy weather) (Source CEREMA and LNE).

7 Other systems in the simulation environment

7.1 IRT SYSTEMX

The main objective of this intermediate deliverable is to provide a view on the Proofs of Concept defined in the project and the means through which the virtual platforms used to for testing can be evaluated.

To this end, one essential brick is the process through which scenarios going to simulation or resulting from simulation can be managed and this is the purpose of the overview given in this sub-section for the MOSAR Scenario Manager.

Through the MOSAR methodology and via the MOSAR Scenario Manager, scenarios can be described by the pertinent POC teams prior to simulation. This is, describing the scenarios at a functional (i.e. situation description) and logical level (i.e. specification of testing ranges for parameters). Conversely, after performing a simulation campaign and provided that feature extraction is performed, results can be retrieved and structured into the same descriptive scenario models already implemented in MOSAR so that statistics on the results can easily be performed by, but not restricted to, the simulation team.

The advantage of structuring scenarios through such a management framework it is compatible with testing of heterogeneous nature (i.e. controlled environment, simulation or other) making comparisons simpler and better structured.

For the purposes of task 2.5 in PRISSMA, the details on the scenario management methodology are documented in [77]. For this first intermediate deliverable only the main envisaged approach is presented in this section as an overview. However, for the following iteration of the deliverable, the goal is to present how for each Proof of Concept in the project scenarios can be defined and managed through the MOSAR methodology and to illustrate how simulation results could be retrieved in the MOSAR Scenario Manager platform to then provide the means for evaluating simulation results.

In future versions of this deliverable each POC should explore the feasibility of the description of the specific scenarios in MOSAR depending on the progress made in their implementations.

7.2 CEREMA

The CEREMA's PAVIN platform is part of the CEREMA infrastructure in Clermont-Ferrand (France). It was developed to investigate the effect of adverse weather conditions on the performances of transportation systems. It can produce artificial rain with different intensities and two different types of fog, with small (radiation fog) or medium droplets (advection fog) [78].

Regarding the work carried out within task 2.3, the PAVIN platform could be used to create artificial fog and rain for a classical road scene, with known targets such as a static vehicule, traffic signs and a dummy model. Validated noise models for each type of sensor could thus be developed within the PRISSMA project (WP2-WP3) by the enhancement of existing and used models or by introducing new models if necessary.

A digital twin of the platform can be created with clear weather conditions by scanning or imaging the platform using a LiDAR and/or a stereo camera. The task will consist in adding fog and rain through numerical simulations with the information from the digital twin using noise models. A review of the different noise models from the literature is presented in the "State of the Art" deliverable from the task 2.1.

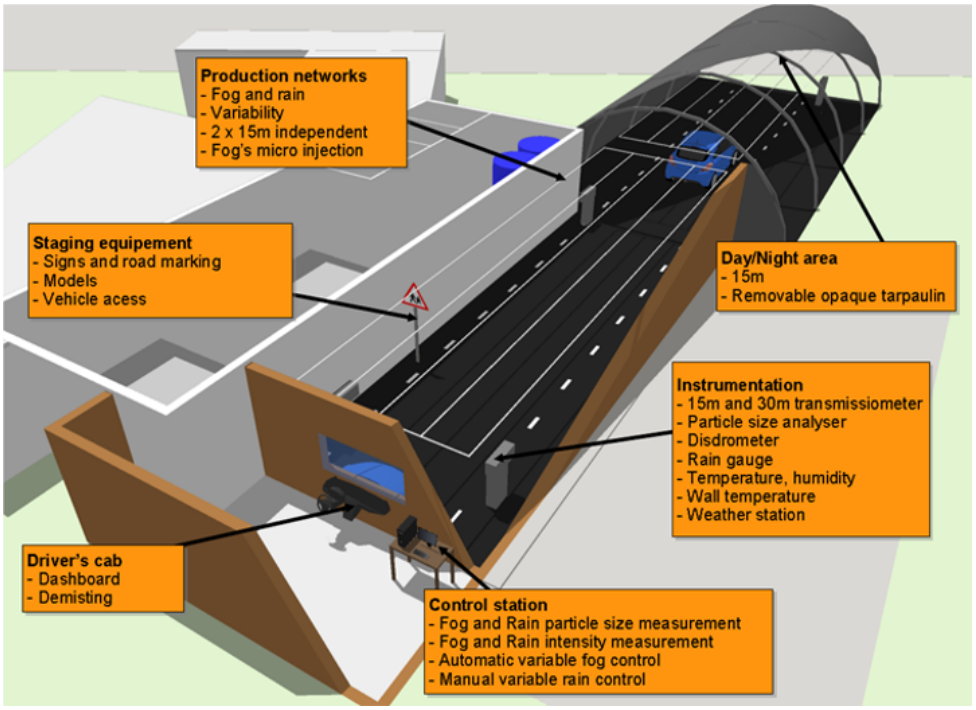


Figure 142: The Cerema PAVIN BP platform (source: Cerema)

Figure 143: Fog producing in the Cerema PAVIN BP platform (source: Cerema)

The idea is to compare the results obtained with artificial rain and fog from the platform, and the results obtained with the simulated rain and fog from the noise models. To do so, statistical methods must be defined to validate the accuracy of the noise models. It can either be using statistics on pixels characteristics or by using pedestrians detection models.

8 Final Considerations (ALL)

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] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," arXiv preprint arXiv:1905.05055, 2019.

[3] Y. Fang, X. Guo, K. Chen, Z. Zhou, and Q. Ye, "Accurate and automated detection of surface knots on sawn timbers using yolo-v5 model." BioResources, vol. 16, no. 3, 2021.

[4] ultralytics, "Yolov5," https://github.com/ultralytics/yolov5/ Accessed May 18, 2020.

[5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in 2016 IEEE international conference on image processing (ICIP). IEEE, 2016, pp. 3464–3468.

[6] Z. Qin, H. Wang, and X. Li, "Ultra fast structure-aware deep lane detection," in European Conference on Computer Vision. Springer, 2020, pp. 276–291.

[7] H. Vanholder, "Efficient inference with tensorrt," in GPU Technology Conference, vol. 1, 2016, p. 2.

[8] W. Xu, D. Gruyer, A. Duminil, and S. song Ieng, "Sivic-adversce: A framework for generating synthetic datasets for visualperception in adverse scenarios and automated driving." in Proceedings of the IEEE 14th InternationalConference on PatternRecognition Systems (ICPRS-2024), 15-18 July, 2024 (London, UK), 2024.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] A. Duminil, S.-S. Ieng, and D. Gruyer, "Quantifying fidelity in synthetic image datasets through a statistical exploration," MDPI Sensors, special Issue "Feature Papers in Intelligent Sensors 2024", vol. 2024, April 2024. [Online]. Available: https://www.mdpi.com/journal/sensors/special_issues/MZXN6Z6H2E

[14] C. blog. (2021) How openpilot works in 2021. [Online]. Available: https://blog.comma.ai/openpilot-in-2021/

[15] L. Chen, T. Tang, Z. Cai, Y. Li, P. Wu, H. Li, J. Shi, J. Yan, and Y. Qiao, "Level 2 autonomous driving on a single device: Diving into the devils of openpilot," arXiv preprint arXiv:2206.08176, 2022.

[16] 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

[17] JORF. (2021, Juin) Décret n° 2021-873 du 29 juin 2021 portant application de l'ordonnance n° 2021-443 du 14 avril 2021 relative au régime de responsabilité pénale applicable en cas de circulation d'un véhicule à délégation de conduite et à ses conditions d'utilisation. [Online]. Available: https://www.legifrance.gouv.fr/loda/id/LEGIARTI000043734793/2021-07-02/

[18] M. Revilloud, D. Gruyer, and E. Pollard, "Generator of road marking textures and associated ground truth. applied to the evaluation of road marking detection," in IEEE ITSC 2012, Anchorage, AK, USA ; 16 Sep - 19 Sep 2012, 2012.

[19] N. Hiblot, D. Gruyer, J.-S. Barreiro, and B. Monnier, "Pro-sivic and roads, a software suite for sensors simulation and virtual prototyping of adas," in Driving Simulation Conference (DSC 2010). Driving Simulation Association (DSA), 2010.

[20] J.-C. Kedzia, P. de Souza, and D. Gruyer, "Advanced radar sensors modeling for driving assistance systems testing," in 2016 10th European Conference on Antennas and Propagation (EuCAP). IEEE, 2016, pp. 1–2.

[21] S. Pechberti, D. Gruyer, and V. Vigneron, "Optimized simulation architecture for multimodal radar modeling: Application to automotive driving assistance system," in 16th International IEEE Annual Conference on Intelligent Transportation Systems (IEEE ITSC 2013), Den Hague, Holland. IEEE, September 2013.

[22] M. Hadj-Bachir and P. de Souza, "Lidar sensor simulation in adverse weather condition for driving assistance development," 2019.

[23] M. Hadj-Bachir, P. de Souza, P. Nordqvist, and N. Roy, "Modelling of lidar sensor disturbances by solid airborne particles," arXiv preprint arXiv:2105.04193, 2021.

[24] 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.

[25] S. Demmel, G. Larue, D. Gruyer, and A. Rakatonirainy, "An ieee 802.11p empirical performance model for cooperative systems applications," in 16th International IEEE Annual Conference on Intelligent Transportation Systems (IEEE ITSC 2013), Den Hague, Holland. IEEE, September 2013.

[26] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt et al., "Towards fully autonomous driving: Systems and algorithms," in 2011 IEEE intelligent vehicles symposium (IV). IEEE, 2011, pp. 163–168.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[28] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Advances in neural information processing systems, vol. 28, 2015.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision. Springer, 2016, pp. 21–37.

[32] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.

[33] ——, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[34] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[35] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in 2017 IEEE international conference on image processing (ICIP). IEEE, 2017, pp. 3645–3649.

[36] J. Illingworth and J. Kittler, "A survey of the hough transform," Computer vision, graphics, and image processing, vol. 44, no. 1, pp. 87–116, 1988.

[37] A. Borkar, M. Hayes, and M. T. Smith, "Robust lane detection and tracking with ransac and kalman filter," in 2009 16th IEEE International Conference on Image Processing (ICIP). IEEE, 2009, pp. 3261–3264.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] W. Xu, D. Gruyer, and S.-S. Ieng, "Generic simulation framework for evaluation process: Applied to ai-powered visual perception system in autonomous driving," in 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2023, pp. 5641–5648.

[43] C. N. Valdebenito Maturana, A. L. Sandoval Orozco, and L. J. García Villalba, "Exploration of metrics and datasets to assess the fidelity of images generated by generative adversarial networks," Applied Sciences, vol. 13, no. 19, p. 10637, 2023.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] A. Duminil, S.-S. Ieng, and D. Gruyer, "Assessing fidelity in synthetic datasets: A multi-criteria combination methodology," in 27th Internation Conference on Information Fusion 2024, 7th-11th July 2024, Venice, Italie, July 2024.

[51] L. Nataraj, T. M. Mohammed, S. Chandrasekaran, A. Flenner, J. H. Bappy, A. K. Roy-Chowdhury, and B. Manjunath, "Detecting gan generated fake images using co-occurrence matrices," arXiv preprint arXiv:1903.06836, 2019.

[52] M. Barni, K. Kallas, E. Nowroozi, and B. Tondi, "Cnn detection of gan-generated face images based on cross-band co-occurrences analysis," in 2020 IEEE international workshop on information forensics and security (WIFS). IEEE, 2020, pp. 1–6.

[53] I. Lelekas, N. Tomen, S. L. Pintea, and J. C. van Gemert, "Top-down networks: A coarse-to-fine reimagination of cnns," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2020, pp. 752–753.

[54] S. R. Richter, H. A. AlHaija, and V. Koltun, "Enhancing photorealism enhancement," arXiv:2105.04619, 2021.

[55] D. Gruyer, S. Pechberti, and S. Glaser, "Development of full speed range acc with sivic, a virtual platform for adas prototyping, test and evaluation," in 2013 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2013, pp. 100–105.

[56] D. Gruyer, M. Grapinet, and P. De Souza, "Modeling and validation of a new generic virtual optical sensor for adas prototyping," in 2012 IEEE Intelligent Vehicles Symposium. IEEE, 2012, pp. 969–974.

[57] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.

[58] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[59] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4340–4349.

[60] B. Li, W. Ren, D. Fu, D. Tao, D. Feng, W. Zeng, and Z. Wang, "Benchmarking single-image dehazing and beyond," IEEE Transactions on Image Processing, vol. 28, no. 1, pp. 492–505, 2019.

[61] G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[62] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[63] 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".

[64] 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.

[65] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, "Highly automated driving on highways based on legal safety," IEEE Transaction on Intelligent Transportation System, vol. 14, pp. 333–347, March 2013.

[66] D. Gruyer, I. Ben Jemma, S. Glaser, P. Desouza, J.-S. Barreiro, and S. Laverdure, "Simulation platform for the prototyping, testing, and validation of cooperative intelligent transportation systems." in 23rd IEEE ITS World Congress, Melbourne, Australia. IEEE, October 2016.

[67] C. technical documentation. (2022) Comma.ai openpilot safety. [Online]. Available: https://github.com/commaai/openpilot/blob/master/docs/SAFETY.md

[68] C. Becker, L. Yount, S. Rozen-Levy, J. Brewer et al., "Functional safety assessment of an automated lane centering system," United States. Department of Transportation. National Highway Traffic Safety . . . , Tech. Rep., 2018.

[69] C. technical documentation. (2022) Comma.ai openpilot limitations. [Online]. Available: https://github.com/commaai/openpilot/blob/master/docs/LIMITATIONS.md

[70] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in International conference on machine learning. PMLR, 2019, pp. 6105–6114.

[71] S. P. Partners. (2022) 3sa – simulation pour la sécurité des systèmes du véhicule autonome. [Online]. Available: https://www.irt-systemx.fr/projets/3sa/

[72] L. Rummelhard, J. Lussereau, J.-A. David, C. Laugier, S. Dominguez, G. Garcia, and P. Martinet, "Perception and Automation for Intelligent Mobility in Dynamic Environments," in ICRA 2017 Workshop on Robotics and Vehicular Technologies for Self-driving cars, Singapore, Singapore, Jun. 2017. [Online]. Available: https://hal.inria.fr/hal-01592566

[73] 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

[74] J.-B. Horel, C. Laugier, L. Marsso, R. Mateescu, L. Muller, A. Paigwar, A. Renzaglia, and W. Serwe, "Using Formal Conformance Testing to Generate Scenarios for Autonomous Vehicles," in DATE/ASD 2022 - Design, Automation and Test in Europe - Autonomous Systems Design. Antwerp, Belgium: IEEE, Mar. 2022. [Online]. Available: https://hal.inria.fr/hal-03516799

[75] 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

[76] 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 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), 2023, pp. 3148–3153.

[77] Q. et al, "Definition of procedures of scenario management and results analysis – initial report," PRISSMA Deliverable, Tech. Rep., 2022.

[78] P. Duthon, M. Colomb, and F. Bernardin, "Fog classification by their droplet size distributions: Application to the characterization of cerema's platform," Atmosphere, vol. 11, no. 6, 2020. [Online]. Available: https://www.mdpi.com/2073-4433/11/6/596