MOTORVEHICLE
UNIVERSITY OF
EMILIA-ROMAGNA

Alma Mater Studiorum Università di Bologna

MASTER'S DEGREE COURSE IN ADVANCED
AUTOMOTIVE ELECTRONIC ENGINEERING

Master's Thesis in Compliance Design of Automotive Systems

Virtual ADAS/AD ECU Validation & Design Scenario Tool in MIL and HIL environment

Author:
Gianvincenzo Daddabbo

University Supervisor:
Prof. Carlo Concari

Company Supervisor:
Ing. Roberto Rabbeni, PhD

A.Y. 2020/2021

Foreword

The thesis that will now be presented will focus on Advanced Driver Assistance Systems hereinafter referred to as ADAS.
This paper is the result of a curricular internship carried out at the Maserati Innovation Lab in Modena.
The activity started with supporting the virtual validation team, a multi-ethnic team operating on a global scale. In particular, the contribution of colleagues from Pomigliano d'Arco, Detroit and Chennai was fundamental.
Subsequently, the testing and validation phase was carried out on the electronic test benches with the help of the Maserati validation engineers.
For reasons of industrial confidentiality, it was not possible to insert in this thesis work several contents and results obtained during the tests. The topics that will be discussed, therefore, do not provide an exhaustive explanation of what the activity carried out was, but contain all the essential details for understanding it.

Abstract

Autonomous Driving is the main trend that automotive industry is following.
Starting from the first driver assistance systems, nowadays car are growing and growing in complexity, becoming a four-wheeled supercomputer that is able to perform a lot of driving actions totally autonomously. This thesis aims to describe what are the functionalities and methodologies that contributes to this amazing progress.
The entire activity is based on a tool for automatic scenario generation. In the first two chapters the context in which the tool is placed is introduced, highlighting the method by which is possible to design its input. Subsequently, the most important aspects of the tool are described, focusing on the graphical interface and the standard it must be compliant with.
Two entire chapters are devoted to the analysis of fundamental testing methodologies applied to the tool, while in the last part of the thesis, the final results are presented and discussed.

Summary

List of Figures

List of Tables

Acronyms

ACC Adaptive Cruise Control.

AD Autonomous Driving.

ADAS Advanced Driver Assistance Systems.

AEB Autonomous Emergency Braking.

API Application Programming Interface.

ASAM Association for Standardization of Automation and Measuring Systems.

ASM Automotive Simulation Models.

BSCM Brake System Control Module.

CADM Central ADAS Decision Module.

CAN Controller Area Network.

CC Cruise Control.

CoG Center of Gravity.

CPG Chelsea Proving Ground.

DIL Driver-in-the-Loop.

DOE Design of Experiment.

ECU Electronic Control Unit.

EPS Electric Power Steering.

FCW Forward Collision Warning.

FPGA Field Programmable Gate Array.

GUI Graphical User Interface.

HAS Highway Assist System.

HIL Hardware-in-the-Loop.

HMI Human-Machine Interface.

I/O Input/Output.

IC Integrated Circuit.

IDE Integrated Development Environment.

IPC Instrument Panel Cluster.

ISO International Organization for Standardization.

L1 Level 1.

L2 Level 2.

L2+ Level 2 plus.

L3 Level 3.

LED Light Emitting Diode.

LKA Lane Keeping Assistance.

MIL Model-in-the-Loop.

OEM Original Equipment Manufacturer.

OpenDRIVE Open Dynamic Road Information for Vehicle Environment.

PIL Processor-in-the-Loop.

PWB Plywood Car Buck.

RANSAC RANdom SAmple Consensus.

SAE Society of Automotive Engineers.

SIL Software-in-the-Loop.

SoC System on a chip.

SW Software.

V-ECU Virtual ECU.

VI-CRT VI-CarRealTime.

VST Virtual System for Testing.

XIL Everyhting-in-the-Loop.

XLS Excel Spreadsheet.

XML eXtensible Markup Language.

XODR ASAM OpenDRIVE.

XOSC ASAM OpenSCENARIO.

Glossary

actuator Component of a machine that is responsible for moving and controlling a mechanism or system.

algorithm Finite sequence of well-defined instructions.

bounding box Coordinates of the rectangular border that fully encloses an object.

calibration Comparison of measurement values delivered by a device under test with those of a calibration standard of known accuracy.

chip Small flat piece of semiconductor material, usually silicon, containing electronic circuits.

controller System supervisor that drives the plant to a desired state.

database Organized collection of data stored and accessed electronically.

ego-vehicle Subject connected and/or automated vehicle, the behaviour of which is of primary interest in testing, trialling or operational scenarios.

end-to-end Technique that tests the entire software product from beginning to end to ensure the application flow behaves as expected.

ethernet Family of wired computer networking technologies.

euclidean distance Length of a line segment between two points.

fellow vehicle Vehicles that are part of the traffic that affect the behavior of the ego - vehicle.

functional target Reference value with which to compare the signals under test.

hands-OFF Functionality that does not require the driver to hold the steering wheel.

hands-ON Functionality that requires the driver to hold the steering wheel.

host PC Computer or other device connected to a computer network.

interface Shared boundary across which two or more separate components of a computer system exchange information.

machine learning Study of computer algorithms that can improve automatically through experience and by the use of data.

matlab Abbreviation of "MATrix LABoratory" is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks.

offline Not in real-time.

plant Dynamic model of the system to be controlled.

protocol System of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity.

radar Short for Radio Detection and Ranging is a detection system that uses radio waves to determine the distance, angle, or velocity of objects.

real-time Process that must guarantee response times within a specified time (deadline), usually a relatively short time.

redundancy Duplication of critical components or functions of a system with the intention of increasing reliability of the system.

sensor Device that produces an output signal for the purpose of sensing of a physical phenomenon.

settling time Time elapsed from the application of an ideal instantaneous step input to the time at which the dynamical system output has entered and remained within a specified error band.

simulink MATLAB-based graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems.

standard An established norm or requirement for a repeatable technical task which is applied to a common and repeated use of rules, conditions, guidelines or characteristics for products or related processes and production methods, and related management systems practices.

stellantis Multinational automotive manufacturing corporation formed in 2021 on the basis of a 50-50 cross-border merger between the Italian-American conglomerate Fiat Chrysler Automobiles and the French PSA Group.

time-to-market The length of time it takes from a product being conceived until its being available for sale.

tolerance Permissible limit or limits of variation in a measured value.

toolchain Set of programming tools.

virtual validation Process of using a virtual model to test, simulate and verify a product's operability before creating a physical prototype.

wiring harness Assembly of electrical cables or wires which transmit signals or electrical power.

workflow Sequence of operations.

Chapter 1

Introduction

The majority of the vehicle accidents are caused by human error, which can be avoided with Advanced Driver Assistance Systems (ADAS). The primary role of ADAS is to prevent deaths and injuries by reducing the number of car accidents and the serious impact of those that cannot be avoided [1].

Essential safety-critical ADAS applications include:

- Pedestrian detection/avoidance
- Lane departure warning/correction
- Traffic sign recognition
- Automatic emergency braking
- Blind spot detection

Moreover, there are plenty of ADAS applications that just help driver monitoring and controlling the driving maneuvers and surroundings. The most common are:

- Adaptive Cruise Control
- Adaptive Light Control
- Automatic Parking
- Autonomous Valet Parking
- Driver Drowsiness Detection

ADAS solutions are partitioned into various chips, called System on a chip (SoC). These chips connect sensors to actuators through interfaces and high-performance Electronic Control Unit (ECU). That means an increasing complexity in the design of hardware architecture of the vehicles. Self-driving cars use a variety of these applications and technologies to gain 360-degree vision (bird's eye view).

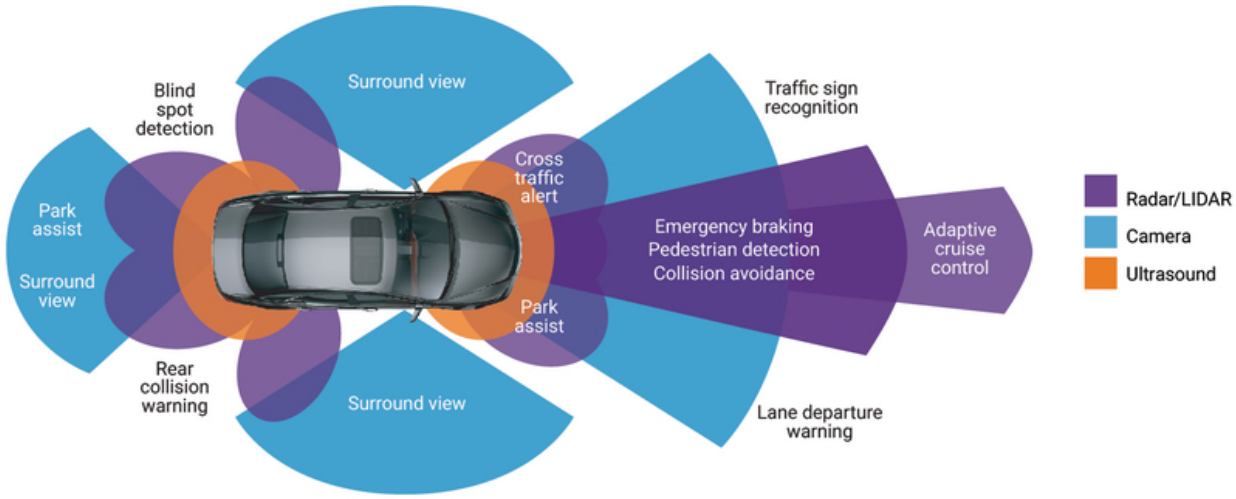


FIGURE 1.1: 360° vision achieved thanks to ADAS sensors

In 2016, as car manufacturers were moving from ADAS toward autonomy, the Society of Automotive Engineers (SAE) published its "Levels of Driving Automation". The goal was to establish common benchmarks as technology progressed toward autonomous vehicles. These levels go from 0 (no autonomy), to 5 (fully autonomous). Recently a new concept, Level 2 plus (L2+), has been introduced into these levels [2].

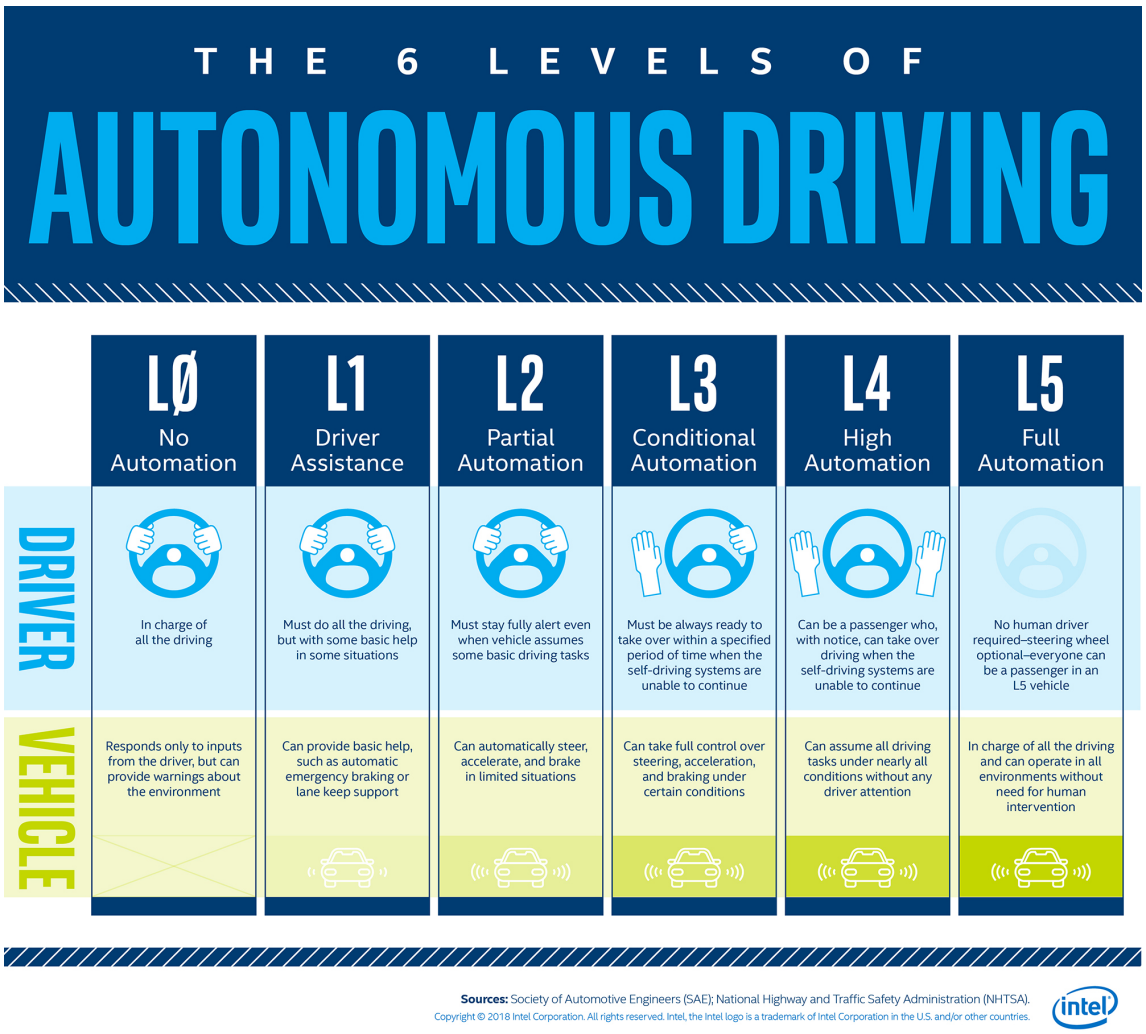


FIGURE 1.2: The six levels of Autonomous Driving

1.1 Level 2+ of Autonomous Driving

Autonomous Driving is the keyword for every automotive engineer today. Examining the established six Levels of Driving Automation reveals that the biggest gap in those levels is between Level 2 and Level 3. This is essentially the crossover from driver assist to some level of autonomy. In the jump between these two levels, liability shifts from the driver to the system. The enhanced functionality of L2+ can be seen in the Lane Keeping Assistance (LKA) feature, for example. In L2+ vehicles, map data enables lane-centering to remain effective even in areas where sensing-only lane centering systems may have a hard time; for example, in areas without visible lane marks or low-quality lane markings, such as ramps with sharp turns, junctions, roundabouts. L2+ also supports automatic lane changes by providing information such as lane-marking types and adjusting the

drive speed according to road speed/curvature. This capability is available day or night, during challenging weather conditions such as fog, low sun, heavy rain, snow or reflecting roads, despite their impact on the front camera's visibility.

FIGURE 1.3: L2+ Lane-keeping assist (left) compared with L2 (right)

In lane-keeping assist and beyond, L2+ does more than simply enhance ADAS capabilities, it delivers some new capabilities altogether. The semi-autonomous features via ADAS technology, such as hands-off driving, are considered safe under certain driving conditions, namely highways. While drivers of vehicles equipped with L2+ technology must always remain alert so they have to be in a position to take over the controls at all times, the enhanced features allow the convenience of advanced semi-autonomous driving capabilities to extend to urban areas and more [3].

Another enhanced functionality of the L2+ domain is the Autonomous Emergency Braking (AEB), a radar-based feature that automatically applies braking torque if there is a standstill vehicle in front.

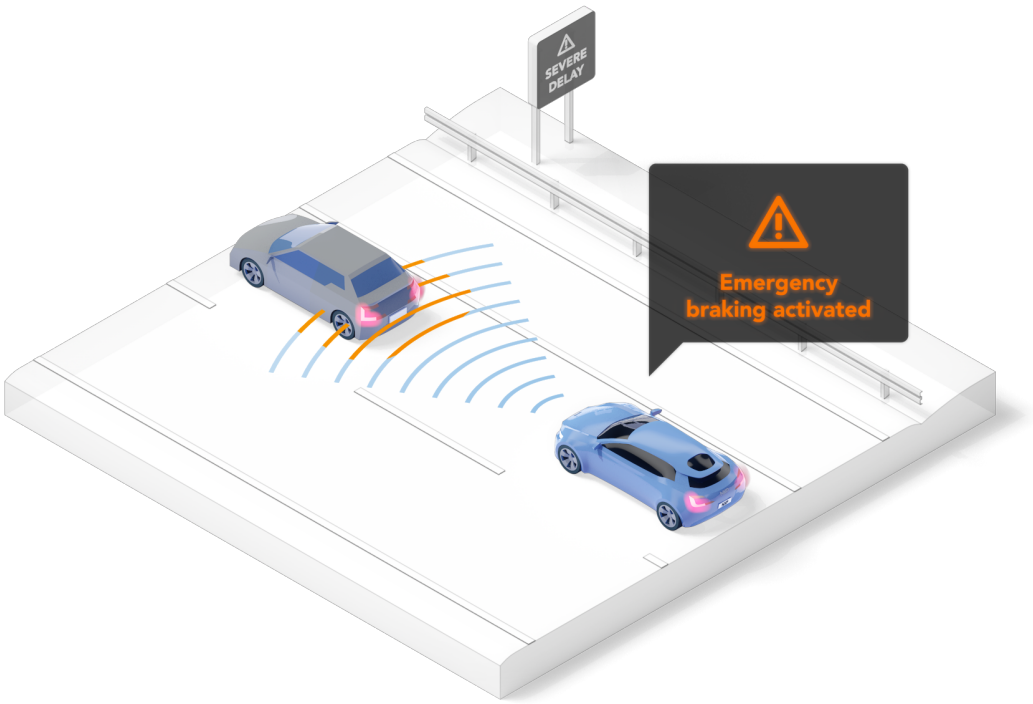


FIGURE 1.4: Higher sensitivity in forward-looking radar gives enhanced performance and improves Autonomous Emergency Braking (AEB) with Forward Collision Warning (FCW)

In conclusion, developing an L3 solution requires substantially more computing power. It requires multiple (in some cases triple) redundancies in the vehicle's sensor suite and related actuators, along with fusing huge amounts of data from multiple sensors. There are also limitations related to the Human-Machine Interface (HMI) with L3, as the driver must come back into the control loop in a very short timeframe. As shown in Figure 1.5, due to the intense competitiveness in the vehicle market, the future trend of the middle/premium OEMs will consist in progressively introducing ADAS functionalities by offering "hands-OFF" driving capabilities in their L2 + vehicles at costs still close to L1/L2, instead of adding complexity and raise the cost.

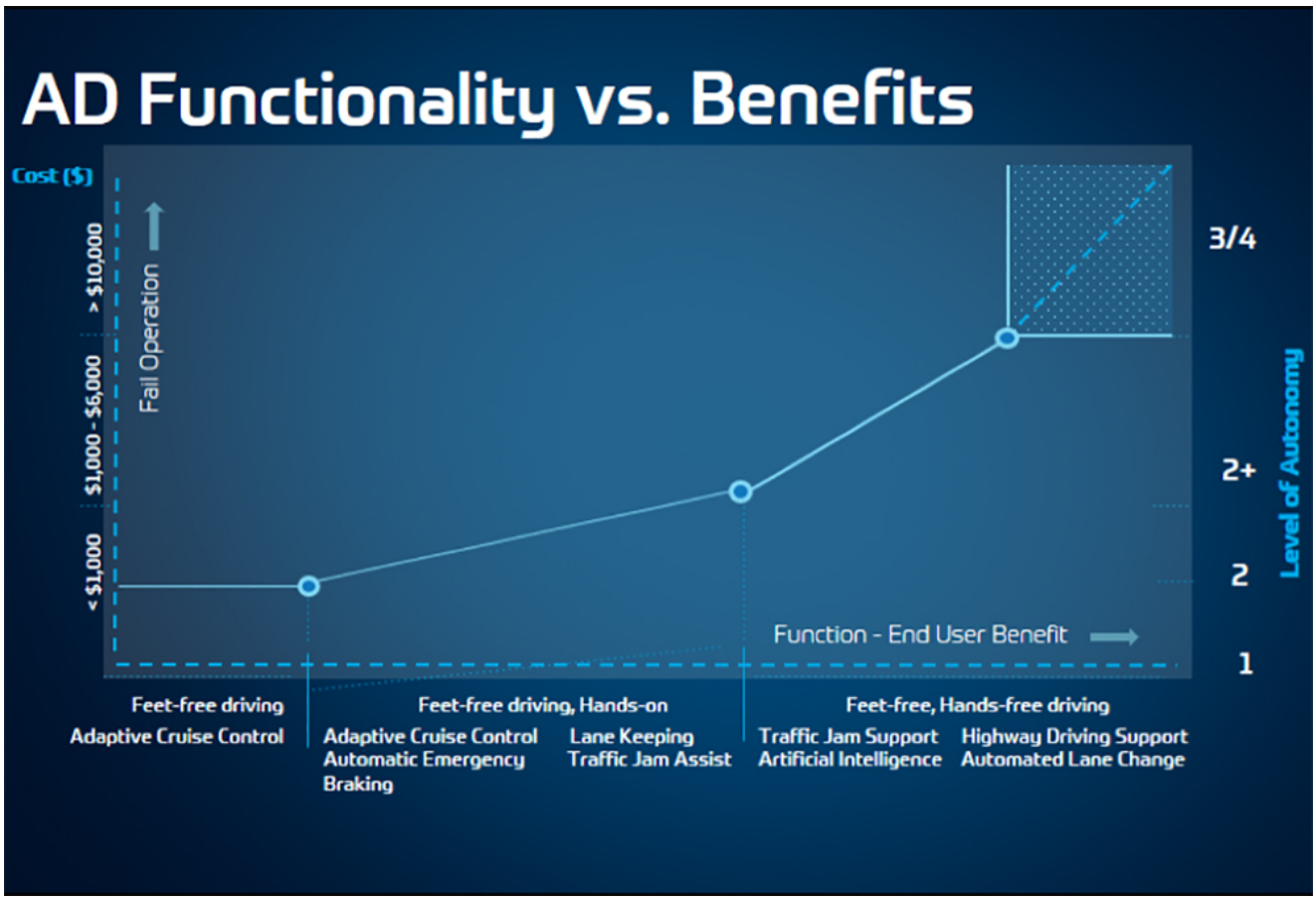


FIGURE 1.5: Enhanced SAE Level 2 automated driving systems avoid the cost, complexity and redundancy of Level 3

Chapter 2

Design of Experiment

The second chapter of this thesis aims to provide the basic knowledge necessary to understand how it was possible to generate the inputs required by the tool. Therefore the following contents constitute a premise to the next chapter.
A designed experiment is a controlled set of tests designed to model and explore the relationship between factors and one or more responses. Design of Experiment (DOE) is defined as a branch of applied statistics that deals with planning, conducting, analyzing, and interpreting controlled tests to evaluate the factors that control the value of a parameter or group of parameters. DOE is a powerful data collection and analysis tool that can be used in a variety of experimental situations.

It allows for multiple input factors to be manipulated, determining their effect on a desired output (response). By manipulating multiple inputs at the same time, DOE can identify important interactions that may be missed when experimenting with one factor at a time. All possible combinations can be investigated (full factorial) or only a portion of the possible combinations (fractional factorial). A strategically planned and executed experiment may provide a great deal of information about the effect on a response variable due to one or more factors [4].

A well-performed experiment may provide answers to questions such as:

- What are the key factors in a process?
- At what settings would the process deliver acceptable performance?
- What are the key, main, and interaction effects in the process?
- What settings would bring about less variation in the output?

An extremely simple example is the one shown in Table 2.1, called Design Matrix, where all the possible combinations of levels for each input factors are represented.

	Factor A	Factor B
Experiment 1	-1	-1
Experiment 2	-1	+1
Experiment 3	+1	-1
Experiment 4	+1	+1

TABLE 2.1: Design Matrix built with 2 factors and 2 levels (-1/+1)

Table 2.1 can be populated with any variables that might affect the result of experiment. For example, in the case we want to analyze the performance of the Adaptive Cruise Control according to the cruise speed of the ego vehicle and the road profile, the Design Matrix becomes the following.

	Ego Vehicle Speed	Upcoming Road Profile
Experiment 1	50 km/h	Straight
Experiment 2	50 km/h	Curve Left
Experiment 3	100 km/h	Straight
Experiment 4	100 km/h	Curve Left

TABLE 2.2: Simple Design Matrix for ACC

It is straightforward that adding more factors and levels, increases the number of possible combination i.e the number of experiments.

DOE has 6 steps that guide the designer through the process of choosing the goal all the way to predicting the best response [5].

1. **Describe**

   The Describe step of the experimental design framework is composed by the following substeps:

   - Identification of the responses (results of interest) and factors (constant parameters that will affect the response).
   - Identification of the goals for the experiment i.e maximize the response, hit a target or simply find which factors have an effect on the response.

2. **Specify**

   Model specification: a good model adequately describes the physical situation subject to one or more constraints.

3. **Design**

   Design generation using inputs, check if the required informations are provided.

4. **Collect**

   Data collection from the first run and record of the response values in the design matrix.

5. **Fit**

   Fitting of the assumed model to the experimental run data. Augmented design could be useful to solve possible model ambiguity.

6. **Predict (Optimize)**

   Use of a refined model to address the experimental goals. Computation of the factor levels that optimize response.

2.1 Orthogonal Array optimization method

An orthogonal array (more specifically a fixed-element orthogonal array) of s elements, denoted by $OA_N(s^m)$ is an $N * m$ matrix whose columns have the property that in every pair of columns each of the possible ordered pairs of elements appears the same number of times [6].

Orthogonal array OA_4 (2^3)

	Column No.		
Row No.	**1**	**2**	**3**
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Orthogonal array OA_8 (2^7)

	Column No.						
Row No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	0	0	0	0	0	0
2	0	0	0	1	1	1	1
3	0	1	1	0	0	1	1
4	0	1	1	1	1	0	0
5	1	0	1	0	1	0	1
6	1	0	1	1	0	1	0
7	1	1	0	0	1	1	0
8	1	1	0	1	0	0	1

TABLE 2.3: Orthogonal Array with 4 rows (left) and 8 rows (right)

Orthogonal arrays can be viewed as plans of multifactor experiments where the columns correspond to the factors, the entries in the columns correspond to the levels of the factors and the rows correspond to the experimental runs. More specifically, the N rows of an $OA_N(s^m)$ can be viewed as a subset of all the possible s^m experiments of a full factorial plan in m factors each having s test levels. Thus, an $OA_N(s^m)$ can be seen as a N/s^m fraction of a complete s^m factorial plan. For example, the four rows of the $OA_4(2^3)$ that are displayed in Table 2.3, can be viewed as a $4/2^3 = 1/2$ fraction of a complete 2^3 factorial plan.

A sub-matrix formed by deleting some columns of an orthogonal array is also an orthogonal array. Thus, by deleting certain columns of a given orthogonal array, it is

possible to generate many different plans of multifactor experiments.

It follows from the definition that an orthogonal array remains an orthogonal array when its (1) rows are permuted or (2) columns are permuted or (3) the elements within a column are permuted. When orthogonal arrays are viewed as plans of multifactor experiments, the row permutation corresponds to reordering of experimental runs, the column permutation corresponds to relabeling of factors, and the permutation of elements within a column corresponds to relabeling of factor levels. Most experimenters realize that the labels of factors, the labels of factor levels and the order of experimental runs are arbitrary, indeed, it is usually randomized. Therefore, two orthogonal arrays are defined to be equivalent if one can be obtained from the other via permutations. This experimental design allows to organize the parameters affecting the process and the levels at which they should be varies. Instead of having to test all possible combinations like the full factorial design, this optimization method tests pairs of combinations. This enables for the collection of the necessary data to determine which factors most affect the result with a minimum amount of experimentation, thus saving time and resources [7].

Chapter 3

L2+ Automation Tool

The core of this thesis is the description of the L2+ Automation Tool. In particular the following points will be discussed:

- Tool workflow and development process
- ASAM OpenSCENARIO and its contribution to Virtual validation

3.1 Tool Workflow

The main purpose of the tool is the automatic conversion of test cases in excel format into scenarios in ASAM standard with predefined road files (XODR) for L2+ ADAS validation. As shown in Figure 3.1, the tool:

1. Imports traffic conditions from the ADAS design validation test cases in XLS format
2. Creates a dSPACE ASM scenario in XOSC format for each test case based on its factors and levels and safety and performance parameters according to the ASAM Standard 1.0 (see Section 3.1.2)
3. Simulates the scenario through dSPACE Toolchain (ASM, ModelDesk, MotionDesk) and outputs simulation results which can be safely executed in the testing environment.

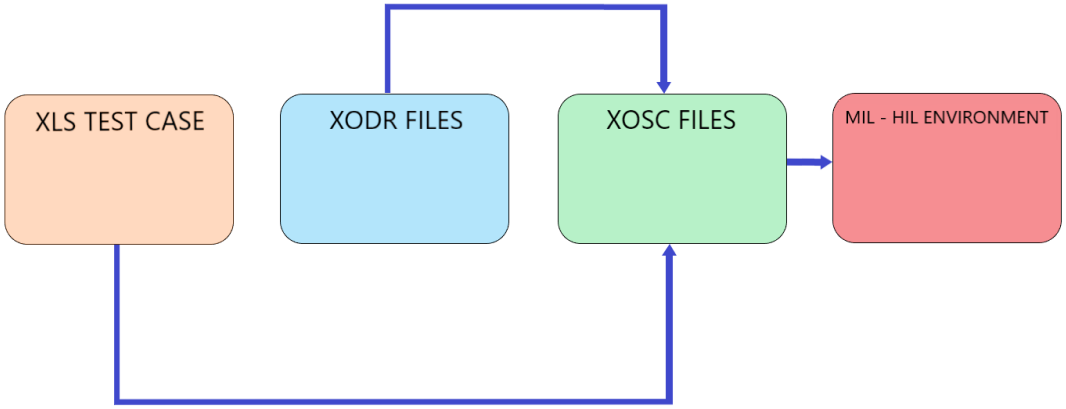


FIGURE 3.1: High Level overall tool development process

After each of the mentioned steps, a proper Matlab script produces the intermediate output needed for the next phase. In particular it's possible to recognize 3 main phases:

1. **Pre-processing**

   In this phase the input excel file, that was developed using the Orthogonal Arrays Method described in Section 2.1, is parsed by means of a pre-processing Matlab script that extracts the necessary factors and levels from the table and converts them into an XML based format called XOSC (ASAM Open Scenario).

2. **Simulation**

   The second phase deals with the run of the scenario in the simulation platform. An intermediate script collects the information coded into the XOSC file and use them to populate a ModelDesk scenario, then the scenario can be downloaded and simulated into MotionDesk.

3. **Post-processing**

   After the simulation, the recording of the signal of interest is post-processed via another script. This phase helps to verify the behaviour of the vehicles involved in the scenario.

3.1.1 GUI

One of the main shortcomings of the tool in its early stages was the lack of a Graphical User Interface. There wasn't a high level interface that made the tool more user-friendly. To face this drawback, the GUI displayed in Figure 3.2 has been developed.

The GUI has been developed using Matlab App Designer: an agile and easy Integrated Development Environment that allows to insert various items and canvases inside the interface and associate to each of them a callback, a function that executes when some action is performed.

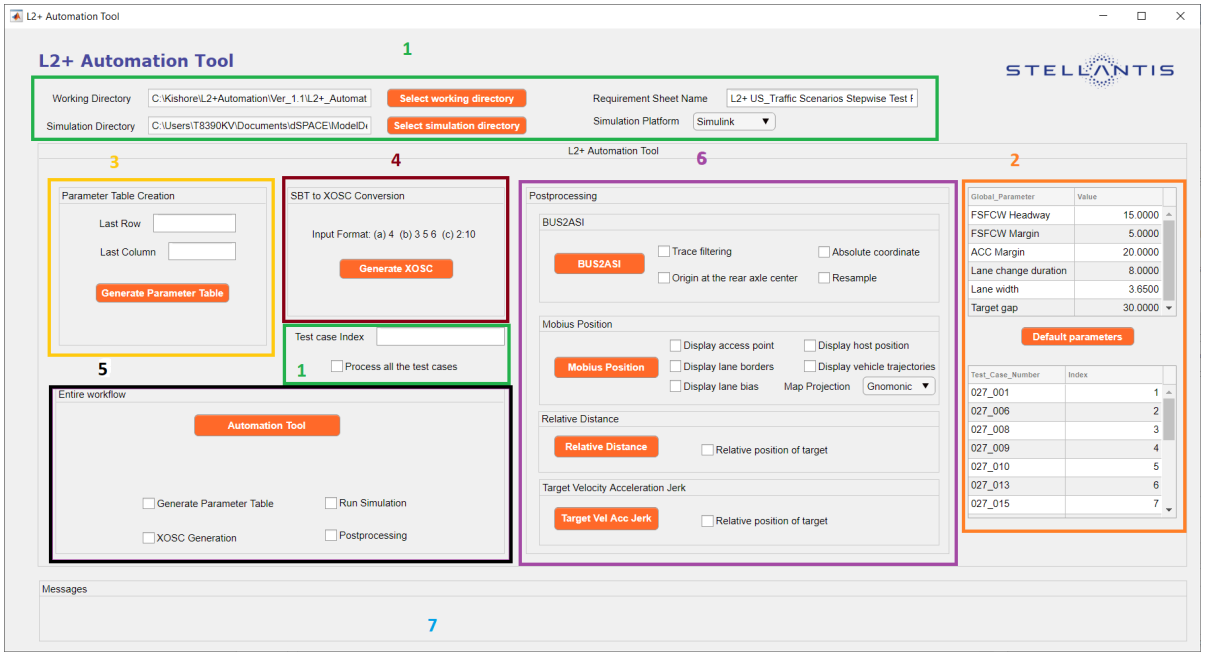


FIGURE 3.2: L2+ Automation Tool GUI

In Figure 3.2 it is possible to recognize:

1. **Input panels**: They are used to feed the tool with the proper inputs i.e. the excel requirement sheet, the index of the test case to be processed and the simulation platform used (Simulink, VEOS , SCALEXIO);

2. **Tables**: The top table allows to modify the global parameters that define the scenario and the maneuvers, instead the bottom one shows a list of the test case with their respective index to be used to feed the test case index panel;

3. **Parameter table field**: This field is used to pre-process the requirement sheet;

4. **Conversion into XOSC**: This button trigger a callback that performs the conversion from XLS to XOSC format;

5. **End-to-end Automation**: In this field it is possible to execute a part or the entire workflow of the tool;

6. **Post-processing field**: This field contains 4 subfields, each of them allows to execute a different post-processing function;

7. **Message box**: Message box is used to display the status, errors and warnings that occur while automation scripts are running.

3.2 ASAM OpenSCENARIO 1.0

During the entire development process of the tool, the main reference has been the OpenSCENARIO Standard 1.0, released by ASAM (Association for Standardization of Automation and Measuring Systems) at the beginning of 2020 [8].
OpenSCENARIO comprises the specification and file schema for the description of the dynamic content in driving simulation applications. The primary use of OpenSCENARIO is the description of complex maneuvers that involve multiple vehicles; it is used in virtual development, test and validation of driver assistance functions, automated and autonomous driving.
According to the standard, a scenario is a description of how the view of the world changes with time, usually from a specific perspective. In the context of vehicles and driving, this encompasses the developing view of both world-fixed (static) elements such as the road layout, and world-changing (dynamic) elements such as weather and lighting, vehicles, objects, people or traffic light states.

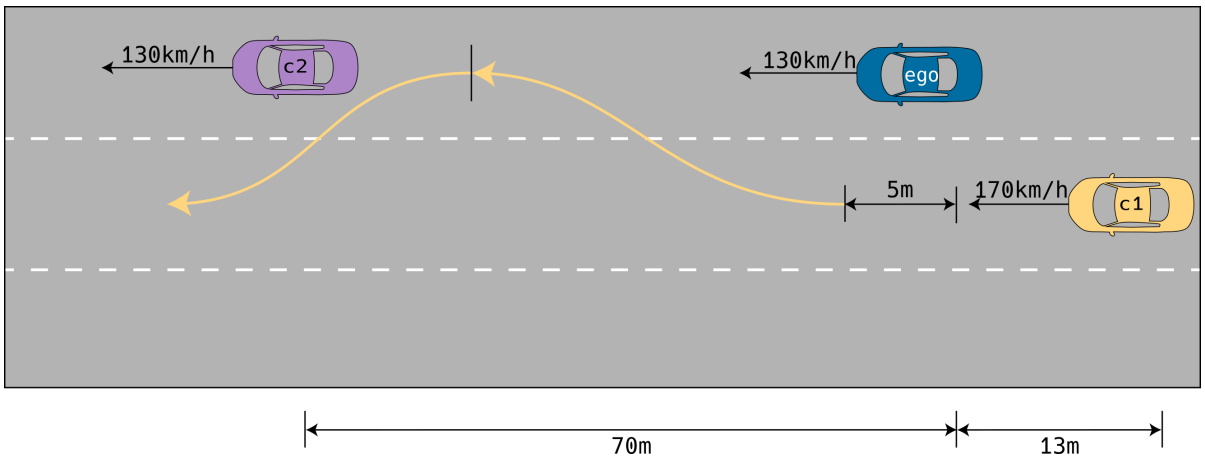


FIGURE 3.3: Scenario example: Double lane changer

OpenSCENARIO defines a data model and a derived file format for the description of scenarios used in driving and traffic simulators, as well as in automotive virtual development, testing and validation. The primary use-case of OpenSCENARIO is to describe complex, synchronized Maneuvers that involve multiple instances of Entity, like Vehicles, Pedestrians and other traffic participants. The description of a scenario may be based on driver Actions (e.g. performing a lane change) or on instances of Trajectory (e.g. derived from a recorded driving Maneuver).
The standard is used to derive XML based schema files i.e. the XOSC format file used in the tool, moreover it can be used together with road network descriptions defined according to the standard ASAM OpenDRIVE (XODR) in a way that they define the entire content required to describe the virtual world for driving simulation, virtual test, development and validation.

In a simulation context a complete scenario is comprised of the following parts:

- Static environment description, including:
  - Logical road network
  - Optionally physical and geometric road and environment descriptions
- Dynamic content description, including:
  - Overall description and coordination of behavior of dynamic entities
  - Optional behavior models of dynamic entities

```
    <RouteCatalog>
        <Directory path="../Catalogs/Routes"/>
    </RouteCatalog>
</CatalogLocations>
<RoadNetwork>
    <LogicFile filepath="../xodr/FCA_ProvingGrounds_0429.xodr"/>
</RoadNetwork>
<Entities>
    <ScenarioObject name="Ego">
        <CatalogReference catalogName="VehicleCatalog" entryName="$EgoVehicle"/>
    </ScenarioObject>
    <ScenarioObject name="T1">
        <CatalogReference catalogName="VehicleCatalog" entryName="$Target1"/>
    </ScenarioObject>
    <ScenarioObject name="T2">
        <CatalogReference catalogName="VehicleCatalog" entryName="$Target2"/>
    </ScenarioObject>
    <ScenarioObject name="T5">
        <CatalogReference catalogName="VehicleCatalog" entryName="$Target5"/>
    </ScenarioObject>
 </Entities>
<Storyboard>
    <Init>
        <Actions>
            <Private entityRef="Ego">
                    <PrivateAction>
                        <RoutingAction>
                    <AssignRouteAction>
                                <CatalogReference catalogName="RouteCatalog" entryName="$Route"/>
                                </AssignRouteAction>
                        </RoutingAction>
                    </PrivateAction>
                <PrivateAction>
                    <LongitudinalAction>
                        <SpeedAction>
                            <SpeedActionDynamics dynamicsDimension="time" dynamicsShape="step" value="0"/>
                            <SpeedActionTarget>
                                <AbsoluteTargetSpeed value="$Ego_speed"/>
                            </SpeedActionTarget>
                        </SpeedAction>
                    </LongitudinalAction>
                </PrivateAction>
                <PrivateAction>
                    <TeleportAction>
                        <Position>
                            <LanePosition laneId="$Ego_laneID" offset="$Ego_offset" roadId="$Ego_roadID" s="$Ego_s"/>
```

FIGURE 3.4: XOSC file example. Note the hierarchical structure and the main classes of objects: Road Networks, Entities, Storyboard

At last ASAM defines a standard for the communication between test automation tools and test benches called XIL API [9]. This API have been used a lot during the validation phases of the tool beacuse they allow to decouple the test software from real and virtual test systems. This makes it easy to reuse test cases for different test systems. The standard also supports test benches at all stages of the development and testing process: Model-in-the-Loop (MIL), Software-in-the-Loop (SIL), Processor-in-the-Loop (PIL), Hardware-in-the-Loop (HIL).

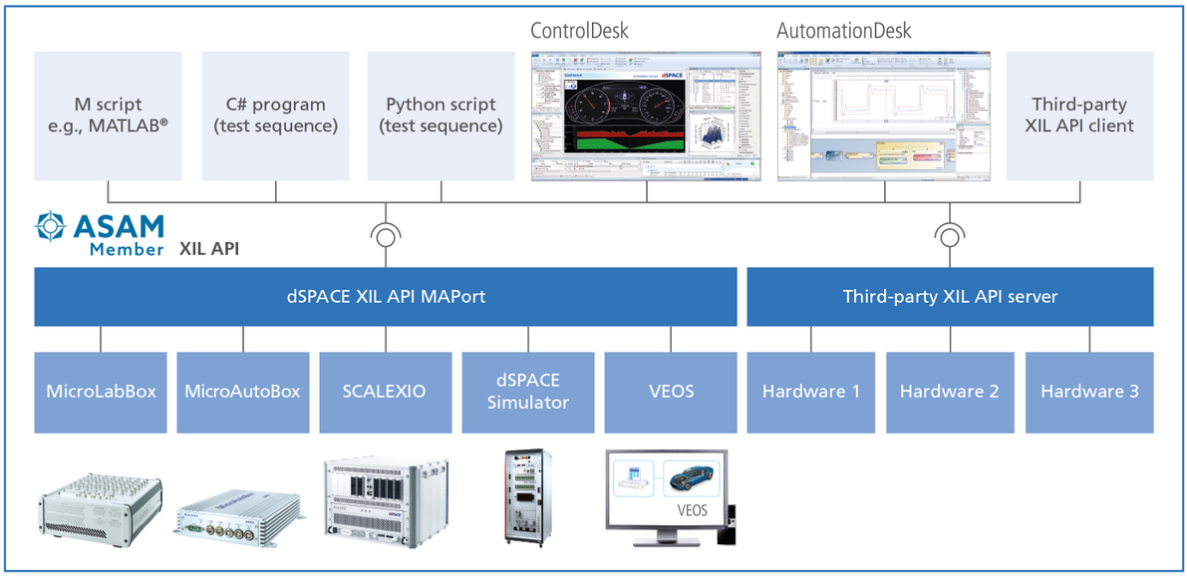


FIGURE 3.5: Support of XIL API in the dSPACE toolchain

3.2.1 Contribution to Virtual Validation

The high innovation potential of modern driver assistance systems is leading to growing volumes of functionality, which are increasingly being assigned to ECU networks. To meet the necessary quality goals and deadlines, the networked functions have to be validated on a PC in interaction with plant models and virtual ECUs at an early phase during software creation [10].

Scenario descriptions are essential for testing, validating and certifying the safety of driver assistance systems and autonomous driving cars. With the help of OpenSCENARIO, large numbers of critical situations can be run across various simulators. Thus, compared to road testing in real traffic, virtual validation presents remarkable benefits:

- Reduced amount of driven test kilometers in field tests
- Reduced time-to-market
- Reduced costs
- Better product or machine performance
- Saving resources: drivers, fuel, materials

Shorter and shorter development times require to make an extensive and methodological use of virtual validation in order to save time, money and in particular to face the current shortage of IC (Automotive-rated) [11]. Through virtual validation, companies can simulate batteries and electronic design systems, test billions of kilometers of driving and millions of lines of code without ever manually building a prototype.

As functional safety standards like ISO 26262 increase in complexity and more proof is required for product certification, virtual validation is expected to grow in need and use over the next decade.

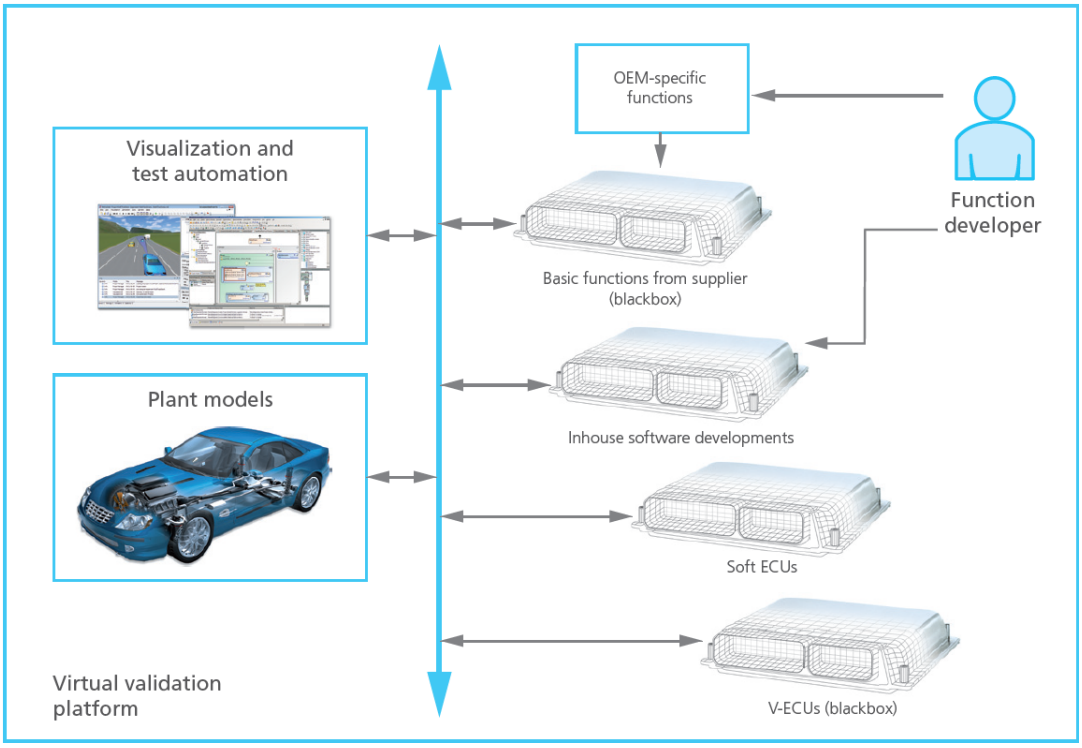


FIGURE 3.6: Virtual validation of networked driver assistance functions on a PC

Chapter 4

Model in the Loop

This chapter will be entirely dedicated to the description of the first testing methodology performed with the L2+ Automation Tool i.e. MIL testing.

4.1 Validation with ModelDesk – MotionDesk

Generally Model-in-the-Loop (MIL) simulation requires the following steps:

- Development of a model of the actual **plant** in a simulation environment such as Simulink, which captures most of the important features of the hardware system

- After the plant model is created, develop the **controller** model and verify if the controller can control the plant as per the requirement

In this way it's possible to test the controller logic on the simulated model of the plant and check if it works as desired.

As discussed in section 3.1, the tool automatically populates ModelDesk environment. Figure 4.1 shows how to implement the Double Lane Change maneuver displayed in Figure 3.3.

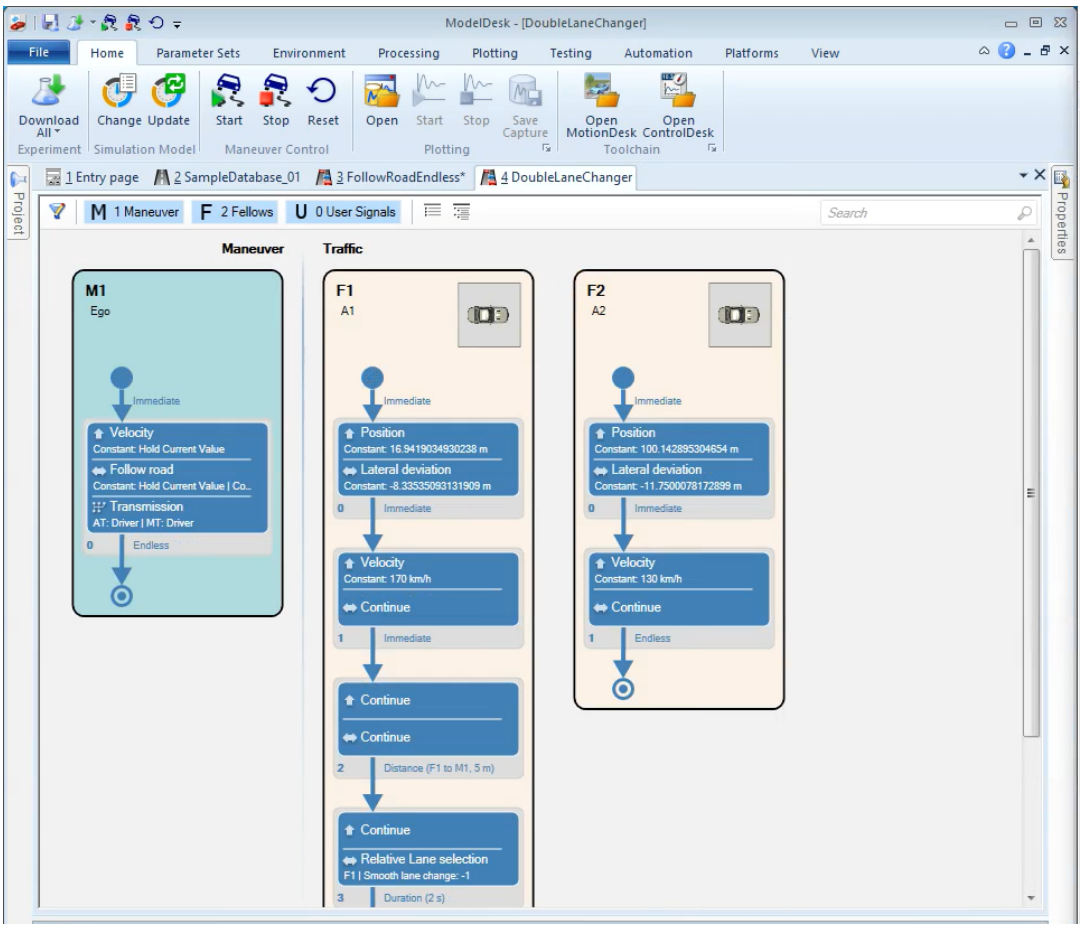


FIGURE 4.1: Example of a maneuver in dSPACE ModelDesk

Notice that ModelDesk classifies the entities of the scenario in 2 categories:

- **Maneuver**: Behaviour of the ego-vehicle
- **Traffic**: Behaviour of the fellow vehicles

Each entity is described by a set of sequences that compose a segment, thus a chain of segments define the entire maneuver of a vehicle. Looking back at the GUI in Figure 3.2, the Global Parameter Table at the rightmost side of the interface, allows to modify the parameters directly in ModelDesk. In order to run and simulate a virtual scenario ModelDesk requires:

1. **Road**: The XODR file that describe the road or the road network topology (number of lanes, lane width, length)
2. **Parameter Set**: The set of parameters that affect the dynamics of the vehicle and the environmental condition

3. **Scenario**: Set of maneuvers performed by each vehicle

Once all the elements have been gathered the simulation is ready to be run on MotionDesk. In this phase the actual MIL validation takes place, as a matter of fact in MotionDesk is possible to download all the elements listed before from ModelDesk in order to visualize the animation of the virtual scenario.

Thanks to MotionDesk is possible to monitor real-time the evolution of the simulation and check the correctness of the maneuver offline on a PC-based simulation.

As an example, if the simulation aims to validate the ACC performance, the animation shows moment by moment the maneuver performed by the vehicles; in this way results very easy to recognize when the ACC is engaged and what impact it has on the vehicle dynamic.

Figure 4.2 shows a typical MotionDesk animation, it is worth noting the following elements:

- **Ego-Vehicle**: Displayed in blue, along with 2 arrows that indicate the amount of lateral acceleration the vehicle is subject to

- **Fellows**: Displayed in white, these are the vehicles of the traffic

- **Road**: The road profile is exactly the one described in the XODR file: number of lanes, lane width, upcoming road and lines topology (dashed, continuous) are precisely represented

- **Gadget**: Numerous gadget helps the tester to access important information about the dynamic of the vehicle such as: tachometer, driven distance, maneuver time

- **Perspective**: Point of view by which the scenario is observed: frontal, lateral, bird's eye view, rear

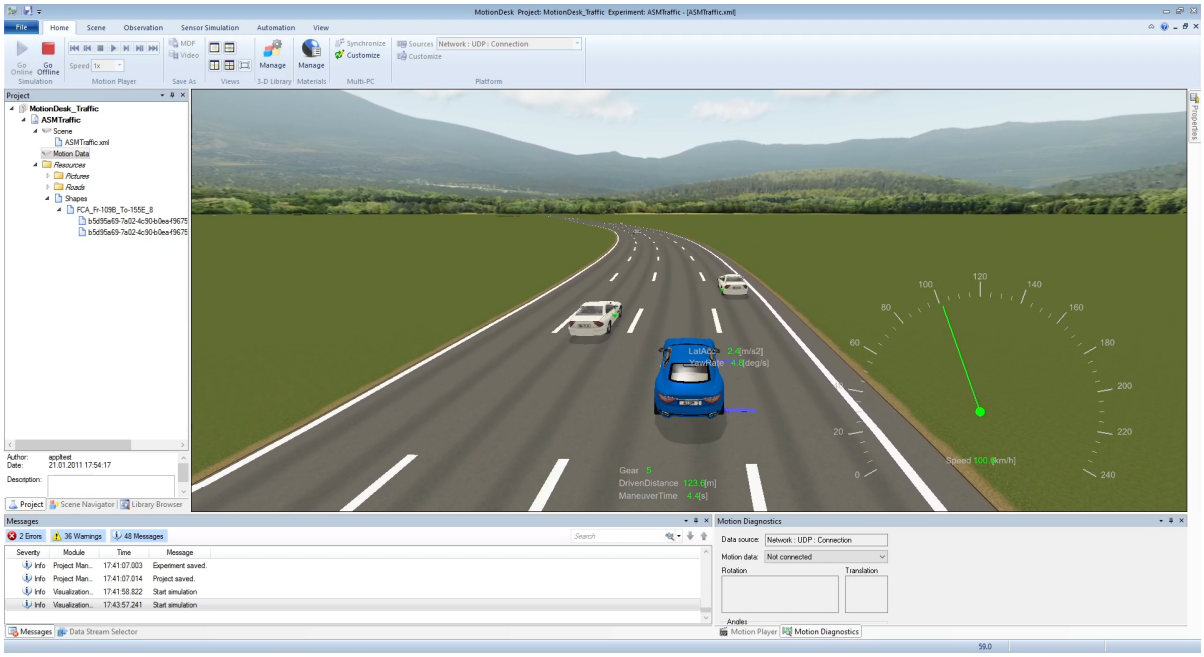


FIGURE 4.2: Example of animation in dSPACE MotionDesk

4.1.1 ASM Model

Automotive Simulation Models (ASM) are Real-time models for vehicle development [12]. This open Simulink model has been the core of the model-based function development of the L2+ Automation Tool. ModelDesk is substantially a graphical interface that allows to parameterize all the variables present in such model and before running the animation in MotionDesk the ASM model is compiled and executed.

In general ASMs are a tool suite which consists of simulation models for automotive applications that can be combined as needed. The models support a wide spectrum of simulation areas and is a flexible solutions to match any needs:

- Combustion engines (gasoline, diesel)
- Vehicle dynamics (passenger car, truck, trailer)
- Electric components (motors, batteries, loads)
- Vehicle environment (road traffic, objects, traffic signs)

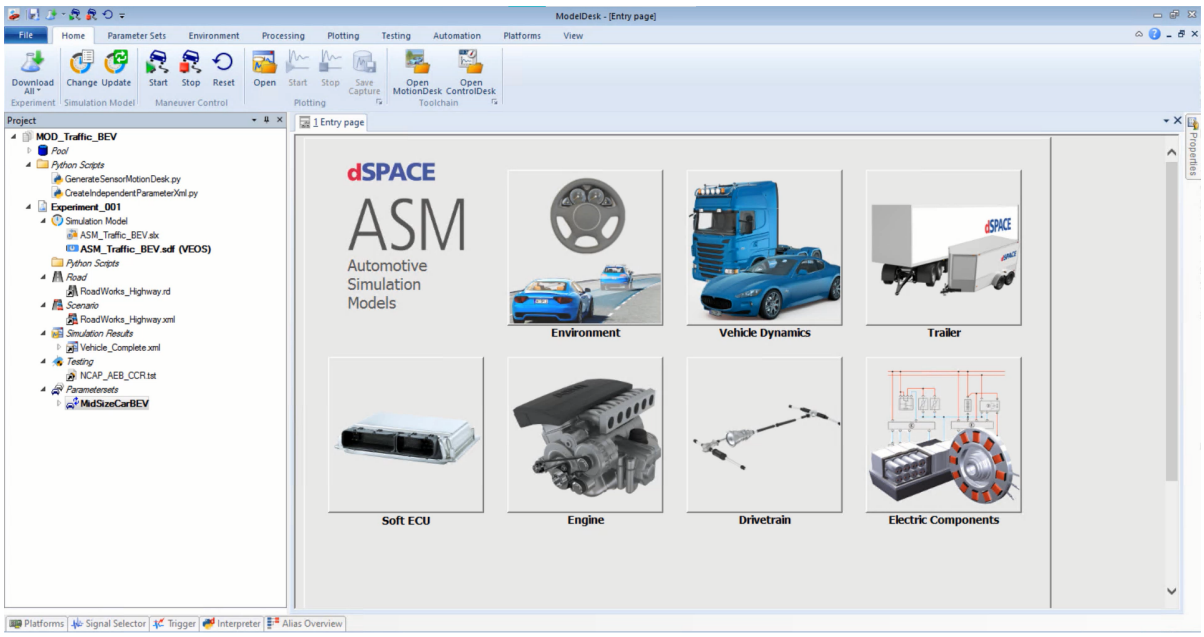


FIGURE 4.3: ASM Models interface in ModelDesk

One of the main benefits of ASM is that the implementation of each model is open and traceable right down to the Simulink basic block level, so it is easy to supplement, replace or customize components with customer-specific models. This means that the properties of each model can be optimally adapted to individual projects. The standardized interfaces of the ASMs make it easy to extend models and even create entire virtual vehicles.

4.2 Validation with VEOS – ControlDesk

dSPACE VEOS is a PC-based simulation platform for ECU software validation in many development stages. It lets the user simulate a multitude of different models: from function models to networks of virtual ECUs, bus systems and vehicle models, even in early phases of development. The VEOS simulation platform provides reliable results independently of the simulation hardware, and it can be used on different PC operating systems as well as in cloud infrastructures.

VEOS has been used as second proof of the validity of the model and the scenarios generated by the tool.

Building the Simulink ASM model automatically generate code that could be exploited to perform SIL testing through VEOS. In this case talk about Software-in-the-

loop testing is inappropriate since it suppose to have all the code available: both embedded ECUs code and model code. For this reason VEOS simulation has been used as counter-proof to check deeply the validity of the maneuver and the scenario still remaining in MIL environment.

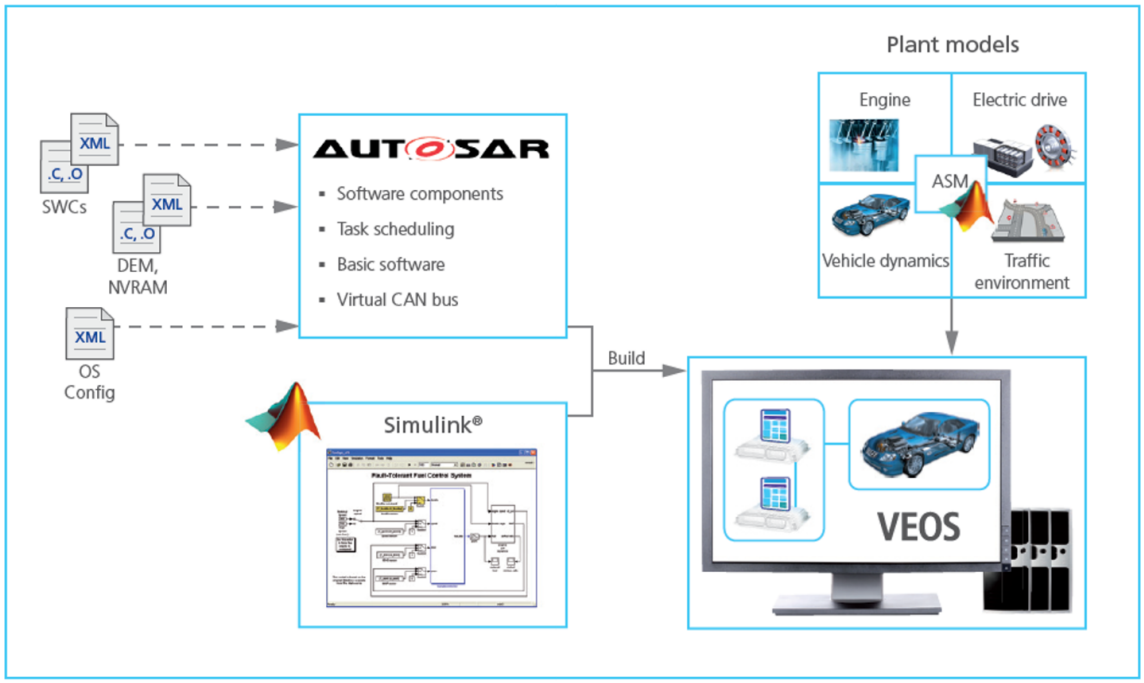


FIGURE 4.4: VEOS Simulation Platform

Another fundamental SW deployed is dSPACE ControlDesk. Thanks to ControlDesk is possible to load the compiled version of the ASM model and directly access to all the signal/buses. ControlDesk is a very effective tool for monitoring more signals at the same time using different layouts. The key benefit is that one or more gadgets (gauges, sliders, LEDs, indicators) can be associated to each signal and according to the type of signal: input or output, they can be set or get respectively.

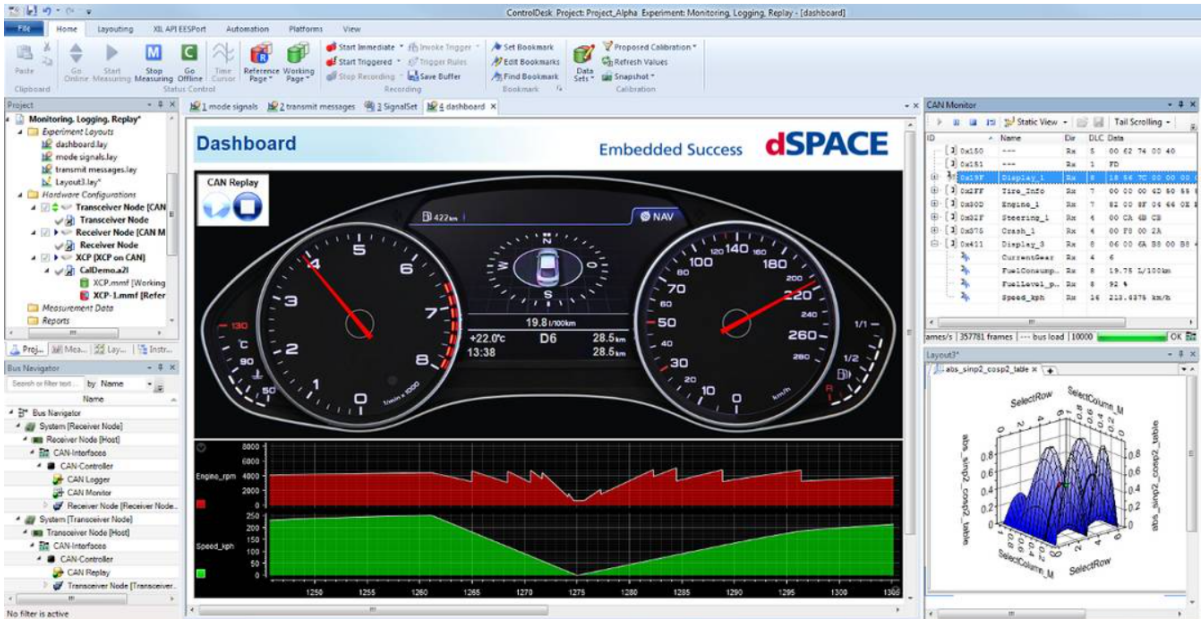


FIGURE 4.5: An example of ControlDesk layout

Summarizing, the main benefit deriving from using VEOS platform together with ControlDesk is that this solution allows to save a lot of time coming from the compilation of the model, in fact being fed with a pre-compiled file, all the signals and variables are just a click away.

Chapter 5

Hardware in the Loop

In the previous chapter the first testing methodology has been discussed. The fifth chapter of this thesis is dedicated to the description of a more challenging methodology i.e. HIL testing.
In general, testing control algorithms can be time-consuming, expensive, and potentially unsafe if it's testing against the real system. To remain competitive and deliver high-quality controller software, test engineers have replaced traditional testing methods with Hardware-in-the-Loop (HIL) testing [13].

Hardware-in-the-Loop testing allows the user to verify the plant/controller design without the complete system hardware. Relying on a real-time plant simulator that acts as a digital twin of the whole system or parts of the system gives several benefits from practicality to lower costs.

5.1 SCALEXIO Real-time system

The modular dSPACE SCALEXIO Real-time system has been used for the HIL testing phase of the scenarios generated by the Automation Tool [14].

The choice of this particular machine was influenced by its many benefits:

- Scalable to any computation and I/O requirements
- Suitable for laboratory as well as in-vehicle applications
- High-performance processor technology for real-time calculation of large and complex simulation models

- Comprehensive, precise, and fast I/O capabilities based on FPGA technology

SCALEXIO was directly connected with the ADAS Plywood Car Buck (PWB) and allowed to perform an online simulation with physical hardware. Unfortunately the physical ECUs present on the PWB belong to an L2 architecture instead of an L2 +, for this reason a couple of scenarios were discarded due to lack of adequate hardware. In particular the physical ECUs are:

1. Central ADAS Decision Module (CADM)
2. Brake System Control Module (BSCM)
3. Instrument Panel Cluster (IPC)
4. Electric Power Steering (EPS)

All the ECUs enumerated are connected to each other and to SCALEXIO by a wiring harness and communicate via CAN protocol.
The SCALEXIO system, connected via Ethernet to the Host PC was able to perform the following tasks:

- Generate the sensor signals which are required by the ECUs
- Measure the signals which are generated by the ECUs for actuator control
- Connect the output signals of the ECUs to loads (internally in the rack or externally) to simulate the actuators
- Receive bus signals which are sent by the ECUs and send bus signals to the ECUs
- Simulate virtual ECUs (V-ECU) not present in the PWB
- Provide the battery voltage to the ECUs
- Simulate electrical errors in the wiring of the ECUs

The behavior of the controlled system is specified by a real-time application which runs on the SCALEXIO processing hardware. The real-time application is created on the host PC and downloaded to the SCALEXIO system via Ethernet and, similarly to what happened with VEOS, its execution can be managed with ControlDesk.

FIGURE 5.1: Scalable SCALEXIO solutions

5.2 VST framework for automatic testing

During the execution of the tests, the VST platform was of fundamental support. Virtual System for Testing (VST) is a test automation framework developed by Stellantis that exploits the ASAM XIL API discussed in section 3.2.

VST stands upon the following tools:

- **Portal HIL**: A web portal that schedules and manages the queue of execution of each test
- **Test Platform Studio**: A desktop IDE for writing test maneuver step-by-step
- **PanDESK**: A GUI that shows test execution progress
- **Database**: A physical centralized database that stores Test Automation core Platform

VST framework is based on several operating principles:

- A centralized system accessible at anytime from anywhere 24/7
- A role and data segregation based system

- Standard metrics for test quality and coverage

- A remote test execution, scheduling and result analysis system

- A keyword based test design methodology (high-level abstraction layer)

- Flexible to any kind of test: Diagnostic, functional, stress test

In particular VST allows to load all the variables contained into real-time application in the centralized database. Then Test Platform Studio extracts the variables saving them in a data dictionary and links each of them to a specific application as keywords. This resulted very helpful during test case creation: it's been very easy to choose which I/O signal to record and manipulate and create trigger for a specific step. Figure 5.2 shows an example of test maneuver performed in HIL environment.
Every test must contains three sections:

- **Init**: The initialization section is used to bring the simulation environment to the initial conditions necessary to start the actual test (e.g. turn ON the Battery)

- **TP**: The TP section is where the actual test is executed, each step of the test maneuver is executed sequentially. In this section, after turning ON the engine and downloading the scenario, the ModelDesk maneuver is executed and visualized in MotionDesk

- **Reset**: The reset section brings the simulation environment back to the original state, thus switches OFF the engine and disconnects the battery.

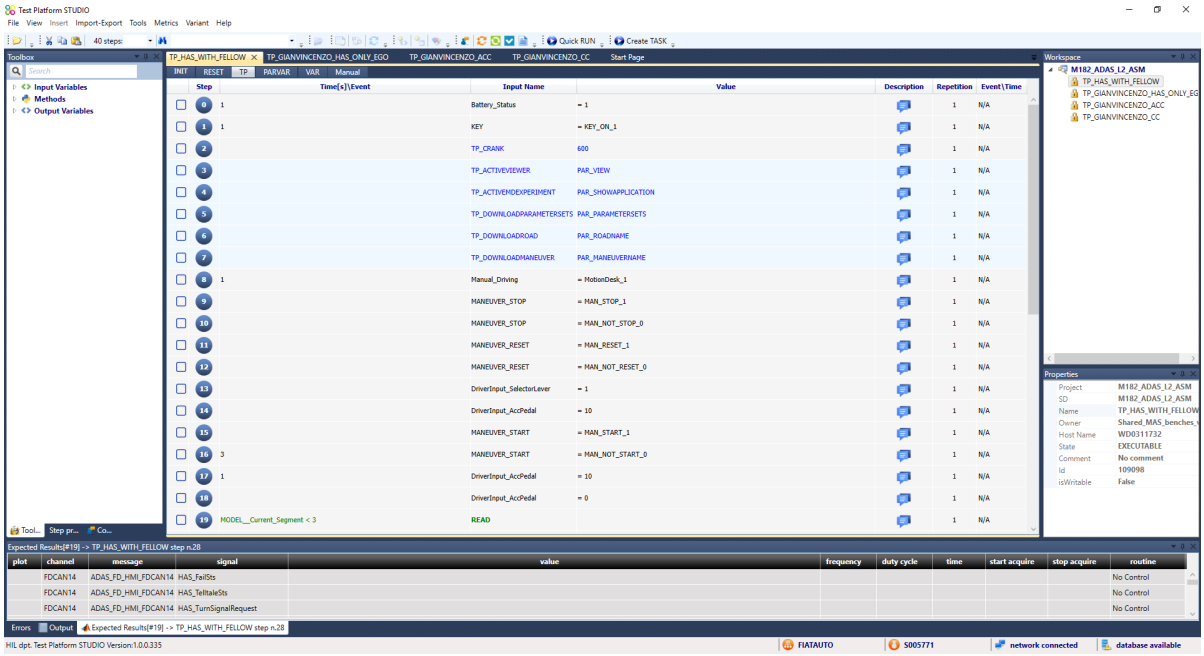


FIGURE 5.2: Test Platform Studio IDE

The core functionality of Test Platform Studio is the possibility to insert post-processing functional targets on one or more variables.

The checks are carried out after the test execution by analyzing the data acquisition recorded during the execution. Every step must meet specific timing constraints, pre-computed and verified during the execution in order to ensure the effective duration of the steps in the face of, for example, a PC load due to stall or slowdown.

The post-processing routines extract from the entire signal acquisition only the segment in which the user has requested a certain check, and on that segment it checks all the samples with respect to the target specified by the user.

With this approach, that will be described in details in the next chapter, has been possible to validate numerous ADAS functionalities. As a matter of fact VST automatically generates a report accessible via PortalHIL in which every post-processed signal is plotted with its respective tolerance range.

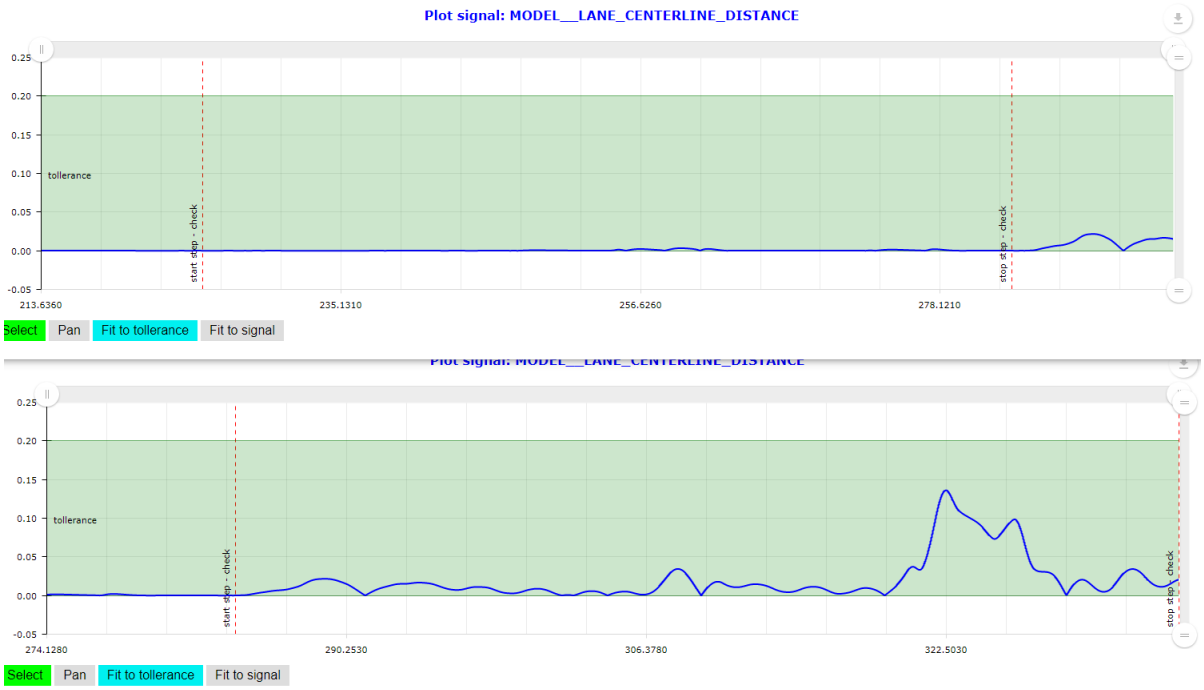


FIGURE 5.3: Example plots of a post-processed signal accessible through PortalHIL

Chapter 6

Results and Discussion

The second-last chapter is entirely focused on the discussion of the results obtained during the two testing methodologies presented in previous chapters.
From the whole set of scenarios that L2+ Automation Tool automatically generates, only 4 of major relevance have been tested both in MIL and HIL environment. These scenarios have been properly chosen in order to stimulate as much as possible the critical ADAS L2 ECU present on the PWB and the machine learning algorithm of the CADM. The scenarios are:

1. **Cruise Control**

2. **Adaptive Cruise Control**

3. **Highway Assist**

4. **Highway Assist with fellow**

Each of the scenarios run on the same racetrack i.e. Chelsea Proving Ground located 60 miles away from Detroit. As mentioned in Section 3.2 the road file has been imported in ModelDesk thanks to an XODR file. Figure 6.1 shows the shape of the proving ground, in particular all tests were performed in the innermost oval (see the blue landmark).

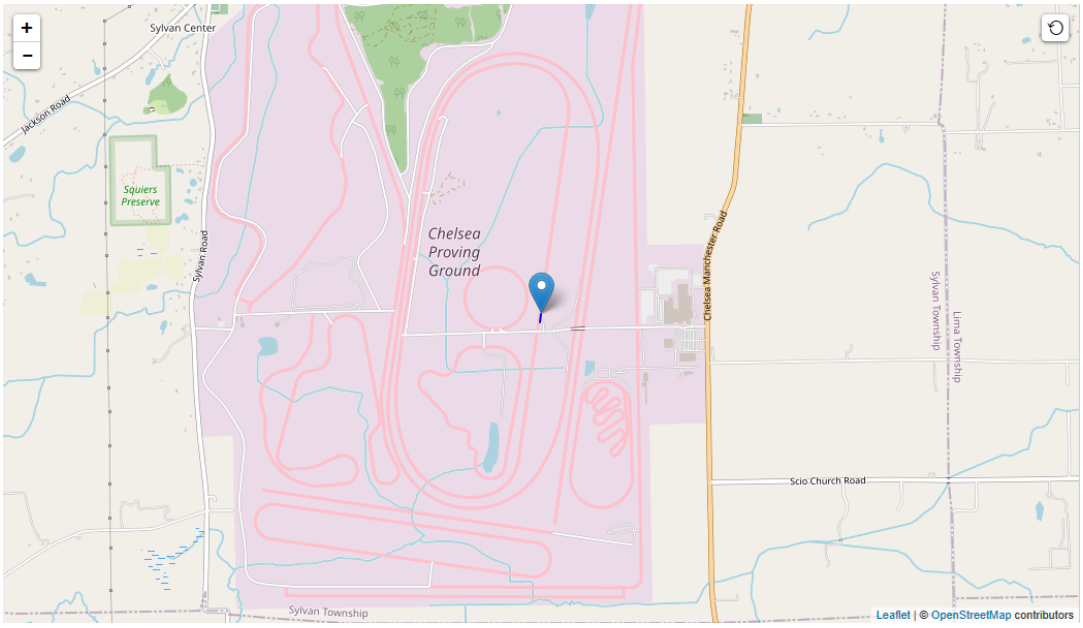


FIGURE 6.1: Top view of the Chelsea Proving Ground

As a preliminary step, before the HIL test, a demo scenario has been run: a straight highway with 3 lanes allowed to calibrate the camera connected to the CADM and let the algorithm to "learn" how to detect the lanes inside a roadway. After ten minutes of calibration, everything was fine and the actual tests could begin.

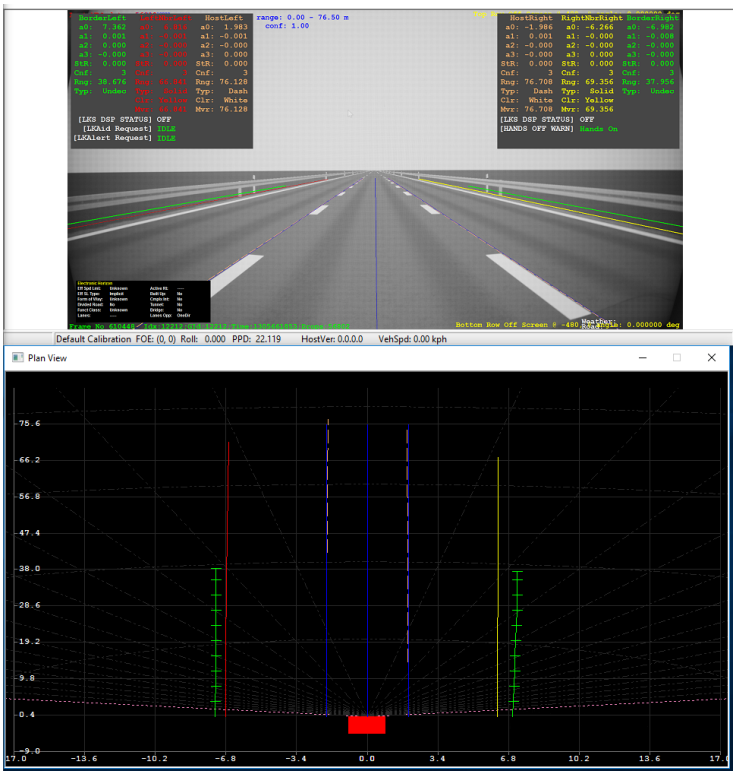


FIGURE 6.2: Calibration of the machine learning algorithm

Focusing on Figure 6.2 it is possible to distinguish the following entities:

- **Solid blue lines**: Lane centerline and its right/left edges
- **Solid yellow line**: Right lane edge
- **Solid red line**: Left lane edge
- **Solid green line**: Eventual guardrails or roadway borders
- **Dashed pink line**: Camera angle of view
- **Red rectangle**: Ego-vehicle position

It is worth noting that the horizontal and vertical scale allows to evaluate the lane width and length respectively.

6.1 Cruise Control Scenario

The first scenario deals with a single vehicle: the ego-vehicle.

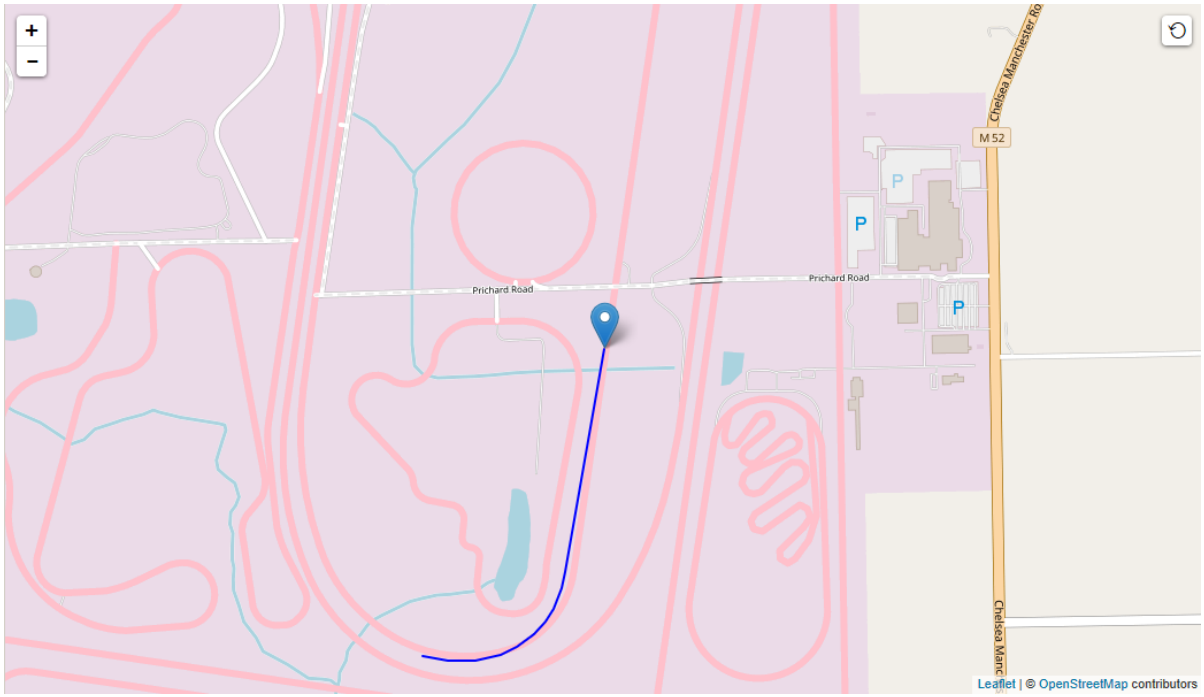


FIGURE 6.3: Ego-vehicle trajectory in CC scenario

The goal of this test was to validate the Cruise Control (CC) functionality and for this purpose 2 functional targets were checked in post-processing phase:

Functional target	Low level description	Validation method
Lane Centering: Upper bound on the lateral deviation from lane centerline	Inequality constraint: $e_{lat} = sqrt[(X_{road} - X_{ego})^2 + (Y_{road} - Y_{ego})^2]$ <= 0.2 m	Check in each point that the euclidean distance between the ego-vehicle position and the lane centerline is not greater than 0.2 m
CC Set Speed: Relative tolerance on the cruise control set speed signal	Inequality constraint: $\|v_{ego} - v_{CC}\|$ <= 3 km/h where: v_{CC} = 40 km/h	When the Cruise Control is engaged the ego-vehicle shall maintain a longitudinal speed of 40 +/- 3 km/h.

TABLE 6.1: Cruise Control functional targets

Figure 6.4 shows how the CADM algorithm have detected the lanes and perceived road curvatures during the execution of CC scenario.
The algorithm is based on an iterative procedure that, in real-time, processes every single frame. Usually the lane detection algorithm is based on the following steps [15]:

1. **Camera calibration**: Remove possible distortion
2. **Perspective transformation**: Method to see the same scene from different viewpoint and angles
3. **Grayscaling**: For this application, grayscale images are enough to perform machine learning tasks
4. **Gaussian smoothing**: Used to reduce image noise
5. **Canny Edge Detection**: Used to identify the edges of an object in an image
6. **Mask region of interest**: Select only the area that may contain the lanes
7. **RANdom SAmple Consensus (RANSAC)**: Algorithm used to fit lane lines and curvature radius calculation

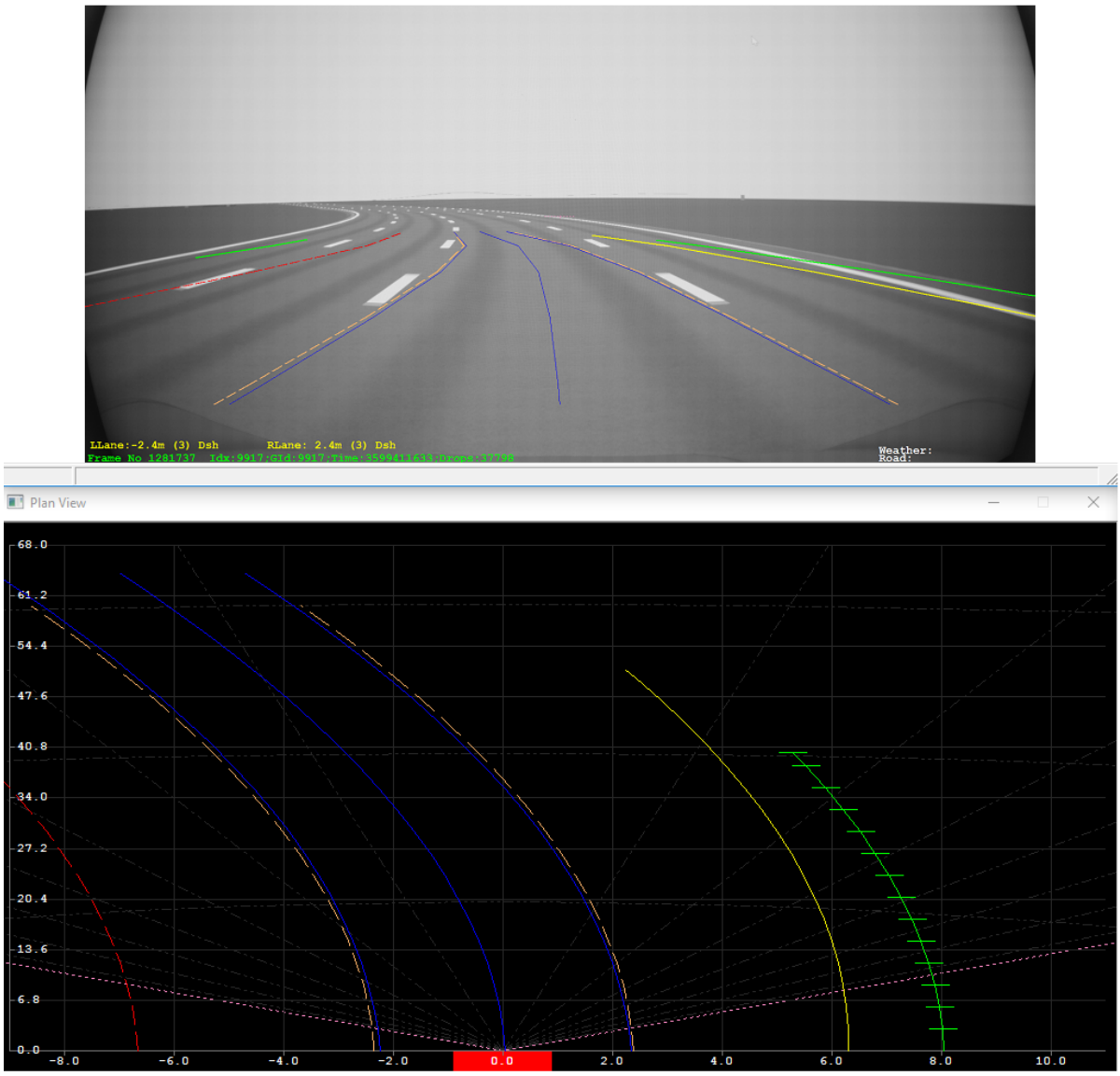


FIGURE 6.4: Lane detection algorithm stimulated by the CC scenario

The results obtained after the run of this first scenario are presented in the 2 figures below.

FIGURE 6.5: Overall lateral deviation profile (CC Scenario)

Figure 6.5 represent the time behaviour of the *LANE CENTERLINE DISTANCE* signal when the CC functionality is engaged. Is it possible to observe that the signal never exceed 0.05 m, therefore the target successfully passed. This is probably due to the smoothness of the scenario: in fact the portion of road on which the ego-vehicle drives is a circle with a high curvature radius that after 800 m turns into a straight way. Moreover the ego speed is relatively low and, for this considerations, the vehicle manages to follow almost perfectly the profile of the road keeping the center of the lane at any time.

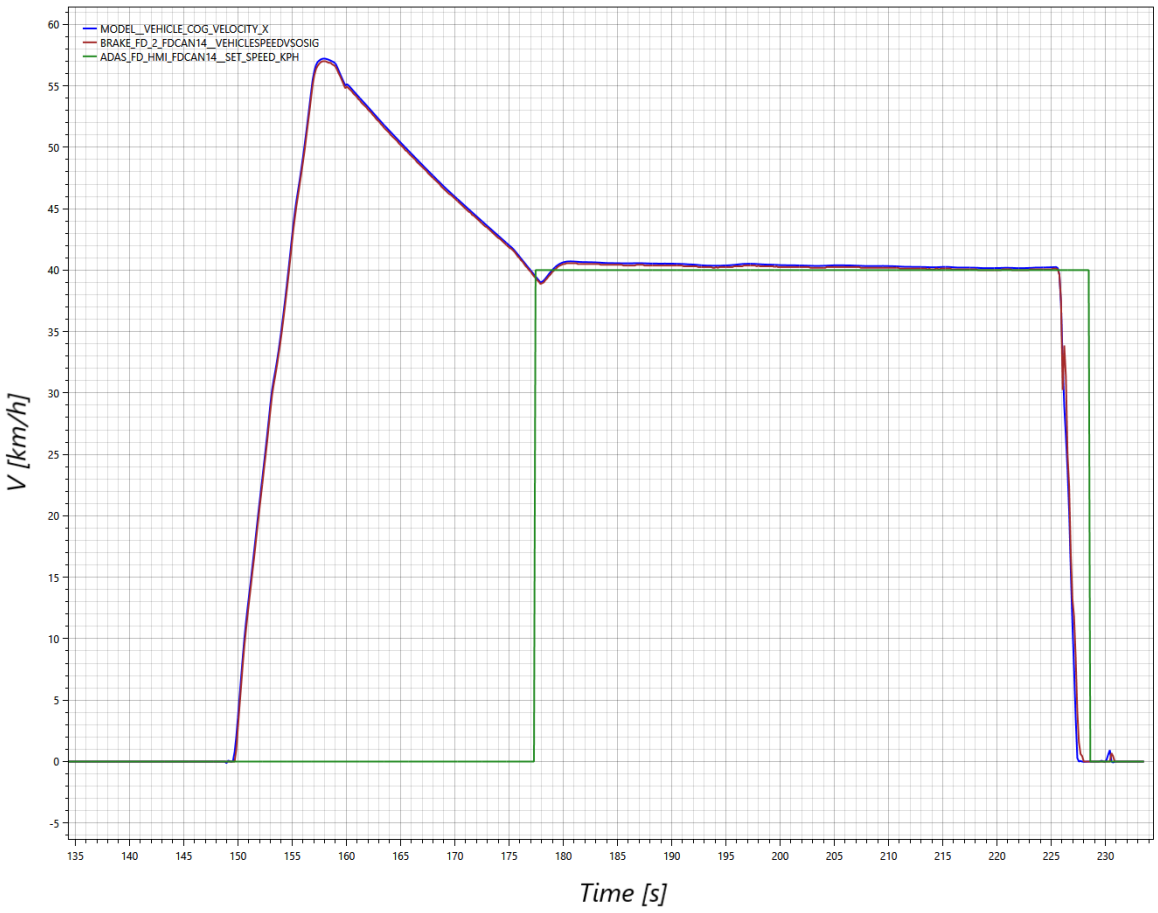


FIGURE 6.6: Overall velocity profiles (CC Scenario)

In Figure 6.6 is possible to distinguish 3 signals:

- *VEHICLE COG VELOCITY X*: Signal coming from the ASM model that represent the longitudinal velocity of the ego-vehicle

- *VEHICLESPEEDVSOSIG*: Signal coming from the BSCM present on the PWB that represent the velocity published by the brake sensor

- *SET SPEED KPH*: Signal coming from the CADM that inform other ECUs on the cruise speed set in km/h

Also in this case the functional target passed.
To further confirm this the following 2 figures shows a zoom of the signal with their respective tolerance band.

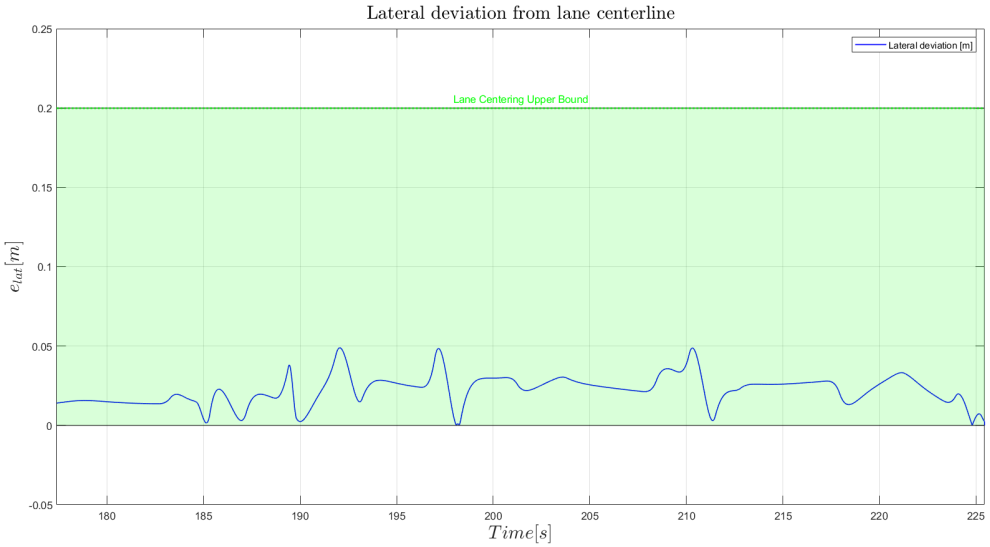


FIGURE 6.7: Lateral deviation signal with its tolerance band (CC Scenario)

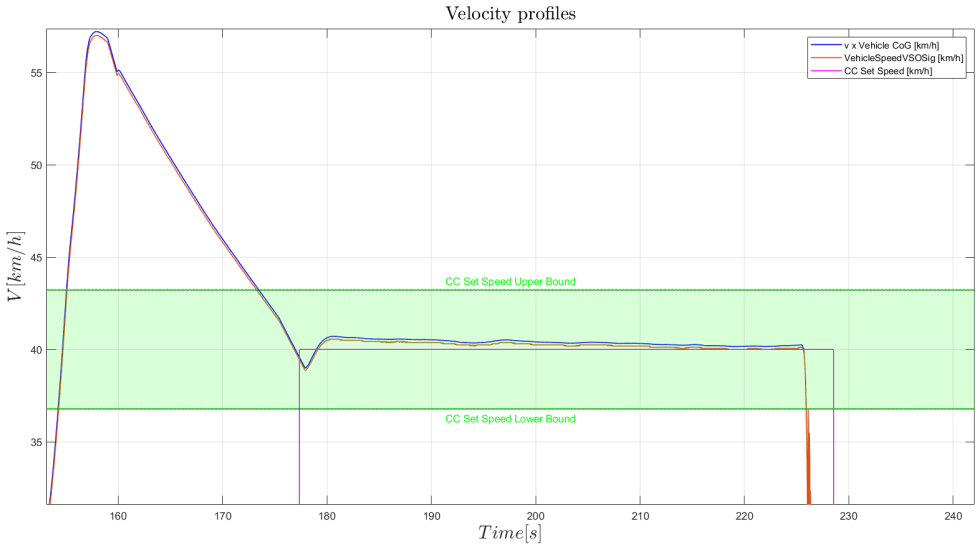


FIGURE 6.8: Velocity signals with their tolerance band (CC Scenario)

It is worth noting that the functional target on the CC set speed is applied only when the CC is engaged thus, looking at Figure 6.8, in the range $[177 \div 229]$ seconds.

6.2 Adaptive Cruise Control Scenario

The second scenario deals with 4 vehicles: the ego-vehicle and 3 fellows.

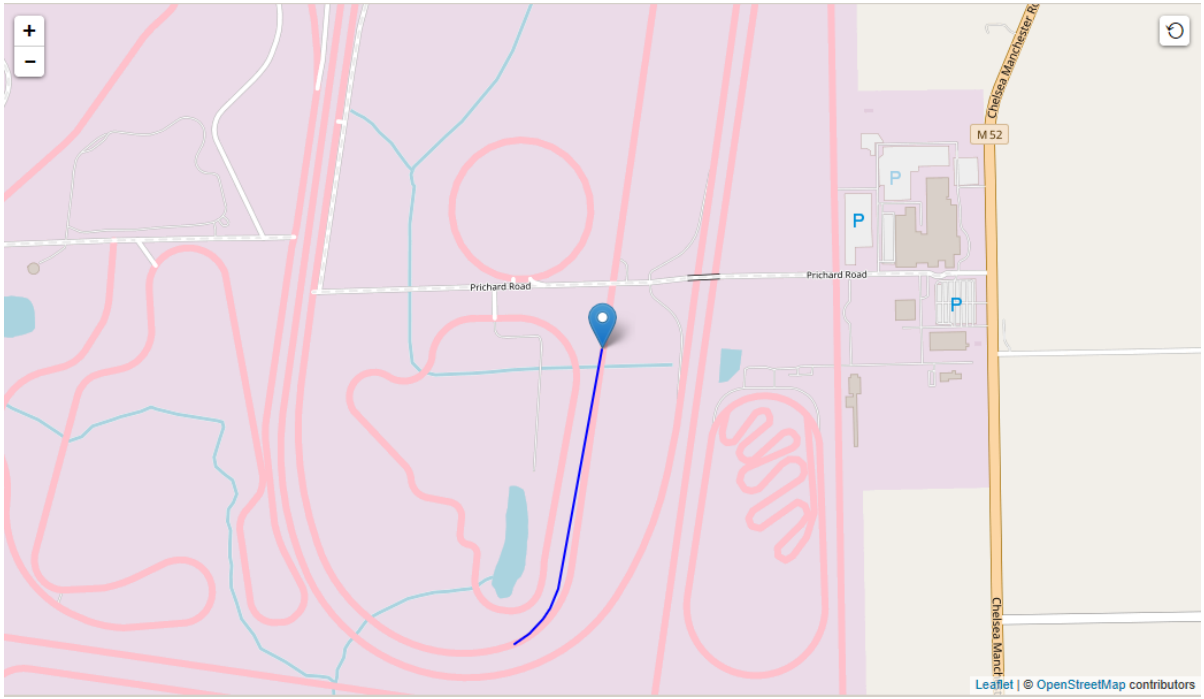


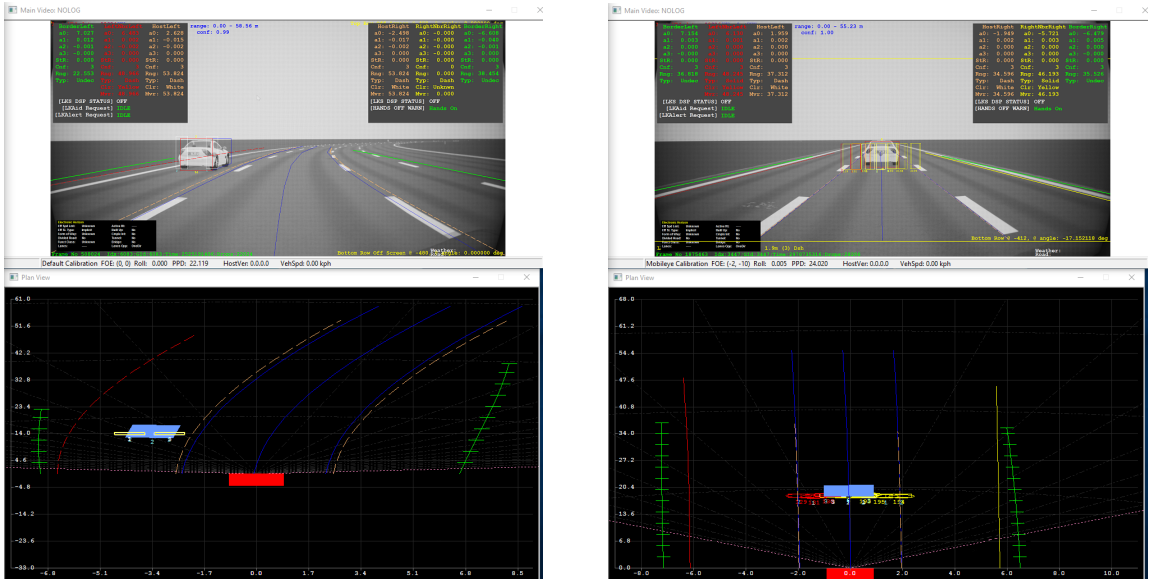
FIGURE 6.9: Ego-vehicle trajectory in ACC scenario

The goal of this test was to validate the Adaptive Cruise Control (ACC) functionality and for this purpose 3 functional targets were checked in post-processing phase:

Functional target	Low level description	Validation method
Lane Centering: Upper bound on the lateral deviation from lane centerline	Inequality constraint: $e_{lat} = sqrt[(X_{road} - X_{ego})^2 + (Y_{road} - Y_{ego})^2]$ $<= 0.2$ m	Check in each point that the euclidean distance between the ego-vehicle position and the lane centerline is not greater than 0.2 m
ACC Following Distance: Upper and lower bound on the safety distance between ego-vehicle and fellow ahead	Tolerance band: $e_{long} = sqrt[(X_{ego} - X_{fellow})^2 + (Y_{ego} - Y_{fellow})^2] \in [17.5 \div 20.5]$ m	When the Adaptive Cruise Control is engaged the following distance between ego-vehicle and fellow ahead must be always contained in the range 19 +/- 1.5 m.
Deceleration Request: Upper bound on the longitudinal acceleration settling time	Inequality constraint: $T_s <= 0.5$ s	The ego longitudinal acceleration shall follow the CADM's requested deceleration rate based on the ego-vehicle speed and the distance between ego and fellow ahead within a settling time of half a second.

TABLE 6.2: Adaptive Cruise Control functional targets

Figure 6.10 shows how the CADM algorithm have detected the presence of another vehicle ahead.



(A) Fellow in the left lane

(B) Fellow in the center lane

FIGURE 6.10: The machine learning algorithm detects the fellow ahead and encloses it in a bounding box

The results obtained after the run of this second scenario are presented in the following figures.

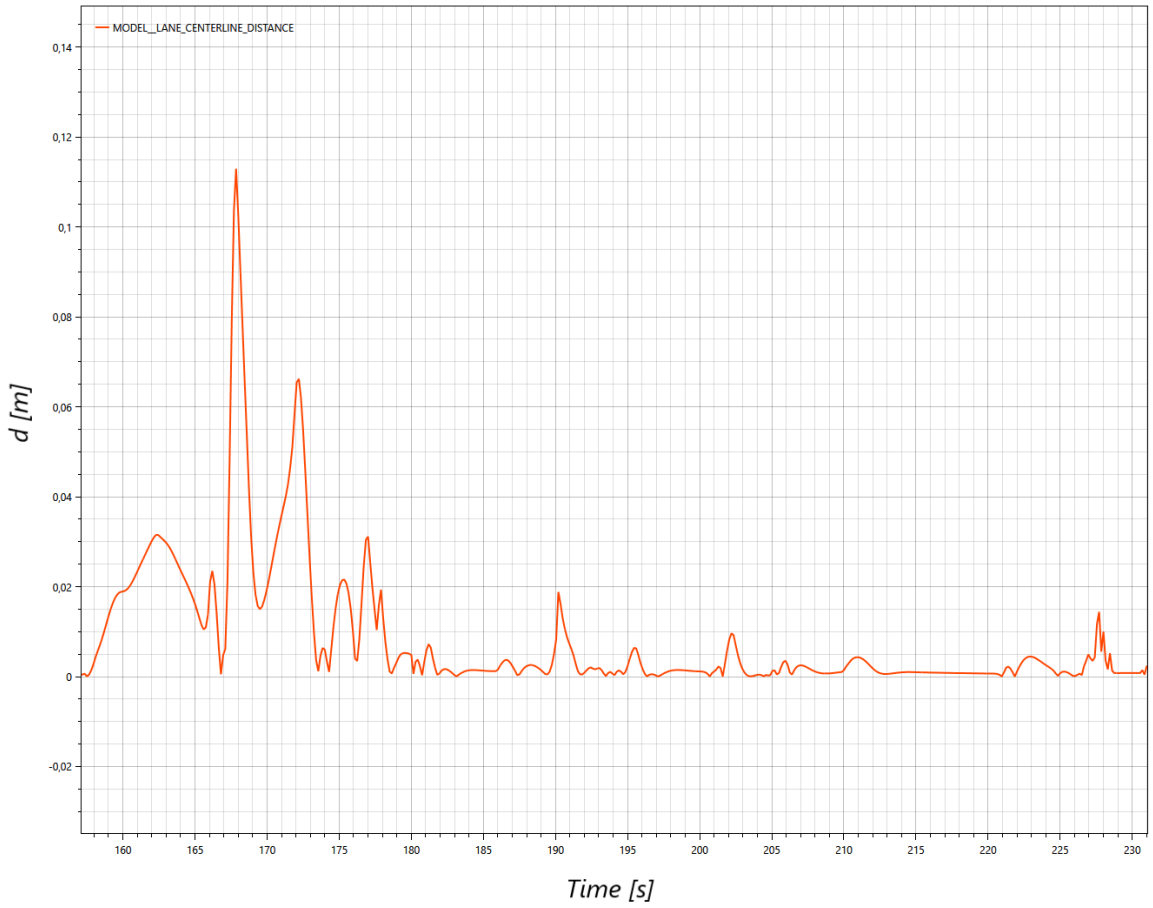


FIGURE 6.11: Overall lateral deviation profile (ACC Scenario)

Similarly to what happened for the CC scenario the *LANE CENTERLINE DISTANCE* signal passed the target once again.

The only consideration regards the peaks at 0.11 m and 0.066 m: this is due to a change in the road curvature. In that moment the ego, that was initially driving on a circular road, enters a straight.

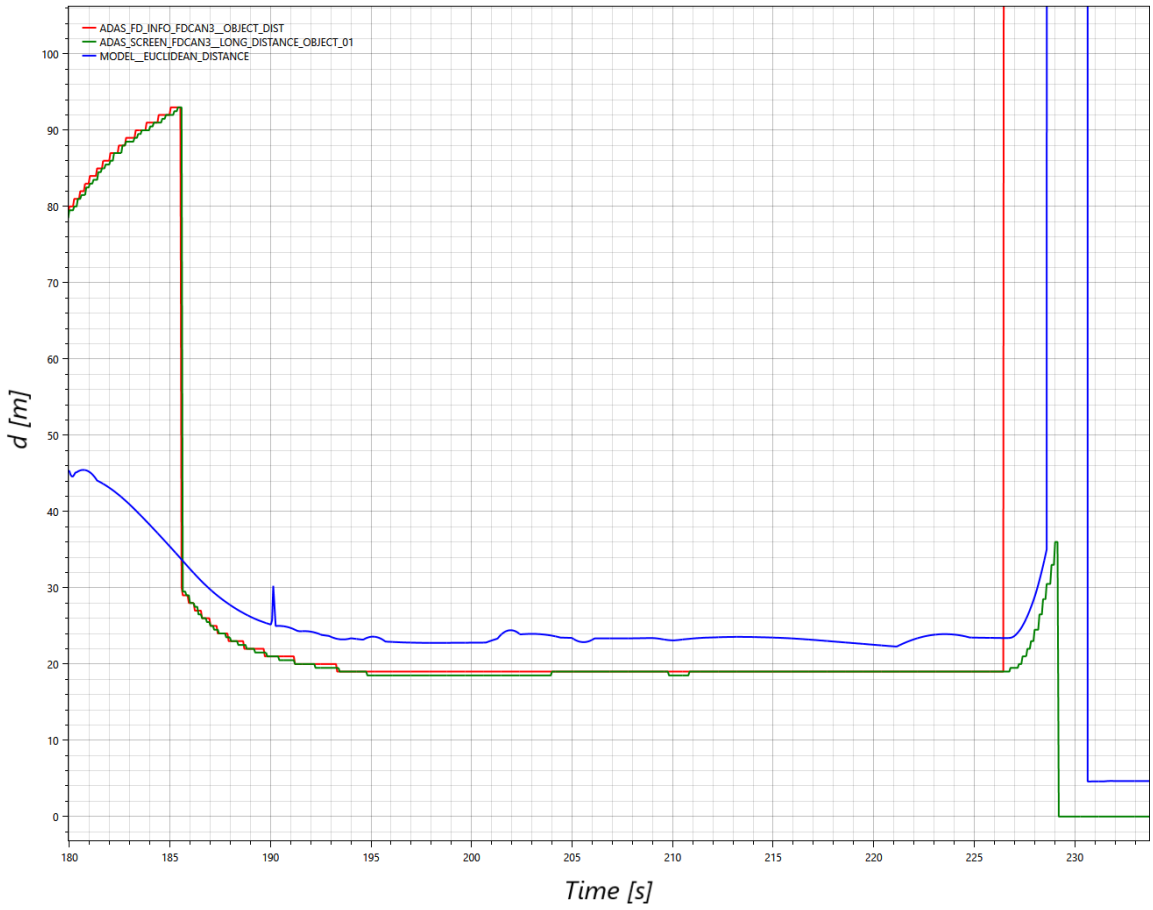


FIGURE 6.12: Fellow distance signals (ACC Scenario)

In Figure 6.12 is possible to distinguish 3 signals:

- *OBJECT DIST*: Signal coming from the CADM that represents the longitudinal distance of the object detected by the Forward Facing Camera
- *LONG DISTANCE OBJECT*: Signal coming from the CADM equivalent to *OBJECT DIST*
- *EUCLIDEAN DISTANCE*: Signal coming from the ASM model that represents the longitudinal distance between ego and the selected fellow

In this case only 2 signals out of 3 passed the functional target. Indeed, although *EUCLIDEAN DISTANCE* has quite the same profile of the other two, it presents an offset of 4 m, thus exceeding the tolerance band. This can be attributed to the function used to compute the distance in ASM model: it evaluates the distance between the ego CoG and fellow CoG, instead the other 2 signals compute the distance between ego front

bumper and fellow rear bumper.

Therefore, in order to be correctly compared, *EUCLIDEAN DISTANCE* signal should be translated downwards by a factor of $(\frac{l_{ego}}{2} + \frac{l_{fellow}}{2})$.

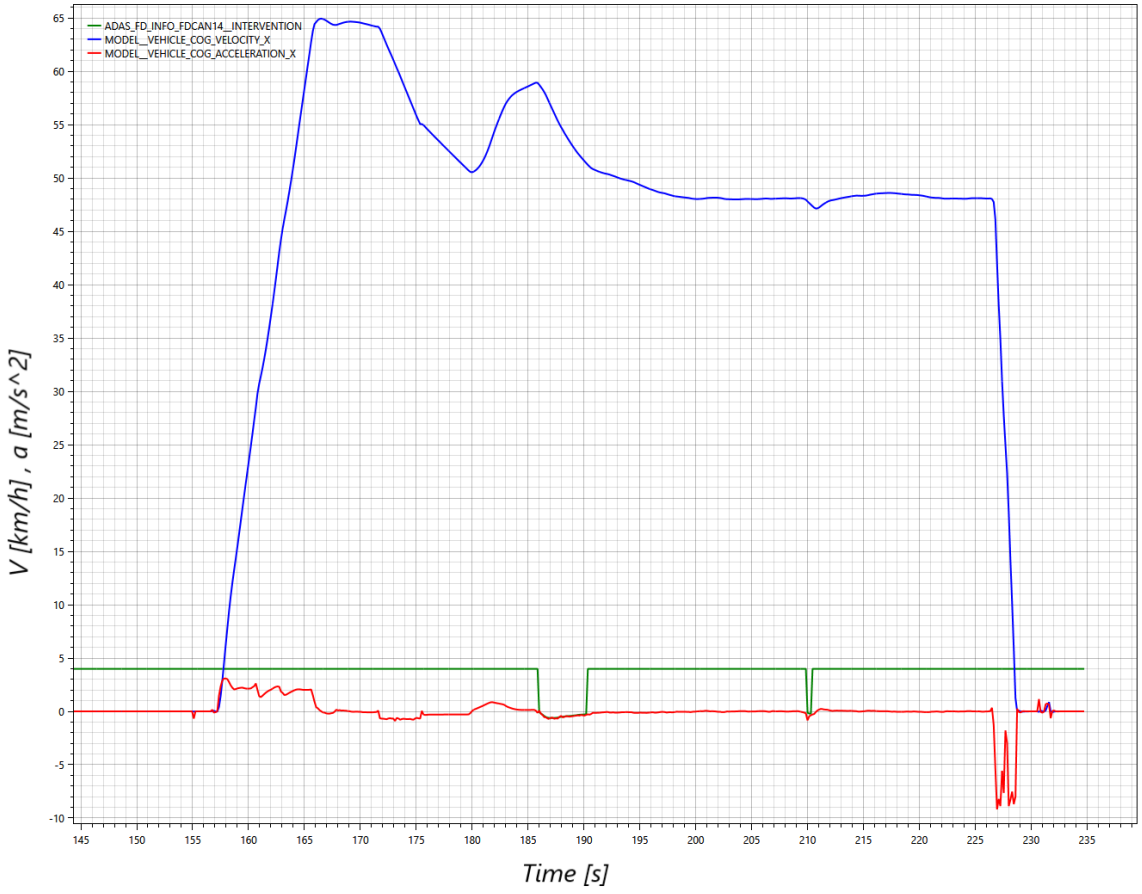


FIGURE 6.13: Velocity and acceleration profiles (ACC Scenario)

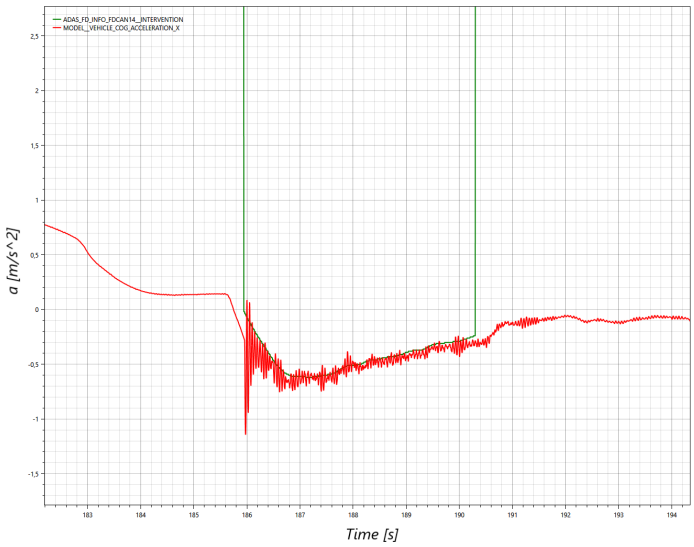
Figure 6.13 displays 2 acceleration signals along with *VEHICLE COG VELOCITY X*:

- *INTERVENTION*: CADM's signal to request a deceleration intervention by the brake
- *VEHICLE COG ACCELERATION X*: Signal coming from the ASM model that represents the longitudinal acceleration of the ego-vehicle

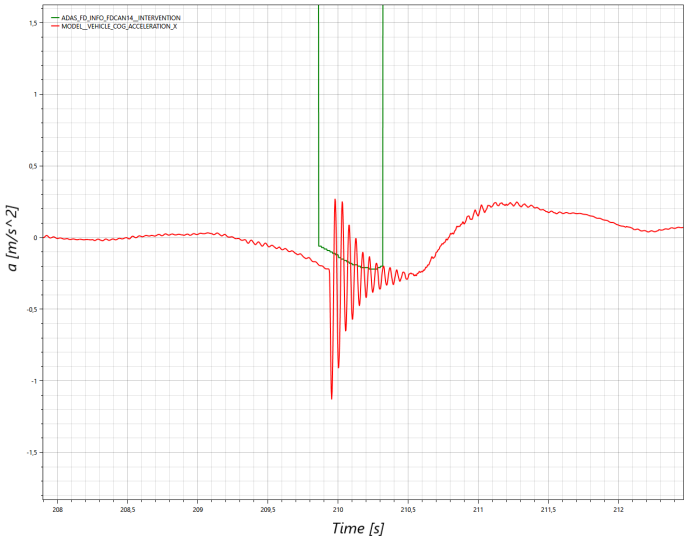
Looking at time instants $t1 = 186$ s and $t2 = 210$ s, is it possible to observe that the velocity rapidly decreases. At the same time the *INTERVENTION* signal present a negative peak.

This happens because a fellow with a lower velocity has been detected in the range of the camera, hence the ego shall decelerate to avoid collision and keep the safety distance. The 2 figures below show a zoom of the 2 intervention requests. In both cases the vehicle acceleration respond to CADM's request with an oscillatory transient which settles in about half a second.

Even the third and last functional target passed.



(A) First intervention request



(B) Second intervention request

FIGURE 6.14: Longitudinal acceleration response to an intervention request (ACC Scenario)

At last figures 6.15 and 6.16 confirms the considerations previously made for the lateral deviation and fellow distance, highlighting also their tolerance.

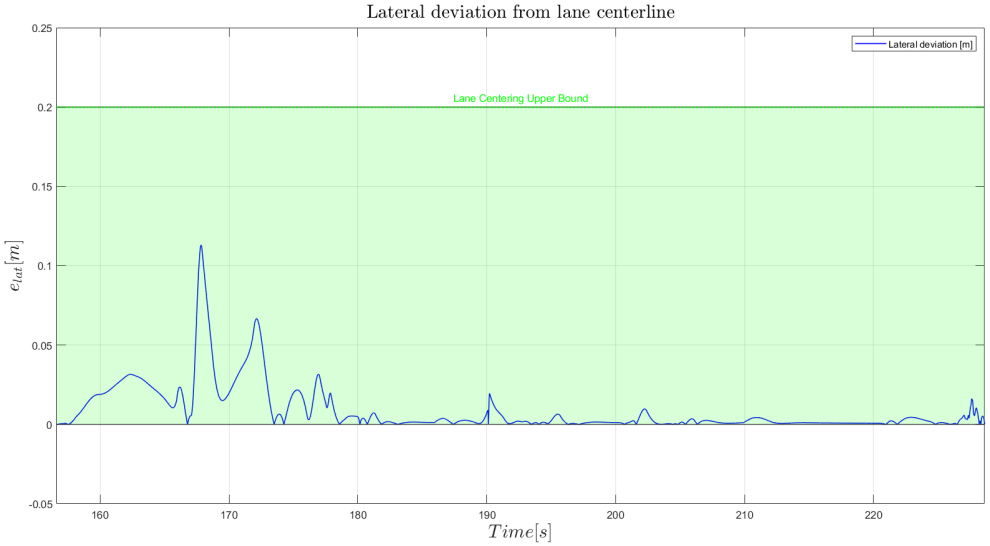


FIGURE 6.15: Lateral deviation signal with its tolerance band (ACC Scenario)

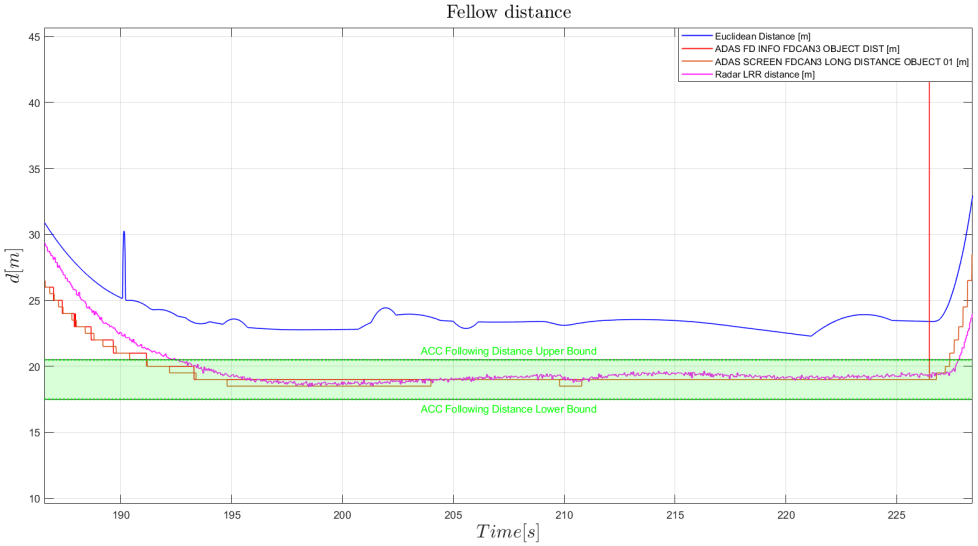


FIGURE 6.16: Fellow distance profiles with their tolerance band (ACC Scenario)

In particular Figure 6.16 shows the offset of the *EUCLIDEAN DISTANCE* signal that is the only one not contained in the tolerance band. In this Figure is plotted also the signal coming from the front radar, called *RADAR LRR DISTANCE*.

6.3 Highway Assist Scenario

The third scenario deals only with the ego-vehicle.

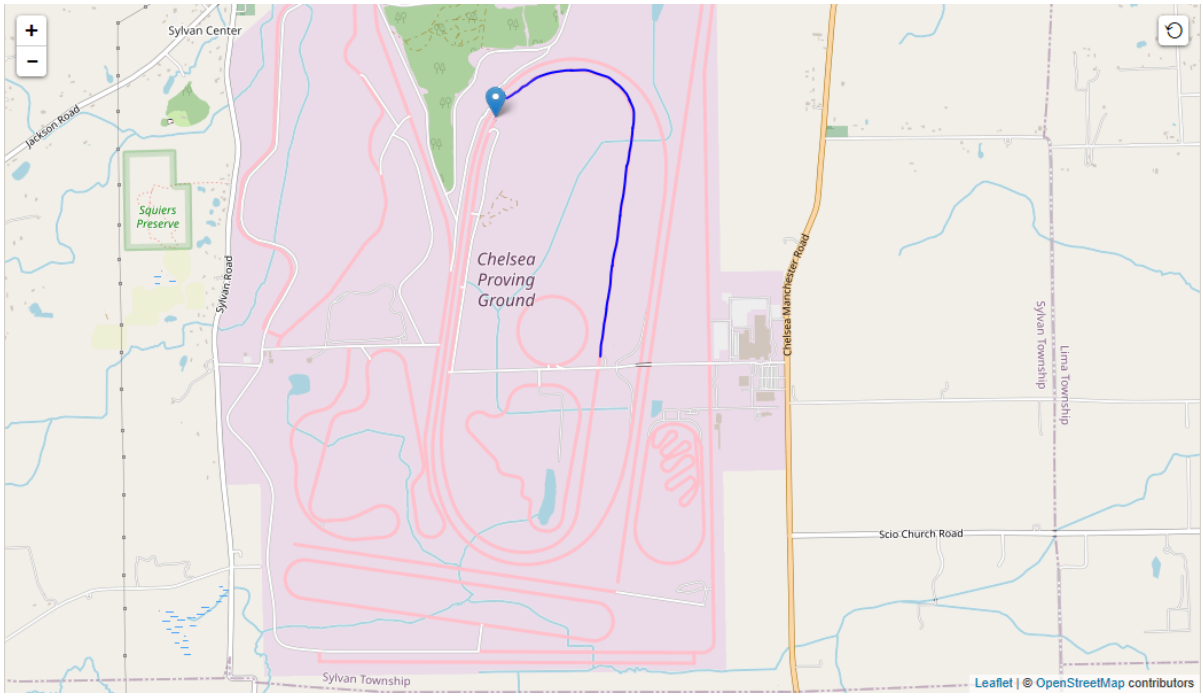


FIGURE 6.17: Ego-vehicle trajectory in HAS scenario

The Highway Assist System is a partially automated driving system, thus an L2 ADAS specific feature.
Picking up information from both a front radar sensor and a forward-facing camera behind the rear-view mirror, HAS performs three crucial functions at the same time [16]:

- Maintains the desired speed (CC)
- Maintains the chosen distance from the vehicle in front (ACC)
- Keeps the vehicle in the center of the lane by actively controlling the steering wheel (LKA)

The goal of this test was to validate the Highway Assist functionality in absence of traffic. For this purpose only one functional target was enough.

Functional target	Low level description	Validation method
Lane Centering: Upper bound on the lateral deviation from lane centerline	Inequality constraint: $e_{lat} = sqrt[(X_{road} - X_{ego})^2 + (Y_{road} - Y_{ego})^2]$ <= 0.2 m	Check that the euclidean distance between the ego-vehicle position and the lane centerline is not greater than 0.2 m when the HAS is engaged.

TABLE 6.3: Highway Assist functional target

HAS is designed to only function on highways or limited access freeways. For this reason the scenario was composed by 2 phases.

1. **ACC** engaged: In this phase, the scenario was not suitable to let HAS work properly, therefore only ACC was engaged

2. **HAS** engaged: In the second phase, the scenario turns into a large straight highway and the HAS could be finally engaged

During the first phase, ego was just left drive and the command was totally in charge of the ASM model; whereas in the second phase the command switched from the model to the CADM.
When HAS is engaged CADM actively controls the steering wheel by sending the information acquired by the camera to the EPS and BSCM. In this way the appropriate steering torque can be applied in order to keep the vehicle centered on its lane.

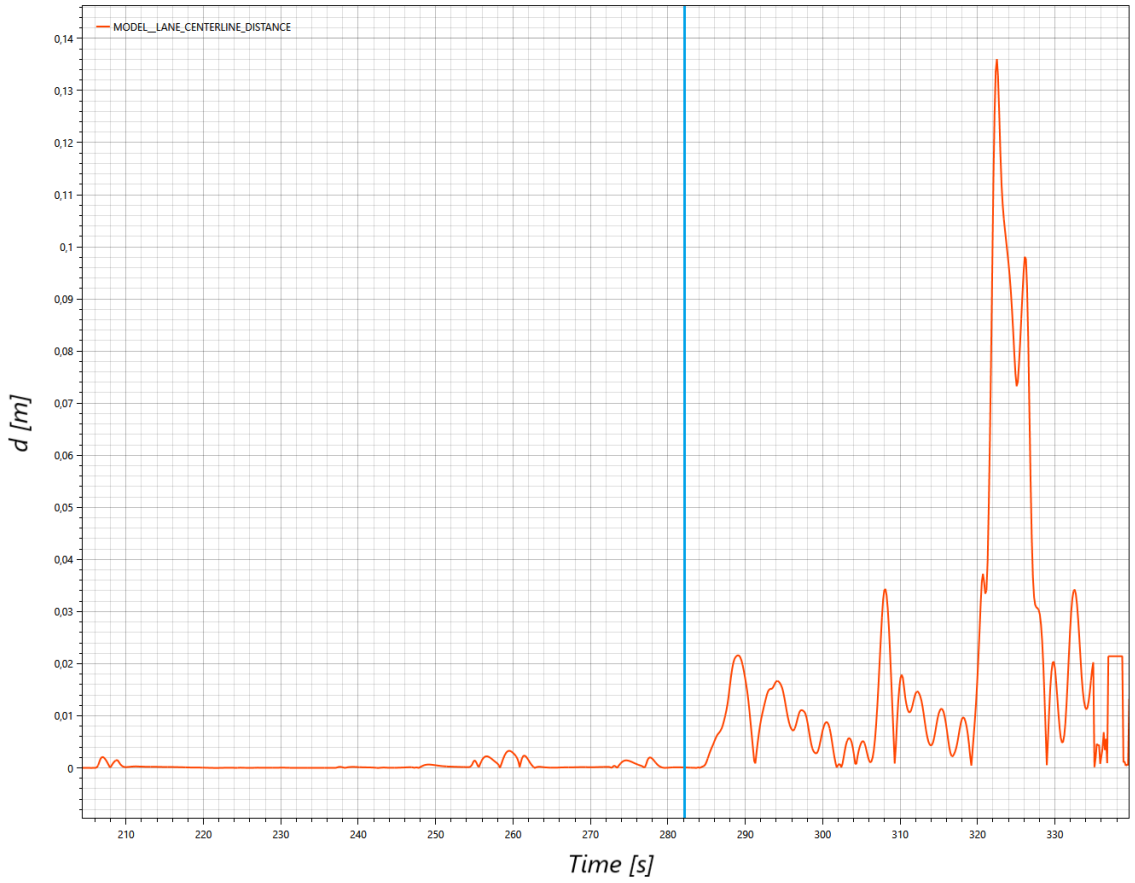


FIGURE 6.18: Overall lateral deviation profile (HAS Scenario)

Observing Figure 6.18 is possible to differentiate 2 regions delimited by a blue vertical line.

The left area represent the first phase i.e. ACC engaged; during this phase the model perform all the longitudinal/lateral actions and controls the entire vehicle dynamics, so the lateral deviation is strictly close to zero.

Instead, looking at the right area of the plot, a series of sharp oscillation affect the vehicle dynamics. In this case HAS is engaged and the control of lateral dynamics is in charge of the physical ECUs.

This finds an answer in the fact that HAS is an L2 feature, therefore it requires "Hands-ON" (see Figure 1.2) i.e. by means of an haptic sensors placed along the steering wheel, the system detects if the driver is holding the steering wheel with at least one hand. In order to engage the HAS in HIL environment, the behaviour of the sensors is simulated and this lead to the following considerations:

- No driver is holding the steering wheel, thus there's no resistance torque

- There are no inertias due to the weight of the car, therefore the system expects to control a much heavier load

Hence, the oscillation obtained when HAS is engaged are bigger w.r.t the ones that could be observed in MIL environment or in a real car.
A last remark regards the 2 peaks at $t \approx 325$ s: these are due to the narrowing of the roadway from 3 to 2 lanes.

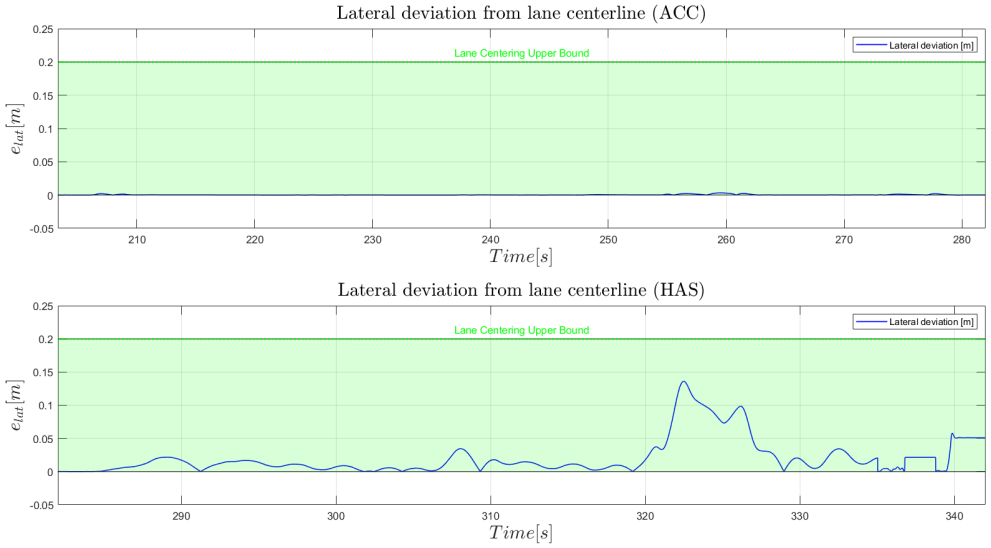


FIGURE 6.19: Lateral deviation profiles comparison (HAS Scenario)

Figure 6.19 shows that in both cases the functional target passed.

6.4 Highway Assist with fellow Scenario

The forth and last scenario deals with 2 vehicles: ego and fellow.

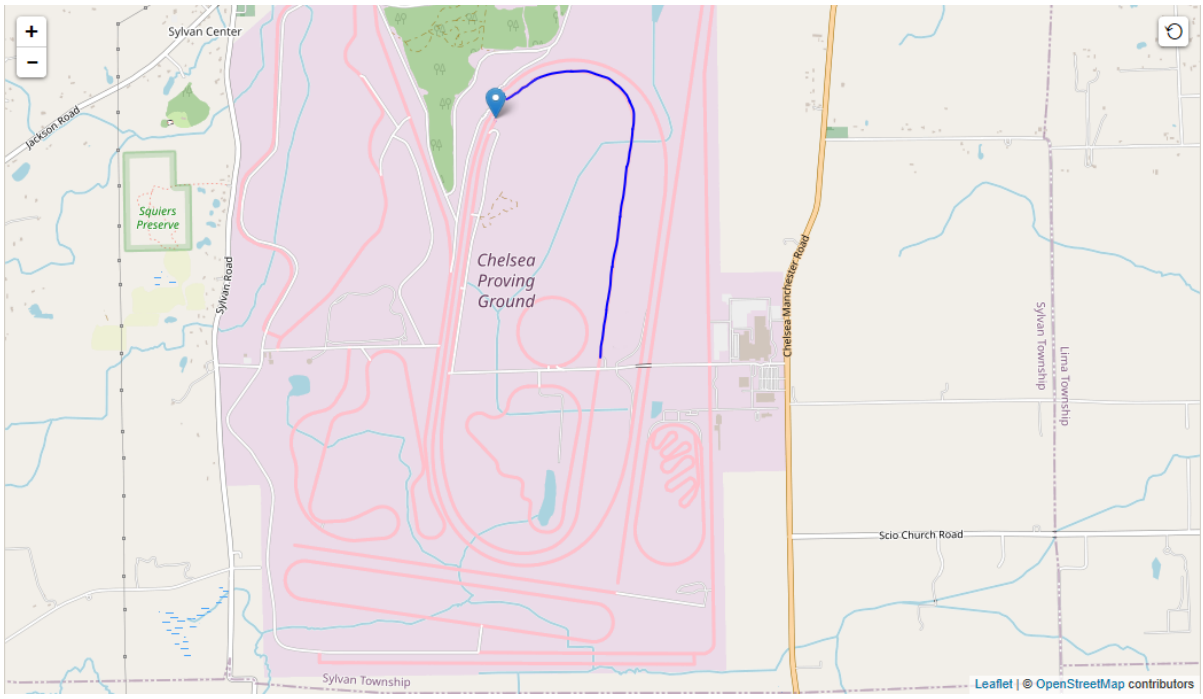


FIGURE 6.20: Ego-vehicle trajectory in HAS with fellow scenario

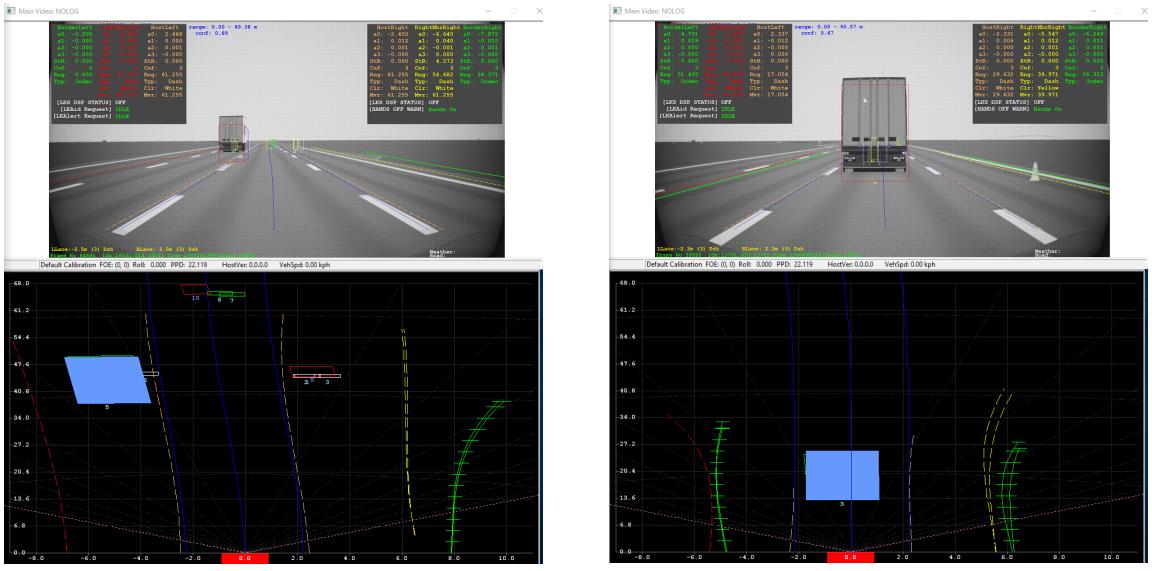
The goal of this test was to validate the Highway Assist functionality in presence of traffic.

For this purpose 2 functional targets were checked:

Functional target	Low level description	Validation method
Lane Centering: Upper bound on the lateral deviation from lane centerline	Inequality constraint: $e_{lat} = sqrt[(X_{road} - X_{ego})^2 + (Y_{road} - Y_{ego})^2]$ <= 0.2 m	Check that the euclidean distance between the ego-vehicle position and the lane centerline is not greater than 0.2 m when the HAS is engaged.
HAS Following Distance: Lower bound on the safety distance between ego-vehicle and fellow ahead	Inequality constraint: $e_{long} = sqrt[(X_{ego} - X_{fellow})^2 + (Y_{ego} - Y_{fellow})^2]$ >= 5 m	When the Highway Assist System is engaged the following distance between ego-vehicle and fellow ahead must be always greater than 5 m.

TABLE 6.4: Highway Assist with fellow functional targets

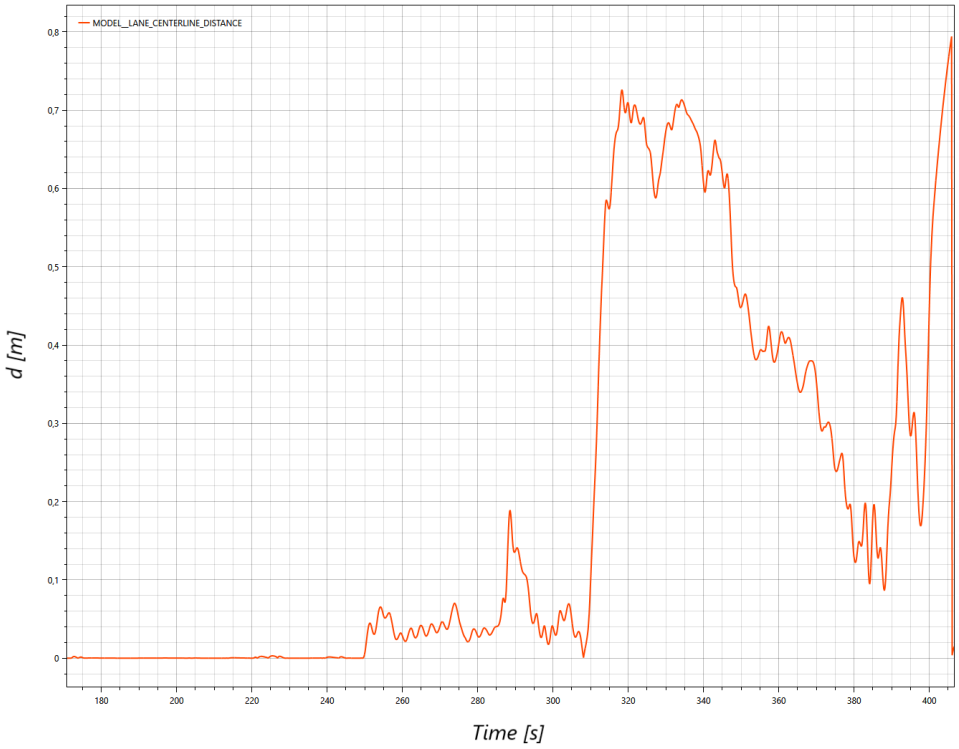
Figure 6.21 shows how the CADM algorithm have detected the presence of another vehicle ahead, in this case a truck.



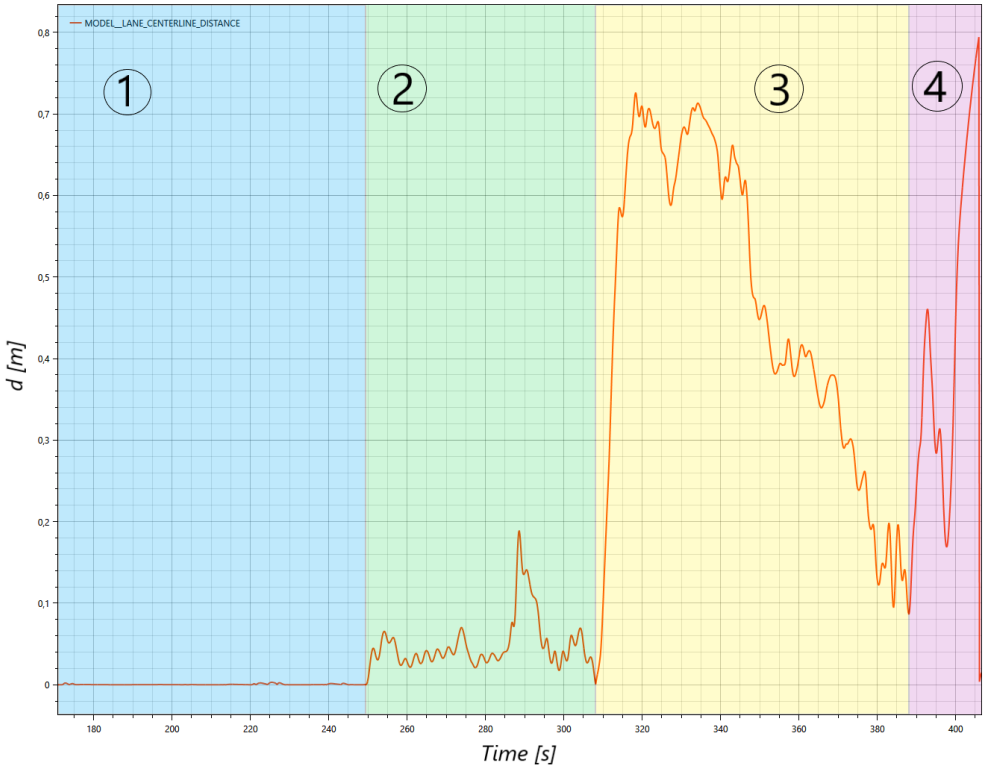
(A) Fellow in the left lane

(B) Fellow in the center lane

FIGURE 6.21: The machine learning algorithm detects the truck ahead and encloses it in a bounding box



(A) Lateral deviation profile



(B) Regions of the lateral deviation profile

FIGURE 6.22: Overall lateral deviation profile (HAS with fellow Scenario)

Referring to Figure 6.22b is it possible to notice four different regions:

1. The first region delimits the time interval in which only the ACC is engaged, thus similarly to HAS scenario, there are no appreciable oscillations and the lateral deviation profile is flat and very close to 0

2. In the second region, the ego-vehicle has reached a portion of the road where HAS could be engaged and has detected the truck in front, thus oscillation begins

3. The third region starts with a quick increasing in the slope and magnitude of the curve that exceeds 0.7 m. In this region, as can be observed in Figure 6.20, the road profile switches from straight to circular with a quite small curvature radius. In order to keep the vehicle centered and at the same time maintain the safety distance from the truck ahead, the system was stressed a lot, causing very big oscillations. Approaching the right edge of region 3 the curve decreases because the ego-vehicle is slowing down to face a deceleration of the truck.

4. Lastly in region 4 the curve performs two sharp oscillations and reaches its global maximum at about 0.8 m. The truck in front braked to a complete stop and the highway assist system applied a very high braking torque to avoid frontal collision and respect the minimum safety distance. This huge braking action caused a partial loss of stability of the vehicle

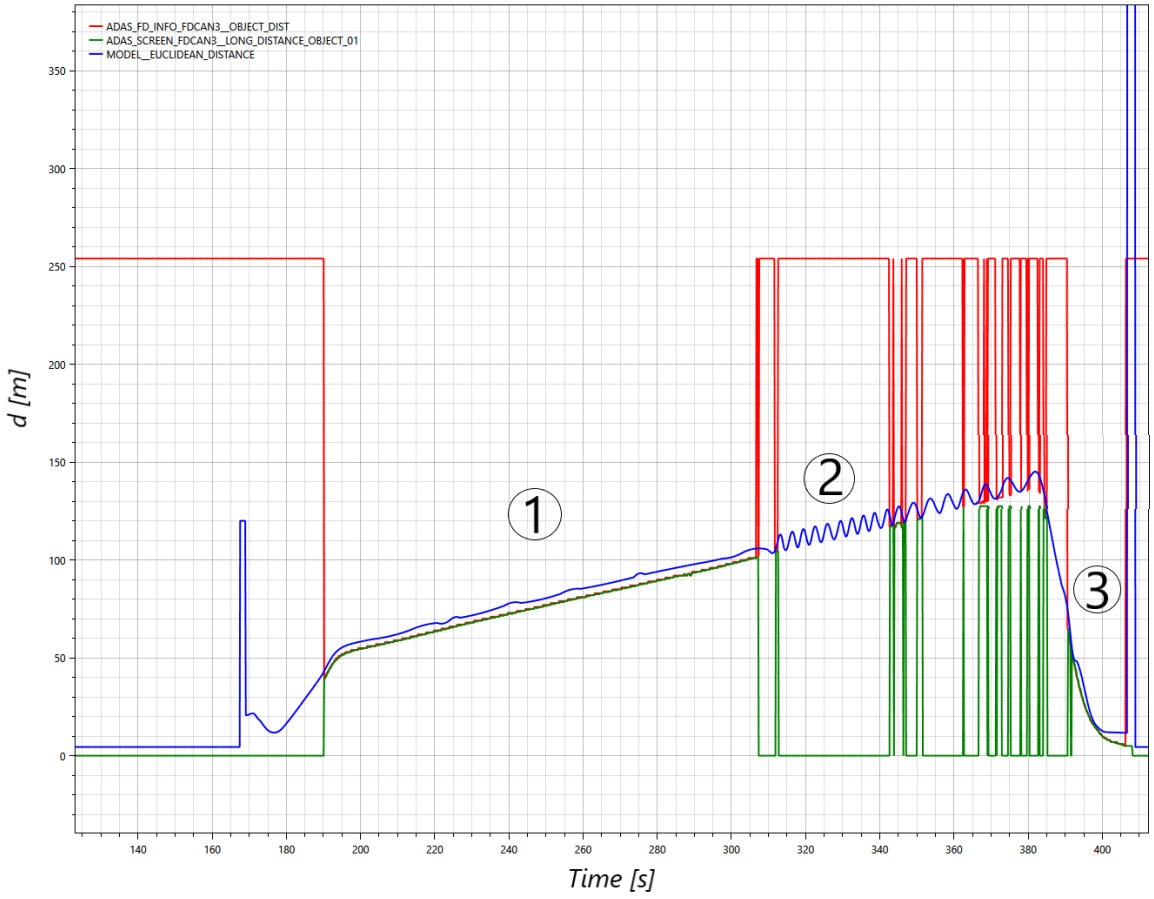
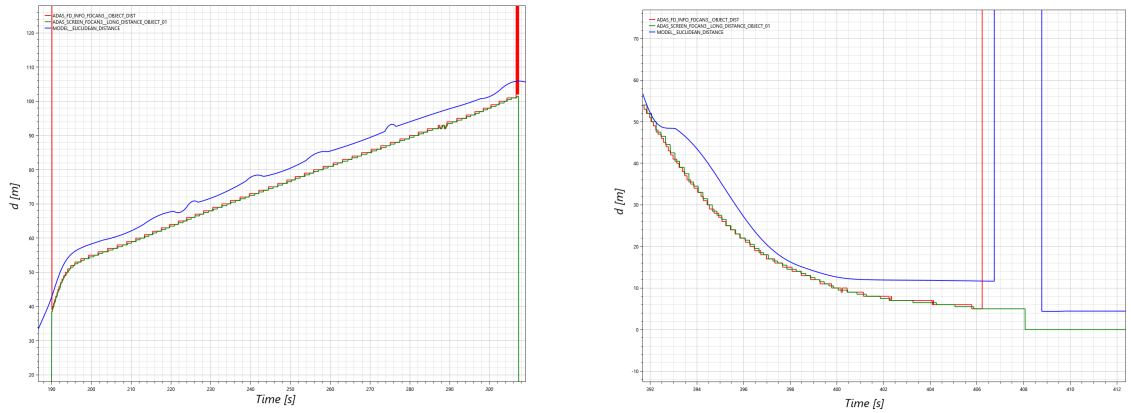


FIGURE 6.23: Overall fellow distance signals (HAS with fellow Scenario)



(A) Fellow is accelerating

(B) Fellow is braking

FIGURE 6.24: Zoom on the fellow distance signals

The three figures in the previous page show the distance between ego and truck by means of 3 signals, the same explained in Figure 6.12. Looking at the overall profile (Figure 6.23), is it possible to recognize three areas:

1. Fellow is detected and is accelerating

2. Fellow accelerates too much and is out of the range of the camera. Only the *EUCLIDEAN DISTANCE* signal is still able to calculate the distance, albeit with some oscillations. While the other two signals return to their default values

3. Fellow brakes firmly until it comes to a complete stop

It is worth noting that the HAS responds well to a change in the truck's acceleration, adjusting the speed of the ego-vehicle accordingly. In particular, looking at Figure 6.24b, is it possible to notice that the distance between the two vehicle never drops below 5 m, thus respecting the functional target.

The two plots below confirm that the functional target on the lateral deviation passed only when ACC was engaged, indeed the bottom plot highlights how the signal exceed the tolerance upper bound.

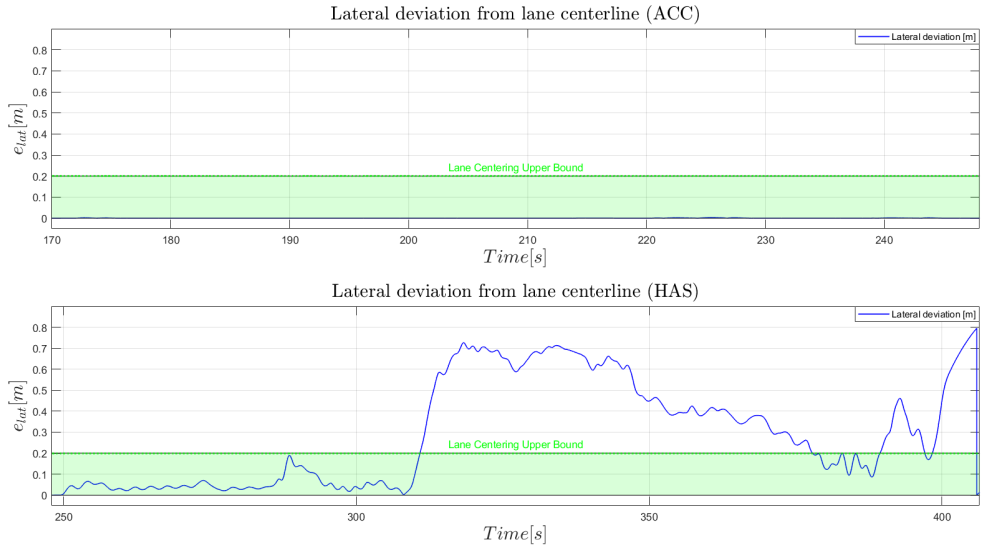


FIGURE 6.25: Lateral deviation profiles comparison (HAS with fellow Scenario)

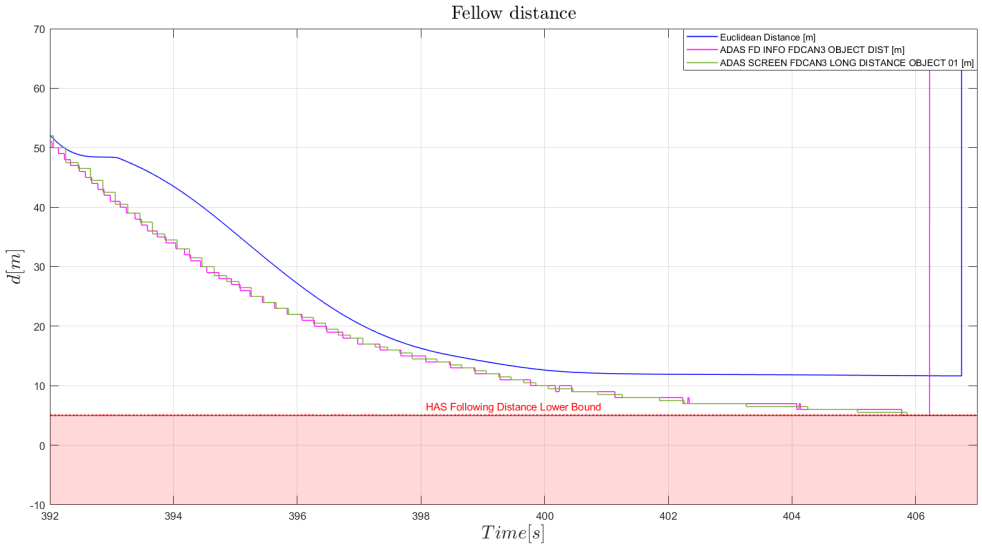


FIGURE 6.26: Fellow distance profiles with their tolerance band (HAS with fellow Scenario)

At last Figure 6.26 shows that each signal stays always above the tolerance lower bound of 5 m.

Chapter 7

Conclusive Remarks

The last chapter of this thesis is devoted to some conclusive considerations. The L2+ Automation Tool has been the heart of this work, it allows to convert the XLS based test cases with predefined road files into open scenario according to the ASAM 1.0 standard. The most relevant aspects are described in the following bullet list:

- **Pros**

    1. The output scenario is flexible to run in any XIL environment potentially
    2. It offers the possibility to perform any kind of test (functional, performance, stress test)
    3. The tool allows to generate a scenario according to any car model (dSPACE-ASM, VI-CarRealTime)

- **Cons**

    1. The automatically generated scenarios are not immediately suitable for the HIL environment but need some manual modifications
    2. Accessing a specific signal in the ASM model is not straightforward and requires a deep understanding of it
    3. The tool doesn't allow to create a new scenario from scratch, it always requires a conversion from an XLS format test case

		Scenario			
		CC	ACC	HAS	HAS with fellow
Functional target	Lane Centering	✓	✓	✓	✗
	Deceleration Request	-	✓	-	-
	Following Distance	-	▲	-	✓
	Set Speed	✓	-	-	-

TABLE 7.1: Functional targets compliance

Table 7.1 recaps the outcome of the execution of the four scenarios discussed in Chapter 6.

The overall result is very good; 6 out of 8 functional target passed, while there was one partial failure (indicated as ▲) and a total failure. Overall the tool fits well different scenario configurations and could give a great support to any user who wants to exploit it for virtual or real validation.

The scenarios selected revealed to be an excellent approximation of the reality thus allowing to save a lot of time and resources and helped to greatly speed-up the L2 ADAS software development process.

7.1 Further Improvements

As a continuation of the work carried out, the following items could be taken into account:

- **DIL Integration**: Several automotive OEMs own a driving simulator, the tool's generated scenarios could be integrated with simulators from different vendors. In this way the driver is directly involved in a safe simulation environment with the highest level of realism
- **Upgrade to oncoming ASAM 2.0**: ASAM 2.0 Standard is going to be released in March 2022 [17]. It will include the definition of more complex scenarios and for this reason the tool needs to be updated to remain standard compliant

Bibliography

[1] Inc. Synopsys. *What is ADAS?* https://www.synopsys.com/automotive/what-is-adas.html.

[2] Mobileye. *Understanding L2+ in Five Questions.* https://www.mobileye.com/blog/understanding-l2-in-five-questions/.

[3] Intel's Public Relations Team. "Defining the 'Plus' in L2+". In: (2019). URL: https://newsroom.intel.com/articles/defining-plus-l2/#gs.ngcwbe.

[4] Keith M. Bower. *What Is Design Of Experiments (DOE)?* https://asq.org/quality-resources/design-of-experiments.

[5] JMP Software. *Design of Experiments DOE Process.* https://www.youtube.com/watch?v=_Rgue-7KDww&ab_channel=JMPSoftware.

[6] Eric S. Lagergren Raghu N. Kacker and James J. Filliben. "Taguchi's Orthogonal Arrays Are Classical Designs of Experiments". In: *Journal of Research of the National Institute of Standards and Technology* 96 (1991), pp. 577–591. URL: https://nvlpubs.nist.gov/nistpubs/jres/096/jresv96n5p577_A1b.pdf.

[7] Peter Woolf et al. "Chapter 14.1 - Design of Experiments via Taguchi Methods - Orthogonal Arrays". In: *Chemical Process Dynamics and Controls.* Ed. by Stephanie Fraley et al. LibreTexts, 2021, pp. 774–784.

[8] ASAM. *ASAM OpenSCENARIO: User Guide Version: 1.0.0.* https://releases.asam.net/OpenSCENARIO/1.0.0/ASAM_OpenSCENARIO_BS-1-2_User-Guide_V1-0-0.html#_foreword.

[9] dSPACE. *Information on ASAM XIL API.* https://www.dspace.com/en/inc/home/applicationfields/our_solutions_for/xil_api/xil_api_standard.cfm.

[10] André Rolfsmeier. "Virtual validation of advanced driver assistance functions". In: *HANSER automotive* (2013). URL: https://www.dspace.com/shared/data/pdf/2013/2013_Article_Hanser_automotive_7-8_Rolfsmeier_E.pdf.

[11] SIEMENS. *What Is Virtual Validation?* https://www.plm.automation.siemens.com/global/it/our-story/glossary/virtual-validation/57459.

[12] dSPACE. *Automotive Simulation Models (ASM).* https://www.dspace.com/en/pub/home/products/sw/automotive_simulation_models.cfm#176_28360.

[13] Speedgoat. *What is Hardware-in-the-Loop Simulation?* https://www.speedgoat.com/applications-industries/applications/hardware-in-the-loop.

[14] dSPACE. *SCALEXIO Modular real-time system.* https://www.dspace.com/en/pub/home/products/hw/simulator_hardware/scalexio.cfm.

[15] Jingwei Cao et al. "Lane Detection Algorithm for Intelligent Vehicles in Complex Road Conditions and Dynamic Environments". In: (2019). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679325/.

[16] Maserati. *Highway Assist System.* https://www.maserati.com/international/en/ownership/guides-and-documentation/safety/highway-assist.

[17] ASAM. *ASAM OpenSCENARIO V2.0.* https://www.asam.net/project-detail/asam-openscenario-v20-1/.

Acknowledgments

A special thanks goes, first of all, to my family who gave me constant support in my first off-site experience during 2 troubled years due to the Covid pandemic.
I say thank you from the bottom of my heart, to my girlfriend Cristina, who in front of so many difficulties and hard months away has always found a way not to make me feel alone. You have been my vital strength and the best soulmate I could wish for. You filled my heart even in the darkest moments and made even the most frustrating days happier and more bearable.
I find it only right to thank Ing. Roberto Rabbeni for believing in me by giving me the opportunity to become part of the Maserati family. In the same way, I thank Gianfranco for the invaluable knowledge he has passed on to me, an exemplary proof of Who an Automotive Engineer is and What he does. In addition, I also thank all the other colleagues in the Virtual Validation team: Arianna, Davide, Matteo, Chiara and Francesco for teaching me to team up and collaborate with each other.
I express my sincere thanks to prof. Carlo Concari who gave me support during my thesis work.
Impossible to forget my coursemates: Ivan, Giuseppe, Lucia who made me feel less alone in a city that was totally new to me.
Finally, there are no words to thank my lifelong friends, those who have never abandoned me and who will always be there: Antonio, Bubu, Davide, Pasquale, Peppinaccio.
I love you deeply guys.