ReliablE in-Vehicle pErception and decisioN-making in complex environmenTal conditionS

Grant Agreement Number: 101069614

D5.1 System integration in the virtual testing setup

Not officially approved by the EC

Document Identification			
Status	Final	Due Date	31/10/2024
Version	1.0	Submission Date	23/12/2024
Related WP	WP5	Document Reference	D5.1
Related Deliverable(s)	--	Dissemination Level	PU
Lead Participant	TECN	Document Type:	Other
Contributors	All WP5 partners	Lead Authors	Leonardo Gonzalez (TECN)
		Reviewers	Michael Buchholz (UULM)
			Jerome Vermersch (APTIV)

Funded by the European Union

This project has received funding under grant agreement No 101069614. It is funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Document Information

Author(s)		
First Name	Last Name	Partner
Peter	Jasinski	APTIV
Anastasia	Bolovinou	ICCS
Emmanouil	Gkigkilinis	ICCS
Javier	Araluce	TECN
Leonardo	Gonzalez	TECN
Barys	Shyrokau	TUD
Thomas	Griebel	UULM

Document History			
Version	Date	Modified by	Modification reason
0.1	04/09/2024	TECN	Empty document with section structure
0.2	28/11/2024	UULM, TECN, TUD, ICCS, APTIV	First input by partners
0.3	04/12/2024	TECN	Complete first draft of deliverable with all sections
0.4	11/12/2024	TECN	Internal review by TECN
0.5	16/12/2024	APTIV, ICCS, UULM	Second input by partners
0.6	16/12/2024	TECN	Integrating all corrections, first version ready for peer review
0.7	17/12/2024	UULM	Review received from UULM
0.8	18/12/2024	APTIV	Review received from APTIV
0.9	19/12/2024	TECN	Integration and final corrections
1.0	20/12/2024	ICCS	Final review ready for submission

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Leonardo Gonzalez (TECN)	19/12/2024
Quality manager	Panagiotis Lytrivis (ICCS)	20/12/2024
Project Coordinator	Angelos Amditis (ICCS)	20/12/2024

Not officially approved by the EC

©EVENTS Consortium 2022-2025

Executive Summary

This document presents the outcomes of Task T5.1 within Work Package 5 (WP5) of the EVENTS project, focusing on integrating automated driving components into virtual environments. The task addressed diverse scenarios, including urban navigation, highway manoeuvres, vulnerable road user (VRU) interactions, and adverse weather conditions. High-fidelity simulation technologies—open-source, commercial, and tailored solutions—were leveraged to replicate real-world complexities and test developments from WP3 (Perception and V2X Communication) and WP4 (Decision-Making and Control). Progressive integration enabled iterative testing loops, ensuring continuous evaluation and refinement of perception systems, control strategies, and decision-making algorithms. Special attention was given to challenges like multi-agent coordination, VRU safety, and dynamic obstacle avoidance through technologies such as model predictive control, machine learning-based prediction, and sensor integration. The virtual environments established in T5.1 provide a scalable, safe, and cost-effective testing foundation, supporting hybrid testing and real-world deployment in the next phases of WP5 and advancing autonomous driving system capabilities.

Not officially approved by the EC

©EVENTS Consortium 2022-2025

Table of Contents

Executive Summary **3**

1. Introduction **9**

1.1 Project aim 9

1.2 Report scope 9

1.3 Structure of the report 10

2. Overall integration Process **11**

2.1 Simulation Environments 12

2.2 Key Differences and Suitability for Automated Driving 13

2.3 List of Experiment Videos 15

3. Integration in simulation environments **16**

3.1 Experiment 1 16

3.1.1 Architecture and Design Considerations 16

3.1.2 Step-by-Step Integration Process 17

3.1.3 Key Challenges and Solutions 18

3.2 Experiment 2 24

3.2.1 Architecture and Design Considerations 24

3.2.2 Step-by-Step Integration Process 29

3.2.3 Key Challenges and Solutions 38

3.3 Experiment 3 40

3.3.1 Architecture and Design Considerations 40

3.3.2 Step-by-Step Integration Process 42

3.3.3 Key Challenges and Solutions 43

3.4 Experiment 6 44

3.4.1 Architecture and Design Considerations 44

3.4.2 Step-by-Step Integration Process 46

3.4.3 Key Challanges and Solutions 48

3.5 Experiment 7 49

3.5.1 Architecture and Design Considerations 49

3.5.2 Step-by-Step Integration Process 51

3.5.3 Key Challenges and Solutions 52

Not officially approved by the EC

3.6 Experiment 8 53
3.6.1 Architecture and Design Considerations 53
3.6.2 Step-by-Step Integration Process 55
3.6.3 Key Challenges and Solutions 57
4. Conclusions 59
References 61

List of Tables

Table 1 Key components and tools per experiment 13
Table 2 List of videos per experiment 15
Table 3 Scenario with randomly spawned pedestrians over 25 experiments 21

List of Figures

Figure 1 Simulation results for Autoware planner. 20
Figure 2 Simulation results for Local Model Predictive Control. 20
Figure 3 Simulation results for Topology-Driven MPC with fallback strategy 21
Figure 4 Trajectories of the four methods in one case with two pedestrians 22
Figure 5 Velocity profile of the four methods for the same scenario. 23
Figure 6 Simulated custom map of Tecnalia's test track. 26
Figure 7 Tecnalia's Facilities 27
Figure 8 Nvidia docker container toolkit. [21] 28
Figure 9 Experiment 2 architecture 30
Figure 10: One vehicle control ROS architecture. 32
Figure 11: health check of a node. 34
Figure 12: Portainer architecture for one vehicle. 35
Figure 13 CARLA environment bird-eye-view snapshot. 36
Figure 14 Virtual testing pipeline a) offline and b) online testing modes 36
Figure 15: Overall architecture of EXP3 modified from D2.2 [37]. 40
Figure 16: Simulation environment for Experiment 3 based on a software-in-the-loop framework [7]. 42
Figure 17 EXP6 Scenario definition 44
Figure 18 Initial Virtual Simulation System architecture 44
Figure 19 Integration of over-drivable OSI objects 45
Figure 20 Final integration architecture 46
Figure 21 Example of implemented debris objects 47
Figure 22 Virtual training Setup of Joint Prediction and Planning system 50
Figure 23 NuPlan Framework High-Level software description where green color denotes modules that had to be adapted by ICCS 51

Not officially approved by the EC

Figure 24 Collision avoidance scenario with two obstacles. 53
Figure 25 An instant of the obstacle avoidance manoeuvre. 55
Figure 26 IPG CarMaker integration into Matlab/Simulink 56
Figure 27 Simulation architecture in Matlab/Simulink. 57
Figure 28 Vehicle trajectories, with the baseline approach represented in red, which results in a collision, and the proposed approach depicted in blue, successfully avoiding the obstacles. 57
Figure 29 The control inputs, specifically the commanded longitudinal force (left) and the road wheel angle (right). 58

Not officially approved by the EC

Abbreviations & Acronyms

Abbreviation / acronym	Description
ACC	Adaptive Cruise Control
AD(F)	Autonomous Driving (Function)
AI	Artificial Intelligence
AL	Alert Limit
API	Application Programming Interface
AV	Automated Vehicle
BP	Behavioural Planner
CA	Consortium Agreement
CAM	Cooperative Awareness Message
CAV	Connected Automated Vehicle
CPM	Collective Perception Messages
DDT	Dynamic Driving Task
DENM	Decentralized Environmental Notification Message
DM	Decision Making
EC	European Commission
EXPs	Experiments
FIS	Fuzzy Inference System
FoV	Field of View
FTP	Fail-degraded Trajectory Planning
GA	Grant Agreement
IR	Integrity Risk
ISO	International Organization for Standardization
I/O	Input(s) / Output(s)
LiDAR	Light Detection and Ranging
MDP	Markov Decision Process
MPC	Model Predictive Control
MRM	Minimum Risk Manoeuvre

Not officially approved by the EC

©EVENTS Consortium 2022-2025

Abbreviation / acronym	Description
MOP	Moving Object Prediction
MOT	Multi-Object Tracking
ODD	Operational Design Domain
OSI	Open Simulation Interface
PE	Position Error
PL	Protection Level
PP	Perception Platform
RADAR	RAdio Detecting And Ranging
REQs	Requirements
RL	Reinforcement Learning
SAE	Society of Automotive Engineers
SMD	Safety-mode Decision
SoTA	State of the art
SPaT message	Signal Phase and Timing message
SPECs	Specifications
TOR	Take Over Request
TP	Trajectory Planner
TSs	Target Scenarios
UCs	Use Cases
VRU	Vulnerable Road User
WP	Work Package

Not officially approved by the EC

©EVENTS Consortium 2022-2025

1. Introduction

This document follows the structure of all the other deliverables, namely, an introduction with the aims of project and scope of the document; after that, the main text, with the specific topic of the document.

1.1 Project aim

Driving is a challenging task. In our everyday life as drivers, we are facing unexpected situations we need to handle in a safe and efficient way. The same is valid for Connected and Automated Vehicles (CAVs), which also need to handle these situations, to a certain extent, depending on their automation level. The higher the automation level is, the higher the expectations for the system to cope with these situations are.

In the context of this project, these unexpected situations, where the normal operation of the CAV is close to be disrupted (e.g., ODD limit is reached due to traffic changes, harsh weather/light conditions, imperfect data, sensor/communication failures, etc.), are called "events". EVENTS is also the acronym of this project.

Today, CAVs are facing several challenges (e.g., perception in complex urban environments, Vulnerable Road Users (VRUs) detection, perception in adverse weather and low visibility conditions) that should be overcome to be able to drive through these events in a safe and reliable way.

Within our scope and to cover a wide area of scenarios, these kinds of events are clustered under three main use cases: a) Interaction with VRUs, b) Non-Standard and Unstructured Road Conditions and c) Low Visibility and Adverse Weather Conditions.

Our vision in EVENTS is to create a robust and self-resilient perception and decision-making system for AVs to manage different kind of "events" on the horizon. These events result in reaching the CAV ODD limitations due to the dynamic changing road environment (VRUs, obstacles) and/or due to imperfect data (e.g., sensor and communication failures). The CAV should continue and operate safely no matter what. When the system cannot handle the situation, an improved minimum risk manoeuvre should be put in place.

1.2 Report scope

This report focuses on the outcomes of Task T5.1 *SW integration in simulation environments*, within the EVENTS project. Task T5.1 addresses the integration of perception, decision-making, and control components developed in previous work packages, into virtual environments to evaluate their performance under diverse and challenging scenarios. These scenarios include interactions with Vulnerable Road

Not officially approved by the EC

Users (VRUs), non-standard and unstructured road conditions, and adverse weather or low-visibility conditions.

The work presented in this document highlights the progressive integration of developments from WP3 (Perception and V2X Communication) and WP4 (Decision-Making and Control) for each of the experiments in the project [1] . The virtual environments used encompass open-source, commercial, and tailored solutions, which provide controlled and scalable testing grounds for automated driving systems. The report also identifies the key challenges addressed during the integration process and the role of virtual environments in preparing for subsequent hybrid and real-world testing phases.

This document excludes detailed analysis of hybrid environments and real-world testing tasks, as these are covered under separate future deliverable *D5.2 System integration in the prototype vehicles*. Additionally, experiments focused solely on hybrid or real-world scenarios, such as EXP4 and EXP5, are not included within the scope of this report.

1.3 Structure of the report

The document is structured as follows:

- Section 1: Introduction provides an overview of the project aims, the scope of this report, and its structure.
- Section 2: Overall Integration Process summarizes the systematic approach taken to integrate automated driving components into virtual environments. This section also introduces the simulation technologies used and their relevance to the experiments.
- Section 3: Integration in Simulation Environments presents the detailed integration process for each experiment in scope (EXP1, EXP2, EXP3, EXP6, EXP7, and EXP8). Each subsection describes the design considerations, step-by-step integration processes, challenges encountered, and key outcomes.
- Section 4: Conclusions summarizes the achievements of Task T5.1, emphasizing the role of virtual environments in supporting the development and evaluation of automated driving systems.

Not officially approved by the EC

2. Overall integration Process

The overall integration process describes the systematic approach adopted for integrating automated driving components into virtual environments across all experiments involved in the project. The objective was to ensure consistency, modularity, and scalability, enabling a robust framework for testing advanced perception, control, and decision-making systems. By leveraging established simulation platforms such as CARLA and IPG CarMaker, alongside tailored in-house solutions like those used by Ulm University in EXP3 and APTIV in EXP6, virtual environments were developed to meet each experiment's specific objectives. This integration work provides a critical foundation for subsequent real-world deployments and hybrid testing to be reported in *D5.2 System integration in the prototype vehicles.*

The integration process encompassed the following key aspects:

1. **Virtual Simulation Frameworks**: High-fidelity simulation platforms were utilized to model diverse and challenging driving scenarios. Tools such as CARLA, IPG CarMaker, and Autoware enabled the replication of urban navigation, highway maneuvers, and adverse weather conditions. Scenario-specific solutions, like Ulm University's internal simulation setup for EXP3, allowed for focused and flexible integration tailored to the experiment needs.
2. **Scalability and Containerization**: Modular architectures based on Docker were implemented in experiments such as EXP2 and EXP7. These setups provided scalability for running multi-agent and multi-sensor systems in parallel while supporting hybrid virtual-physical environments. By isolating components such as V2X modules, tracking systems, and motion planners into separate containers, experiments ensured efficient deployment, parallel execution, and streamlined troubleshooting.
3. **Integration of Machine Learning and Optimization**: Machine learning components and optimization tools from WP3 and WP4 were integrated to enable advanced perception and decision-making functionalities. For example, ML-based predictive models were incorporated in EXP7 for highway joint prediction and planning tasks using PyTorch, while real-time optimization of controllers was achieved in EXP8 using ForcesPro for nonlinear MPC. These integrations supported the development of efficient and reliable control strategies in simulated environments.
4. **Performance Monitoring and Health Checks**: Automated performance monitoring and health checks were implemented in experiments such as EXP2 to manage the execution of complex multi-agent systems. Metrics such as task duration, collision events, and computation times were logged to assess the system's behavior within the virtual environments. Tools like ROS2 and

Not officially approved by the EC

Scenario Runner provided interfaces for automating and monitoring simulation performance.

5. **Real-World Compatibility Focus**: Experiments were designed with an emphasis on integration into virtual environments while maintaining outputs compatible with real-world systems. For instance, real sensor models (LiDAR, radar, cameras) and standardized message formats such as collective perception messages (CPMs) were integrated into simulations. Platforms like the CARLA-ROS bridge ensured seamless data exchange between virtual agents and real-world components, facilitating hybrid testing planned in T5.2.

The integration work achieved in this task (T5.1) ensures that components developed in WP3 (Perception and V2X Communication) and WP4 (Decision-Making and Control) were successfully implemented in virtual environments. By adopting standardized simulation frameworks, scalable architectures, and modularized designs, the project delivered robust experimental setups across multiple use cases. These virtual environments are a crucial step towards enabling comprehensive hybrid and real-world testing in subsequent project tasks.

2.1 Simulation Environments

The simulation environments used in the EVENTS project played a key role in ensuring consistent and reliable integration of automated driving components. Each environment was carefully selected or tailored to match the requirements of specific experiments, tooling varied for each experiments, ranging from full on frameworks for automated driving like Autoware, to automated driving simulators, such as CARLA.

1. **Autoware** [2]: Used in EXP1, Autoware provides a modular open-source software stack based on the ROS2 environment. It allows a seamless integration of motion planning and perception components in urban driving environments.
2. **CARLA** [3]: Used in EXP2, provided a highly configurable and realistic simulation platform for testing urban navigation, and perception systems. Its ability to generate dynamic traffic scenarios and integrate with ROS2 made it ideal for multi-agent and V2X applications.
3. **IPG CarMaker** [4]: used in EXP8, due to its high-fidelity vehicle dynamics environment which made it suitable for the research work performed on nonlinear control tasks like collision avoidance on slippery roads. The combination of advanced tire models and real-time simulation made it the simulator chosen for testing vehicle control strategies under adverse conditions.
4. **Nuplan simulator** [5]: To be used in EXP7, for conducting closed loop (re-active agents) simulations in urban and highway driving scenarios with ML-powered

Not officially approved by the EC

ego and non-ego driving models with predictive and planning capabilitites.Note: This is a new tool to be adopted by ICCS team for ongoing/future work that was taken over in the context of WP4 based on the second project amendement (approval is pending at the time of writing).

5. **dSPACE ASM** [6]**:** Used in EXP6, because it provides the environment simulation model, which includes the streets, weather conditions and vehicle dynamics of the host vehicle. Furthermore it provides the OSI groundtruth data which is used as input for the APTIV RADAR Sensor Model.
6. **In-House Solutions** [7]: Ulm University deployed a tailored simulation environment for **EXP3** to integrate modules such as self-assessment systems and V2X.

The Table 1 below summarizes the key components and tools integrated within each experiment:

Experiment	Perception	Control	Decision-Making	Virtual Environment Tools
EXP1	VRU detection	Topology-Driven MPC (T-MPC++)	Urban Navigation Planning	Autoware [2], ROS2 [8]
EXP2	V2X Perception (CPMs) and self-asssesment	Platooning Control	Coordinated Roundabout Navigation	CARLA [3], ROS2 [8], Autoware [2]
EXP3	Self-Assessment, V2X Fusion	-	Behavioral Decision-Making	In-house simulator [7]
EXP6	Radar Sensor Models for small objects	-	-	dSPACE ASM [6]
EXP7	ML-Based Prediction	Joint Prediction and Planning	Highway Behavioral Planning	nuPlan Simulator [5]
EXP8	Obstacle detection on adverse weather	Nonlinear MPC, Delft Tyre model	Collision Avoidance in adverse weather	IPG CarMaker [4]

Table 1 Key components and tools per experiment

2.2 Key Differences and Suitability for Automated Driving

The selection of virtual environments in the EVENTS project addressed a variety of automated driving scenarios, including urban navigation, highway maneuvers, and adverse weather conditions. Each platform offered unique advantages aligned with

Not officially approved by the EC

the specific needs of the experiments, ensuring the successful integration of perception, control, and decision-making components.

CARLA was used for its flexibility and configurability in urban driving environments. Its support for sensor models such as LiDAR and cameras, combined with integration with ROS2, alongside the benefits of being open-source with an active community, made it ideal for integrating perception developments. In EXP2, CARLA enabled the simulation of coordinated platooning in a roundabout, supporting V2X-based communication and vehicle control. Other virtual environments were also leveraged, specifically "at-home" developments such as Ulm University's simulation platform in EXP3. This environment was used to integrate self-assessment modules and V2X-based collective perception systems, focusing on improving perception reliability. Similarly, APTIV's radar-centric virtual environment in EXP6 allowed precise radar perception testing with targeted capabilities for radar-focused applications.

The commercial tool IPG CarMaker was used in EXP8 for its high-fidelity vehicle dynamics modeling, particularly for scenarios requiring precise simulation of nonlinear tire-road interactions. The platform's advanced tire models and ability to represent real-world vehicle dynamics made it suitable for testing control strategies under adverse conditions, such as reduced road friction. This, alongside integration with the ForcesPro solver, allowed real-time optimization of nonlinear MPC algorithms, enabling accurate simulations of evasive maneuvers on slippery roads caused by heavy rainfall.

Autoware, an open-source framework for automated driving, was also leveraged. Employed in EXP1, it served as a modular simulation platform specifically for urban navigation and motion planning. It provided a complete software stack integrating perception, decision-making, and control components. Its modular design allowed for the exchange of project-specific developments while maintaining a consistent base architecture. Autoware's compatibility with HD maps and simplified vehicle dynamics models made it suitable for replicating real-world urban environments, ensuring consistency in testing motion planners and perception pipelines.

The combination of platforms such as CARLA, IPG CarMaker, and Autoware, along with tailored virtual environments and benchmarks like nuPlan, ensured comprehensive coverage of the project's technical requirements. This integration provided a robust foundation for testing automated driving components in virtual environments, supporting the transition to hybrid and real-world testing in subsequent project tasks.

Not officially approved by the EC

©EVENTS Consortium 2022-2025

2.3 List of Experiment Videos

The following table summarizes the video links for all experiments conducted in T5.1, showcasing the integration of automated driving components in virtual environments. While it is not necessary, not we as a consortium are obliged per the Grant Agreement, to provide demonstration videos of the software integration in the simulations environments, we have deemed it both useful and illustrative to provide such videos on certain experiments. It was not though feasible to provide a video for each of the experiments, either due to technical or proprietary reasons. The latter reason is even more valid since this deliverable is a public document.

Experiment	Description	Video Link
EXP1	Interaction with VRUs in complex urban environment	EXP1 Video
EXP2	Re-establish platoon formation after splitting due to roundabout platooning in a roundabout	EXP2 Video
EXP3	Self-assessment and reliability of perception data with complementary V2X data in complex urban environments	EXP3 Video
EXP8	Driving minor road under adverse weather conditions including perception self-assessment	EXP8 Video

Table 2 List of videos per experiment

Not officially approved by the EC

©EVENTS Consortium 2022-2025

3. Integration in simulation environments

This section describes the integration of automated driving components into virtual environments for each experiment conducted in Task T5.1. The experiments address diverse scenarios, including urban navigation, highway maneuvers, vulnerable road user (VRU) interactions, and control strategies under adverse weather conditions. Each experiment leverages specific virtual environments, either open-source, commercial, or tailored, to test and refine the developments from WP3 and WP4 in a controlled, repeatable manner.

It is important to note that EXP4 and EXP5 are planned for hybrid environments and rely on real-world data, which falls outside the scope of the virtual environment integration focus of Task T5.1. Thus, this chapter focusses on the remaining experiments.

3.1 Experiment 1

The motion planner developed in this experiment is designed to navigate urban environments while accounting for the presence of Vulnerable Road Users (VRUs). The following section outlines the virtual test environment used to evaluate the performance of the planner against several baseline approaches in scenarios involving pedestrian interactions.

3.1.1 Architecture and Design Considerations

The key requirement for the simulation of EXP1 is to minimize the gap from simulation to reality and to assess the motion planner with decision-making in real-time. The developed approach is integrated with Autoware software [9]. The Autoware software provides a baseline planner and can incorporate the real-world environment, for this reason, the Autoware Planning Simulation has been used. We adapted this simulation environment to match as closely as possible the real-world experimental settings, using our own HD map collected on the real test track and adapted vehicle parameters to match those of the used test vehicle.

With the developed approach integrated in Autoware, our software is compatible with the existing sensing, perception and control pipelines available in Autoware.

The simulation environment is defined as an urban environment consisting of a two-lane road without clear lanes, and where pedestrians are frequently crossing. In this setting, the developed motion planner with decision-making needs to drive safely but assertively to progress along the road as a human would do when driving in such an environment.

Not officially approved by the EC

3.1.2 Step-by-Step Integration Process

As mentioned previously, our simulation environment builds on top of the Autoware Planning Simulation. We first adapted the simulation to our real-world environment as follows:

- **Vehicle**: The vehicle model parameters are configured to match the TUD test platform based on the Toyota Prius. The dynamics are simplified but already model delays and dynamic limits on velocity, acceleration, etc.
- **Map:** The HD map of our real-world test location is loaded into the simulation. This provides LaneLet [2] annotations describing the road centreline and boundaries. By using this map, our simulation environment closely matches the real-world environment.
- **Objects:** Autoware provides functionality to spawn dummy pedestrians, including a point cloud and camera images that can be detected by the perception pipeline. Spawning functionality is limited, however, for automated testing.

Beyond configuring the basic Autoware simulation environment with our scenario, we further improved the environment with more realistic pedestrian behaviour:

- **Timing:** We implemented a GPS triggered gate that starts pedestrian motion when the vehicle drives through. This enables us to start virtual pedestrians with the same timing in the simulation as in the real-world.
- **Pedestrians:** Our pedestrian simulation was enhanced with several features.
    - **Output:** Pedestrians are simulated in a separate simulator. The output of the simulator is converted to Autoware dummies to simulate point clouds and the interaction of the perception stack with the planner.
    - **Model:** Pedestrians use the social forces model [10]. This model has interaction, causing pedestrians to evade each other when they are close together, resulting in more realistic pedestrian motion.
    - **Uncertainty:** We simulate a Gaussian noise distribution on top of the social forces model to simulate the uncertainty associated with the uncertain, non-deterministic motion of pedestrians.
    - **Randomized Scenarios:** We spawn pedestrians in a random area annotated on top of the HD map. Pedestrian spawn and goal locations are different for each scenario but consistent between the different planning methods.

Not officially approved by the EC

- **Automated Testing:** The simulation environment is extended for automated testing. First, the vehicle is spawned in a start location on the HD map. Second, a goal position is passed along the road and the motion planner with decision-making is activated. When the goal is reached, data relating to the experiment is saved, and the vehicle is respawned at the start location. Simulations are repeated up to a configured number.

- **Analysis:** Each simulation generates a file with metrics, including:
  - Task duration, s
  - Collisions (if any), [-]
  - Timeouts (when the vehicle did not reach the goal), [-]
  - Average velocity [m/s]
  - Minimum distance to pedestrians, m
  - Computation time, s
  - Positions of the vehicle and objects, m

Replacing and adapting the Autoware component

Since our proposed motion planner with decision-making replaces the Autoware planning component, we adapted the sensor and actuator ROS2 topics to receive and send data to the software stack. On the output side, the Autoware control component expects a trajectory with a history to ensure the smoothness of the vehicle's behaviour. The code for maintaining the history was adapted from the Autoware planning stack leading to smooth planning in simulation.

A major adaptation was made to convert the route planned by Autoware to the format of the developed motion planner with decision-making. The route consists of a sequence of lanelets including their boundaries. Specialized software was developed to model the road centreline and its boundaries as continuous cubic splines that can be handled by the Model Predictive Controller (MPC) described in D4.2 [11]in detail.

3.1.3 Key Challenges and Solutions

The following describes the virtual tests of our developed motion planner with decision-making compared to existing baselines.

Baselines

We used two primary baselines. The first baseline is the Autoware standard planner, tuned and configured to enable dynamic collision avoidance. This stack relies on several components:

- First, the dynamic collision avoidance component in the behavioural planner [12] implements rule-based dynamic collision avoidance. It projects laterally the motion plan away from a collision point, steering to avoid a collision. This

Not officially approved by the EC

planner has a known limitation that it can only steer a limited amount to avoid pedestrians and is not incorporated by default in the Autoware software stack.

- If the avoidance planner cannot evade pedestrians, a braking component takes over control. This module simply waits for the pedestrians to pass before continuing to drive. The two behavioural planner components can conflict and lead to collisions.

- The plan of the behavioural layer, which may or may not be avoiding collisions, is passed to a motion planning component that refines the plan. In particular, it uses a convex MPC with collision avoidance constraints to satisfy the vehicle dynamic constraints.

The Autoware planner is known to be conservative and is not designed to deal with multiple dynamic obstacles but is available open source.

The second baseline is Local Model Predictive Contouring Control (LMPCC) [13]. LMPCC uses MPC to optimize a single trajectory. It was introduced to navigate around static and dynamic obstacles. The objectives and constraints are equal to those of our proposed planner.

Simulation results

We compare these two baselines against two variants of our proposed algorithm Topology-Driven MPC (T-MPC++):

- **T-MPC++**, optimizes multiple distinct trajectories in parallel.

- **Fallback-enhanced T-MPC++**, also optimizes trajectories that do not pass obstacles to have a safe fallback option available.

All MPC-based planners use the same cost and safety constraints. T-MPC++ adds constraints to maintain the passing behaviour of trajectories locally.

The simulated environment considers scenarios with 0, 2 and 4 pedestrians to compare how safe and efficient the planners are when evading dynamic objects. Simulations were run on a laptop with an Intel i9 CPU @ 2.4 GHz 16-core CPU. The snapshots of the baselines and the proposed approach in the simulation are shown in Figure 1, Figure 2 and Figure 3. Figure 1 is related to Autoware planner using the dynamic collision avoidance and obstacle stop planner. The plan shown with green (fast) – blue (slow) colours plans to steer and brake to pass the obstacles. Figure 2 demonstrates the simulation results for Local Model Predictive Control optimizing a single trajectory. In this case, the planned trajectory got stuck in poor behaviour, failing to pass the obstacles efficiently. Figure 3 shows the results for Topology-Driven MPC with fallback strategy optimizes several trajectories in distinct passing behaviors

Not officially approved by the EC

(visualized with light green, dark blue and red lines). Even in crowded environments, the planner finds high-quality trajectories.

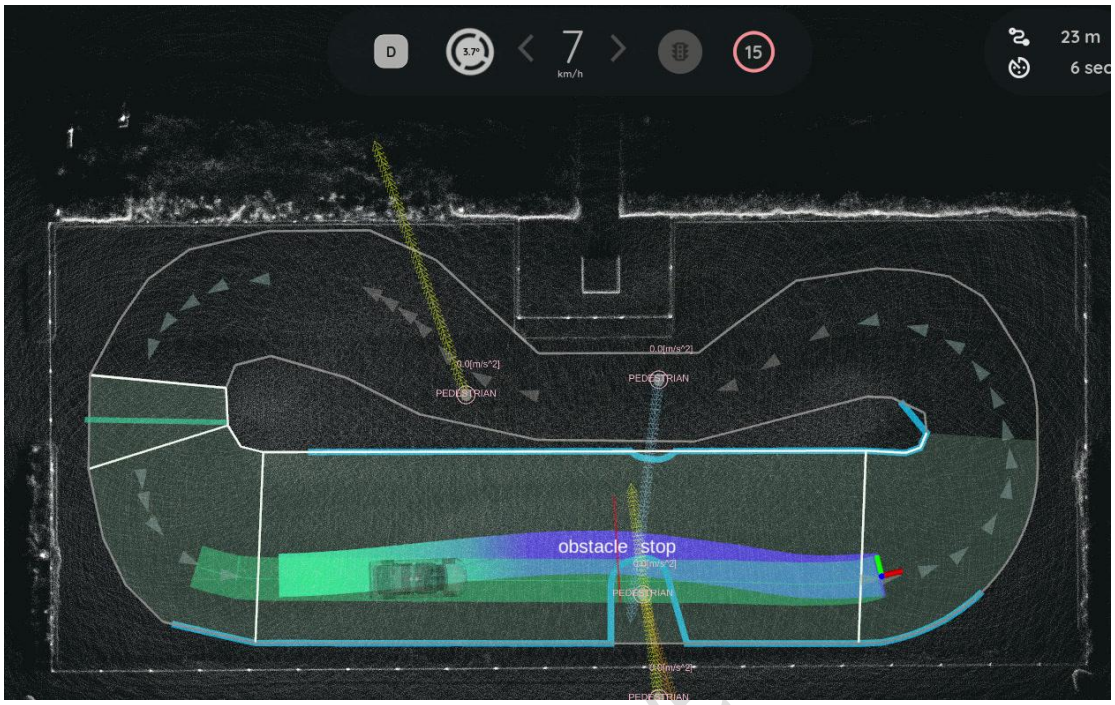

Figure 1 Simulation results for Autoware planner.

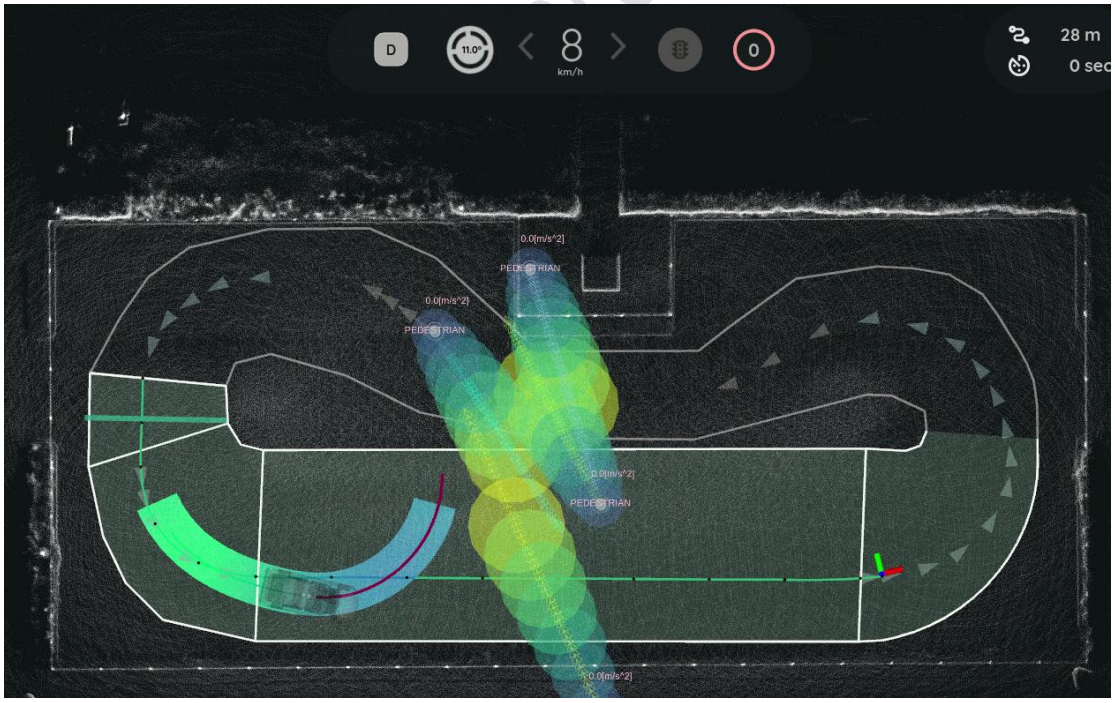

Figure 2 Simulation results for Local Model Predictive Control.

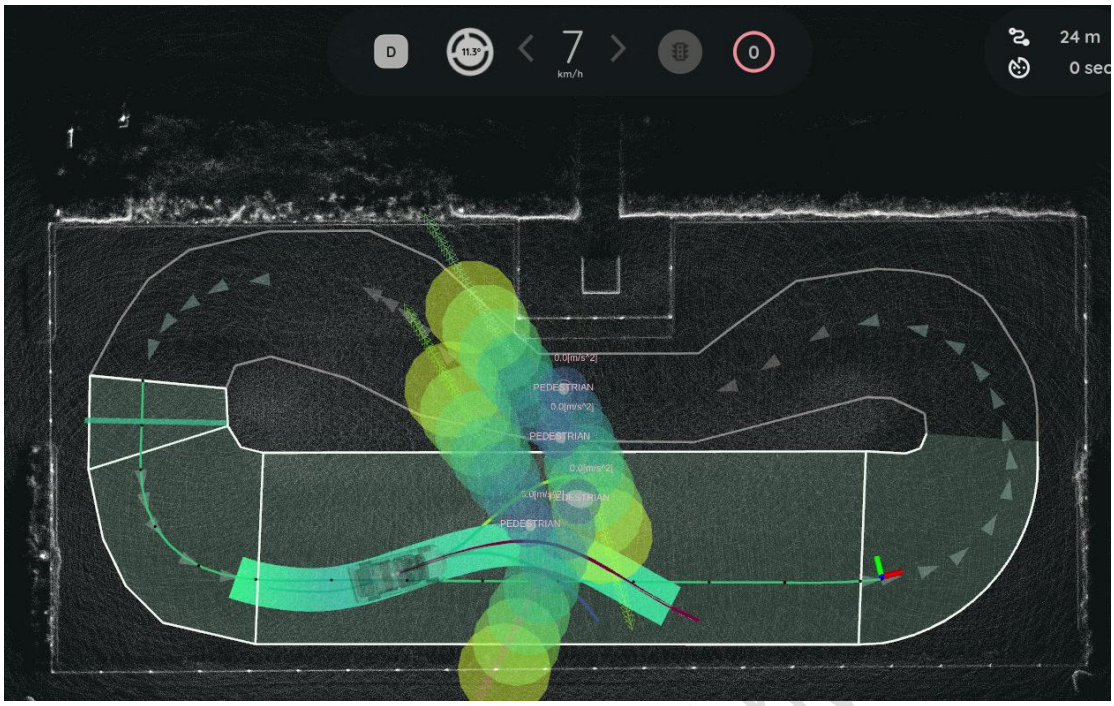

Figure 3 Simulation results for Topology-Driven MPC with fallback strategy

Pedestrians	Method	Duration[s]	Collisions	Timeouts	Avg. Velocity [m/s]
0	Autoware	19.5 (0.1)	0	0	1.87
	T-MPC++	19.4 (0.3)	0	0	1.86
2	Autoware	26.0 (6.0)	4	0	1.45
	LMPCC	20.9 (1.4)	1	0	1.75
	T-MPC++ (w/o fallback)	21.0 (1.1)	1	0	1.74
	T-MPC++	20.1 (1.2)	1	0	1.83
4	Autoware	28.4 (4.4)	5	0	1.31
	LMPCC	24.0 (3.2)	0	0	1.57
	T-MPC++ (w/o fallback)	24.2 (4.8)	3	1	1.56
	T-MPC++	21.0 (1.7)	0	0	1.78

Table 3 Scenario with randomly spawned pedestrians over 25 experiments.

The quantitative results are summarised in Table 3. The following key performance indicators are reported: task duration [mean (std)], collisions, timeouts (vehicle did not reach the goal in time) and average velocity. The developed algorithm T-MPC++, with a fallback strategy, outperforms the other methods in task duration without compromising on safety.

Not officially approved by

The Autoware baseline is the slowest planner and often collides. Autoware's collision avoidance planner is experimental and does not guarantee collision avoidance. Furthermore, two motion plans in Autoware (collision avoidance and braking) are operated concurrently. When the collision avoidance module cannot guarantee overtaking of the obstacle, the braking planner is activated too late, leading to a collision. LMPCC (i.e., single trajectory MPC) performs better, it does not have collisions and is driving faster on average. Performance is still suboptimal, as it optimizes a single trajectory that may be far from the globally optimal solution. This is clear from the results of T-MPC++, particularly looking at the increase in task duration with an increasing number of obstacles. With two obstacles, LMPCC takes 1.6s longer, while T-MPC++ takes only 0.4s longer. T-MPC++, therefore, passes the obstacles more efficiently. Without fallback, the planner has to take emergency action (i.e., perform hard braking) when the nominal planner fails. With fallback, an alternative is available that may not require heavy braking and, as a result, can result in faster navigation.

Trajectories of the four methods in one scenario with two pedestrians are shown in Figure 4 . Trajectories of the ego-vehicle and obstacles are shown in red and green, respectively, with increased transparency over time. The Autoware planner brakes for the obstacles, notice that it does not steer and stays stationary. LMPCC finds a poor local optimum and also does not steer to pass the pedestrians. T-MPC++ does steer to evade the pedestrians from behind, completing the task faster (visible in the less transparent trajectory around x=20). T-MPC++ with fallback strategy performs the same manoeuvre, but its safety is improved (demonstrated by less collisions in Table 3).

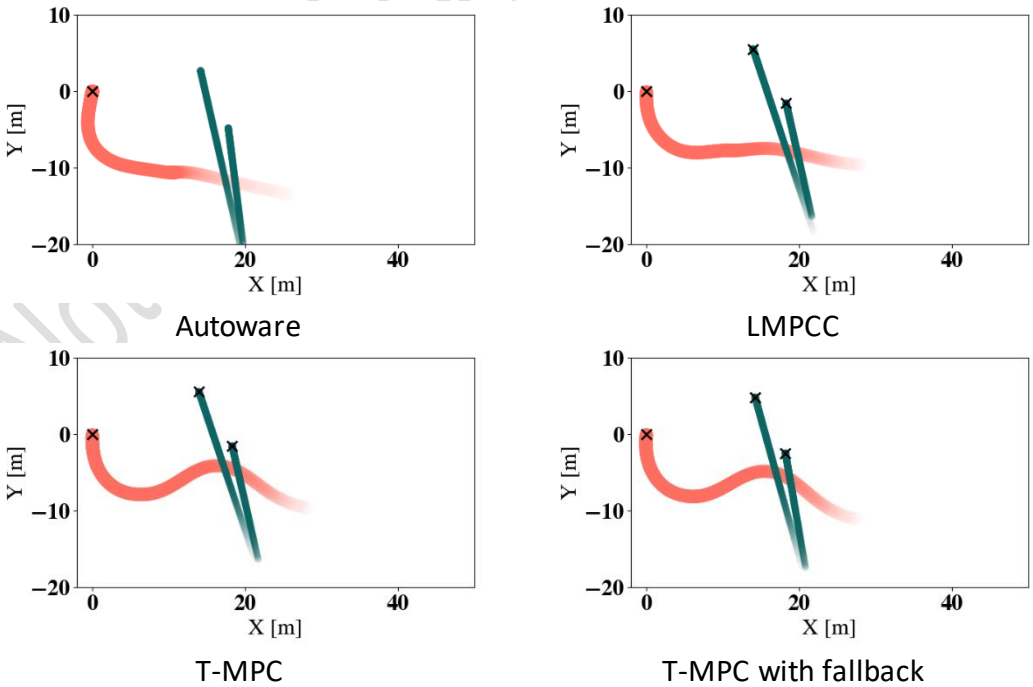

Figure 4 Trajectories of the four methods in one case with two pedestrians.

Further comparison of the methods is provided Figure 5, where the velocity profiles planned by the methods are compared for the same scenario. The Autoware planner brakes for the obstacles, leading to slow navigation. LMPCC finds a slow trajectory but keeps moving. T-MPC++ with and without a fallback strategy achieve similar performance, evading the obstacles around the reference velocity. All trajectories lead to a smooth velocity profile. In Figure 5, the reference velocity is denoted by the black dashed line. “tmpcnf” denotes T-MPC++ without fallback strategy, “tmpc” denotes T-MPC++.

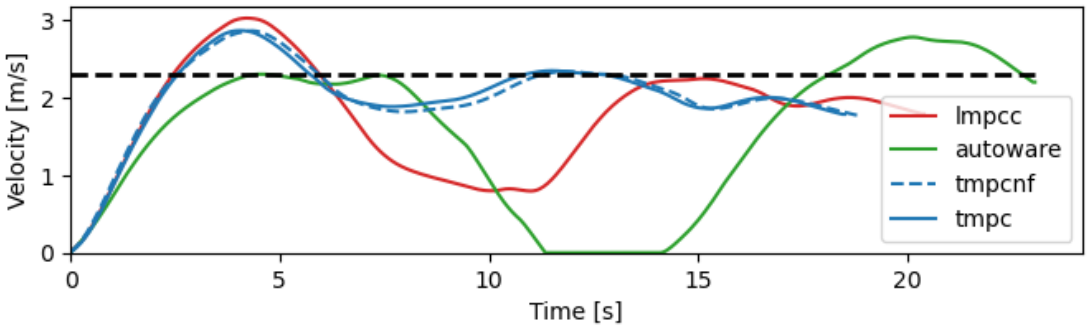

Figure 5 Velocity profile of the four methods for the same scenario.

Not officially approved

3.2 Experiment 2

The objective of Experiment 2 is to evaluate the behaviour of an autonomous vehicle platoon as it navigates a roundabout, specifically focusing on maintaining platoon cohesion despite intermittent interruptions. The experiment consists of three CAVs entering and exiting a roundabout in a way that allows the platoon to remain together, even if one or more vehicles are temporarily required to yield to external traffic, leading to brief separation within the platoon.

The experiment will be conducted in a hybrid environment using a real vehicle and CARLA simulator, which enables precise control and observation of the vehicles' behaviour. Additionally, the perception system on each vehicle includes detection, tracking, and motion prediction modules, which will work in unison to ensure that each AV can autonomously sense, interpret, and respond to its surroundings. The perception system runs on the simulator because the adversarial agents are only simulated. The experiment is designed to assess the platoon's ability to manage temporary disruptions while maintaining coordinated movement through the roundabout.

Experiment 2 represents a collaborative effort between two teams, Tecnalia and ICCS, an onsite and offsite integration respectively for virtual environments. The contributions have been coordinated and structured to reflect the expertise and focus of each team. This section details the integration of their respective developments into a cohesive virtual environment, highlighting the combined advancements in platooning control and V2X-based communication.

3.2.1 Architecture and Design Considerations

Simulation integration requires some specific considerations: the simulation environment chosen, which allows us to create a scenario and a custom map based on our real test track to replicate the experiment. In addition, the experiment has many components that need to run simultaneously, so we need to isolate the processes. Due to the high computational demands of this system, a High-Performance Computing (HPC) system is required to run the simulation and automate three vehicles. Finally, we chose the Robot Operating System (ROS) for the communication between the nodes.

Simulation environment

Automated vehicle simulators have become an essential tool in the development, testing and validation of autonomous driving systems. These simulators provide virtual environments, where developers can evaluate vehicle behaviour, test algorithms, and experiment with different driving scenarios without risking real-world consequences. They enable the simulation of different driving conditions, such as

Not officially approved by the EC

varying weather, lighting conditions and terrain, which are critical for comprehensive testing of autonomous systems.

CARLA (Car Learning to Act) [3] stands out as an ideal choice for EXP2 due to several reasons:

- CARLA provides highly realistic environments, with detailed textures, dynamic weather, and lighting effects. This realism is critical when simulating complex driving scenarios, where visual cues, sensor accuracy, and environmental details can significantly affect vehicle behaviour. It is particularly useful when we want to analyse complex events that do not occur during normal driving.
- CARLA provides an extensive API that allows for fine-grained control over the simulation. It allows developers to manage all the behaviours that occur during the simulation to force critical situations, as was done in EXP2.
- Developers can create custom maps, control individual sensors, simulate traffic, and configure vehicle dynamics. This flexibility is essential for testing a wide variety of conditions and vehicle behaviours. Custom maps are particularly useful when we want to test specific roads or environments, such as EXP2, where Tecnalia's test track has been recreated in a simulation environment.
- CARLA supports a comprehensive range of sensors that can be used in autonomous vehicles, including LiDAR, GPS, IMUs, radar, and cameras. These sensors are configurable in terms of position, orientation, and sampling rates, allowing for accurate and realistic simulation of perception systems.
- CARLA integrates well with other simulation and development platforms, especially ROS. This compatibility makes it easier to incorporate CARLA into larger development pipelines and collaborate with other tools commonly used in autonomous vehicle research. Compatibility with ROS allows our architecture to be used without any special function to transform the parameters and transfer them to a real platform. ROS is the framework used by ICCS and Tecnalia to communicate developments that work together in EXP2.

In summary, CARLA's combination of high realism, configurability, extensive sensor support, and accessibility makes it an excellent choice for developing and testing autonomous vehicles in simulated environments. Its robust community and continuous development further ensure that CARLA remains a powerful and versatile platform for advancing autonomous driving technology.

Not officially approved by the EC

Scenario

To manage the proposed scenario in EXP2, we used the Scenario Runner package developed by CARLA's developers, which is specifically designed to facilitate the execution of complex, scripted traffic interactions like the one presented in our experiment. Scenario Runner supports multiple formats, allowing for standardized and reproducible testing; in this case, we chose for the OpenSCENARIO [14] format to standardize our scenario description.

Custom map

The CARLA simulator's flexibility in scenario and map customisation allowed us to develop a tailored map (Figure 6) replicating the Tecnalia's facilities (Figure 7), which can be seen in more detail in the EXP2 video. This customised environment ensures that we can test the experiment within a highly accurate, relevant setting. Moreover, it allows for hybrid simulations that integrate real sensor data and vehicle inputs, which will be developed in the context of T5.2, bridging the gap between virtual and physical testing for enhanced experimental fidelity.

A fountain has been added to the custom map in the middle of the roundabout to make sure there are occlusions to challenge the perception system.

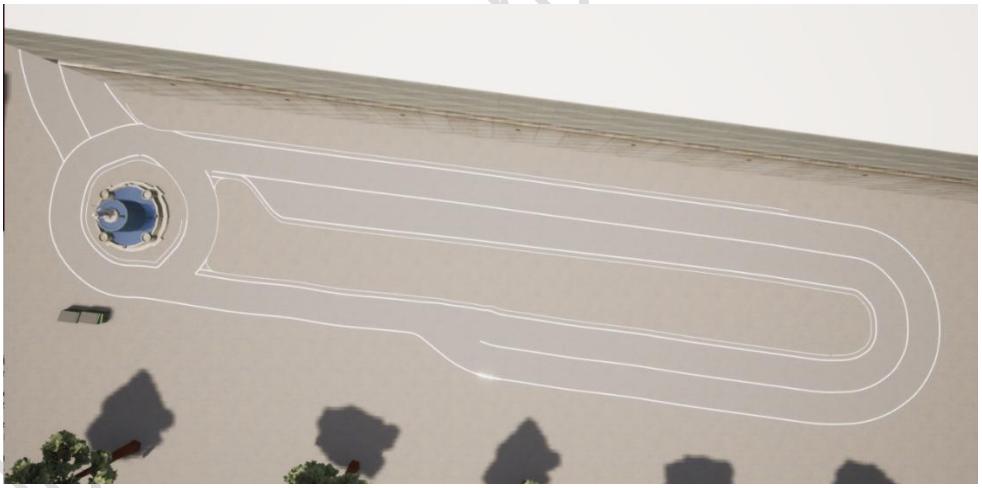

Figure 6 Simulated custom map of Tecnalia's test track.

©EVENTS Consortium 2022-2025

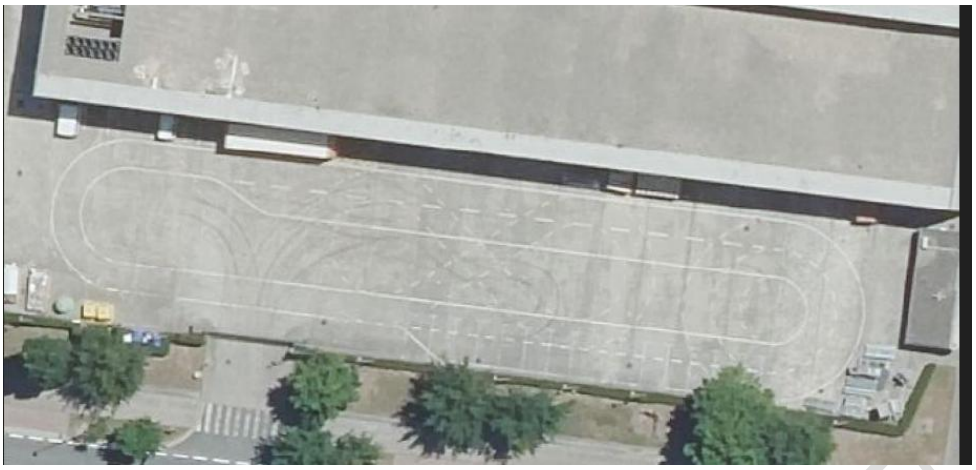

Figure 7 Tecnalia's Facilities.

The creation of the custom CARLA map began with lane design, accomplished using Mathwork's RoadRunner tool [15] . Leveraging OpenStreetMap data and orthophotos by PNOA [16] (Spanish National Aerial Topography Plan), we generated a georeferenced OpenDRIVE [17] file to accurately represent the road network. Building models were created in Blender, using additional OpenStreetMap data to ensure structural accuracy, before being exported alongside the RoadRunner-generated files into Unreal Engine. Finally, we selected a pre-existing static fountain asset available within the CARLA project to complete the environment, integrating it seamlessly with the custom-built components.

Isolation and containerization

The entire architecture has been designed leveraging the versatility of Docker [18] containers, allowing for efficient modularization and isolation of each component. By compartmentalizing all elements of the system within separate containers, we maximize the benefits of isolation, ensuring that each module (such as perception, control, and communication) runs independently without interference from others. This approach not only enhances the stability and security of the system but also simplifies deployment, scaling, and troubleshooting, as each container can be managed, updated, or replaced individually. Docker's containerized environment thus provides a robust foundation for developing, testing, and running the platoon's architecture in a controlled and reproducible manner.

We have built two main images, the first one runs CARLA Simulator and the control architecture, which has its own dependencies. And the second one oversees the perception suite, which needs a specific CUDA environment [19] and some deep learning libraries that have their own requirements.

Both images take advantage of Nvidia Docker [20] container (Figure 8). NVIDIA Docker, or more commonly referred to as NVIDIA Container Toolkit, is a tool that

Not officially approved by the EC

allows Docker containers to use NVIDIA GPUs for accelerated computing. This setup is especially useful for applications that require GPU support, like machine learning, data science, and autonomous driving simulations. Standard Docker containers do not have native GPU support, so the NVIDIA Container Toolkit is essential to bridge this gap by enabling GPU resources within containers.

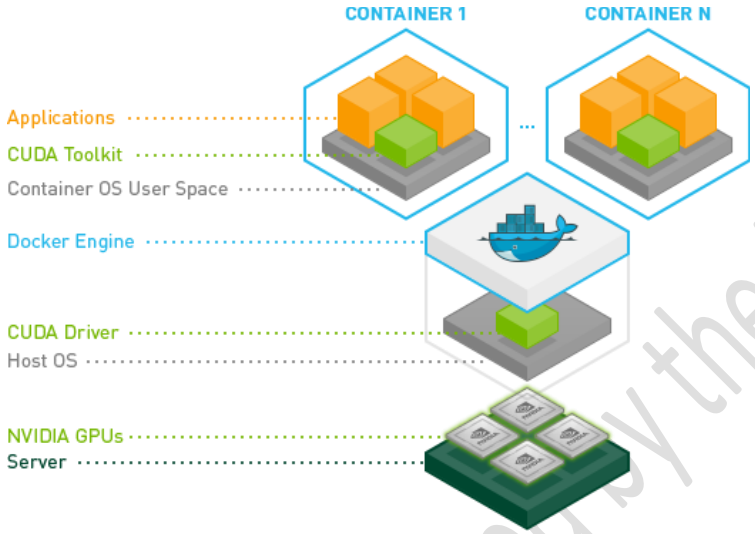

Figure 8 Nvidia docker container toolkit. [21]

High-Performance Computing (HPC)

Due to the computational requirements of EXP2, we have leveraged our computing units with an server grade HPC. The system in place has 64 threads of AMD EPYC 9124 16-Core Processor CPU at 3.7 GHz, 768 GB of RAM at 4800 MHz and an Nvidia L40 with 46 GB of VRAM.

Developers connect to the server using ssh protocol [22] and can use the display interface thanks to a remote desktop. We have created a network system that allows more than one developer to connect at the same time, each with their own display, to use the simulator without disturbing other colleagues.

To achieve this milestone, we are using Docker's ability to isolate networks to be able to use the same Docker image and code without having to change all the CARLA network ports (server, traffic manager, ...). Each user has their own network, which is automatically created when they deploy all the containers.

Robot Operating System (ROS)

Our automated vehicle architecture is based on ROS2, because it allows the workflow to be divided into nodes that can work in parallel. The communication between the nodes is done through topics, which contain all the information that needs to be transferred to the next segments of the architecture. These topics can contain raw data (images, point clouds, ...) or processed information (detections, control

Not officially approved by the EC

commands, ...). We use ROS2 Humble [8], which has an end of life (EOL) date of May 2027, allowing us to use this architecture for several years with security updates.

Additionally, we integrated the CARLA ROS bridge [23], which includes services for reloading and controlling scenarios directly through ROS. This integration ensures seamless interaction with the CARLA environment, enabling efficient Software-in-the-Loop (SiL) testing and rapid scenario reloading. By leveraging this setup, we achieved an optimal testing environment where the proposed scenario could be consistently recreated, supporting robust analysis, and testing of platoon behaviour under varying traffic conditions in a roundabout.

3.2.2 Step-by-Step Integration Process

Onsite Integration

The following section details the steps/subsystems that make up EXP2. We have based our experimental integration on two Docker images (CARLA Image and Perception Image), which form the basis of various containers that launch some nodes.

Figure 9 shows the complete architecture of this proposal, where eighteen containers are launched. Three are responsible for the CARLA server (CARLA World, Scenario Manager and Scenario). These are followed by five containers for each vehicle (control, detection, tracking, motion prediction and data collector).

Not officially approved by the EC

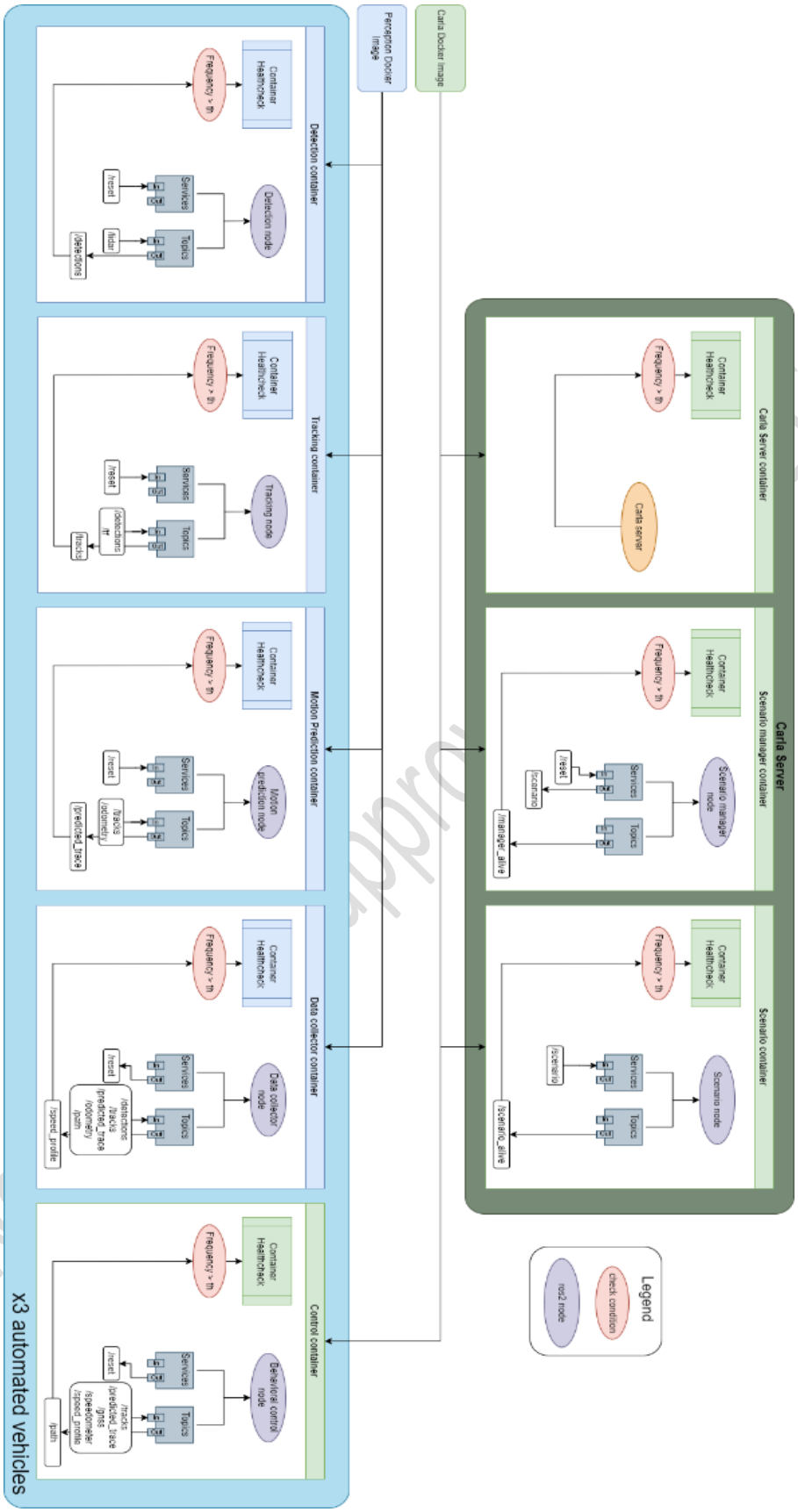

Figure 9 Experiment 2 architecture.

CARLA Image

We have created a custom image that met all the simulation requirements. We installed the CARLA binary distribution using a custom one based on CARLA 0.9.14 and Python 3.10 [24]. This last requirement is due to the fact that we need Ubuntu 22.04 [25] for ROS2 Humble and it uses Python 3.10 natively. When building the image, we need to share the user id of the host to run and modify the code, which is hosted in a shared volume.

When we run the container, we share some environment variables that are needed to run the simulation. We share $DISPLAY, $ROS_DOMAIN_ID and the $EVENTS_PATH, the last variable being the shared path where all the code is allocated. We also share some volumes and files where the code is stored, and the custom network built for the user.

One vehicle control

Each ego vehicle needs a total of 8 nodes to coordinate the control:

- **Global planner:** There are two modes integrated for global path generation, relevant to this experiment. If the vehicle is on the leader position the global path is generated using Dijkstra algorithm [26] on the CARLA map information for an initial and goal position. If it is not, the global path updates with the position of the leader vehicle with a frequency of 20Hz.
- **Behavioural:** This node manages Platoon Management Messages (PMM) and Platoon Control Messages (PCM) to determine, first, if the ego vehicle is in a platoon and second, if it is, which is its position.
- **Localization:** This node acts as a parser of the GNSS sensor to the relative position inside the map. Additionally, it composes the state of the vehicle by reading IMU and Speedometer data form the CARLA ROS bridge.
- **Trajectory generator:** the trajectory of the vehicle is generated using information from the global planner node, as well as the information of the surrounding vehicles in the platoon. Interaction with obstacle is added through the communication with the perception stack.
- **Longitudinal control:** This is the low-level control that generates the throttle and brake values based on the speed error.
- **Lateral control:** This node reads the local path created in the trajectory generator node as a reference to calculate the steer value.

Not officially approved by the EC

- **Control Bridge:** This node parses the control values to feed the CARLA ROS bridge.
- **Communication**: The communication node, in this case acts as a virtual communication node, where virtual Cooperative Awareness Messages (CAM), Decentralized Environmental Notification Messages (DENM), PCM and PMM messages are sent, received and parsed into manageable information for the other nodes.

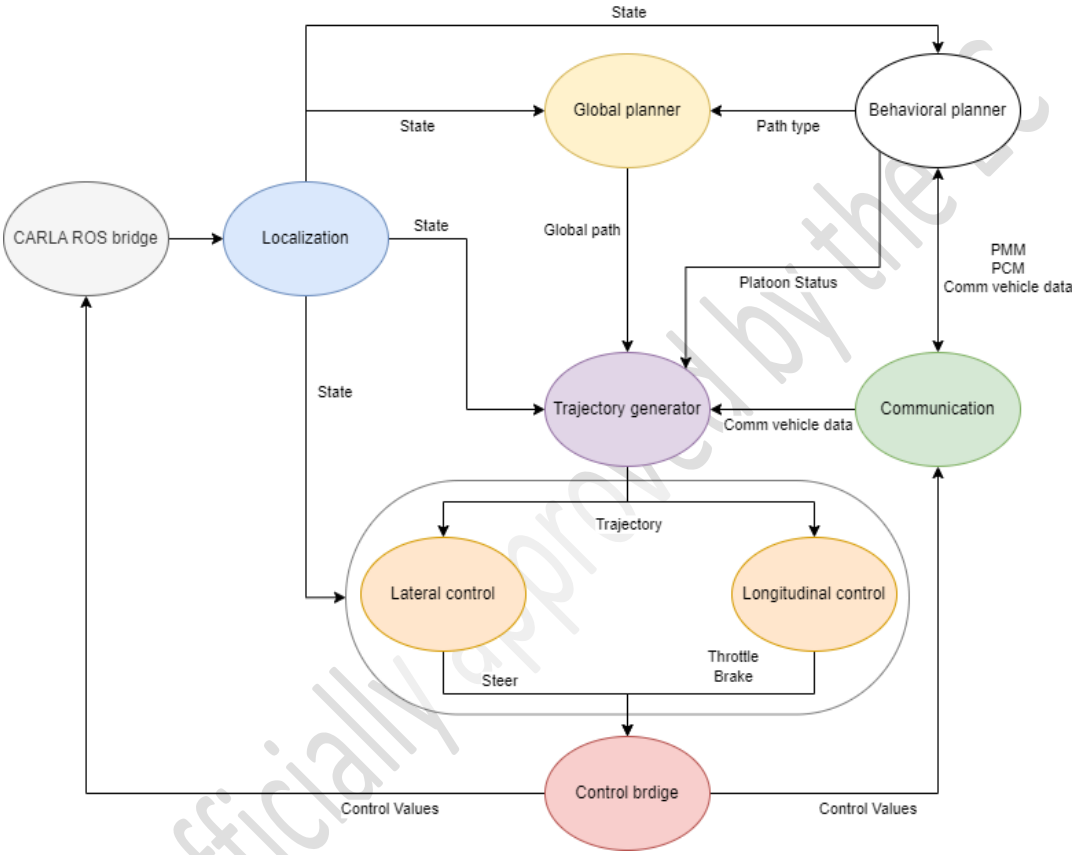

Figure 10: One vehicle control ROS architecture.

Perception image

The perception image requires some libraries to run the deep learning models for detection and motion prediction. We have combined both requirements into one docker image to use the same for both tasks. It is based on an Nvidia image with CUDA 12.0 and cudnn 8 with Ubuntu 22 for ROS2 Humble.

One vehicle perception

Each vehicle has its own perception suite, which consists of three main nodes, all built from the perception image. The first is responsible for detection and has LiDAR input from the vehicle. The second performs tracking from the environment and requires the detections from the previous node and the global transformation. Finally, the last

Not officially approved by the EC

node is responsible for motion prediction, which was developed in the EVENTS project. It needs the tracking objects, the position of the ego-vehicle and the global transformation.

Scenario

The OpenScenario [14] file has been developed using QGIS [27]with a plugin that let us connect to a CARLA server, load the OpenDrive [17] information from the active map and customize the scenario with the available vehicles or pedestrians. Moreover, the plugin allows for vehicle behaviour customization, which contributes to the complexity of the scenarios created.

In the specific case of EXP2, the scenario counts with three ego vehicles set in a line before the roundabout and 2 other vehicles that act as obstacles, driving at a constant speed around the roundabout guided by an array of waypoints.

Reinforcement Learning integration

Integration with the reinforcement algorithm required a special setup, as training this tool requires many steps to find the solution.

First, the information is fed into the algorithm via ROS2, which synchronises all the data from different sources.

With all this data synchronised, we faced the problem that the whole setup would not be able to run during long runs. So, we prepared some health checks for each container and some global checks to restart the simulation.

Container HealthChecks

As mentioned earlier, each container has its own health check to assess the performance of the node, and if there is an error or bug, the node is restarted.

We will detail the health check for each node:

- CARLA Server: check the world frame rate to assess the node. If this frame rate is below a threshold, the node will be restarted.
- Scenario: a variable is published to indicate that the node is alive. If this variable is not published with a frequency greater than a threshold, the node will be restarted. The CARLA server is also checked to restart the node, because if the server is restarted, the scenario node should also be restarted to connect the two environments.
- Scenario manager: a variable is published to indicate that the node is alive. If this variable is not published with a frequency greater than a threshold, the

Not officially approved by the EC

node will be restarted. The CARLA server is also checked to restart the node, because if the server is restarted, the scenario manager node should also be restarted to connect the two environments.

- Ego-vehicle control: check the world frame rate to assess the node. If this frame rate is below a threshold, the node will be restarted.
- Detection: the output topic must have a frequency above a certain threshold, otherwise the container will be restarted.
- Tracking: the output topic must have a frequency above a certain threshold, otherwise the container will be restarted.
- Motion Prediction: the output topic must have a frequency above a certain threshold, otherwise the container will be restarted.

Figure 11 shows an example of a health check of the motion prediction container. This behaviour can also be seen in the EXP2 video.

In addition to these considerations, there is a global variable that checks that the whole system is running correctly; if this variable is true, the whole system is restarted to continue training.

Restarting the nodes is made by a community docker called autoheal [28].

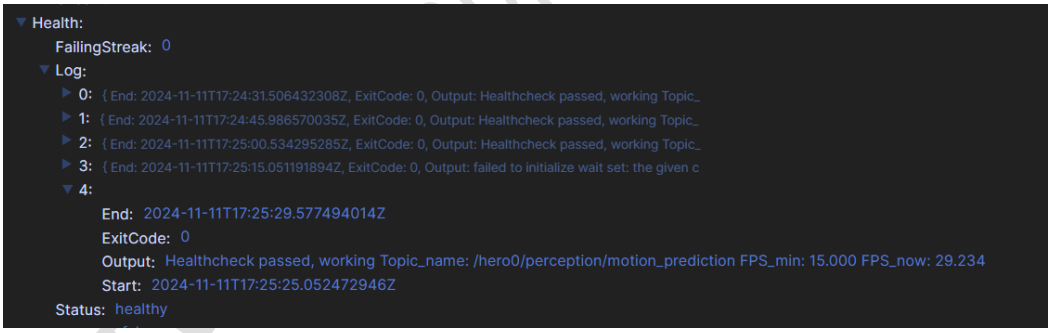

Figure 11: health check of a node.

With this system, we can check the logs of the containers to know the actual status of each one. For this purpose, we use a tool called Portainer, which is a container management tool (Figure 12). This tool is demonstrated in the EXP2 video, which

Not approved by the EC

shows full orchestration.

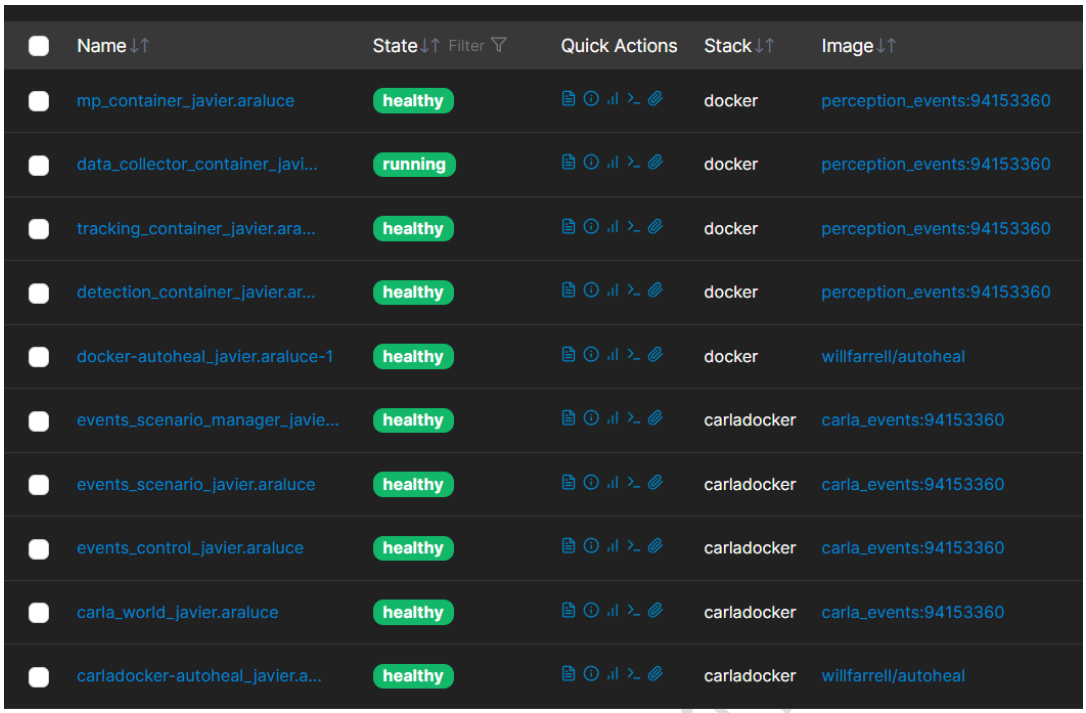

Figure 12: Portainer architecture for one vehicle.

Three vehicles integration

To extend this architecture to three vehicles, we need to triple the following nodes: ego-vehicle control, detection, tracking, motion prediction and data collection.

We then share the information from each agent with the others to take advantage of the V2X perception system.

Offsite Integration

As described in the section above, an additional virtual deployment has been setup to aid in the tasks related to V2X communication of the a coordinated platooning, more specifically based on Collective Perception (CP) information exchange among the agents forming the platoon. For virtual testing of the CP module, an additional simulation environment has been setup in CARLA (v.0.9.14)[1] to support the following core functionality:

- Custom scenario editing in CARLA;
- Simulation and sensor data recording mechanism for multiple agents: CARLA simulation data (groundtruth), sensor raw data, sensor data as rosbags ;
- Data replay in CARLA;

[1] https://carla.org/2022/12/23/release-0.9.14/

Not officially approved

- Interfacing the CP module with CARLA for real time experiments via CARLA-ROS bridge;
- Support for hybrid testing via real-time communication of a real agent in ICCS test track with EVENTS simulation environment.

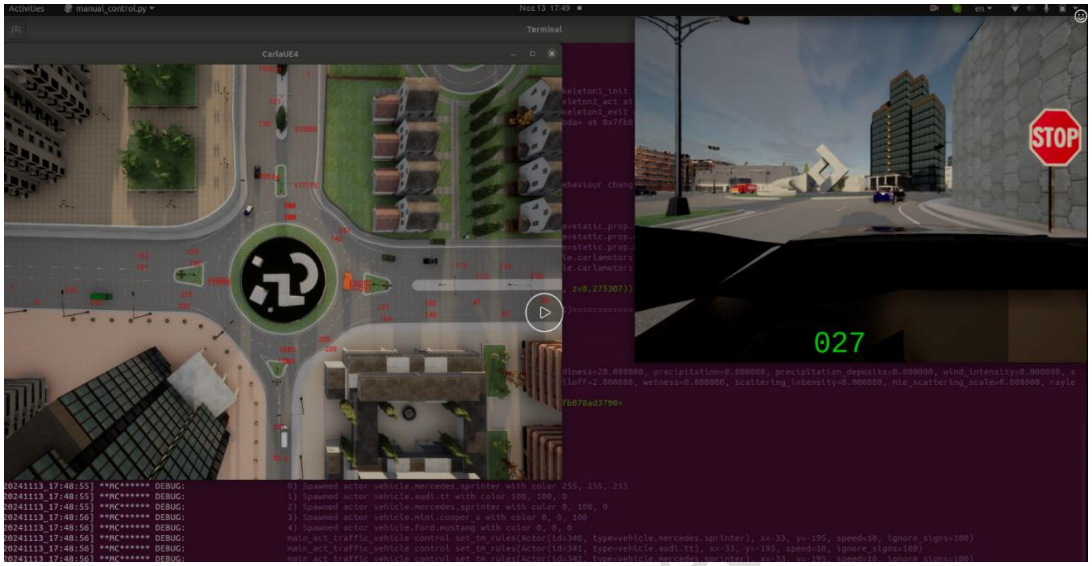

Figure 13 CARLA environment bird-eye-view snapshot.

In Figure 13, a roundabout scenario with occlusions (buildings, other vehicles, roundabout static) is executed with agents capable of generating sensor data: on the left window a snapshot of the bird's eye view roundabout scenario is depicted; on the right window, the RGB camera output of the black vehicle approaching from the north is depicted – other agents' camera output can also be depicted (our setup accomodates up to 7 agents visualized simultaneously).

Two testing modes, one offline and one online, are supported by the ICCS virtual testing pipeline as shown in Figure 14 below.

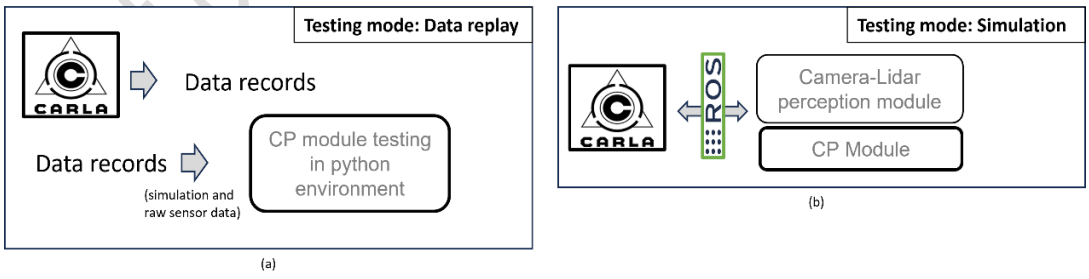

Figure 14 Virtual testing pipeline a) offline and b) online testing modes.

Notes: Main CP testing is via data replay; CPMs are encoded in a custom format similar to ETSI CPM to be consumed by CP python module.

©EVENTS Consortium 2022-2025

Data replay support (offline testing)

Simulation data replay happens via CARLA replaying system[2], while sensor data replay is supported either via Rosbags or via raw sensor data recordings. Those recordings feed the CP module developed in WP3 for testing purposes.

CARLA environment setup

***CARLA custom scenario editor*:** A python-based library was created to simplify scenario editing in CARLA with the requirement to support scenario creation for perception layer testing, where one ego is configured through initial position and route (set of waypoints) and all the other agents follow their pre-set route and drive relative to that ego (i.e., no collisions occur and speeds are relevant to the speed of the ego). Spawning of objects (static or dyamic) anywhere in the scene is easily supported by CARLA. In our setup, apart from the connected vehicles entering a roundabout, a static camera is also used, spawned at an elevated location at the center of the roundabout. This is used for gathering groundtruth scene data.

CARLA LiDAR configuration: the configuration of the CARLA emulated Velodyne Lidar parameters (sensor.lidar.ray_cast[3]) to match our real LiDAR (VLP-16 Velodyne): {'range': '50', 'rotation_frequency': '30', 'points_per_second': '300000'}).

***CARLA synchronous and asynchronous mode of operation**:* Synchronous mode of operation is used for all EVENTS virtual experiments in order to ensure repeatability of scenario-based testing process. Asynchronous mode may be selected in the case of a hybrid experiment setup, where data from a real agent shall be displayed in a CARLA scenario with multiple virtual agents.

CARLA KPIs logging

A small set of execution KPIs are automatically logged in each run:

- Scenario execution duration
- Collisions (de-activated but available)
- Average velocity of each agent [m/s]
- Minimum TTC and distance to other agents
- Computation time

***CARLA Hardware**:* The experiments ran on a laptop ASUS ROG AMD Ryzen 9 7940HS; 32GB Ram DDR5 5200MT/s; IntegratedGPU Radeon 780M Graphics; DedicatedGPU

[2] https://carla.readthedocs.io/en/0.9.7/recorder_and_playback/
[3] https://carla.readthedocs.io/en/latest/ref_sensors/#lidar-sensor

Not officially approved by the EC

NVIDIA RTX4060 mobile; Manjaro Linux 24.1.2; Linux Kernel 6.12; NVIDIA driver 550.120.

***Interfacing an external ROS-based module**:* In order to support real-time simulation and CP testing, two external modules had to be interfaced with CARLA via CARLA-ROS bridge, as shown in Figure 14.b .The two modules are the on-board perception module developed by ICCS using camera-based perception (YOLO [29] was used) verified by lidar and the CP module responsible for the fusion of all agents' CPMs and the creation of a scene CPM which is transmitted back to the connected agents.

Hybrid testing support (future work for experiment that is also part of T5.2)

CARLA real world interface: within EXP2 an additional experiment, part of T5.2, includes a real vehicle deployed on ICCS premises, communicating in real-time with the ICCS CARLA environment. For this purpose, message queuing services (using gRPC or MQTT protocols) will be supported at the side of the real node (ICCS prototype vehicle) offering real-time connectivity with the CARLA PC which will integrate an appropriately configured connectivity gateway (gRPC adapter).

Importing OSM map data to CARLA : A prerequisite for hybrid testing is that real test track topology, lanes, environment can be reproduced inside CARLA via a custom map importing. For this purpose the functionality of CARLA v.0.9.15[4] will be used while a digital map of ICCS premises (NTUA campus) is already available from T3.1 (see project deliverable D3.2 [30]) using the RoadRunner-CARLA pipeline for map creation from OpenStreetMap format.

3.2.3 Key Challenges and Solutions

The key challenge we faced was to achieve a smooth rendering experience in CARLA while multiple sensors generating big size data (RGB and Lidar data from multiple agents) could be recorded simultaneously. To do that the following actions were needed:

- CARLA modules that we do not need for the experiment and require computing resources are deactivated: that is 'lane keeping' and 'collision notification' sensor.
- Logging on a MMAP file on a tmpfs (RAM filesystem) for quick logging: we use a shared memory file mechanism for logging the data in a tmpfs (RAM) filesystem file, reducing the time required for writing all the amount of data from all the agents and sensors to an m.2 NVME SSD. By reducing the data storing time we can achieve simultaneous storing and human perception-

[4] https://carla.readthedocs.io/en/0.9.15/adv_digital_twin/

Not officially approved by the EC

suitable rendering quality. RGB images are rendered and logged at a resolution of 800x600 pixels.

One of the main problems with full integration was the large number of modules that needed to work together. They also needed to be stable and able to be restarted if they failed. In addition, there are often two deep learning models per vehicle plus the CARLA simulation that need to run on the GPU. So we built everything on an HPC server, which does not allow the kind of development that can be done on a desktop PC. To solve the above situations, we take the following actions:

- Each module has its own container, which runs in isolation and is tested every 20 seconds to make sure it is working properly. We use Docker Compose and its tools to test the health of the containers. In addition, the test checks assess the frame rate of various variables using ROS2 to evaluate the performance of the container. If it is not running properly, it is restarted to continue. This behaviour is critical for the RL integration, where the simulation needs to run for hours at a time.
- The entire simulation environment runs on an HPC with a dedicated NVIDIA GPU for server platforms, which has no display outputs. We simulate these displays and build a custom one for each developer using the server.

Not officially approved by the EC

©EVENTS Consortium 2022-2025

3.3 Experiment 3

EXP3 focuses on the self-assessment of the onboard perception system within the ego vehicle. This newly developed self-assessment approach [31], [32], [33], [34], [35], [36] will be integrated into Ulm University's test vehicle. Before implementation on the test vehicle, the approach is tested and validated through simulations, which replicate the entire architecture and software stack. These simulations include the onboard perception self-assessment system, complementary V2X data in the form of CPMs from an infrastructure pilot site, the fusion of onboard perception with external CPMs, and behavioral decision-making to support trajectory planning.

3.3.1 Architecture and Design Considerations

The architecture for EXP3 is illustrated in Figure 15. To implement and integrate this architecture into the software stack, key modules had to be developed and validated. These modules include:

1. Object tracking with self-assessment,
2. V2X data integration via CPMs,
3. CPM and track list fusion, and
4. Behavioral decision-making.

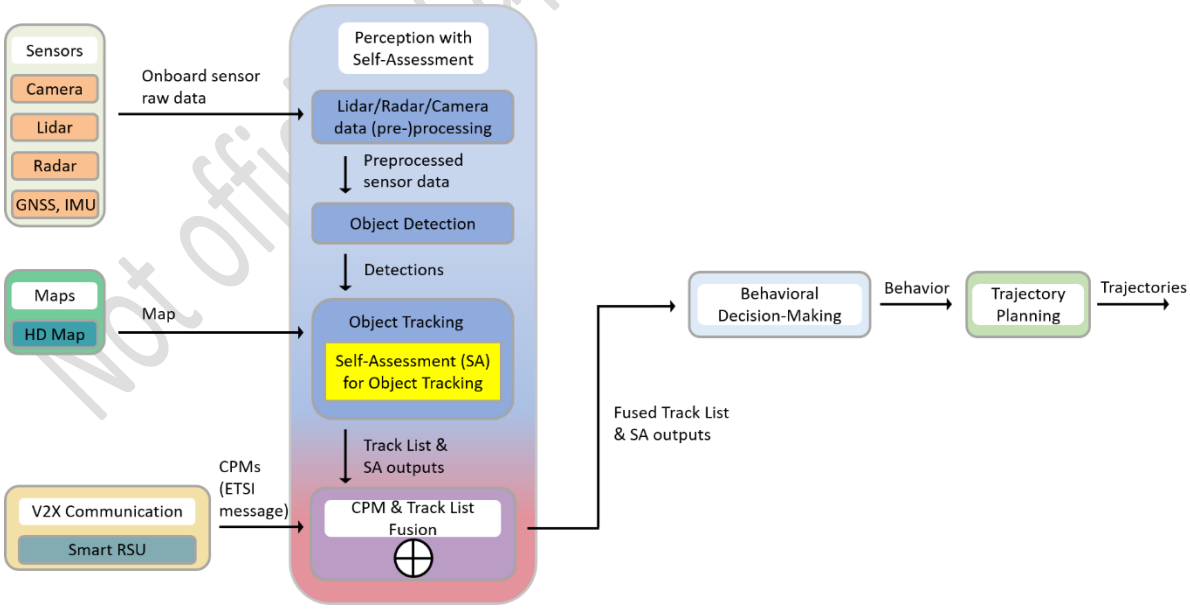

Figure 15: Overall architecture of EXP3 modified from D2.2 [37].

The **object tracking with self-assessment module** underwent initial testing in isolated and simplified simulation environments, such as the tracking simulation framework

Not officially approved by the EC

[38]. This phase focuses on analyzing various potential errors in object tracking and validating the self-assessment module's ability to monitor and identify these errors or disturbances. Results from this stage are documented in [33], [34], [35], [36].

After demonstrating its functionality, the self-assessment module was implemented within the comprehensive simulation environment for EXP3. This integrated internal virtual environment, developed at Ulm University and basically described in [7], allows for the testing of the full software stack. Within this environment, all modules—including V2X data handling, fusion processes, and decision-making—are combined to evaluate the architecture holistically under simulated conditions. The tool allows to simulate human driven vehicles with realistic behavior as well as CAVs on a provided map in lanelet2 [39] format. The used map reflects the roads at the pilote site in the suburban area of Ulm-Lehr, where the evaluation in real traffic will be performed.

The used simulation environment is illustrated in Figure 16. This figure provides an example of various elements involved in the creation of the CPMs in EXP3, offering insights into the simulator and its functionality. Note, however, that the simulator and its simulations are primarily demonstrated visually in the corresponding video created for this deliverable. The cameras' fields of view of the intersection used in EXP3 are represented as rectangles that indicate the coverage areas of the cameras. In addition, the 3D Tracks, depicted as solid blue bounding boxes, represent the object tracks in 3D space, which are used to generate the CPMs. These tracks are based on the detections appear as light blue outlined bounding boxes, showing the detected objects extracted from camera images in 2D. Note that the detections are mostly hidden under the tracks, making them hardly visible in the figure. The ground truth, which is avaible in simualtion, is shown in light green, representing the true 2D bounding boxes for the objects. Also, the ground truth is hardly visible behind the 3D tracks, which indicates good tracking performance.

Not officially approved by the EC

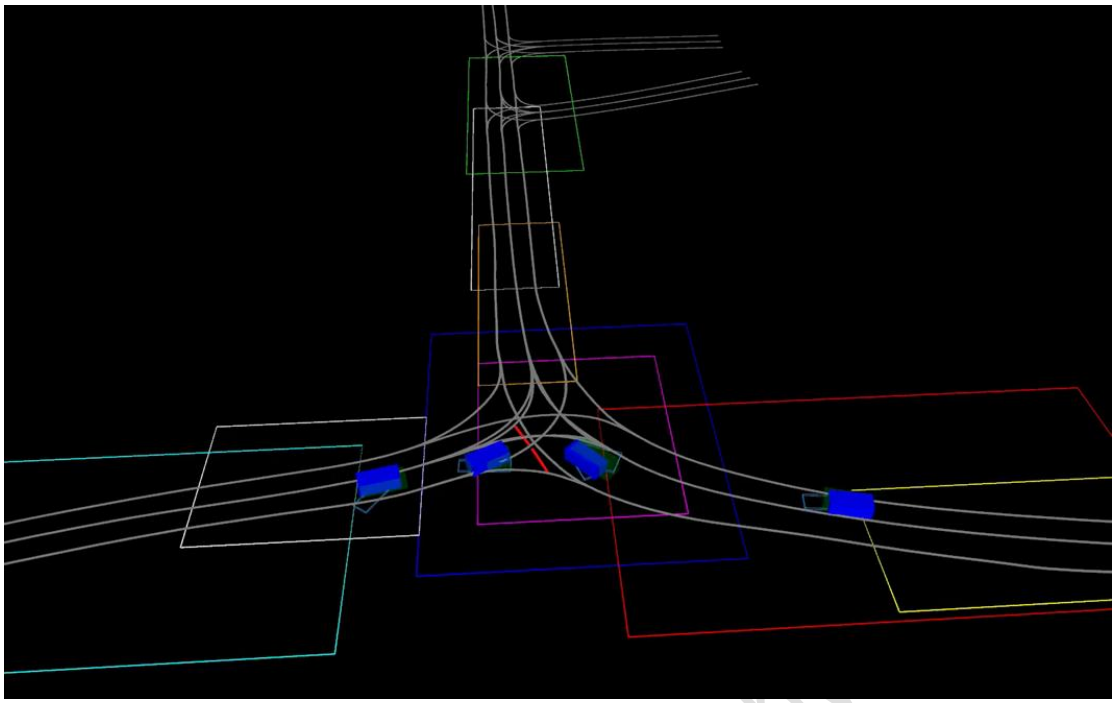

Figure 16: Simulation environment for Experiment 3 based on a software-in-the-loop framework [7].

3.3.2 Step-by-Step Integration Process

This subsection outlines the step-by-step procedures followed to integrate EXP3 into the simulation environment, covering the key tasks, software modules, and challenges encountered during the process.

The key tasks for the integration process are:

1 **Module Development and Testing:** The core modules for EXP3 were developed and validated independently to ensure functionality. This included object tracking algorithms and self-assessment mechanisms. Early testing was conducted in simplified simulation environments such as the tracking simulation framework [38].

2 **Simulation Environment Setup:** The simulation environment was established using Ulm University's virtual simulation environment, based on [7]. This environment allowed the integration of Experiment 3 alongside complementary modules from the whole software stack.

3 **Integration of EXP3 Components:** V2X data in the form of CPMs was incorporated into the simulation. This data was fused with an onboard object tracking outputs to augment the perception field of view.

Not officially approved by

3.1 **Object Tracking with Self-Assessment:** The object tracking module, equipped with self-assessment capabilities, was developed in WP3 (will be described in deliverable D3.2) to monitor and evaluate the reliability of detected objects in real-time. This module serves as the foundation for detecting and addressing tracking errors, ensuring accurate perception.

3.2 **V2X Data Incorporation:** Cooperative Perception Messages (CPMs) were integrated as an essential V2X data source. These CPMs, generated from an infrastructure pilot site, provided an extended perception range beyond the onboard sensors.

3.3 **CPM and Track List Fusion:** A dedicated fusion module was implemented to combine the onboard object tracking outputs with external CPM data. This fusion process augmented the vehicle's perception field of view and enhanced the accuracy and robustness of the perception system.

3.4 **Behavioral Decision-Making Module:** This component processes the fused perception data to enable informed decision-making. It incorporates information from the self-assessment on the reliability of the ego vehicle's perception to determine whether the trajectory planning module can operate normally or should prioritize a safe stop point, utilizing V2X data.

4 **System Validation:** The integrated system underwent iterative testing to evaluate the interaction between modules, focusing on robustness and error handling in various simulated scenarios.

3.3.3 Key Challenges and Solutions

1. **Interfacing Modules:** Ensuring seamless communication between the self-assessment module and other components, particularly during error flagging, required extensive debugging and iteration.
2. **Error Modeling:** Accurately simulating various object tracking errors to test the self-assessment module required significant effort to achieve realistic conditions.

Despite these challenges, the integration of EXP3 into the simulation environment was successfully completed, providing a robust platform for further testing and validation of the developed modules.

Not officially approved by the EC

3.4 Experiment 6

The goal of this experiment is to propose a perception system, which can detect an object at a far range in adverse weather conditions. The system has to estimate the relative distance and classify the object as over-drivable or non-drivable. In the scope of EXP6, scenario is the following , the ego vehicle is driving straight forward to the debris as shown in the Figure 17. Differents parameters will be introduced as : ego vehicle speed , orientation of debris in regards to road axis , lateral position offset of the debris in regards to ego vehicle position , and debris's material, shape and size.

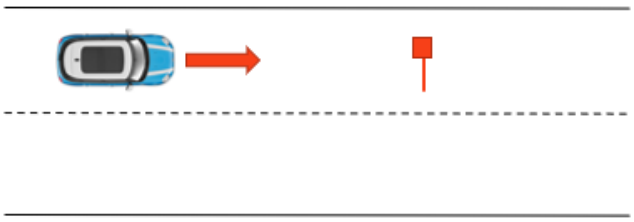

Figure 17 EXP6 Scenario definition

3.4.1 Architecture and Design Considerations

To guarantee a working and effective architecture ,the base for the realisation of the experiment, an existing virtual simulation system composed of of a third party tool and an Aptiv software was used. This base is shown in Figure 18. The system consists of a Realtime Environment Simulation Provider. In this case, a dSPACE System was used, which provides ground truth to other system components via OSI (Open Simulation Interface) SensorView and offers CAN/ETH interfaces if closed Loop functionality is required.

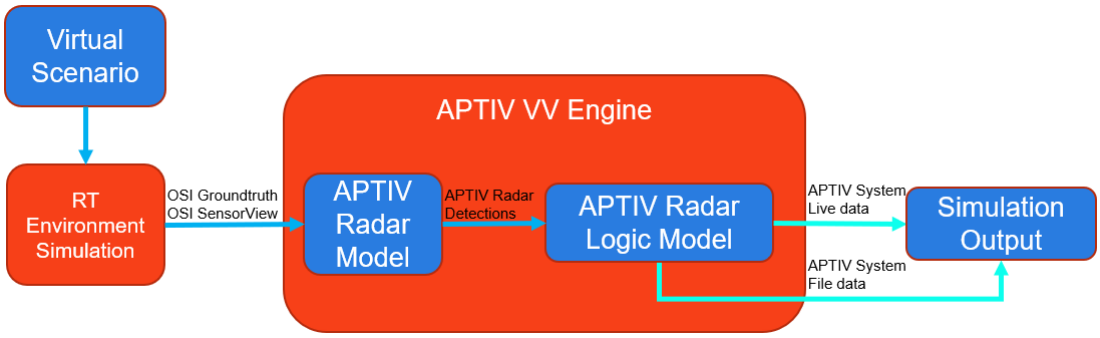

Figure 18 Initial Virtual Simulation System architecture

The ground truth is used by the Virtual Environment (VE) Engine that runs two different Models. The first one is realtime APTIV Radar Model, which generates APTIV Radar specific detections based on the moving and static objects in the OSI SensorView. The Aptiv Radar detections then are fed into the APTIV Radar Logic

officially approved by the EC

Model, which contains all components that are part of the APTIV Radar SW. Depending on the requirements, this contains a Radar Tracker, Radar Feature Functions, and the representation of the physical communication interfaces (CAN, ETH). The Environment Simulation is parametrized with a fitting Scenario, which consists of a Road Environment and a driving scenario.

To integrate the over-drivable feature into the proved solution, in the first step, the Virtual Scenario and the APTIV Radar Model needs to be adapted to contain objects that fall under the consideration of the over-drivable feature. As OSI SensorView is the interface to the APTIV Radar model, those objects need to be configured with OSI object definitions as well. Once Environment Simulation and APTIV Radar Model are updated with the new Objects, the APTIV Radar Model will provide detections for those objects which are handed to a standalone SW block of the over-drivable function. This architecture provides first results and can be used to evaluate the performance of both the APTIV Radar Model and the over-drivable feature function. The modification is depicted in Figure 19.

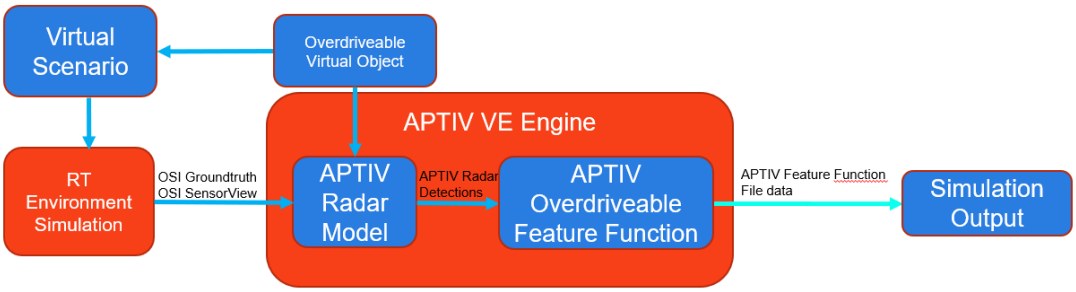

Figure 19 Integration of over-drivable OSI objects

As the final step, the standalone over-driveable Feature Function needs to be integrated into a APTIV Logic Model, to be executed next to all other features of the APTIV SW.

Not officially by the EC

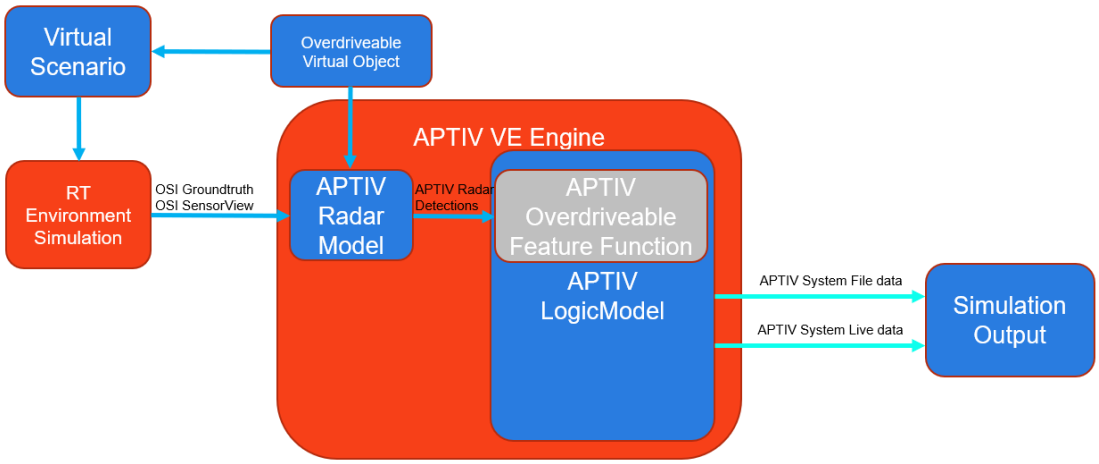

Figure 20 Final integration architecture

3.4.2 Step-by-Step Integration Process

From the above defined architecture, the different integration steps are as follows:

1. Environment Simulation
    a. Debris objects definition : during this step we defined all relevant Objects with OSI specification in Environment Simulation Tool
    b. Scenarios definition : we created Scenarios and Roads including relevant Objects
2. Radar Model
    a. Support for relevant OSI objects : we added mesh support for relevant OSI Objects
    b. Radar Model Detections validation : we compared test track data to simulation data
    c. Radar Model Detections : we cooperated with Tracker team (team who develop and integrate sensor data algorithm fusion block) to ensure detections are valid for static object feature
3. Logic Model
    a. static object feature into Logic Model Integration : we integrated the algorithm block
    b. Validation : we validated output data of Logic Model block by checking same behaviour seen by our Tracker team
4. System Test : we tested the global setup to check performances

Not officially approved by the E

Below, the different involved components are explained in detail and the steps leading to the final system are described.

Environment Simulation

The simulation environment used in this experiment was dSPACE ASM (Automotive Simulation Models) in combination with a dSPACE Scalexio Real Time (RT) Hardware. The dSPACE Scalexio is running a combined model for environment, road, traffic and host vehicle ego motion, so that the wide range of required parameters can be considered.

This setup enables the creation of realistic virtual scenarios, which are used within the Realtime Environment Simulation to feed radar models with OSI ground truth data (Open Simulation Interface). Accurately modelling these scenarios is critical to providing realistic input for sensor perception.

A specific scenario was designed to incorporate debris object positioned at a varying distance. In this scenario, the ego vehicle travels at a constant speed of 65 km/h on a road devoid of guardrails, starting at a distance of 350 meters.

To evaluate the radar perception algorithm's capability in assessing driveability, diverse types of debris objects where introduced. The objects shown in Figure 21 were selected for this purpose: These objects, beside their localisation and visualisation inside of the Environment Simulation, were parametrized as OSI objects.

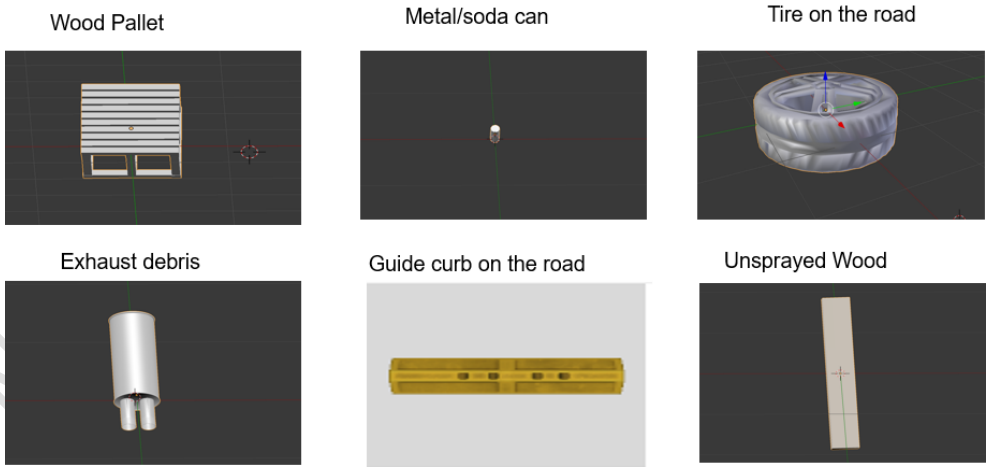

Figure 21 Example of implemented debris objects

Radar Model Development

The core focus of the radar model development is to replicate the physical behaviour of radar sensors. It uses OSI ground truth data to generate perception point clouds that simulate radar detections.

©EVENTS Consortium 2022-2025

The primary goal for this experiment was to integrate objects relevant for the SG Feature into the existing Aptiv sensor model. This included:

- Integration of Debris OSI Objects into APTIV Radar Model.
- Validation of radar sensor model vs. real world data to cover use-case's small objects with high fidelity based on initial real world data

Beside from the adaptions made for the APTIV Radar Model, the consequently step was collaborating with the Static Geometry Feature developers to adjust the model parameters to achieve maximum alignment with real-world conditions. Following that we conducted extensive testing and validation to ensure the simulated radar detections function as intended.

Logic Model: The APTIV Logic Model is tasked with integrating APTIV Radar Features as runnable Functional Mock-up Unit (FMU). It takes the APTIV Radar Model Detections as input and runs all other features that are part of the Radar system. Together with the APTIV Radar Model, it is the virtual representation of an APTIV Radar System. It allows for a full evaluation of the Radar SW components via Live data injection and generated debug files or Live Data output.

For this experiment, the functionality of the Static Geometry (SG) feature was integrated into an existing Logic Model, so that it can run together with Tracker and ADAS Features and as part of the Realtime system, making it an equivalent alternative for real world Vehicle tests. Fully integrated, the Logic Model allows to validate components depending on the SG Feature like Motion Planning algorithms.

3.4.3 Key Challanges and Solutions

To get the proper output, the existing APTIV Radar Model for the APTIV Front Looking Radar had to be tuned to match the behaviour of the used SW/HW combination. Parameters like field of view, range, and mounting positions had to be adapted to the virtual Host vehicle.

The Environment simulation does not contain debris objects by default. Although it is possible to define new OSI objects and define their parameters, those 3D Objects had to be implemented in the Sensor Model as well to have a proper representation.

Whenever a new functionality is implemented, it takes time until the tooling support is established. Thus, it was necessary to use an alternative approach for validation to ensure results can be created and analysed.

Not officially approved by the EC

3.5 Experiment 7

As part of Experiment 7 extended description*, ICCS aims to integrate a Machine Learning module which jointly conducts prediction and planning (Joint PP) for autonomous vehicles testing within a virtual simulation environment. We consider various scenarios in highway environment with traffic which test the autonomous vehicle under various critical conditions including cut-ins, breaks and merges. The open-source nuPlan simulation environment was selected due to its closed-loop simulation support and its popularity in the automotive research community. In this deliverable, the integration software details are reported, with the algorithmic and mathematical counterparts presented technically in deliverable D4.3 [40].

*Note: As it is already reffered in the introductory part of this deliverable, this is a new part of ICCS work that was taken over in the context of WP4 based on the second project amendement (approval is pending at the time of writing) and which is still under development. What is reported here is the progress we made till this deliverable's submission date.

3.5.1 Architecture and Design Considerations

Figure 22 below demonstrates the virtual training setup, which uses samples from the nuPlan database from various scenarios. The number of simulation steps K and reaction steps K' is a hyperparameter, to be chosen dataset-dependent. The simulator logs trajectories and vectorized maps to feed the ML systems (prediction, planning). Specifically, the input is training samples consisting of vectorised agent and map locations (i.e., logged 2D trajectories) and semantic information (traffic states, vehicle types and identities, etc.) logged in the given database as annotations. The simulation steps analytically are:

- *Input Preprocessing*: Vector objects are extracted from the Perception database and arranged in a tabular fashion (and min-max scaling is applied as required by the following deep learning modules. Two submodules are considered one for Agents and one for Maps which extract feature representations that feed into the next ML components of the framework for prediction.
- *Joint Prediction and Planning (PP) call*: Forward pass of (learned) features by our proposed system that outputs a traffic-compliant time-series of non-ego agent trajectories (*predictions*) and ego agent trajectories (*plans*) for the next K' frames, given K simulation step input previously.
    - *Differentiation Engine*: Framework with automatic differentiation capabilities (PyTorch) to backpropagate gradients and update the weights of each Deep Network in the Joint PP system.
    - *Simulator*: Closed-loop simulator accepting the prediction and planning outputs of Joint PP system. The simulator executes the motion plan for ego

Not officially approved by the EC

vehicle using the control system equations and computes the prediction and planning metrics for the current training segment and visualizes results during runtime.

- *Metric logger*: Logging module monitors and plots the prediction and planning metrics online during the inference section of time window T_p for the current segment. This section outlines the architectural framework and design principles guiding the integration of prediction on the NuPlan simulator, focusing on system requirements and interoperability with other/existing components.

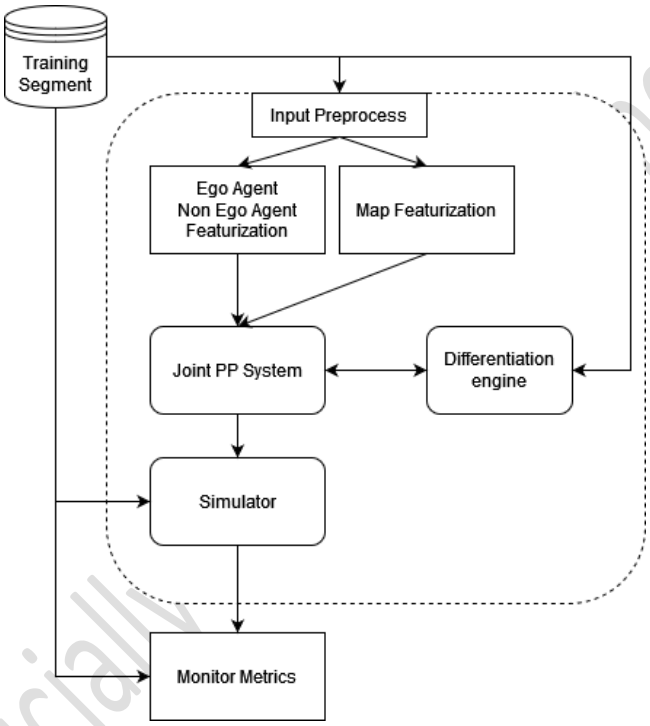

Figure 22 Virtual training Setup of Joint Prediction and Planning system

Not officially the EC

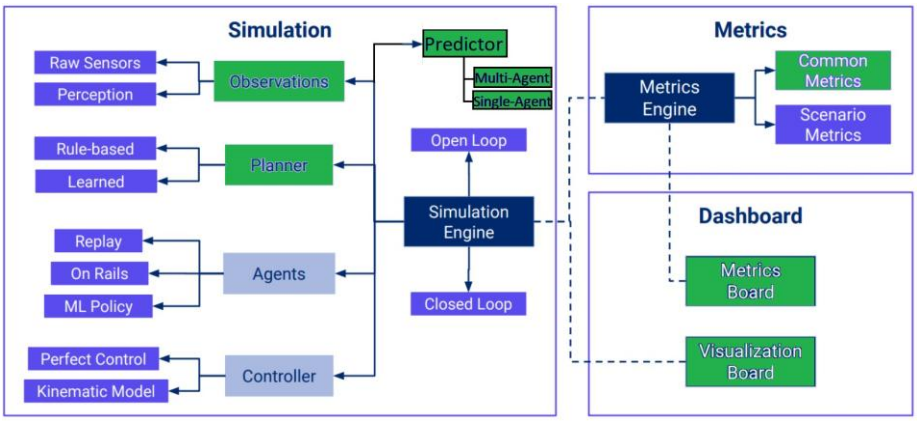

Figure 23 NuPlan Framework High-Level software description where green color denotes modules that had to be adapted by ICCS.

3.5.2 Step-by-Step Integration Process

As observed from Figure 23, we have added extra predictive functionality with ML components for the non-ego vehicles, together with an integrated package for merging prediction and planning in a modular, differentiable manner. NuPlan provides a configurable environment via Hydra configuration files, which we edit to add to the Joint PP approach. Specifically: ICCS changed the ”Observations”, which correspond to the input features consumed by the planner module to conduct inference on ego agent future trajectories, by adding additional prediction features (non-ego agent trajectories) as a custom class attribute by changing also the custom Hydra configuration file for the NuPlan simulator. The prediction module continuously updates a ”prediction trajectory” field stored for each agent as an “Observation” class in simulation. On a high-level, the framework at each iteration step assigns the predicted trajectory state sequence, together with historical tracks of each agent as observations (for details see deliverable D4.3 [40]) and the feature builders of NuPlan construct features for planning accordingly. We further change the planners by expanding their input dimensionalities to consume non-ego agent tracks (3D pose information), for most planners as flattened vectors on the regression head. Further, we expand on the metrics for evaluation, including those relevant for Joint PP evaluation (see D4.3 [40]). Finally, the vizualisation board was edited, to show non-ego trajectory prediction lines, next to the actual controller output.

The key objectives for the integration process are listed below:

1. **System Integration**. NuPlan adopts a modular pipeline in terms of hardware resource allotment for each component for building model features and targets for learning(preprocessing) and training, hence correct assignment of data into CPU for the preprocessing steps and GPU for training, validation and testing ensures correct allocation of memory, avoiding high disk memory pressure.

Not officially approved by the EC

2. **Scenario Filtering**: NuPlan logs and vehicle tracks are divided into various scenario tags, specifically: { ”medium magnitude speed, near construction zone, near multiple vehicles , on intersection, stationary in traffic, traversing traffic light intersection”}.A custom filter module is being developed to retrieve scenarios which best meetsthe operational design domain of the EXP 7.

3. **Correct Assignment of Features for Preprocessing**: Selecting the correct features (object tracks as poses, HD map features) and assigning them as ”observations”, which are fed to preprocessing modules, ensures seamless communication of the simulator with the feature building modules of the planning models in NuPlan and correct training of ML models.

4. **Module Development and Validation**: Each module of EXP7 was trained and tested individually, i.e., Predictor, Planner, in the simplest core configurations in terms of: {scenario, controller, map, perception model} in open and closed loop simulation protocol to validate standalone performance prior to integration.

5. **Behavioral Decision-Making and Trajectory Prediction**: Development and integration of trajectory forecasting module presented in D4.3 [40] which regresses the non-ego waypoint time-series given features supplied by the simulator during runtime. Meeting time restraints for predictor inference is critical to meet the planners input data requirements and ensure quick simulation iteration.

6. **NuPlan Dashboard trajectory vizualization**: The NuPlan visualization component, nuBoard, was modified to show non-ego trajectories forecasted by the Prediction module and the Ego Trajectory computed by the standalone NuPlan vs the integrated NuPlan to be compared qualitatively.

3.5.3 Key Challenges and Solutions

The NuPlan simulator is originally designed for planning of the ego vehicle without considering non-ego vehicle trajectory predictions, which was previously shown to be of high importance in decision making. Hence a multitude of challenges was faced in understanding complex undocumented code responsible for feature and ground truth extraction and complementing with code necessary for prediction of non-ego vehicles.

Further, the metric structure of the evaluation had to be modified, as well as the vizualization code for the dashboard, to include key metrics necessary for joint prediction and planning, including dynamic ADE (Average Displacement Error), FDE(Final Displacement Error) etc.

Not officially approved by the EC

3.6 Experiment 8

EXP8 focuses on emergency evasion manoeuvre on slippery road under rain conditions. The objective is to avoid collisions (e.g., pedestrians or cyclists) in poor weather conditions on slippery roads. The scenario replicates a classical double-lane change manoeuvre involving two static / slowly-moving obstacles, which the vehicle must avoid. Heavy rainfall reduces the road friction coefficient to 0.5, thereby increasing the manoeuvre's complexity and requiring precise control strategies, see Figure 24.

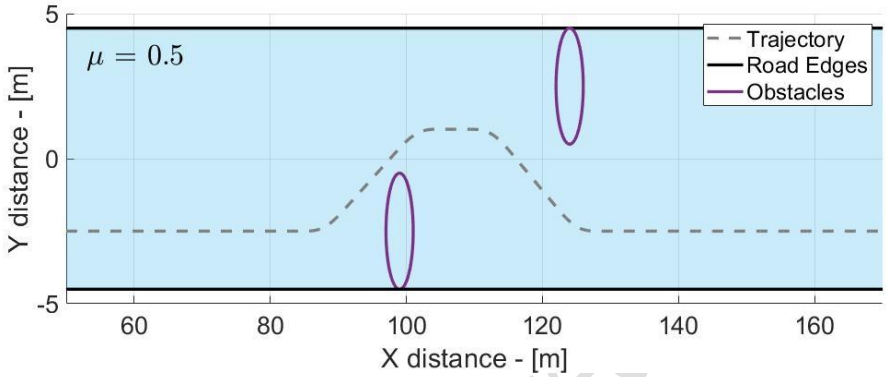

Figure 24 Collision avoidance scenario with two obstacles.

In this scenario, the vehicle encounters unexpected obstacles with limited visibility due to adverse weather conditions, necessitating an immediate and aggressive response. The evasion manoeuvre involves several critical stages. First, the vehicle executes heavy braking to reduce speed rapidly while initiating a sharp cornering manoeuvre to bypass the obstacles. Following the initial evasion, it must perform another cornering event to return to its original lane while simultaneously accelerating to mitigate the risk of a rear-end collision. This sequence of actions inherently involves complex interactions between longitudinal and lateral tyre forces, creating a highly nonlinear control problem due to tyre-road interactions in the presence of reduced friction.

3.6.1 Architecture and Design Considerations

To address the scenario's complexity, our virtual environment incorporates unique characteristics that ensure accurate modelling and testing:

1. **High-Fidelity Vehicle Plant Model**: The nonlinearities arising from suspension kinematics and dynamics, tyre forces, and the braking system necessitate a highly accurate and robust vehicle plant model. We use a high-fidelity simulation environment such as IPG CarMaker [4]. The vehicle model was parameterized using mass-inertia data from a vehicle inertia measurement facility, suspension kinematics, and compliance characteristics obtained from

Not officially approved

testing on a Kinematics & Compliance test rig for wheel suspension analysis. It was then validated through field tests conducted on a proving ground [41].

2. **Advanced Tyre Modelling**: The coupled dynamics of longitudinal and lateral tyre forces require an accurate and robust tyre model. For this, we employ the Delft Tyre Model [42], recognized for its effectiveness in capturing tyre dynamics and nonlinear behaviours, even under adverse conditions, and also experimentally validated.

3. **Optimized Model Predictive Control (MPC) Framework**: The proposed control solution leverages an MPC framework to address nonlinear control challenges, integrating real-time motion planning, path tracking, and stability constraints while considering powertrain and braking limits, using ForcesPro for fast, reliable nonlinear optimization solutions.

4. **Real-Time Feasibility Testing**: To ensure real-time feasibility, the controller is developed in MATLAB using ForcesPro [43]. For real-time testing, it is translated into C++ for faster execution, providing lower-level control compared to interpreted environments.

The complexity and unique requirements of EXP8, including the need for a high-fidelity vehicle plant model and an advanced tyre model, necessitate the creation of a unique architecture for developing and validating the proposed control solution.

The fast development architecture is focused on three essential requirements to support efficient control strategy design. First, it must provide access to a high-fidelity simulation environment validated with experimental data. For this, we employ IPG CarMaker [4] in combination with the Delft-Tyre model [42], ensuring that simulations closely mirror real-world dynamics and accurately capture nonlinear tyre-road interactions. This high-fidelity modelling is critical for robust control design. Second, the architecture must support rapid evaluation and iterative testing of design concepts within a unified multidomain simulation environment. This capability accelerates the development cycle, allowing for the refinement of control strategies in a flexible, adaptable manner. Third, the architecture must facilitate the generation of C++ code suitable for deployment on the embedded systems used in EXP8, ensuring smooth transitions from simulation to implementation.

MATLAB/Simulink is serving as the environment for developing and validating the proposed Model Predictive Contouring Control (MPCC) framework (described in D4.2 [11] in detail) in combination with using high-fidelity simulations like IPG CarMaker integrated with the Delft-Tyre model. This flexible approach allows for the exploration and validation of different control strategies while isolating potential inaccuracies from other aspects of the automated driving pipeline. The iterative evaluation process

Not officially approved by the EC

ensures rigorous validation before integration into more complex system environments. Additionally, MATLAB's capabilities for generating C++ code streamline the transition from simulation to real-world testing, reducing potential implementation errors and enhancing efficiency. Figure 25 illustrates a simulated scenario within IPG CarMaker and Simulink, demonstrating the controller's ability to successfully avoid two obstacles while operating at the limit of handling.

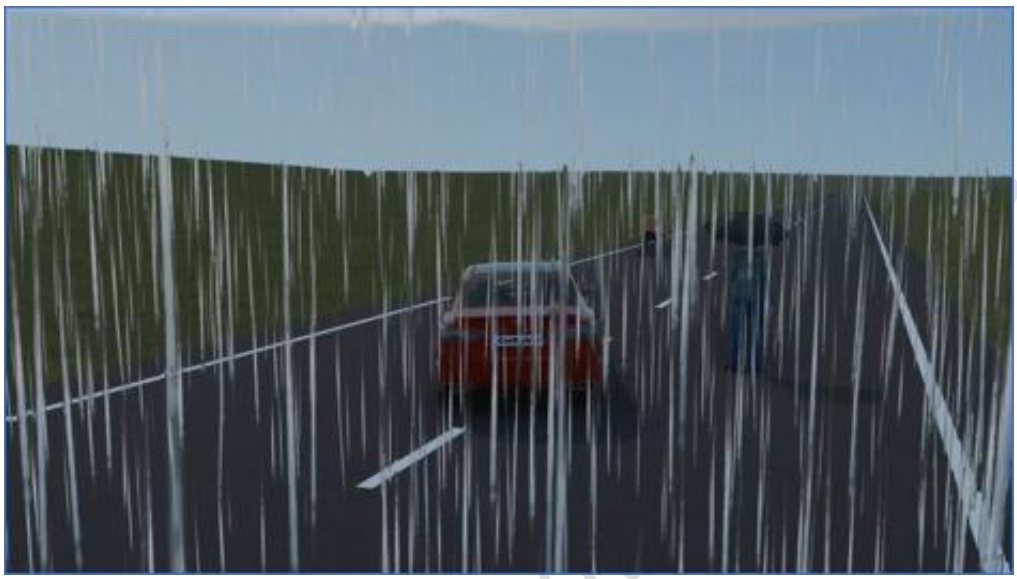

Figure 25 An instant of the obstacle avoidance manoeuvre.

Under normal operating conditions, the proposed MPCC [44], [45] tracks the planned trajectory generated by the motion planner when the vehicle is safely distanced from obstacles. The primary goal is smooth, precise path tracking that follows the planned route closely, leveraging MPCC's predictive capabilities to minimize deviations and optimize vehicle dynamics by considering coupled lateral and longitudinal dynamics. This predictive approach significantly enhances real-time response and path accuracy.

As scenario complexity increases, such as when the planned trajectory brings the vehicle close to obstacles or when motion plans become infeasible due to oversimplified predictive models, the MPCC's dynamic capabilities become critical. In such cases, it activates a dynamic motion replanning process to ensure obstacle avoidance while maintaining stability and safety. Real-time trajectory adjustments allow the vehicle to navigate unexpected challenges, such as sudden changes in road friction or hazardous conditions, without compromising control and stability. This adaptability is essential for managing nonlinearities inherent in tyre-road interactions, especially in constrained friction environments.

3.6.2 Step-by-Step Integration Process

This subsection outlines the procedures to integrate EXP8 into the simulation environment:

Not officially approved

1. **Implementation of the High-Fidelity Model.** The initial step involved incorporating a high-fidelity vehicle model within the IPG CarMaker simulation environment. Specific tuning of the Delft-Tyre model's scaling factor was performed to accurately capture the reduced friction conditions and dynamic changes associated with heavy rain, reflecting the unique characteristics of EXP8. Figure 26 shows how the IPG CarMaker is incorporated into Simulink.

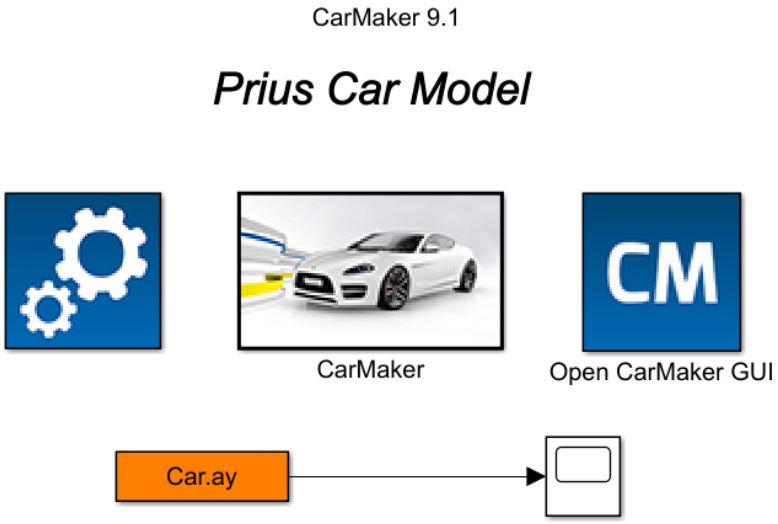

Figure 26 IPG CarMaker integration into Matlab/Simulink.

2. **Inclusion of Brake and Steering Dynamics Delays.** Brake and steering dynamics delays were incorporated into the high-fidelity model to realistically represent response lags inherent in automotive systems. This adjustment was crucial for developing and testing control algorithms on a more accurate simulation basis.

3. **Implementation of the Controller in ForcesPro.** The proposed Model Predictive Contouring Control framework was implemented using ForcesPro, an advanced optimization software for high-speed, nonlinear problems [43]. Figure 27 illustrates the integration of ForcesPro within the Simulink simulation architecture.

4. **Validation of the Prediction Model.** The prediction model was validated by comparing its performance against the high-fidelity model within the simulation environment. This allowed for fine-tuning to ensure that the prediction model accurately mirrored vehicle behaviour under challenging conditions.

Not officially approved

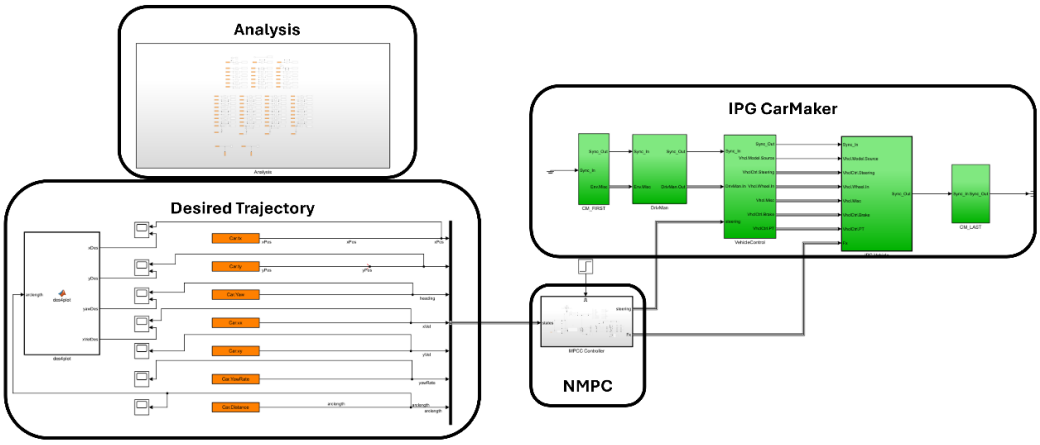

Figure 27 Simulation architecture in Matlab/Simulink.

5. **Scenario Generation for Experiment 8.** A challenging test scenario was generated for Experiment 8, placing obstacles strategically to push the controller to the limit of handling. This scenario evaluated the controller's performance in critical situations.

3.6.3 Key Challenges and Solutions

Iterative testing and refinement of the controller were conducted to enhance performance and robustness across different conditions. The results, presented in Figure 28 and Figure 29, demonstrate the effectiveness of the proposed controller using the wet tire parameterization. This approach successfully avoids both obstacles, whereas the baseline controller (shown in red) fails to avoid the second obstacle, resulting in a collision.

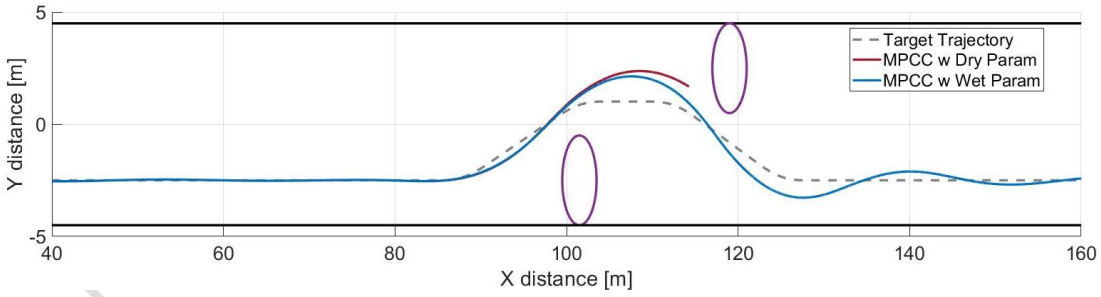

Figure 28 Vehicle trajectories, with the baseline approach represented in red, which results in a collision, and the proposed approach depicted in blue, successfully avoiding the obstacles.

©EVENTS Consortium 2022-2025

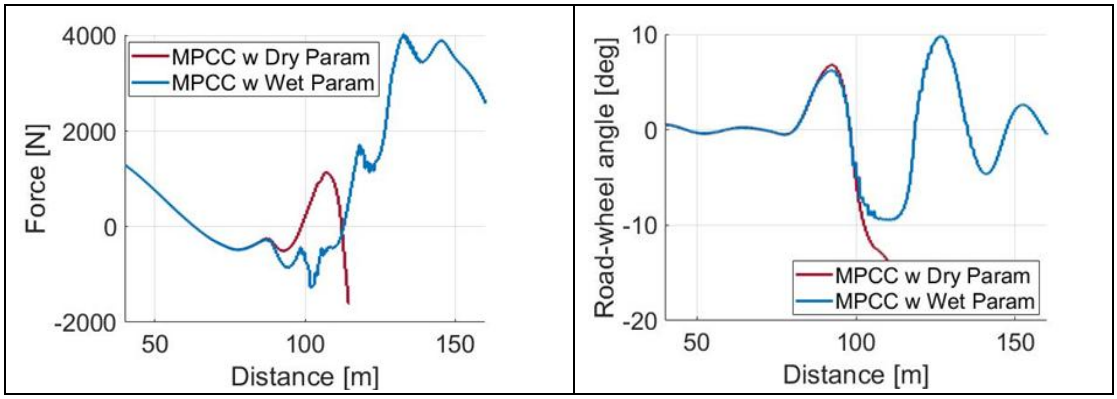

Figure 29 The control inputs, specifically the commanded longitudinal force (left) and the road wheel angle (right).

Not officially approved by the EC

4. Conclusions

The EVENTS project successfully integrated automated driving components into virtual environments, addressing a wide range of automated driving scenarios, including urban navigation, highway manoeuvres, interactions with vulnerable road users (VRUs), and adverse weather conditions. Through a systematic approach emphasizing modularity, scalability, and consistency, the project ensured effective testing and evaluation of perception, control, and decision-making systems. This work provides a robust basis for hybrid and real-world testing in subsequent phases of the project.

A critical achievement of the project was the ability to replicate real-world challenges within virtual environments, including dynamic multi-agent interactions, VRU-focused scenarios, sensor integration, and complex decision-making processes. For instance, VRU-focused experiments in urban environments tested motion planning systems to safely navigate around pedestrians. These environments enabled the integration of advanced perception technologies, such as LiDAR- and radar-based detection, alongside control and decision-making strategies, including model predictive control and machine learning-based approaches.

The developments carried out in WP3 (Perception and V2X Communication) and WP4 (Decision-Making and Control) were progressively integrated into the virtual environments in parallel to their development. This approach created an iterative testing loop, where components were evaluated and refined continuously as new functionalities emerged. By enabling early integration and testing, the project accelerated the development cycle and ensured that modules were compatible and robust before deployment. Ultimately, this process culminated in setting up the final virtual environments, which now serve as a foundation for hybrid and real-world integration in subsequent tasks.

The use of containerization technologies significantly enhanced the scalability and efficiency of the integration process. Multi-agent systems, which required parallel execution of processes like perception, control, and communication, benefited from containerization's ability to streamline deployment, ensure process isolation, and simplify troubleshooting. Performance monitoring and automated health-check mechanisms further ensured reliable execution of complex systems, enabling consistent and repeatable testing across diverse scenarios.

Machine learning technologies played a pivotal role in tasks such as prediction and decision-making, particularly in dynamic highway environments. Virtual frameworks provided structured testing grounds for evaluating reactive systems under conditions like cut-ins, merges, and other critical interactions. High-fidelity modeling further

Not officially approved by the EC

enabled precise evaluations of control strategies under challenging conditions, such as nonlinear tire-road interactions on slippery roads.

The combination of open-source frameworks, commercial tools, and tailored virtual environments allowed the project to address diverse experiment requirements effectively. Urban navigation scenarios involving VRUs required configurable environments capable of replicating realistic sensor inputs and dynamic interactions. Meanwhile, experiments focusing on control strategies under adverse weather relied on detailed physical models to simulate road conditions with high accuracy. Tailored virtual environments complemented these tools by addressing unique challenges, such as integrating self-assessment modules or enabling radar-based perception systems.

In conclusion, the EVENTS project demonstrated the critical role of simulation technologies in integrating and testing automated driving components across a variety of scenarios. The progressive integration of WP3 and WP4 developments, combined with iterative testing loops, ensured that perception, control, and decision-making systems were evaluated and refined efficiently throughout the project. This process culminated in the creation of robust virtual environments, providing a strong foundation for hybrid and real-world testing. By replicating complex real-world challenges, including VRU interactions, multi-agent coordination, and control under adverse conditions, the project offers valuable insights for advancing automated driving technologies. Virtual environments remain a vital enabler for safe, cost-effective, and repeatable testing, bridging the gap toward real-world deployment in subsequent phases of the project.

Not officially approved by the EC

©EVENTS Consortium 2022-2025

References

[1] EVENTS Project, “EVENTS Deliverable D2.1: User and System Requirements for selected Use-cases,” 2023.

[2] Autoware Foundation, “Autoware Universe#,” [Online]. Available: https://autowarefoundation.github.io/autoware.universe/main/. [Accessed 19 12 2024].

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017.

[4] IPG Automotive, “CarMaker,” [Online]. Available: https://www.ipg-automotive.com/en/products-solutions/software/carmaker/. [Accessed 19 12 2024].

[5] Motional, “nuPlan,” [Online]. Available: https://www.nuscenes.org/nuplan. [Accessed 19 12 2024].

[6] dSpace, “Automotive Simulation Models (ASM),” [Online]. Available: https://www.dspace.com/en/pub/home/products/sw/automotive_simulation_models.cfm. [Accessed 19 12 2024].

[7] J. Strohbeck, J. Müller, H. A. and M. Buchholz, “DeepSIL: A Software-in-the-Loop Framework for Evaluating Motion Planning Schemes Using Multiple Trajectory Prediction Networks,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, 2021.

[8] Open Robotics, “ROS 2 Documentation,” [Online]. Available: https://docs.ros.org/en/humble/index.html. [Accessed 19 12 2024].

[9] The autoware foundation, "Autoware Documentation - Planning simulation," [Online]. Available: https://autowarefoundation.github.io/autoware-documentation/main/tutorials/ad-hoc-simulation/planning-simulation/. [Accessed 30 11 2024].

[10] D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” *Physical Review E,* vol. 51, p. 4282, 1995.

[11] EVENTS Project, “EVENTS Deliverable D4.2 Motion Planning,” 2024.

[12] The Autoware Foundation, “Autoware Universe Documentation - Avoidance module for dynamic objects,” [Online]. Available: https://autowarefoundation.github.io/autoware.universe/main/planning/behavior_pat

Not officially approved by the EC

h_planner/autoware_behavior_path_dynamic_obstacle_avoidance_module/. [Accessed 30 11 2024].

[13] O. d. Groot, B. Brito, L. Ferranti, D. Gavrila and J. Alonso-Mora, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments," *IEEE Robot. Autom. Lett.,* p. 5389 – 5396, 2021.

[14] ASAM, "ASAM OpenSCENARIO 2.0," 20 07 2022. [Online]. Available: https://www.asam.net/static_downloads/public/asam-openscenario/2.0.0/welcome.html. [Accessed 30 11 2024].

[15] Mathworks, "Roadrunner," [Online]. Available: https://www.mathworks.com/products/roadrunner.html. [Accessed 19 12 2024].

[16] Plan Nacional de Ortofotografía Aérea, "Visualizadores y Servicios web - PNOA," [Online]. Available: https://pnoa.ign.es/pnoa-imagen/visualizadores-y-servicios-web. [Accessed 19 12 2024].

[17] ASAM, "ASAM OpenDRIVE®," [Online]. Available: https://www.asam.net/standards/detail/opendrive/. [Accessed 19 12 2024].

[18] Docker, "Docker Documentation," [Online]. Available: https://docs.docker.com/. [Accessed 19 12 2024].

[19] NVIDIA, "CUDA Toolkit," [Online]. Available: https://developer.nvidia.com/cuda-toolkit. [Accessed 19 12 2024].

[20] NVIDIA, "Nvidia container toolkit - Github," [Online]. Available: https://github.com/NVIDIA/nvidia-container-toolkit. [Accessed 19 12 2024].

[21] Nvidia, "Github - NVIDIA Container Toolkit," [Online]. Available: https://github.com/NVIDIA/nvidia-container-toolkit. [Accessed 30 11 2024].

[22] T. &. L. C. Ylönen, "The Secure Shell (SSH) Transport Layer Protocol," 2006.

[23] CARLA, "The ROS Bridge package," [Online]. Available: https://carla.readthedocs.io/projects/ros-bridge/en/latest/run_ros/. [Accessed 19 12 2024].

[24] Python Software Foundation, "Python 3.10.0," 4 10 2021. [Online]. Available: https://www.python.org/downloads/release/python-3100/. [Accessed 19 12 2024].

[25] Ubuntu, "Ubuntu Desktop Guide - Documentation," [Online]. Available: https://help.ubuntu.com/22.04/ubuntu-help/index.html. [Accessed 19 12 2024].

Not officially approved by the EC

[26] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik,* vol. 1, no. 1, p. 269–271, 1959.

[27] QGIS, “QGIS Documentation,” [Online]. Available: https://www.qgis.org/resources/hub/. [Accessed 19 12 2024].

[28] [Online]. Available: https://github.com/willfarrell/docker-autoheal. [Accessed 30 11 2024].

[29] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[30] EVENTS Project, “EVENTS Deliverable D3.2: Perception System and Self-Assessment,” 2024.

[31] T. Griebel, J. Müller, M. Buchholz and K. Dietmayer, “Kalman Filter Meets Subjective Logic: A Self-Assessing Kalman Filter Using Subjective Logic,” in *2020 IEEE 23rd International Conference on Information Fusion (FUSION), pp. 1-8, doi: 10.23919/FUSION45008.2020.9190520*, Rustenburg, South Africa, 2020.

[32] T. Griebel and e. al., “Self-Assessment for Single-Object Tracking in Clutter Using Subjective Logic,” in *2022 25th International Conference on Information Fusion (FUSION), pp. 1-8, doi: 10.23919/FUSION49751.2022.9841294*, Linköping, Sweden, 2022.

[33] T. Griebel, J. Heinzler, M. Buchholz and K. Dietmayer, “Online Performance Assessment of Multi-Sensor Kalman Filters Based on Subjective Logic,” in *2023 26th International Conference on Information Fusion (FUSION)*, Charleston, SC, USA, 2023.

[34] T. Griebel, N. Dehler, A. Scheible, M. Buchholz and K. Dietmayer, “Self-Assessment for Multi-Object Tracking Based on Subjective Logic,” in *2024 IEEE Intelligent Vehicles Symposium (IV)*, Jeju Island, Korea, Republic of, 2024.

[35] T. Griebel, J. Müller, M. Buchholz and K. Dietmayer, "Adaptive Kalman Filtering Based on Subjective Logic Self-Assessment," in *2024 27th International Conference on Information Fusion (FUSION)*, Venice, Italy, 2024.

[36] A. Scheible, T. Griebel and M. Buchholz, "Self-Monitored Clutter Rate Estimation for the Labeled Multi-Bernoulli Filter," in *2024 27th International Conference on Information Fusion (FUSION)*, Venice, Italy, 2024.

[37] EVENTS Project, “EVENTS Deliverable D2.2: Full Stack Architecture & Interfaces,” 2023.

[38] J. Barr, O. Harrald, S. Hiscocks, N. Perree, H. Pritchett and S. e. a. Vidal, “Stone Soup open source framework for tracking and state estimation: enhancements and

Not officially approved by the EC

applications," in *Signal Processing, Sensor/Information Fusion, and Target Recognition XXXI (Vol. 12122, pp. 43-59). SPIE.*, 2022.

[39] F. Poggenhans, J. H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[40] EVENTS Project, "EVENTS Deliverable D4.3: Behavioural decision-making.".

[41] N. Chowdhri, L. Ferranti, F. Iribarren and B. Shyrokau, "Integrated nonlinear model predictive control for automated driving," *Control Engineering Practice,* p. 104654, 2021.

[42] "TNO Automotive MF-Tyre/MF-Swift 6.2," Helmond, 2012.

[43] A. Zanelli, A. Domahidi, J. Jerez and M. Morari, "FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control,* vol. 93, no. 1, pp. 13-29, 2020.

[44] A. Bertipaglia, D. Tavernini, U. Montanaro, M. Alirezaei, R. Happee, A. Sorniotti and B. Shyrokau, "Model Predictive Contouring Control for Vehicle Obstacle Avoidance at the Limit of Handling Using Torque Vectoring," in *International Conference on Advanced Intelligent Mechatronics*, 2024.

[45] A. Bertipaglia, M. Alirezaei, R. Happee and B. Shyrokau, "Model Predictive Contouring Control for Vehicle Obstacle Avoidance at the Limit of Handling," in *Advances in Dynamics of Vehicles on Roads and Tracks III*, 2024.

[46] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta and others, "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[47] Openstreetmap, [Online]. Available: https://www.openstreetmap.org/. [Accessed 19 12 2024].

Not officially approved by the EC