Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Meccanica

Tesi di Laurea Magistrale

ADAS systems using advanced technologies in a simulated environment

Relatore

Prof. Mauro Velardocchia

Co-relatore

Ing. Giacomo Paolieri

Candidata

Valentina Abate

Aprile 2021

SOMMARIO

ABSTRACT

The thesis project has been carried out in collaboration with Danisi Engineering, a society that takes care purely of the development of engineering solutions and prototyping of vehicles for the automotive industry.

One of the main activities of interest to the company is the research and development of ADAS (Advanced Driver-Assistance systems) techniques, autonomous driving systems that will be mandatory on all vehicles registered from 2022, which are made through the use of the state-of-the-art dedicated software and hardware.

The aim of this thesis activity is to implement an autonomous driving maneuver, specifically a Lane-Keeping, in a Matlab/Simulink environment and to ensure that this is perfectly realized by a real custom vehicle, created in VI-CarRealTime, on two kinds of realistic scenarios, one urban and one extra-urban, designed using a simulation software called Software X, for copyright reasons.

In the first period the attention has been focused on the study of the ADAS world: a brief look at the history of ADAS, the knowledge of both SAE levels of driving automation and the state of the art of ADAS techniques, the study of electronic systems to realize them, such as Camera, Lidar and Radar sensors and the analysis of the legal aspects and further developments listed in the European projects taken as a reference point.

By studying and understanding the operation of Software X, the two basic scenarios were created.

The scenarios reflect in detail the features required in three European projects, the purpose of which is to create virtual platforms that provide highly automated operations for advanced vehicles based on safety and they will be the basic environment on which the Lane-Keeping maneuver will be introduced.

After that, the control logic for the generation of the target maneuver, the mathematical approach and the parameters to be controlled have been implemented, and a real vehicle model has been realized in VI-CarRealTime, to replicate the autonomous driving system on the two target scenarios.

This is a real Software-in-the-Loop logic realized by interacting with each other, three simulation software: Matlab/Simulink, Software X and VI-CarRealTime.

Finally, the results and further developments are discussed.

INTRODUCTION

The "Vienna convention on road traffic" [1] is an international treaty born in 1968 to organize road traffic and increase the drivers' safety through the adoption of uniform traffic rules among signatory countries.

In this document, the driver assumes a very important rule:

Article 8 [paragraph 1]: "Every moving vehicle or combination of vehicles shall have a driver".

Article 8 [paragraph 5]: "Every driver shall at all times be able to control his vehicle".

Article 8 [paragraph 6]: "A driver of a vehicle shall at all times minimize any activity other than driving".

Nowadays, the OEMs market is in continuous evolution and most of the investments are made for the research and the development of new technologies, including the improvement of autonomous driving, through the ADAS (Advanced Driver Assistance Systems) techniques.

The aim is to entrust the driving of the vehicle to automated systems to reduce road accidents and increase human safety and achieve the complete absence of the human pilot at the wheel.

For these reasons, the autonomous driving may appear in conflict with the rules listed in the Vienna document, where the driver was the most important element but in reality, it is an evolution of the latter, which must necessarily adapt itself to the needs of the modern automotive world. Today we rely more on the intervention of automated system, which are such a great help to any driver, that they can even completely replace it.

Infrastructure organization is another very important aspect to enable automated systems to read signals and act accordingly.

It is clear that ADAS systems will be the pillars of the automotive revolution but, in order for them to achieve the desired reliability and safety objectives, a long period of study, bench testing and road testing is required and therefore it is also necessary to update the legislative framework.

This thesis activity is focused on the description of the steps needed to achieve an autonomous driving maneuver in a realistic virtual environment and to carry out tests to increase the safety of drivers and the reliability of the vehicle even in case of human distraction.
It is divided into the following sections:

Chapter 1: ADAS WORLD

This chapter introduces ADAS world.
A brief excursus of ADAS history to understand the goals of the OEMs, a further information on the SAE levels of driving automation and a technical knowledge of the ADAS state of art and the electronic systems used.

Chapter 2: EUROPEAN PROJECTS

Three important European projects have been taken as reference point, for the following thesis activity.
The aim of these latter is to increase road safety and minimize accidents by studying more frequently dangerous traffic situations and limiting them through autonomous driving manoeuvres.

Chapter 3: SIMULATION SOFTWARE

This chapter will highlight the utility of exploiting a simulation software for the generation of basic scenarios on which to implement autonomous driving manoeuvres and will be described in detail Software X, so called for copyright reasons, which is the system used in this thesis project and its tools.

Chapter 4: CUSTOM SCENARIOS

In the chapter 4 there is a description on how to create the base scenario and the scenario as sequence of actions within the Software X.
The two types of scenario realized, one urban and one extra-urban, will be those in which the Lane-Keeping manoeuvre will be implemented.

Chapter 5: MANOEUVRE IMPLEMENTATION

The aim is the implementation of a logic of control that reproduces a universal manoeuvre of Lane-Keeping on a trajectory made in a Matlab/Simulink environment and with a model of custom vehicle realized in CarRealTime.

Chapter 6: SOFTWARE-IN-THE-LOOP

This chapter analyses the integration among three simulation environments, Software X, CarRealTime and Matlab/Simulink. The loop is closed by inserting the model of customized vehicle made in CarRealTime in a Matlab environment and it implements the maneuver of Lane-Keeping generated in Simulink, on both urban and extra-urban scenarios created through the Software X. The links among the simulation software and the different operating systems on which these latter works, generate a complete SiL (Software-in-the-Loop).

Chapter 7: CONCLUSIONS

It is the last chapter in which there is a description of both the complete project and the goals achieved and a list of any further developments.

1. ADAS WORLD

1.1. ADAS history

In the second half of 20s century, the automotive industry attended the initiation of ADAS systems, when General Motor Research Group released an automatic vehicle on test tracks, focusing on the lateral control of the steering and the longitudinal control of the speed and breaking.

This kind of AVCS (Advanced Vehicle-Control System) included the management of the full vehicle motions, not only about the safety warnings or the assistance of the vehicle controls.

In 1960s, Ohio State University and the MIT, elaborated new techniques based on longitudinal and transversal controls needed to solve urban transportation problems.

In 1968, it was published a MIT's project called METRAN (Metropolitan Transportation), in which a group of researchers developed an integrated, evolutionary transportation system for Urban Areas, with a fully autonomous vehicle and automatic control capabilities.

In the same years, similar studies were conducted in Europe and in Japan.

In the UK, the Road Research Laboratory was the first one to anticipate a hard traffic scenario on the motorway and to elaborate a strategy to prevent the jam. This experiment examined the possibility to create a driverless, fully automatic vehicle guidance and control technology. The project foresaw that every road vehicle would have to be equipped with electro-mechanical auto-drivers, for use on motorways and main roads. In this way, a central computer would check the position and the progress of individual vehicles, their steering, distance and cruising speeds, reducing the traffic jam and ensuring greater speed and safety. The goal has been achieved, because, assuming that both the average speed and the volume of traffic would be constant, the use of automation could increase the capacity of the roads by at least 50 per cent and avoid the 40 per cent of accidents.

The Ministry of International Trade and Industry (MITI), in Japan, started the most sophisticated PRT (Personal Rapid Transit) system, called CVS (Computer-controlled Vehicle System). The presentation of this project took place in 1970, at the Osaka World Exposition, where ten specially designed electric vehicles operated in a guideway network with intersections every five meters, controlled by a central computer. The cars communicated with this computer, through an underground communication channel. Then, a second phase of the CVS system was implemented and for this reason, a large test track was built outside of Tokyo. Nevertheless, the Ministry of Land, Infrastructure and transport, decided to close the project for the short headway distances and the lack of safety rules.

With the development and technological advancement in electronic and computer field, the most important R&D project was the EUREKA PROMETHEUS (PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety), a European activity started in 1987, in which Mercedes-Benz installed cameras in a W140 S-Class, to transfer image sequences to microprocessors. These electronic systems could detect real road and traffic conditions and control the steering, accelerator and brake. This was the beginning of the "smart" cars.

During the same period, on the other side of the world, American researchers with the California Partners for Advanced Transit and Highways (PATH) Program, tried to solve traffic congestion, which was an acute public concern, studying the impact of navigation, automation and roadway electrification. They created an improved network for traffic information and they introduced ITS (Intelligent Transportation Systems) in roadway-powered electric vehicles.

In the 21st century, the autonomous vehicle field was constantly evolving, particularly in Europe, where the most important initiatives were:

- VIAC (VisLab Intercontinental Autonomous Challenge) (2007-2010): a twining project between Parma and Shanghai;
- SPIT (2008-2011): in Holland;
- HAVEit (2008-2011): it will be described later;
- Cybercars-2 and CityMobil (2005-2008 and 2008-2011);
- GCDC Competition (2009-2011): it was the first European competition among 10 teams of autonomous vehicles, from different countries.

HAVEit was a Volkswagen group's project, in which a first generation of autopilot was tried. It could control the speed at 130 km/h, stop and start the vehicle in traffic.

In 2005, in the United States, another interesting project was the Gran DARPA (Defence Advanced Research Projects) challenge, in which 23 teams followed a 212 km off-road course with driverless vehicles, passed through three narrow tunnels and navigated more than 100 sharp left and right turns, near the California/Nevada state line. The winner was the Stanford University, but all the participants ended the challenge.

Then, in 2007, there was the urban DARPA challenge and the scenario changed. The vehicles had to compete among buildings, streets, cars and the city in general. It was a success too and the search went on.

Some of the first advanced semi-autonomous systems in automotive field were developed by Mercedes, which was the first automaker to introduce ABS in 1978, ESP in 1995 and a DISTRONIC system in 1998.

Then, in 2013, an evolution of DISTRONIC was installed on the new Class S. This was the birth of DISTRONIC plus, a software able to manage the steering and to maintain a safe distance on the highway; it controls the vehicle's speed

and recognizes bends, intersections and traffic conditions, through Radar and Camera sensors with a visual range of 250 m.

In the same period, Lexus presented the AASRV (Advanced Active Safety Research Vehicle), a car equipped with GPS, a 360° Lidar (Laser Imaging Detection and Ranging), three cameras, radar, accelerometers and inertial platforms.

Through Lidar, the vehicle could detect pedestrians up to 70 m away and, using cameras, it could monitor traffic conditions and control the nearby vehicles, in the blind spot [2].

To date, the aim of all the OEM in the world is to experiment new intelligent driving systems and homologate them on commercial vehicles, to prevent accidents in urban and extra-urban scenario and to increase human safety and reduce air pollution too.

Therefore, the purpose of both these projects as well as the ADAS systems is not to create a completely autonomous vehicle, but to ensure the lateral and longitudinal control of the car. In this way, the driver can distract himself for a few seconds, but without leaving the driving experience completely.

1.2. SAE levels of driving automation

There is a ranking of the different levels of automation. It was lodged by SAE (Society of Automotive Engineers) international, in 2014 and its name is J3016 *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems.*

This classification not depends on the technical capabilities of the vehicle but it was born following the "helps" that the car needs and considering the attention to the pilot during the driving.

The table has been updated in 2018 and it was modified as follow:

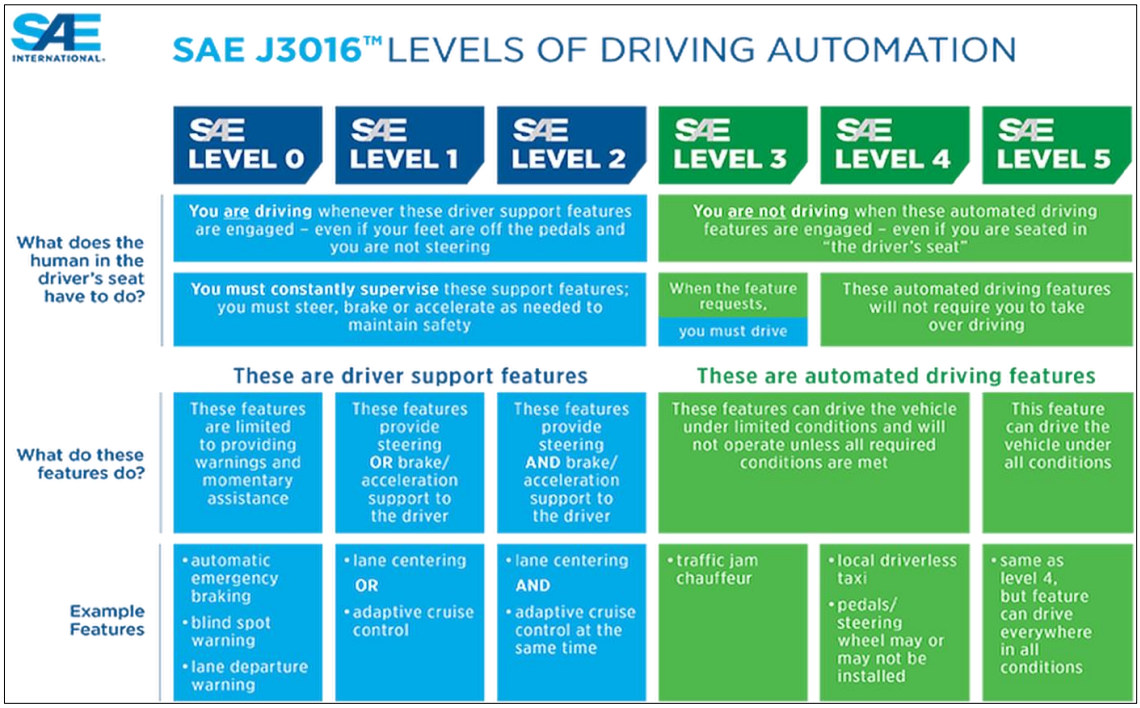


Figure 1 – SAE levels table

As we can see, there are six different levels of driving automation [3]:

- Level 0 (no automation):
  - The driver has not an electronic support and he needs to control every driving manoeuvres;
  -
- Level 1 (driver assistance):
  - The driver has the complete control of the vehicle. The latter is equipped with electronic systems, that can alert about dangerous situations or adverse conditions with visual or acoustic signals and it can control the vehicle speed and the steering;

- Level 2 (partial automation):
  - In this level, there is the first driving integration. The driver has to pay attention both to the lateral and longitudinal direction of the vehicle and to the traffic conditions, while the car manages breaking and acceleration, through particular electronic systems. When there is a clearly visible horizontal road signs, the steering can be partially automated, through different kinds of sensors and other electronic devices;

- Level 3 (Conditional automation):
    - In ordinary weather conditions, the vehicle independently manages accelerator, brake and steering, but it requires driver oversight to control system problems or particular traffic situations.

      The electronic systems can monitor the driving environment.

- Level 4 (highly automated):
    - Finally, the vehicle drives itself. The driver becomes a passenger and he does not need to have the accelerator and brake pedals and the steering.

      The only limitation of these kinds of systems is that they work better under particular conditions, like good weather or dry asphalt.

- Level 5 (completely automated):
    - It is an advanced Level 4, in which the pedals and the steering are still absent and every electronic system and device works well in different weather conditions, status of asphalt or traffic jam.

Today, in Europe, it is allowed a second level of autonomous driving, because the higher levels of automation will take effect when technologies will be more advanced to ensure road safety.

Simultaneously, in North America, the law is less stringent; in fact, there are many projects with vehicles, which use an automation driving of fourth and fifth level [4].

On the market, there are different models of vehicles with a 3rd level autopilot, like Audi 8 or Tesla model S and the Tesla founder, Elon Musk, was able to make an autopilot of fourth and fifth level. So, these new technologies will be exploited when Europe will decide to legalise the highest-level SAE systems [4].

1.3. ADAS: State of art

The different kinds of ADAS systems use electronic devices to capture and elaborate a road scenario, pedestrians, bikers and other possible driving obstacles and traffic signs.

Below, a list of ADAS sensors:

- Cameras;
- Radar;
- Lidar;

Figure 2 – Electronic systems

1.3.1 Camera

These based-camera sensors are used to help the driver during the driving experience and are positioned outside the vehicle, on the front, back and sides to capture images of the road, pedestrians, bikes, street signs, etc. The images are analysed by supporting software and the information then triggers a response to improve safety.

1.3.2 Radar

Radar works on the principle of transmitting and receiving radio waves after reflection. It is able to determine the position of objects, measuring:

-elevation: it is possible to receive an echo from the directive antenna when the radar radiates in the direction of it;

-distance: measuring the delay with which the echo arrives;

It has many advantages over other sensors, like:

- The weather conditions do not affect radar-based systems;
- It works seamlessly under different lightning conditions, night or day;
- Long range radar systems can see very far, between 30 to 250 metres range;
- Through radio waves is easier to measure speed, distance and exact position of an obstacle;
- Radar can differentiate between stationary and moving objects and detect multiple objects simultaneously;

There are also many shortcomings, like:

- Detecting small objects with Radar is relatively difficult for shorter wavelengths;
- There are radar-based ADAS systems which not work well in closed environment such as tunnels and they go into stand-by mode;
- These systems have some problems to recognize and classifying objects;
- They could have interference with other sensors;

Radar is cheaper and provides wider range of applications than other technologies. For car manufacturers and OEM, It will become one of the best sensors for ADAS techniques.

1.3.3 Lidar

This sensor has the same working principles of radar. It emits electromagnetic wave pulses and collects those reflected by objects to extrapolate their characteristics. Lidar sensors emit invisible laser lights to scan and detect near and far objects and create a 3D map of the obstacles; this is the main difference compared to Radar sensors. The LIDAR is increasingly used to measure distances, both horizontal and vertical and it is a technique used in laser altimetry, as it allows to measure correctly the distance between the measuring point and the destination point, with a fast and precis way.The light has a shorter wavelength than radio waves and this increases detection resolution and obstacle definition.

There are two kind of automotive LIDAR sensors:

- Electro-Mechanical LIDAR:

  It is the first-generation Lidar sensor in automotive field. The disadvantages are that it is bulky, very expensive and prone to wear and tear in though terrain. They are installed on the top of the vehicle and

rotate to scan the surroundings of the vehicle and typically cover a long range;

- Solid State LIDAR:

  In this case, all the components like emitter, receiver and processors are integrated on a single chip. They are fixed in front, rear, sides of the vehicle and have optical emitters, which send a burst of laser photons. The light collides with objects in the way and bounces back to the system's receiver. The processor fetches this data and produces a real 3D map of the vehicle surroundings.

Between these two electronic sensors, the automotive world is increasingly oriented towards the implementation and the use of Solid State Lidar.

1.3.4 ADAS techniques

The different ADAS techniques are clearly shown in figure 3 and they can be realized with the sensors described above:

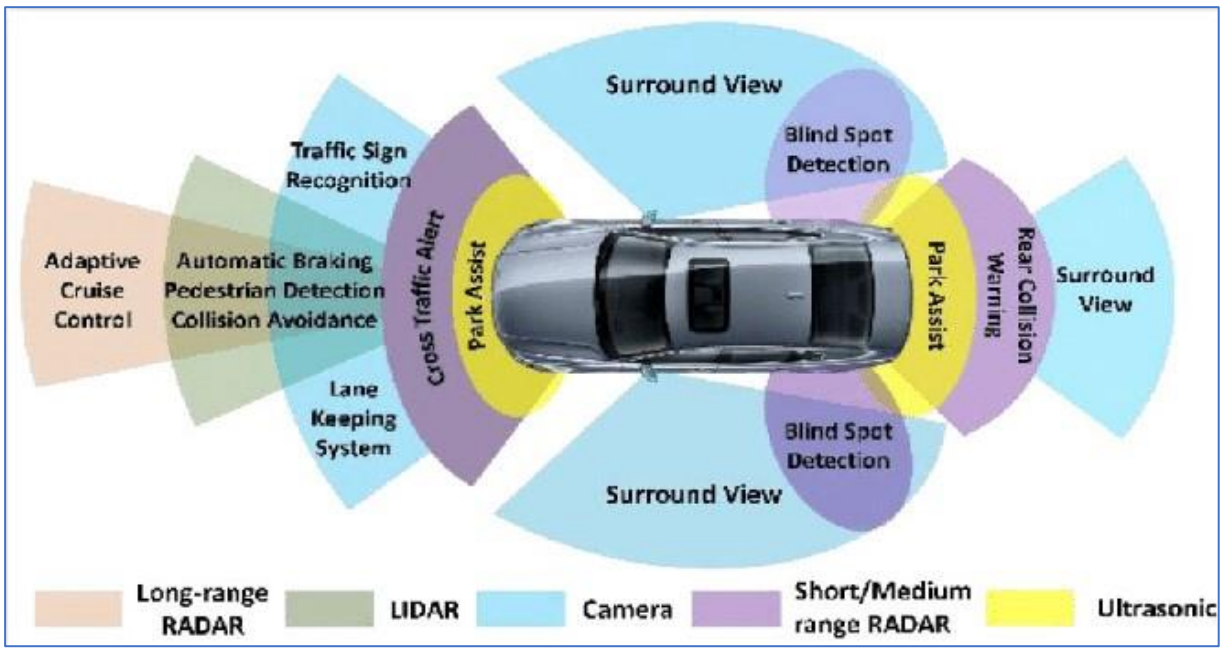


Figure 3 – ADAS state-of-the-art

The automotive field is investing heavily in the research and development of these systems. Some of them are:

- *Adaptive Cruise Control*

It is a cruise control system, that uses a radar sensor and a front camera to monitor the distance with the vehicle travelling in front and it reduces the speed of the car, when the speed of the front vehicle drops below a certain threshold. Then, when the street is free, the cruise speed set by the driver is restored

- *Advanced Emergency Braking*

  This system automatically brakes if it detects a potential danger situation.

  It is activated with cars and motorbikes, but the most evolved are able to identify children, cyclists, animals that had to cross the road (this happens within 50 km/h). If the situation is "at the limit", the system helps the driver to apply the maximum braking force on the central pedal. This technology works through a camera, usually mounted on the windshield and, as soon as the computer connected to it detects the rapid approach of an obstacle, the driver is alerted by an audible warning and the car stops immediately and autonomously.

- *Traffic sign recognition*

  The system uses a front camera that frames the road and recognizes the road signs when the car on which we travel meets them. If the driver is distracted, the device is able to warn him with a visual or audible warning.

- *Blind Spot Detection*

  It is a system of protection made by Volvo, to prevent car collisions. A camera is installed on the rear-view mirrors of the vehicle and monitors the nearest cars.

  When the other vehicles enter in the monitoring area, warning lights come on in the central rear-view mirror, to warn the driver and to keep him at a safe distance. The Volvo system can see not only the behind vehicles, but also those surpassed by your own.

- *Lane keeping systems*

  There are three different kind of these systems:

  - *Lane departure warning* (LDW), which warn the driver if the vehicle is leaving its lane with visual, audible and/or vibration warnings;
  - *Lane keeping assist* (LKA), which warn the driver and automatically take steps to ensure the vehicle stays in its lane;
  - *Lane centring assist* (LCA), which help in oversteering, keeping the car centering in the lane and asking the driver to take over in challenging situations.

- *Parking Assistant*

  The driver can make the parking manoeuvre without touching the steering wheel. The vehicle is able to measure the distance between two parked cars and to understand if it is possible the parking manoeuvre.

  The system is autonomous and the driver only needs to reverse and control the gas.

According to the European parliament, the market of autonomous driving vehicles will grow exponentially, leading to the creation of new jobs and an increase in profit in both the automotive and electronic sectors.

From 2022, all the vehicles approved in the year must have a level two of automation, with the standard ACC (Adaptive Cruise Control) and LKA (Lane Keeping Assist).

Self-driving cars, with third and fourth levels of automation, are currently undergoing testing and will enter the market between 2020 and 2030, while fully self-driving vehicles with level five of automation should be ready for 2030 [5].

2. EUROPEAN PROJECTS

The European Union has financed many projects focused on the assessments of impacts, benefits and costs of connected, cooperative and automated road-related driving systems with the only aim to increase driver's safety and vehicles reliability by minimizing fatal accidents.

The reference projects for the execution of this thesis activity are three:

- NEWCONTROL deals with highly automated operations for new generation vehicles;
- AUTOMATE develops, evaluates and demonstrates the concept of «Teammate Car» as one of the leading highly automated vehicle enablers;
- SAFESPOT deals with cooperative vehicles and road infrastructure for road safety ;

2.1 The NEWCONTROL project

NewControl [6] is a European project started on 2019 that will last 3 years.

It will develop virtualized platforms for vehicular subsystems that are essential to highly automated driving, realizing functions such as perception, cognition and control, to enable mobility-as-a-service for next generation highly automated vehicles.

Its overarching goal is to provide an industrially calibrated trajectory towards increased user-acceptance of automated control functions, through an approach that is centered on the premise of safety by design.

Newcontrol will deliver:

- Fail-operational platform for robust holistic perception through a combination of Lidar, Radar, and sensor fusion;
- Generalized virtual platform for stable and efficient control of propulsion systems;
- Cost- and power-efficient, high-performance embedded compute-platforms for in-vehicle perception, cognition, and control;
- Robust approaches for implementing, verifying, and certifying automated control for safety-critical applications several demonstrators will be built to display the project's findings and their capability to facilitate perception, cognition and control of next generation highly automated vehicles.

The developments in NewControl will facilitate significant cost reductions for essential modules necessary for future automated vehicles. Concomitantly, these developments will improve the safety and reliability of automated systems to levels necessary for mass-market deployment. These innovations will leverage the expertise of industrial and research partners along the complete semiconductor, automotive, and aviation value chains, providing Europe with a competitive edge in a growing market.

Importantly, NewControl's innovations will improve the market penetration of safety-centric automation systems, contributing directly to the European goal of zero road fatalities by 2050.

There is a list of eight WP (Work Packages), each of which contains objectives to be achieved.

The characteristics chosen for the basic scenarios realized in this thesis activity were taken from the WP4 (Work Packages 4).

The objective of this work package is to develop algorithms, methods and technologies to allow implementation of reliable cognitive perception, efficient signal processing and advanced adaptive and predictive control and signal processing algorithms as well as operation under faulty condition of individual components or the computing platform itself. The developed solution will enable novel functionalities for automotive system. Novel functionalities require more computing power and thus the emphasis will be on parallelization to take advantage of the new multicore architectures. The design of the architecture of the controller and embedded platforms is another crucial point that will be addressed in this work package. Attention will be paid especially to the following points:

• Utilization of predictive and reactive adaptive control strategies to increase the efficiency and lifetime of powertrain components;

• Design and utilization of virtual twins as a virtualized co-simulation platform for testing certification and validation;

• Development of algorithms for LIDAR and solid-state mirror LIDAR data processing and multispectral optical data fusion.

The aim is to implement and validate algorithms for human-like motion planning in complex urban scenarios.

The two complex scenarios that have been decided to study and will be described in the following chapters, are the urban traffic light crossing and the intrusion in the roundabout.

The goal is to implement a maneuver that highlights the fundamental interaction between man and the automated vehicle even in complicated cases such as those chosen.

2.2 The AUTOMATE project

The main purpose of the AutoMate European [7] project is to build a "TeamMate system", in which the human and the automation cooperate with each other to achieve a safe, pleasant and efficient driving, using a Human-Machine Interface (HMI). Man-machine interfaces are designed with a warning-based approach, in which the purpose of the system is to alert the driver to change his behaviour or to inform him of imminent risks.

The direct interaction between man and vehicle is still very important because there are some scenarios and some circumstances that are difficult to understand for electronic systems and the presence of human driver is essential for his own safety.

The vehicle is not able to predict the wrong behavior of a driver in traffic or irregularities on the road surface and to overcome from these aspects, it is essential to study the way in which the automated vehicle and the human driver can interact with each other easily.

The aim of this project is the building of the "TeamMate car" that is able to replicate the driver's attitudes, through the recognition and the online learning of human intentions and capabilities.

These independent agents (driver and car) are part of a team and aim to carry out all the driving activities needed to obtain a safe and efficient driving in different traffic conditions.

The project objectives are six:

- To develop solutions to monitor, understand, assess and anticipate the driver, the vehicle and the traffic situation;
- To develop solutions for flexible, gradual and smooth distribution of tasks between driver and automation to better handle critical driving situations;
- To develop solutions allowing the TeamMate Car to plan and execute driving maneuvers in a human expert-like way;
- To develop solutions to assess and guarantee safety of all manual and automatically generated maneuvers at any time;
- To develop solutions for optimized human-machine interaction;
- To develop demonstrators to test the safety, efficiency and effectiveness of the TeamMate technologies in real life conditions and consider security, legal and social issues.

Among the different types of scenario taken as a reference point to analyse the Human-Machine interface, the inrusion into the roundabout with automated mode has aroused interest of study.

Into the AutoMate project, this scenario is called "Eva Scenario" [7] and it is divided into two uses cases:

- Use case one (support in perception): when the TeamMate car approaches a roundabout, it detect high amount of traffic and can affect the efficiency of the automated manoeuvre. The vehicle may take some time to enter the roundabout or the automated system may not be able to perform the desired manoeuvre. This is where the interaction between man and machine comes into play. To speed up the manoeuvre, the Team Mate asks Eva to check the available space and to provide a trigger to start the introduction on the road. Eva checks the traffic and gives the confirmation when entering the roundabout. Thus, the TeamMate car notifies the feedback and enters the roundabout in automated mode.
- Use case two (support in actions): The TeamMate vehicle perceives the information from the maps and so it is aware of the presence of a roundabout for the time needed to enter and for the absence of clear signs. The car asks Eva for help which will choose when entering the roundabout and will take care of the lateral control of the machine. The driver (Eva) has to accept the request of cooperation with TeamMate car and his attention is essential.

Therefore, entering a roundabout is a well-known issue for the autonomous driving in term of efficiency and comfort and is one of the most interesting manoeuvres for the world's car manufacturers.

2.3 The SAFESPOT project

SAFESPOT [8] is an integrated research project co-funded by the European Commission Information Society Technologies whose aim is to create dynamic cooperative networks where the vehicles and the road infrastructure communicate to share information gathered on board and at the roadside to enhance the drivers' perception of the vehicle surroundings.

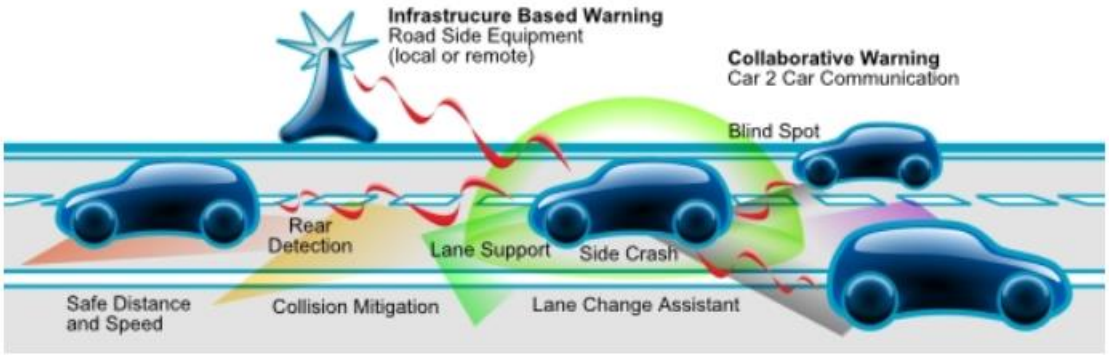


Figure 4 – SAFESPOT infrastructure

The SAFESPOT objectives are as follow:

- To use the infrastructure and the vehicles as sources and destinations of safety-related information and develop an open, flexible and modular architecture and communication platform;
- To develop the key enabling technologies: ad-hoc dynamic network, accurate relative localization, dynamic local traffic maps;
- To develop and test scenario-based applications to evaluate the impacts on road safety;
- To define a sustainable deployment strategy for cooperative systems for road safety, evaluating also related liability, regulations and standardization aspects;

The co-operative networks are composed by the following communicated elements:

- intelligent vehicles equipped with on board co-operative systems;
- intelligent infrastructure including road side units;
- safety centers or Traffic centers that are able to centralize or forward safety information coming from the intelligent vehicle or the intelligent infrastructure;

This project was born for the following reasons:

- to increase road safety for all road users;
- to support drivers preventively to the proper maneuvers in the different contests;
- to optimize the intervention of vehicle controls with respect to critical situations;
- to enable the development of new safety applications based on the cooperative approach;

A large number of study applications were considered in the SAFESPOT project and for this thesis activity, it was chosen to realize the Lane Detection maneuver in two scenarios of European interest, as described in the reference projects:

- An urban scenario containing two interesting study applications at European level, that are traffic light intersection and roundabout

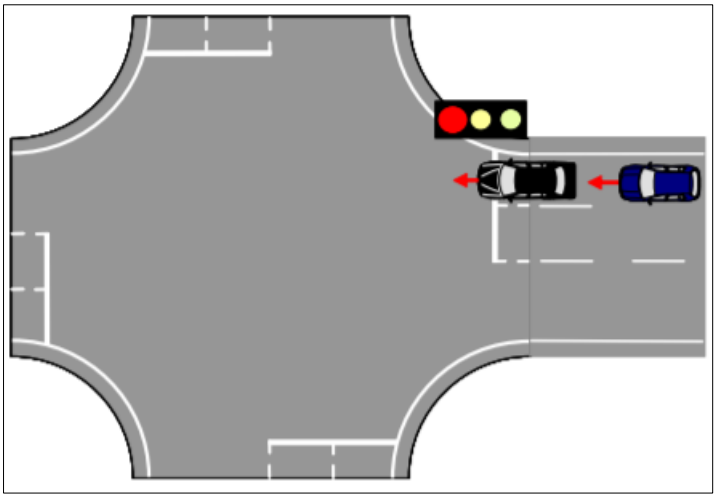


Figure 5 – Traffic-light intersection

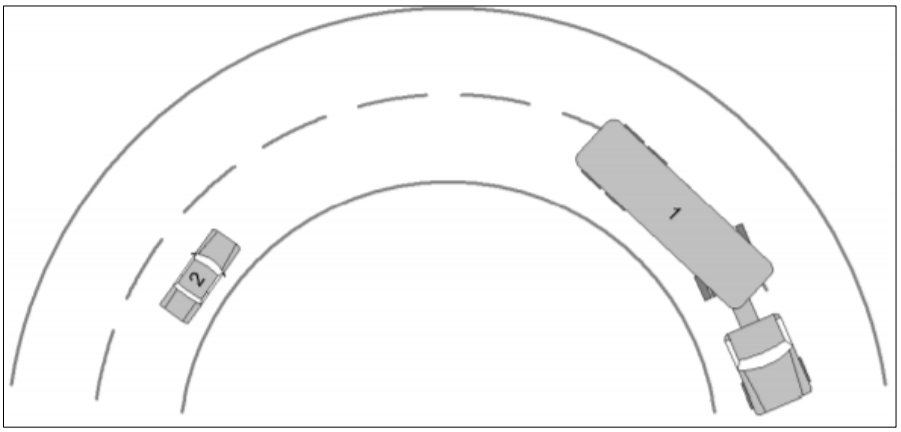


Figure 6 - Roundabout

- An Extra-urban scenario with two and more curved lanes and different speed limits

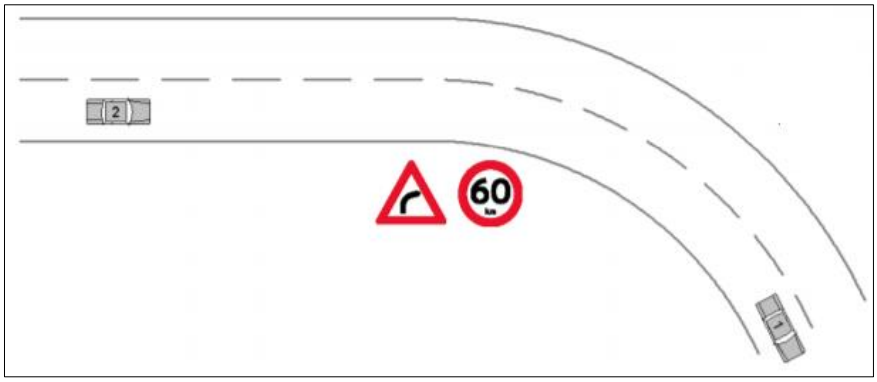


Figure 7 – Curved road

3. SIMULATION SOFTWARE

Given the reliability and safety issues at stake, developing autonomous driving manoeuvres is a very complex task [9].

To make the manoeuvre safe and reliable in any different driving situations that is possible to face on the road every day, it is important to carry out continuous tests to verify complex cases, thing that involves a large investment of money and time.

The number of situations to be tested on the road should be infinite, following the data volume, the criticality of the constraints involved and the sheer number of parameters at work.

It is difficult to replicate traffic, roadway and weather conditions, intentions of other road users, Euroncap safety requirements, etc.

For these reasons, physical road tests are replaced by simulation, through virtual driving.

During these processes, technologies and algorithms developed through simulation software are validated on digital simulators and compared with electronic representations of real-life situations previously recorded on the road, with the aim of developing the system to provide an optimal response.

In the autonomous vehicle development process, massive simulation assesses millions of scenarios corresponding to literally billions of miles of real-life road testing. It evaluates the reaction of detection systems, control algorithms and vehicle actuators and identifies types of problem-limiting situations that require physical testing.

It is very complicated to set up real-life traffic situations involving vehicle failures, but through digital simulation, the autonomous virtual prototypes vehicles can be immersed in an environment that reproduces what would otherwise be an inherently unpredictable type of traffic situation.

After that, it is important to perform the next physical tests to confirm the settings chosen and modify them if necessary, especially for situations that may prove difficult to model.

At the moment, simulative tests are not able to reproduce any condition of the real life, such as the way lighting conditions could affect a camera sensor or how fog inhibits sensor perception. That is why, the physical tests on the road are important too.

In addition to allowing development teams to comply with project schedules, the fact that physical testing can be reserved for validation of system response in limiting situations also leads to substantial cost savings, since simulation is much less expensive than road tests in real life and also involves fewer risk situations

for professional drivers. On average, a single physical test is worth about a thousand virtual tests.

For the reasons just exposed, the OEMs are investing more and more in the study and use of simulation software for ADAS systems because the autonomous driving is constantly expanding.

3.1 Software X

Software X [10], so called for a copyright question, is created for the realization of a virtual realistic environment to develop ADAS, automated driving systems, and has been implemented for Linux environment only. The Windows version is being studied.

The study focused on the use of two tools made available by the software, called Tool 1 e Tool 2 for matter of copyright:

- Tool 1: dedicated to the generation of logical road networks and to the creation of the surrounding environment (Baseline scenario);
- Tool 2: dedicated to the configuration of swarm traffic and to the creation of the route to be followed by the vehicles;

3.1.1 Tool 1

The first tool deals with the creation of the base scenario and the surrounding environment.

There are three different ways to build the desired track:

- In the first case, it is important to define the reference line of the track, which describes the track course from the top view.

  This tool supports lines, curves, clothoids, splines and parametric cubic curves to represent the reference line of a track and it is possible to create the desired path manually (green lines and yellow connection).

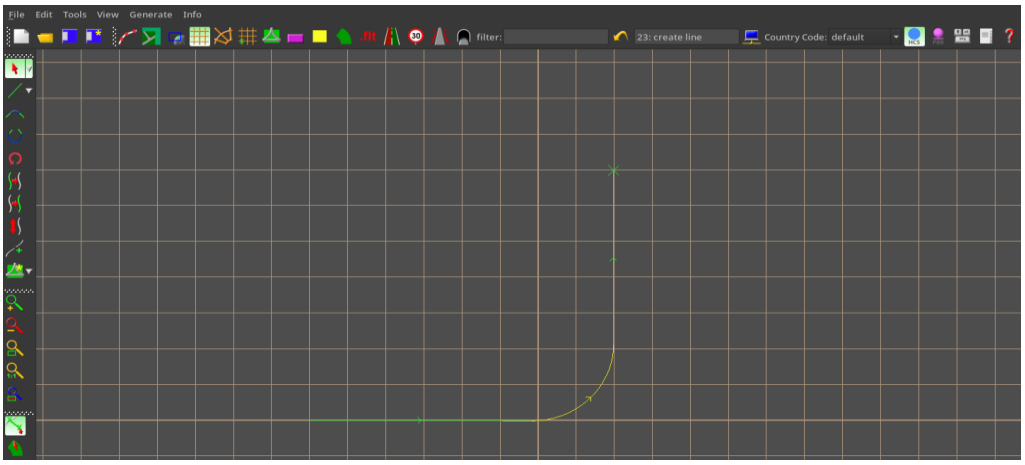


Figure 8 – Reference lines of the track

- Another possibility is to use existing roads or areas by extracting them from the satellite map and it is very interesting if the purpose is to realize a project with a real road and specific features.

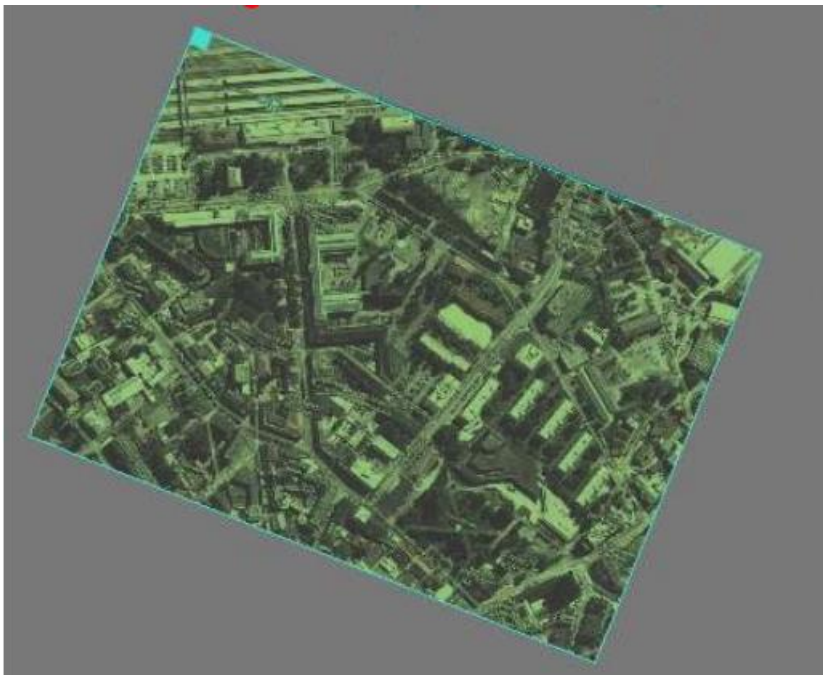


Figure 9 – Satellite map of the roads

- The last way to realize the baseline track of a project is to use a database into the software, where there are "street pieces" such as rural intersections, urban roundabouts, long straight motorway roads and it is possible to create logical connections to achieve the appropriate scenario.

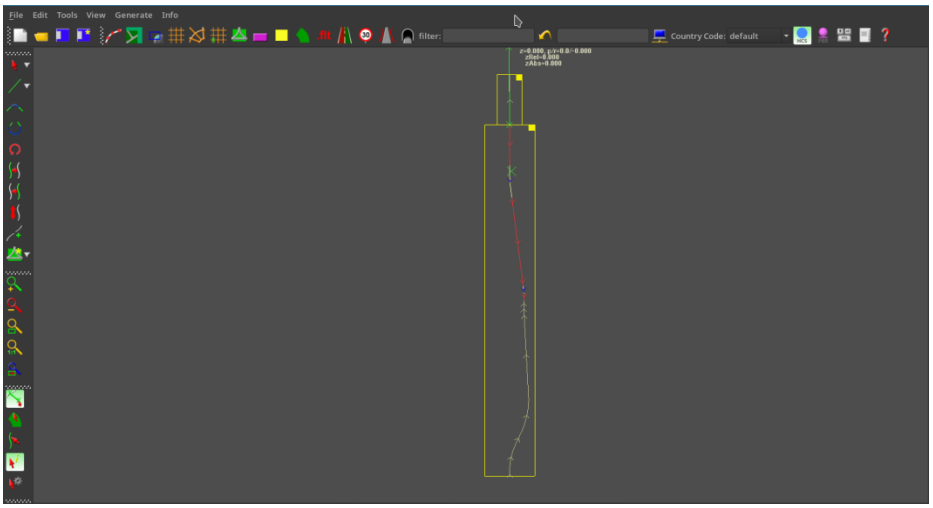


Figure 10 – Database roads

When the base scenario is ready, it is important to add the logical properties to customize it.

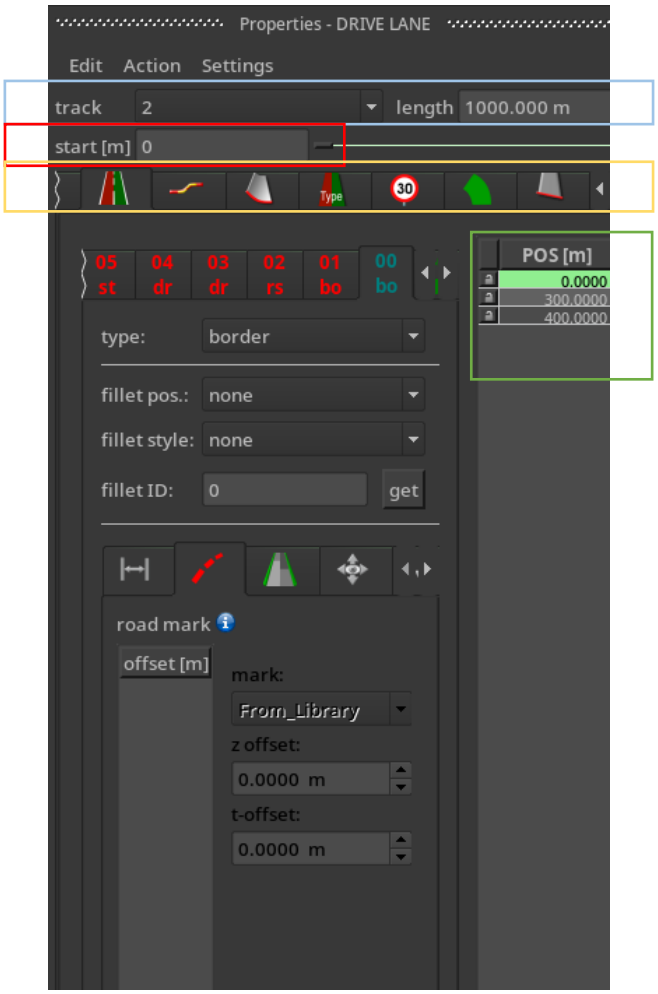


Figure 11 – Logical properties

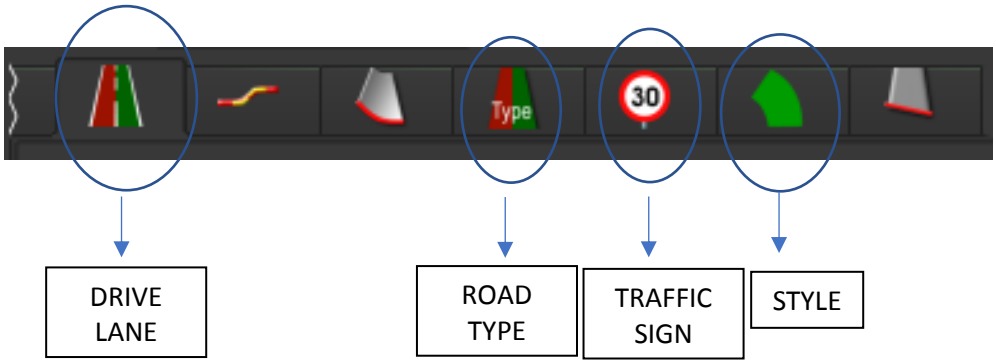
-In the blue box, there are the ID and the length of the track to customize.

-In the red box, there is the position where the base track should start.

-In the yellow box, there is the possibility to edit all the desired properties.

-In the green box is possible to change or modify roads, section by section.

By zooming on the yellow box, it is possible to understand how to edit the scenario properties:



- Drive Lane: lanes have to be define in order to be able to drive on a track.

  Each lane is identified with an ID and it is possible to edit the lane width, the type of road surface, the presence of continuous or stretched lines and the presence of pavements or green areas on the roadsides;

- Road Type: in this way, the user can choose the desired type of road, with a custom number of lanes, with rural or motorway surface, with pavement on the roadsides or with crash barriers.

- Traffic Sign: there is a big database of static traffic signs and road-marks. Many categories encompass a wide range of them, from rural to motorway signals.

  They can be configured by choosing the orientation, that is the direction in which they must be seen by traffic, the distance from the reference line, the signals ID and the heading, pitch and roll angles according to them rotation.

  It is possible to place the signals in the desired position of the track.

- Style: there are two ways to create a landscape:

  -using a predefined scenario;

-using the big database in which there are many realistic objects to realize a surrounding environment.

There are three predefined scenario categories:

-Bavarian Village;

-Motorway Railing;

-Town;

The database is complete and it is possible to create several detailed landscapes.

Figure 12 – Extra-urban landscape

Figure 13 – Urban landscape

3.1.2 Tool 2

This tool focuses on the study of dynamic elements that may be present in a realistic environment, such as vehicles, pedestrians, animals, etc.

The introduction of "Ego Car" is the first step for the realization of the scenario as a sequence of actions.

The Ego Car is the vehicle of reference for all the dynamic elements present in the scenario, that move according to the Ego car position.

It is possible to add the number of favourite vehicles to form the traffic jam and each time a car is included in the base track, the software automatically determines the orientation of it from lane information and traffic direction.

The technical specification of each vehicle can be chosen together with the car model and its position in the route.

Pedestrians, animals and other objects are dynamic elements too and it is possible to create paths and assign them to the latter.

In the figure 14, a graphical representation of how these activities is carried out.

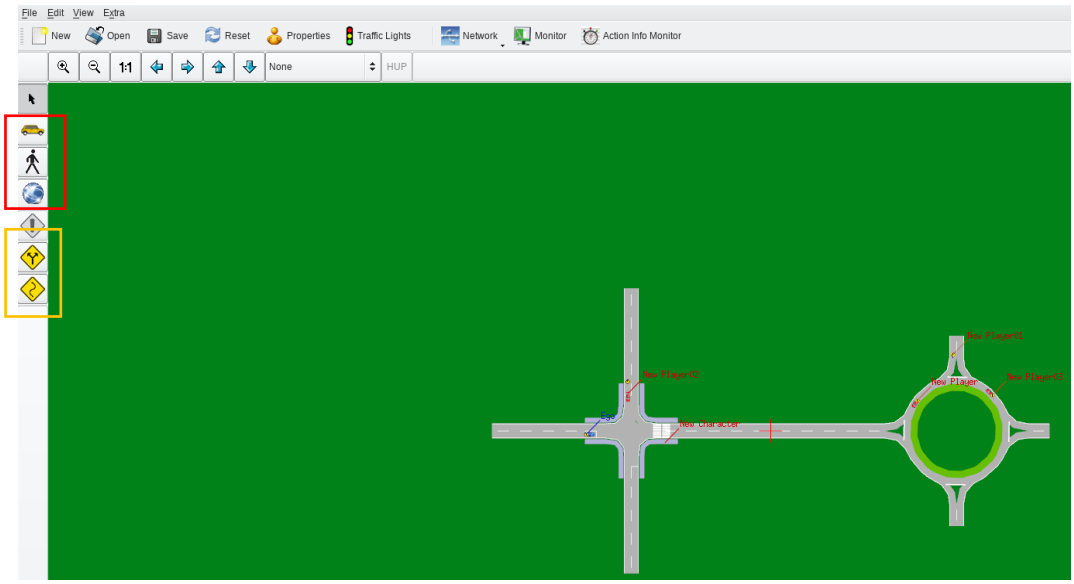


Figure 14 – Tool 2 interface

In the green area, there is the base scenario created with the Tool 1.

The dynamic elements to insert can be seen in the red box and on the track are defined "New Players". The Ego Car is in blue and the vehicles, referred to it, are in red.

In the yellow box there are the two ways used to achieve the desired path.

In the first case, a series of waypoints create the preferred path. It is important to give a different name to each route and then to assign it to the specific element in the track.

In the second case, the path shape can be determined following three different forms:

-Polyline;

-Spline;

-Clothoid;

The insertion of dynamic elements and the creation of paths to be followed by the latter define the scenario as a sequence of events.

3.2 Real sensor simulation

The Software uses an inner protocol for sending and receiving data to/from an external environment for example Matlab/Simulink or other simulation software. In the following paragraphs will be explained in detail how the data within the scenario are read and decoded by sensors placed on the Ego Car, to be understood and processed by an external software.

3.2.1 Data receiving from the software

To read all the signals during the software simulation is possible to use a "PerfectSensor module" which is positioned in an arbitrary location of the referred vehicle.

This sensor is so called, because its visual range consists of a shaped cone with a uniform sensitivity. It is an ideal sensor not able to reproduce all the technological limitations typical of a real sensor, such as the presence of dead zones or the frequencies in which it is possible to notice a reduction in sensitivity but it simulates a real sensor, which can read all the data needed and send them using an internal protocol to the software, in the form of packages to be decoded.

In the Software X, there are a lot of useful information, which represent the output data of the PerfectSensor, used for the implementation of a control of logic for autonomous driving manoeuvre, such as:

- Dynamic elements (vehicles, pedestrians, etc.) data, for example their category and ID, Heading, Pitch and Roll angles or their relative speed and position;

- Details of vehicles, their size, mass or axle distance, information about drivetrain (speed, torque, gear), news about driver-vehicle-interface (pedals, steering wheel), wheels and status of the vehicle lights;
- Details of road sections, such as the position of road-marks or traffic-signs or lane information;
- Choice of weather conditions, the status of the road surface, time-of-day, sky status, visibility and the information about the graphic image generation.

The reference system against which the sensor is able to see the objects, present in the scenario, is shown in the figure below:

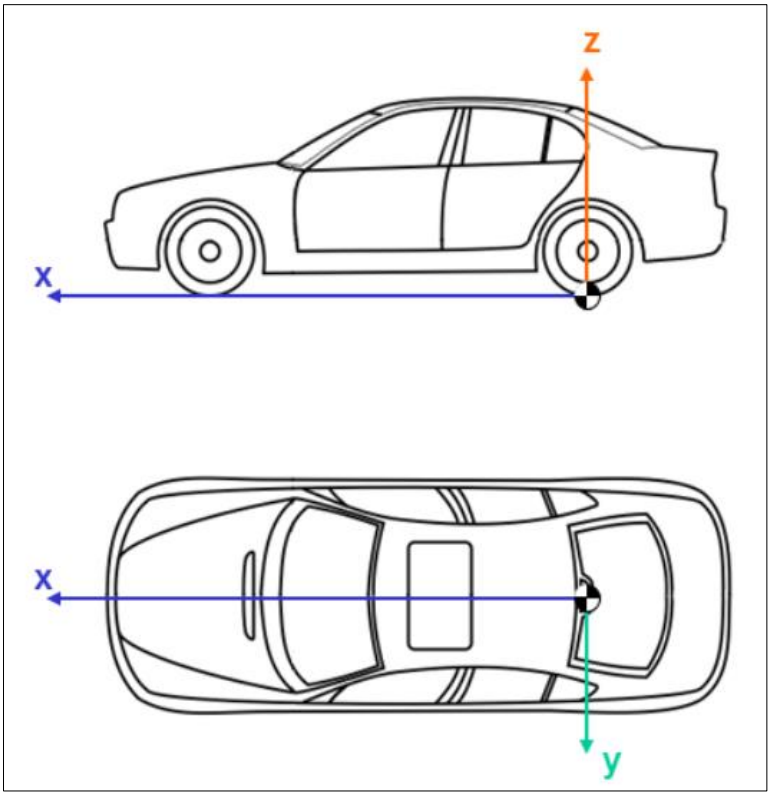


Figure 15 – Used co-ordinate system

The centre of gravity is the middle of the rear axle, on street level. The system is linked to the chassis, so that elements that are positioned in vehicle co-ordinates will also perform pitch and roll motions of the vehicle accordingly.

This co-ordinate system is typically used for the following reasons:

- Position of sensors and detected objects;
- Position of camera sensor;
- Position of symbols;

The software allows choosing between different references systems, changing the position of the source:

- Inertial system;
- Sensor system;
- RelativeToPlayer system;
- GPS system;

Figure 16 – Reference systems

The visual area of the PerfectSensor is called "sensor frustum" and it has a cone-shaped with a uniform sensitivity.

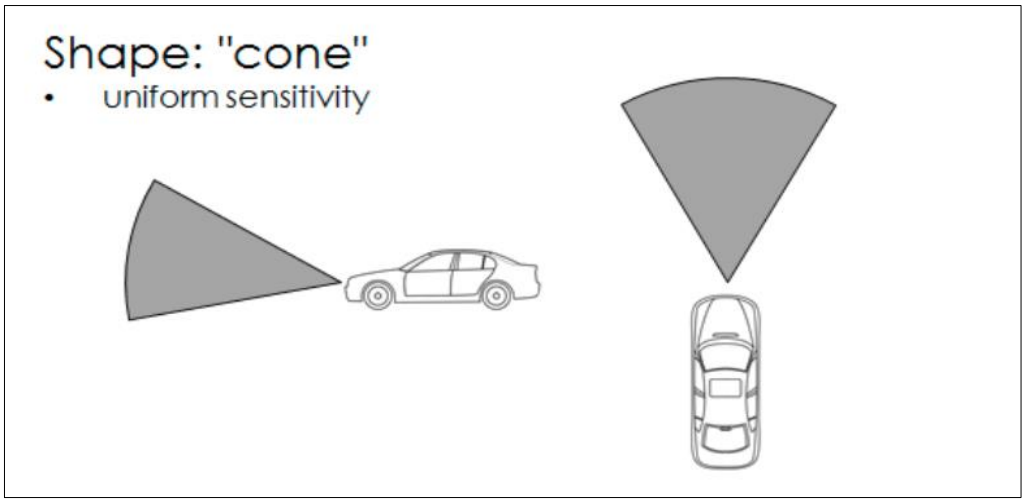


Figure 17 – Sensor cone-shape

The frustum parameters can be customized:

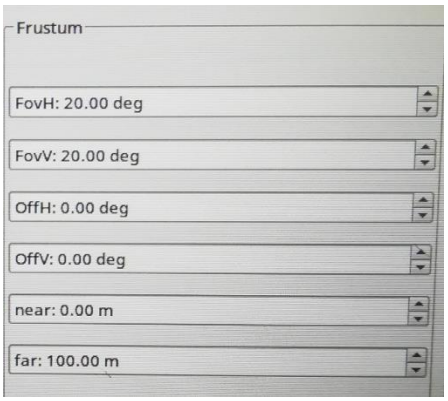


Figure 18 – Sensor frustum features

- FovH: cone angle to the top [deg];
- FovV: cone angle to the bottom [deg];
- OffH: cone angle to the left [deg];
- OffV: cone angle to the right [deg];
- Near: near clipping plane in [m];
- Far: far clipping plane in [m].

The PerfectSensor reads all the information available in the scenario and sends them to a default port established for the internal protocol to the software.

Then, these data must be decoded, as will be explained later, to be used in the implementation of logic control.

It is possible to select filters, on the data read by the sensor, ticking them from the graphical interface based on the signals needed as sensor output:

Figure 19 – Sensor filters

For example, in figure 19, two filters are selected and during the simulation, the sensor will read the laneInfo and the roadMarks signals only.

The sensors' efficiency depends on weather and road surface conditions. The Software allows editing these two aspects to realize realistic scenarios and test the manoeuvres implemented in the best possible way.

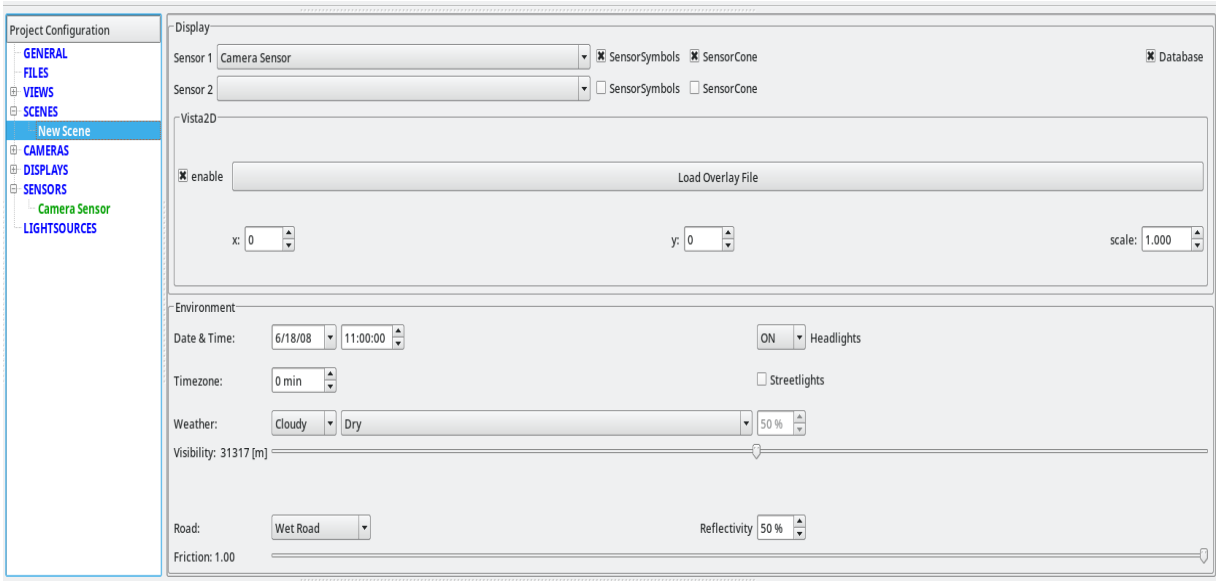


Figure 20 – Sensors features

By changing the parameters present in the section "environment", it may be possible to establish the type of lighting that characterizes the scenario, the shadows, the visibility and the friction of the ground. It all depends on the day and time.

Through the "display" is possible to activate the custom sensor and its symbols on the simulation screen.

4. CUSTOM SCENARIOS

4.1 Base scenarios

The creation of basic scenarios is essential for the implementation of control logic for the reproduction of autonomous driving manoeuvres.

The choice of landscape features can be a great contribution to deal with situations of real danger, on which are concentrated both today's OEMs and the various European projects on the autonomous driving.

Among the scenarios of interest at European level, two types were chosen to be analysed:

- Urban scenario;
- Extra-urban scenario;

4.1.1 Urban scenario

The urban scenario is composed of a traffic light crossing, a long straight road and a roundabout and through Tool 1 the basic track has been realized with the connection of the "street pieces":

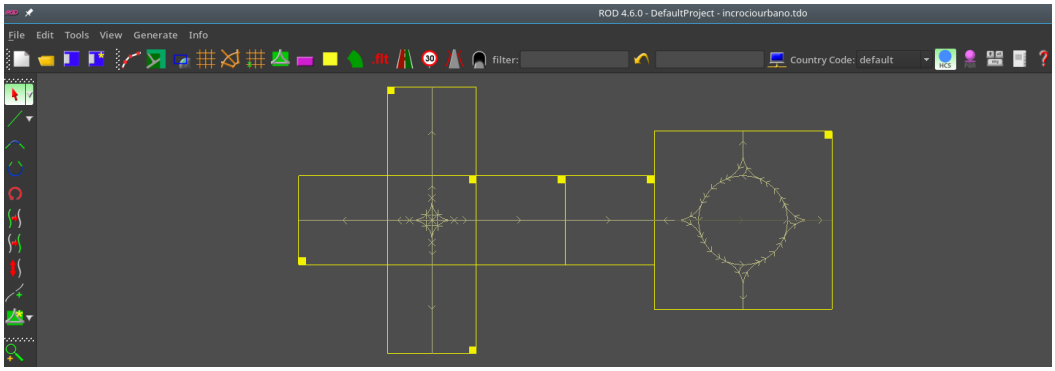


Figure 21 – Urban basic track

The most recent reference standard for the construction of new roads, with regard to the dimensional aspects of the different categories of roads and any associated service roads, is the D.M. (Ministerial Decree) "Norme funzionali e geometriche per la costruzione delle strade" [11].

According to this decree, the target trajectory is sized following the dimensions of Category E (District Street); a single carriageway street with at least two lanes, pavements and paved platforms.
These types of road connect each other surrounding neighborhoods or in case of large cities, they connect extreme areas of the same district.

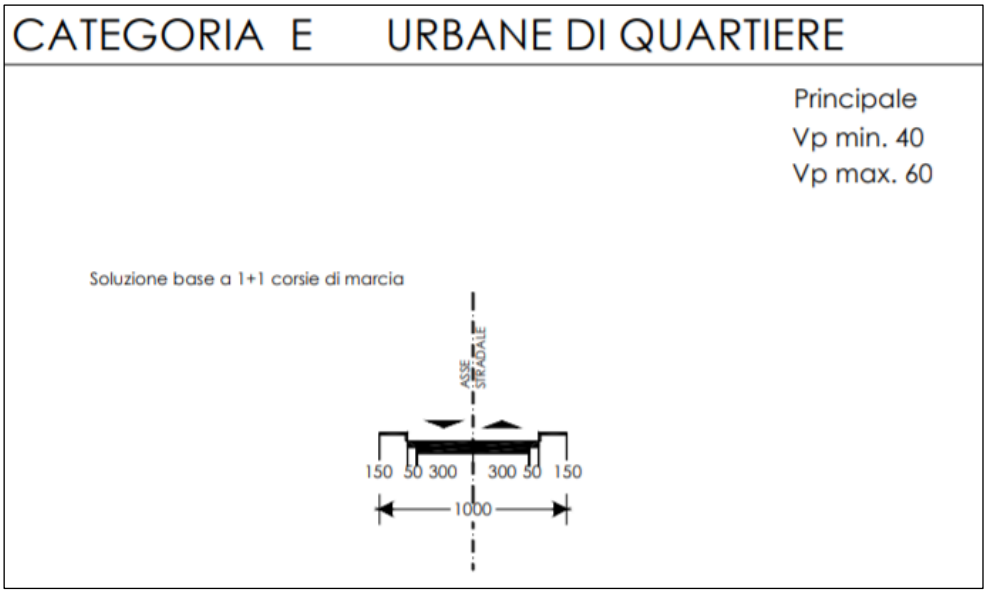


Figure 22 – Urban road dimensions

As shown in Figure 22, the neighborhood urban street of Category E has a unique roadway, with one or more lanes in the direction of travel with 3 meters wide each, right platform from 0.5 meters and pavements from 1.50 meters.

After that, the scenario has been enhanced through the introduction of the road signs:

- Traffic lights at the urban crossings (figure 23).

- The crosswalks before every traffic lights and before the introduction into the urban roundabout (figure 23 and figure 24).

- The signal to give precedence before the introduction in roundabout (figure 24).

-the prescription signal indicating the presence of an area where rounding is required in the direction indicated (figure 24).

Figure 23 – Traffic-lights urban intersection

Figure 24 – Urban roundabout

Subsequently the surrounding landscape was created. It represents a residential urban environment very similar to the real one of a small town, with detached houses, green areas and parks, clothing shops and markets.

Figure 25 - Intersection and straight urban roads

Figure 26 – Residential environment

Figure 27 – Residential roundabout

4.1.2 Extra-urban scenario

The extra-urban scenario has been designed by connecting together a six-lane curved road and a straight road with the same number of lanes, three for each side.

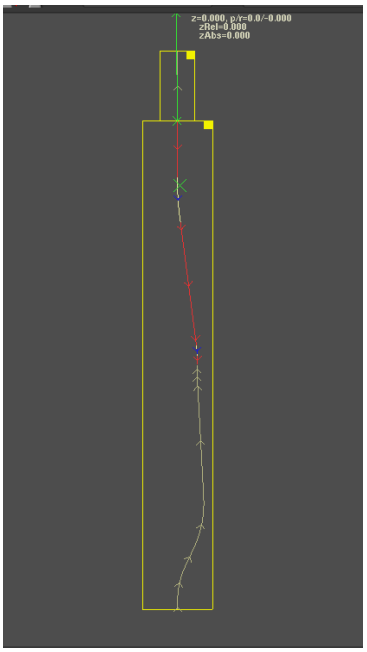


Figure 28 – Extra-urban basic track

In the first road piece of the Figure 28, there is a section highlighted in red that indicates the presence of work in progress and the reduction of the number of operative lanes.

This structural choice will allow verifying the success of the implementation of the objective maneuver, which will be described later.

According to the decree of the Ministry of infrastructure mentioned earlier, the extra-urban track has been designed in line with the dimensions of the category B:

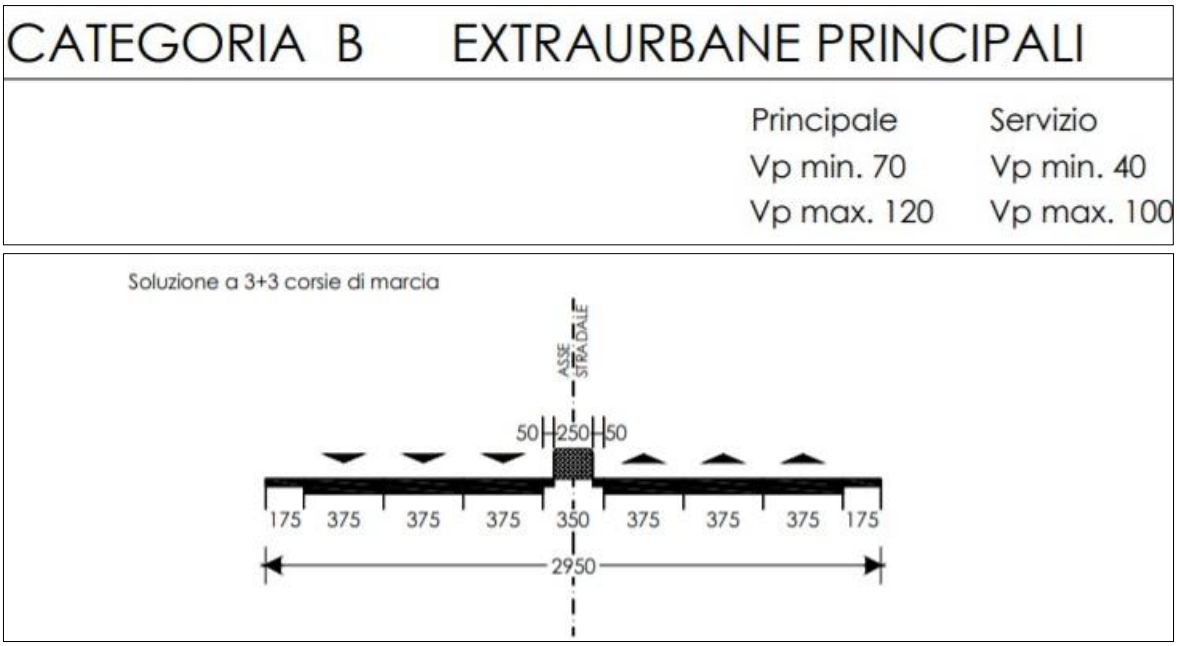


Figure 29 – Extra-urban road dimensions

The main extra-urban road is a road with independent lanes from 3.75 meters wide each and paved platform on the right from 0.5 meters.

There are three lanes separated by a bollard from 3.5 meters form each direction, for a roadway almost 30 meters wide.

Later, all needed road signs were introduced in the landscape [12]:

- Closed lane signals to indicate the reduction of the number of lanes for work in progress, the distance in which this will happen and the maximum wide and the maximum width of the transports that can be accessed (figure 30).

- Work in progress signals installed near construction sites (figure 32).

- Indication of speed limits, which are reduced from 120 km/h to 60 near the site (figure 31).

- Introduction of yellow lines to delimit all the working area (figure 32).

- No overtaking signs near the reduction of the number of lanes (figure 32).

The extra-urban base scenario is complete after the design of the typical landscape presents in the suburban areas characterized by many green spaces with a rich vegetation alternating with purely industrial elements.

Figure 30 – Extra-urban reduction of the number of lanes traffic signs

Figure 31 – Speed limit traffic signs

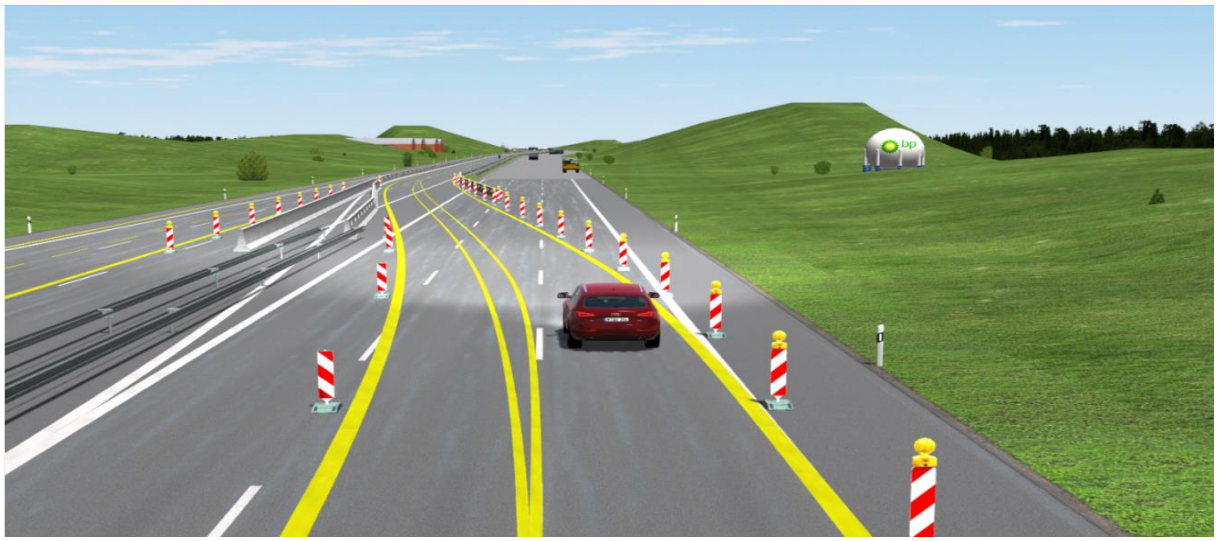


Figure 32 – Work in progress traffic signals

4.2 Scenario as a sequence of actions

The introduction of dynamic elements is required to complete the creation of scenarios and make them close to the real ones.

Both in urban and extra-urban environment the Tool 2 is used to create the traffic jam referred to the red Ego Cars.

Each vehicle introduced in the scenario has its own internal logic defined by the software and it is possible to choose one of the driver behaviors listed among the "driver properties"; in these cases, it was used a "Default driver".

The Default Driver is a driver with a perfect driving behavior, respects the speed limits and manages to keep the vehicle stable without lateral oscillations even in U-turn or sharp corners.

To generate the desired autonomous driving maneuver and introduce it in the created environments, it is important to take as reference vehicle the red Ego Cars and place the sensors which will be used on the latter.

These sensors are then activated and customized according to both the data they must read from the simulation environment and the control logic that must be realized for the implementation of the maneuver.

A "PerfectSensor" was introduced at the rear-view mirror of the Ego Car, both in the urban and in the extra-urban scenarios and all the available filters in the software have been activated.

The two Ego cars in red with PerfectSensor can read information about other vehicles in traffic, traffic signs and road marks belonging to the visual range of the sensor (yellow cone) and they can transmit these data on the simulation display, in real-time as can be seen in the figure 33 for the urban scenario and in the figure 34 for the extra-urban scenario:

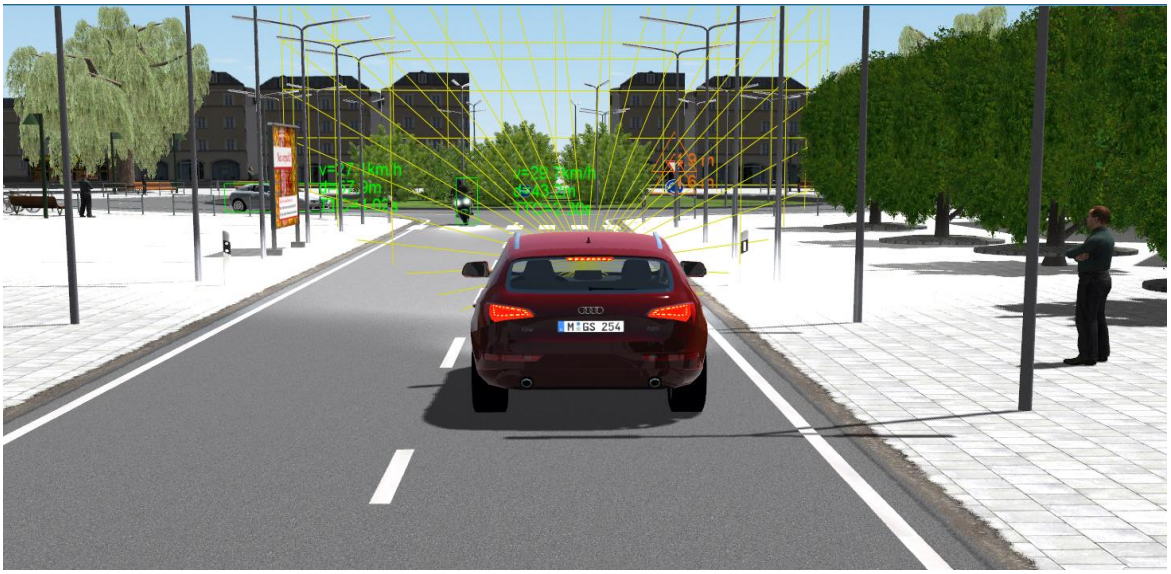


Figure 33 – Ego car actions in urban scenario

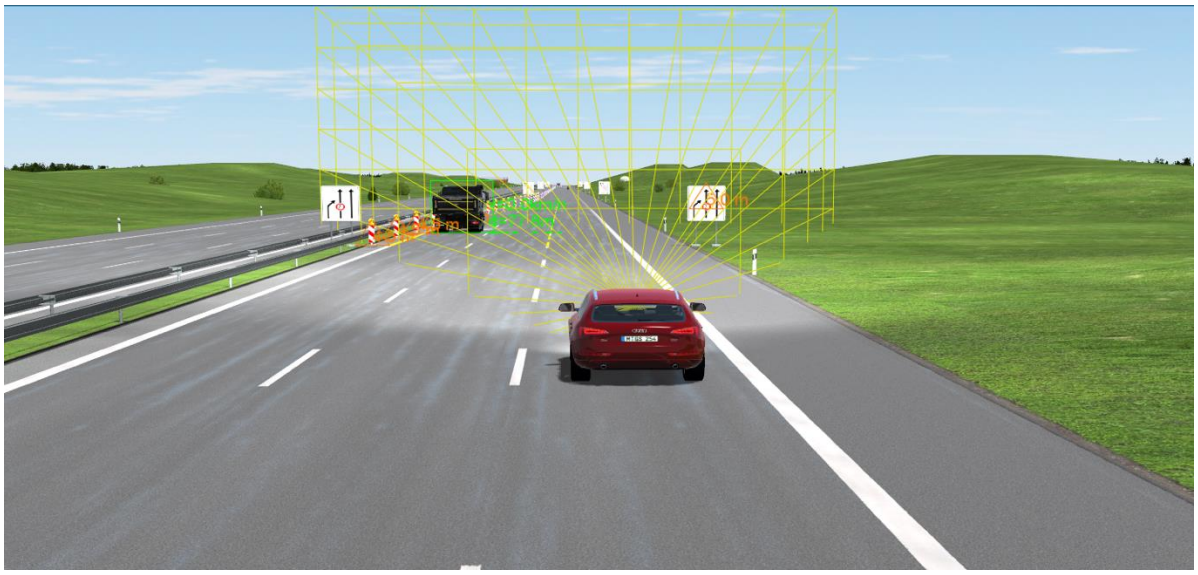


Figure 34 – Ego car actions in extra-urban scenario

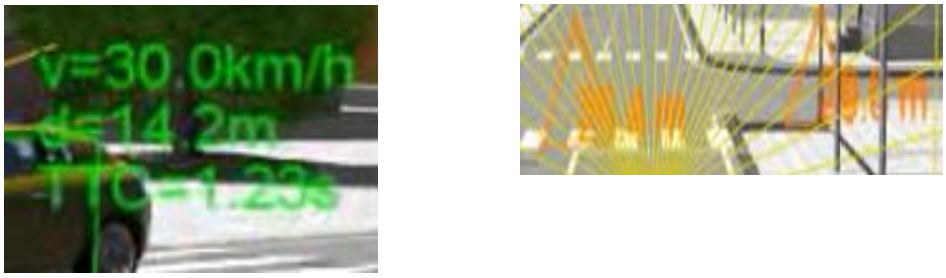


Figure 35 – Real time display data

When the sensor detects the presence of another vehicle in the yellow area, three information can be read on the real-time display (green data in figure 35):

- Speed of the vehicle in Km/h;
- Relative distance from the vehicle in meters;
- TTC (Time to Crush) in seconds.

When the sensor detects the presence of both vertical and horizontal traffic signs, instead, the distance in meters between the signals and the Ego car is calculated in real-time on the screen (orange data in figure 35).

All other data perceived by the sensors and not highlighted in real-time on the display are collected and saved in the default port of an internal protocol to the software for receiving and transmitting data.

At this point, the scenarios are complete and can be used for the implementation of an autonomous driving maneuver through ADAS techniques, inserting a real external vehicle model driven by a real driver.

5. MANOEUVRE IMPLEMENTATION

At first, a universal lane-keeping maneuver has been studied, realized by a model of custom vehicle in CarRealTime and totally implemented in the Matlab/Simulink environment.

The implementation of a universal maneuver is useful to understand the parameters on which the control should focus and to build validation tests in order to make it effective and efficient.

5.2 Model vehicle with 15 degrees of freedom

The vehicle custom model used in CarRealTime simulation is a model with 15 degrees of freedom, consisting of three masses each having their own degrees of freedom:

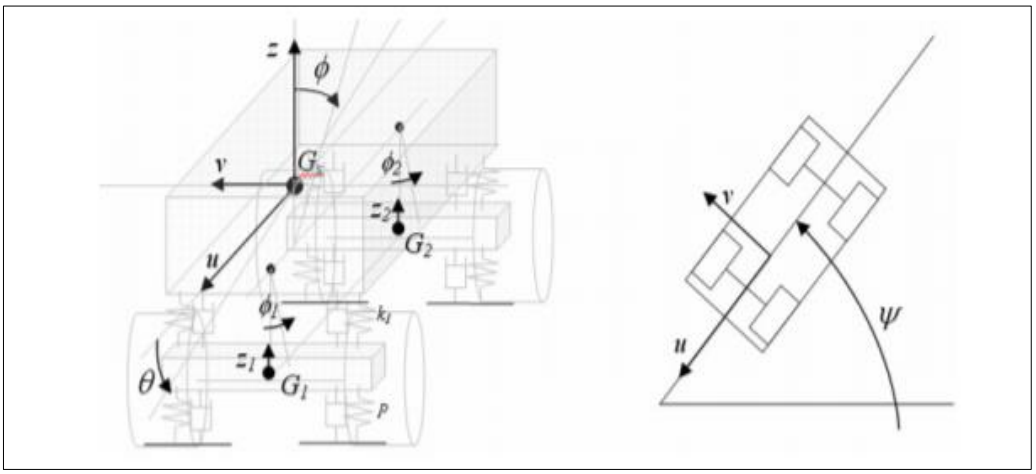


Figure 36 – sprung and unsprung mass

- Six degrees of freedom for the sprung mass; usually three components of the displacement define the position of the vehicle and three rotations define its orientation in space.
- Unsprung mass has four degrees of freedom, two for the front axle and two for the rear axle and other four degrees of freedom for the rotational speeds of the wheels, one for each independent suspension.
- One degree of freedom shall be assigned to the study of the dynamic behavior of the steering wheel, and therefore to its position.

In this case, the maneuver is called “free controls and rigid steering maneuver”, in which the pilot must counter the torque acting on the steering wheel to maintain the desired trajectory [13].

The total number of degrees of freedom is then 6 + 2m, where m is the number of axles. To these basic degrees of freedom, it is possible to add:

- The steering angle δ of the steering wheels. It may be considered as:

  – A given quantity, or an input, if the motion is studied with locked controls and the compliance of the steering system is neglected.

  – a known function of a single variable, the angle at the steering wheel δv, if the motion is studied with free controls but the compliance of the steering system is neglected.

  – a variable, linked by an equation expressing the compliance of the steering system, to the angle at the steering wheel δv that is a given quantity, or better, an input, if the motion is studied with locked controls and the compliance of the steering system is accounted for.

  – an independent variable, to which a further independent variable, the angle at the steering wheel δv is added, if the motion is studied with free controls and the compliance of the steering system is accounted for [14].

In CarRealTime the reference vehicle follows a straight road of about 100 m.

5.2 Target trajectory

The goal is to ensure that the reference vehicle, with the characteristics listed above, can follow a target-curved trajectory created in Matlab.

It is a trajectory made point by point formed by:

- A straight line of 60 meters, similar to that generated in CarRealTime;
- A curved line with a radius of 50 meters;

To achieve the desired lane-keeping maneuver it is necessary to analyze the parameters to be controlled for the chosen vehicle model to follow the curved trajectory:

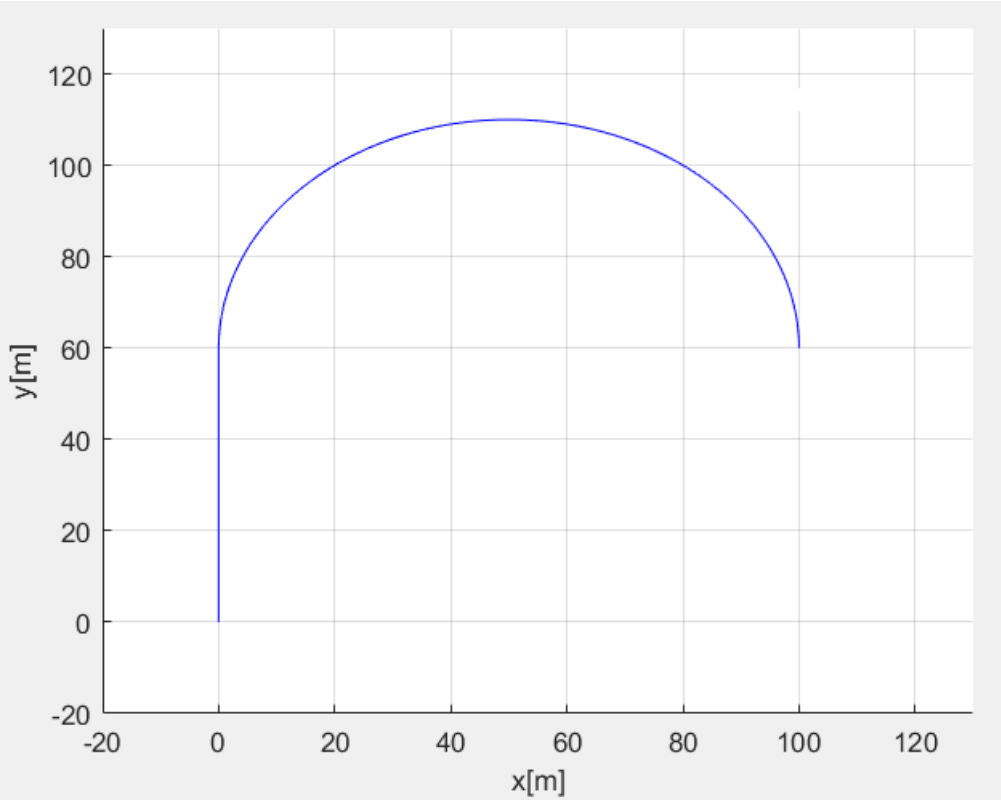


Figure 37 – Matlab target trajectory

5.3 Lane-keeping algorithm

In CarRealTime, the root body of the tree representing the rigid-body system is the Global Frame.

The vehicle body is an inertial frame with the origin point called $\mathbf{N_0}$ and unit vectors equal to $\mathbf{n_x}$, $\mathbf{n_y}$, $\mathbf{n_z}$.

The vehicle model is positioned with respect to the global frame.
The origin of the Vehicle Reference System $\mathbf{S_0}$ is located at $\mathbf{Z=0}$ of Global Reference Frame and at half front vehicle track.
The triad is oriented, at design time, as the Global Reference Frame with:

- X positive axis pointing forward in the direction of motion.
- Y positive axis pointing leftward;
- Z positive axis pointing upward;

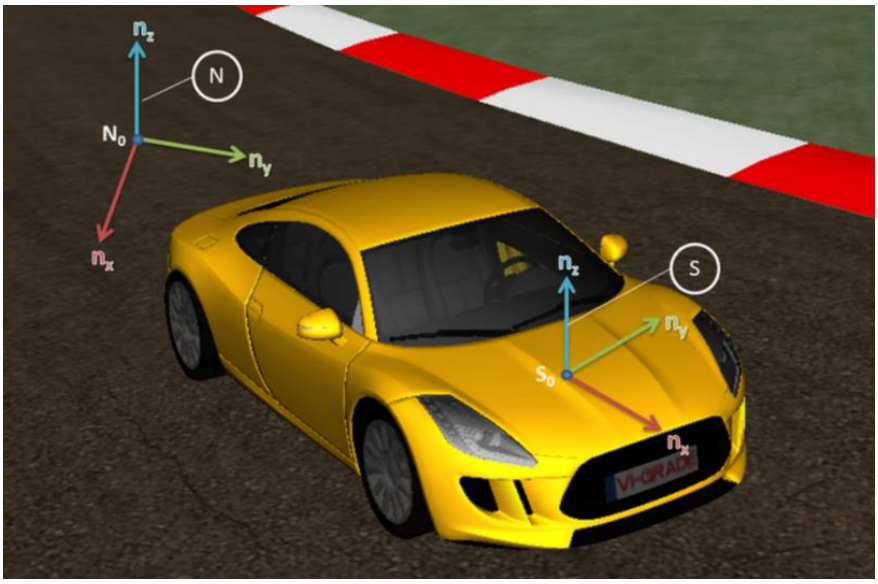


Figure 38 – CarRealTime reference system

The relative position of the vehicle to an identified line is described by two quantities:

- lateral deviation, which is the distance between the vehicle and the target centerline;
- yaw error, which is the angle between the longitudinal axis of the vehicle and the tangent to the centerline;

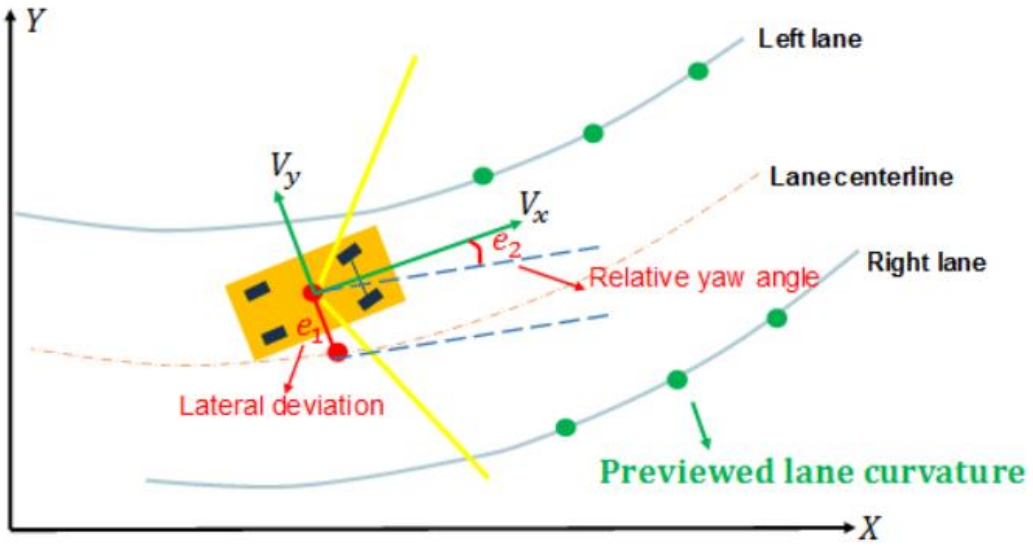


Figure 39 – vehicle model on the lane

The goal for the Lane-Keeping Controller block is to keep the vehicle in its lane and follow the curved road by controlling the front steering angle ∂.

The lateral deviation has been calculated by applying the Pythagoras theorem and by controlling, instant by instant, the minimum distance between the center of gravity of the moving vehicle and the points of the objective trajectory.

$$\text{Lateral deviation} = \sqrt{(y - y_{CG})^2 + (x - x_{CG})^2}$$

Where:

- (x, y) are the target trajectory coordinates;
- (x_{CG}, y_{CG}) are the center of gravity coordinates of the vehicle;

At the minimum calculated quantity, the tangent to the centerline is plotted point by point to calculate the heading angle of the road:

$$\text{Heading angle} = \tan^{-1}\left(\frac{x - x_1}{y - y_1}\right)$$

After that, the yaw error can be generated by calculating the difference between the yaw angle of the vehicle and the heading angle of the road to follow:

$$\text{Yaw error} = Yaw_{CG} - Heading\ angle$$

To keep the vehicle on the desired path and minimize the track errors, it is needed the use of a PID controller, with the specific characteristics listed in the Appendix 1.

The PID controller shall be calibrated appropriately to minimize the lateral deviation and the yaw error by imposing a steering angle ∂ on the vehicle, to prevent deviation from the target centerline.

This control of logic implemented in Matlab/Simulink environment will be the same that will be used to integrate the lane-keeping maneuver with the urban and extra-urban scenarios created with software x and it can be represented in this block diagram:

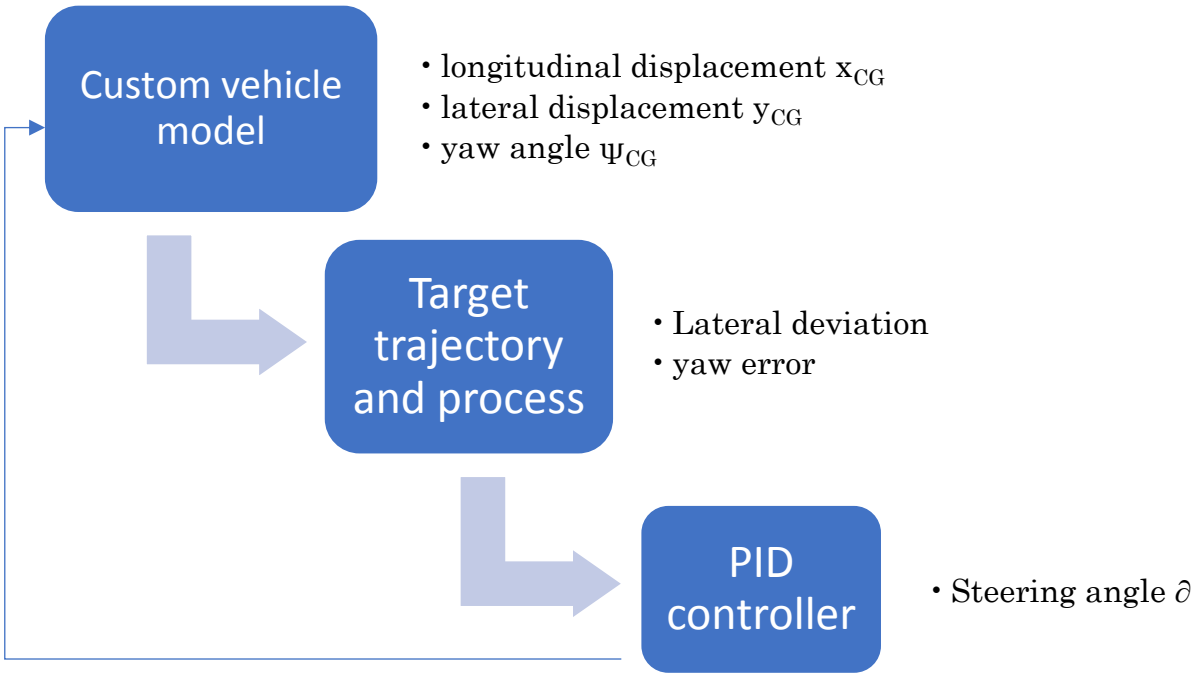


Figure 40 – Universal Lane-Keeping blocks

As can be seen in the figure 41, the vehicle follows perfectly the trajectory made in the Matlab environment and this highlights the efficiency of the control:

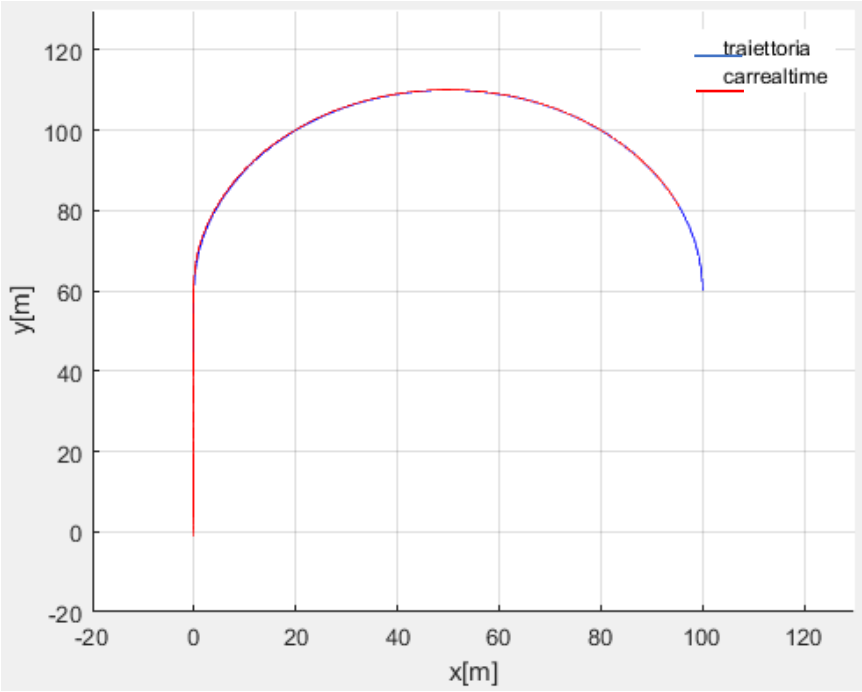


Figure 41 - overlapping of trajectories

The following is the instantaneous behavior of the lateral deviation and yaw error even before being minimized by the PID controllers.

During the first section, the vehicle runs along the straight about 60 meters and both the lateral deviation y_0 [m] and the yaw error ψ are null.

From about 7 seconds onwards, the vehicle must bend to follow the target trajectory and then the lateral displacement and yaw error begin to grow, following the trend shown in the Figure 42 and Figure 43.

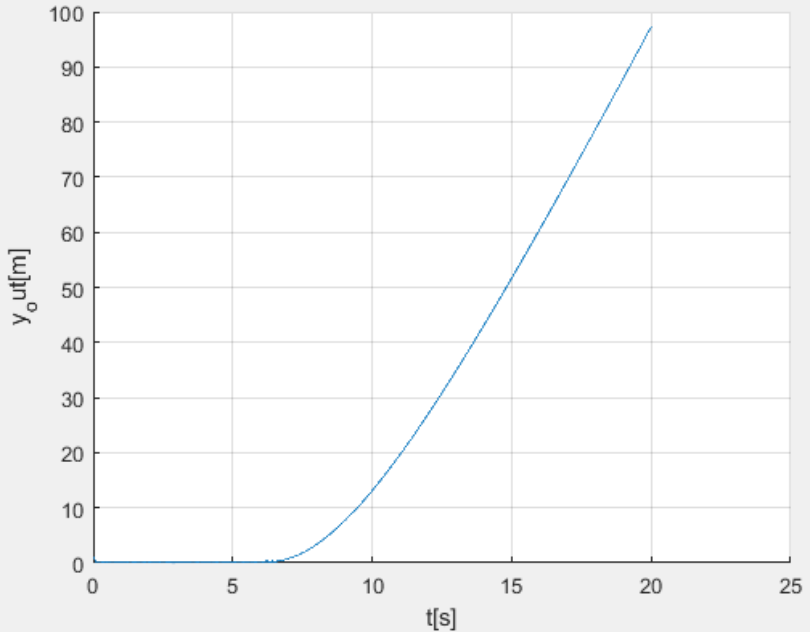


Figure 42 – lateral deviation

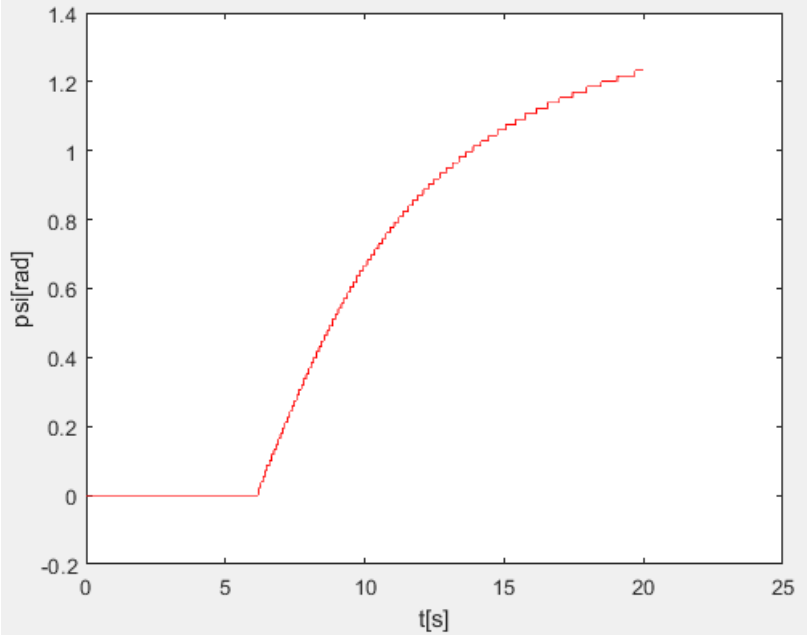


Figure 43 – yaw angle

PID controllers will tend these quantities, y_0 and ψ, at a value of zero.

6. SOFTWARE-IN-THE-LOOP (SiL)

The basis of the SiL activity used for the study of virtual guide is the integration between the lane-keeping maneuver, implemented in the previous chapter, and the custom vehicle model, made in CarRealTime, to be inserted on the two types of scenario created through Software x.

The Software-in-the-Loop testing is used to describe a test methodology to verify the electronic control units, which use full software emulation of the system for which they are intended, for example the sensor fusion data algorithm in a real-time simulation.

Many current ADAS developments make use of offline Software-in-the-Loop (SiL) simulations to run through myriad sensor calculations or ECU settings. This creates evaluation subsets for later confirmation with Hardware-in-the-Loop (HiL) or focused prototype vehicle testing with real drivers.

The advantage of this approach is the possibility to have virtual environments for testing using standard hardware, such as PC, avoiding the use of expensive physical simulation systems, such as physical prototype or test benches.

The diagram below perfectly describes the loop that must be realized to test the maneuver of lane-keeping on a simulation software:

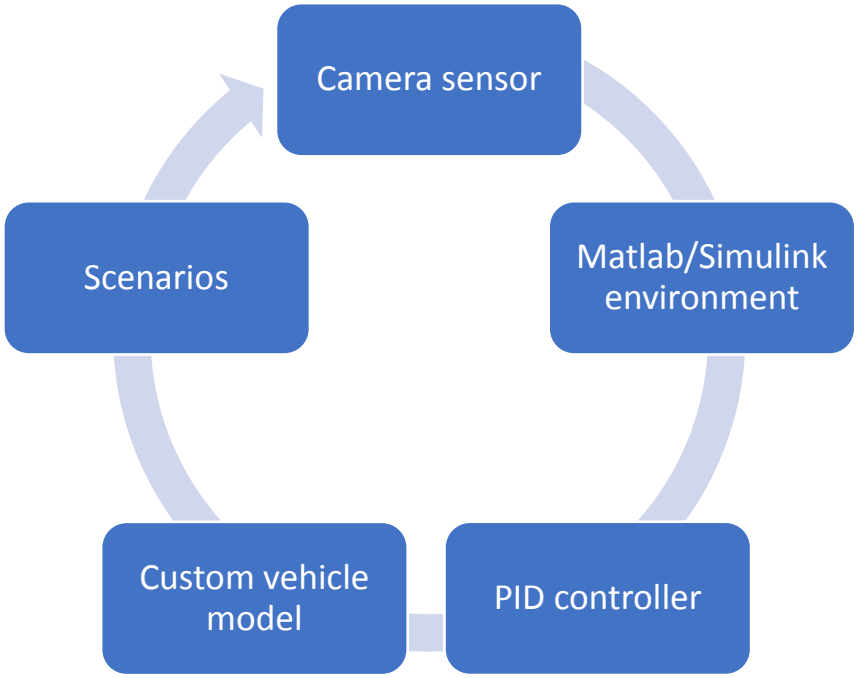


Figure 44 - SiL

6.1 Custom vehicle model connection with Software X

Software X provides two interfaces, which are used for sending and receiving data. The former one is used for high frequency data transmission whereas the latter one is used for event-based data only.

For this thesis activity was used the protocol of transmission and reception of the run-time high frequency data.

Through this protocol, the data that could be sent are:

- Object information, for example vehicle positions and states.
- Environment information.
- Sensors information.
- Road information.
- Weather conditions.
- Lights sources.
- Other custom packages.

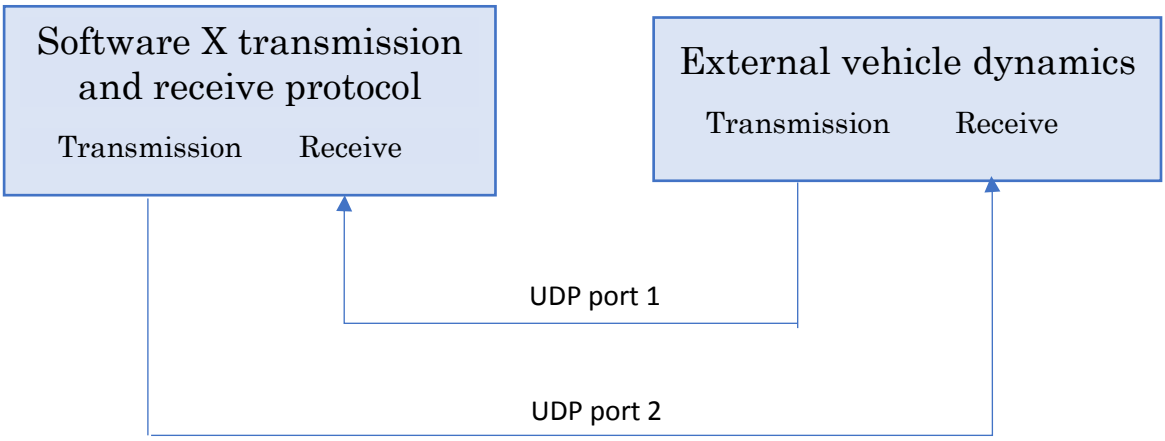
The custom vehicle dynamic model realized in CarRealTime may be connected to Software X, through the protocol described above via TCP or UDP.

In this case, an UDP connection was preferred because it is a connectionless protocol, with packet data exchange between sender and recipient, which does not require the preliminary operation of creating a physical or virtual circuit, on which to route the entire data stream in a predetermined and orderly manner over time.

It is very fast and it is generally used for applications for which a delay package is invalid, for example real-time audio-video transmission.

UDP provides only the basic transport level services, such as:

- multiplexing of the connections, obtained through the port allocation mechanism;
- error verification by means of a checksum;



UDP port connection

Software X was developed only to work in Linux environment, because the Windows version is being studied, while the custom vehicle model was created in CarRealTime, in Windows environment.

Similarly, the control logic to be integrated into the Software X is a logic implemented in Matlab/Simulink, then in the Windows environment.

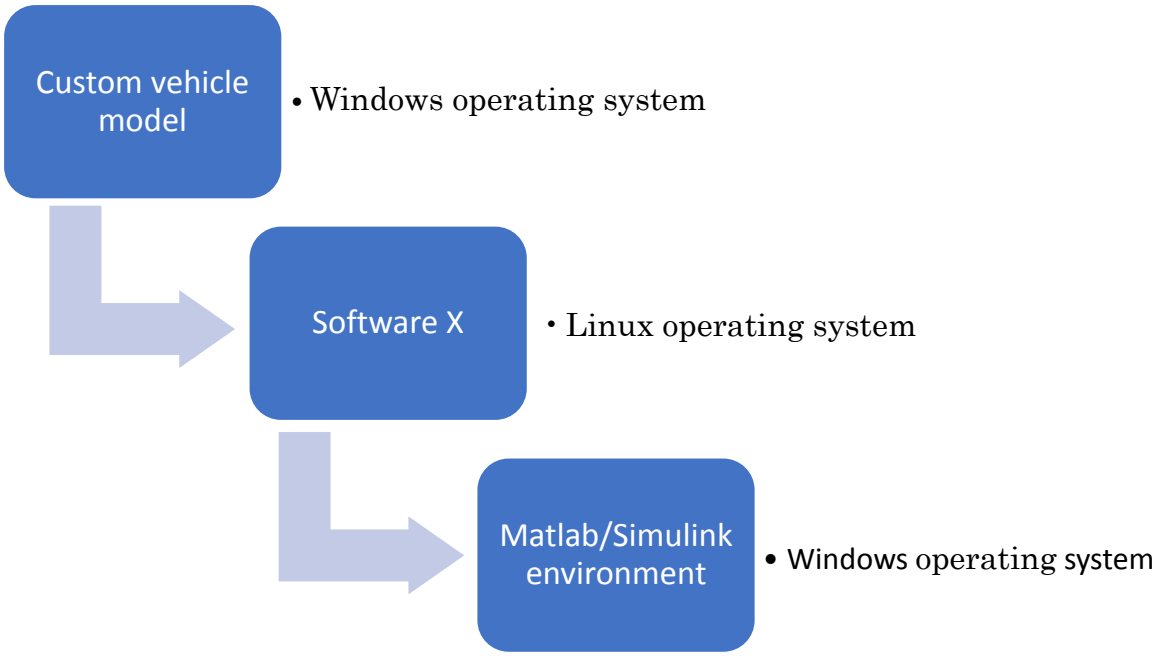


Figure 45 – Operating systems connection

Thus, to create the connections among the used software and to implement the Lane-keeping maneuver in the created scenarios with a custom vehicle model, it is essential to connect physically the two computers with the two operating systems via IP address and a straight cable, as can be seen in the Figure 46:

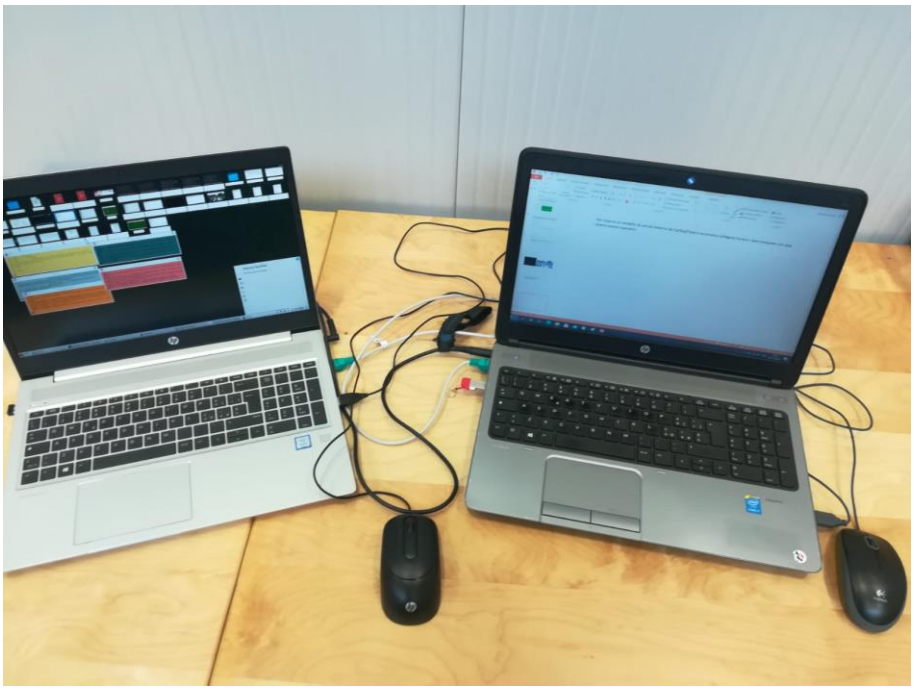


Figure 46 – PC connection

The physical connection is an Ethernet connection, which is been configured to run either via UDP:

Figure 47 – custom vehicle and Software X connection

It is essential to deactivate the vehicle model inside the software X to avoid a competing instance with the external custom dynamic.

After that, all the vehicle features, needed to realize the control, are sent via UDP to Software X and UDP block must be configured appropriately, setting the right IP address and the number of ports to send signals to.

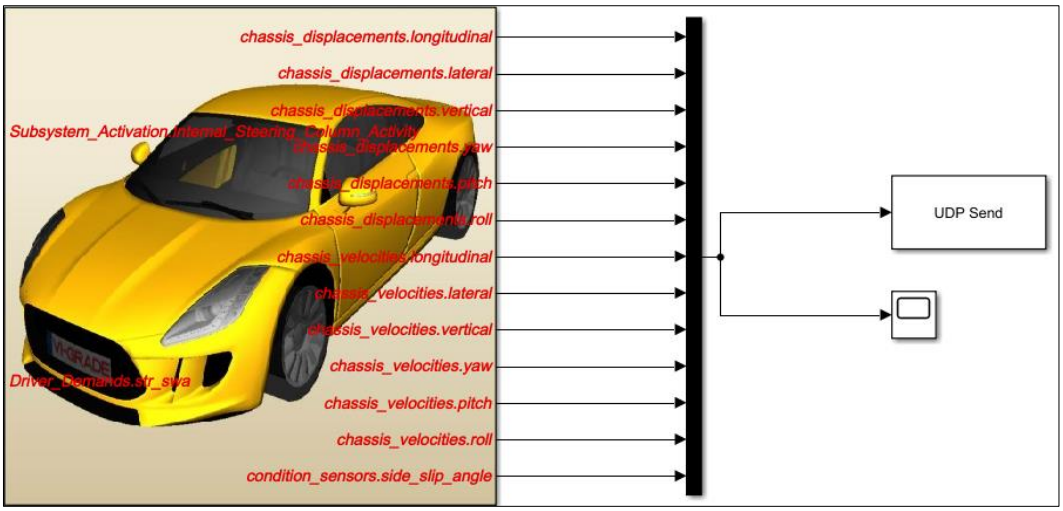


Figure 48 – CarRealTime send information

Software X receive these features through the UDP receive block in the same number of port set in the previous send block, on Linux environment.

All these information must be:

- Decoded and written into a file.c understandable for the software and compiled in a writer block;
- Transported from the decoding s-function to software X via UDP transmission block, by setting the port number from which the software will read the data;

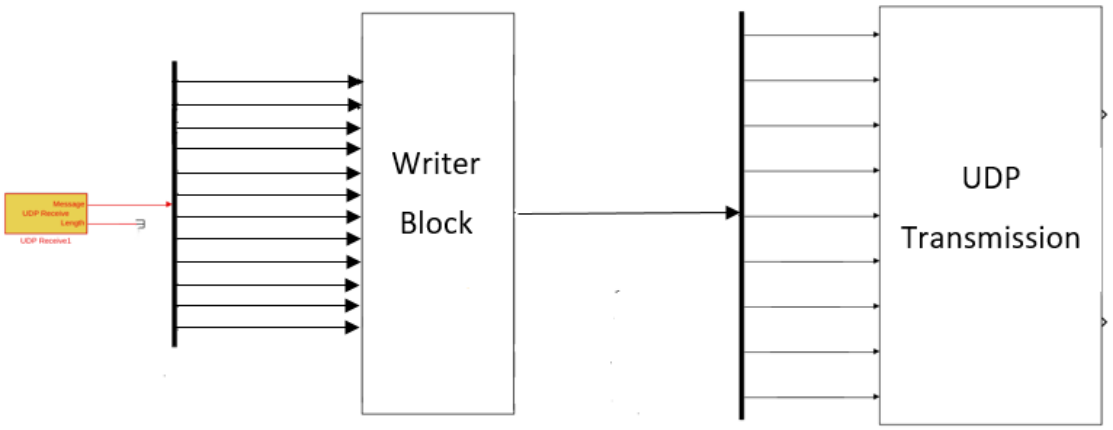


Figure 49 – Software X send blocks

6.2 Mathematical model of the Lane-Keeping maneuver

The Lane-Keeping maneuver relies on the lateral control of the vehicle dynamic model and on the use of a camera sensor to obtain lane, marks or other vehicles information.

It is interesting to refer to the mathematical model of the lane and to extrapolate the parameters necessary for the development of the control.

The most important features for our needs is lane detection and other elements such as obstacles, vehicles, road-marks, etc. are lane delimiters.

As discussed in the chapter 5, the relative position of the vehicle to an identified line is described by both the lateral error and the yaw error.

The reference frame (X_0, Y_0) for these quantities originates from a point on the delimiter line, designated by the perpendicular projection of the center of the vehicle's rear axle. The X_0 axis coincides with the tangent of the line at the origin, pointing to the direction of travel while the Y_0 axis is directed to the left. As this frame moves along the delimiter line, it will be called co-moving coordinate system in Figure 50.

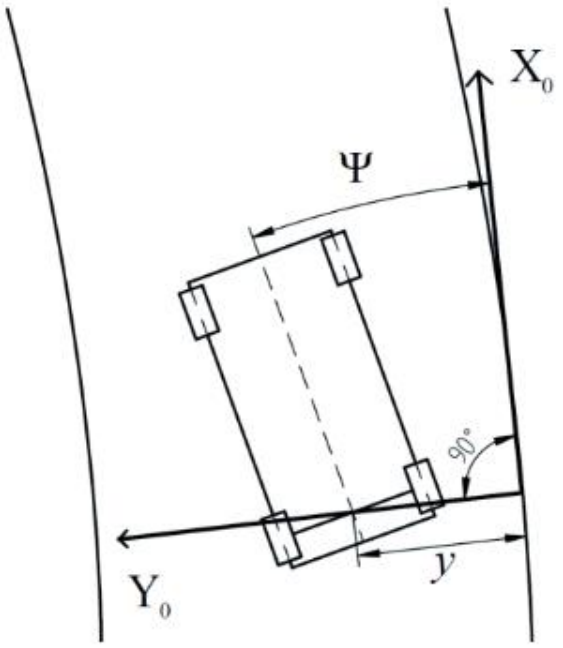


Figure 50 – Co-moving coordinate system

The theoretical road model used by the camera for curve fitting is the Euler spiral, which is a common track transition curve. The spiral cannot be handled conveniently by analytical methods, but since only the middle part of the whole spiral has practical usage, simple polynomial approximation can be used [15].
The third-degree function:

$$f(x) = \frac{C_1}{6}x^3 + \frac{C_0}{2}x^2$$

interpreted in the co-moving coordinate system can equally be used for approximating transition curves or circular arcs and for describing straight lines. The coefficients in the polynomial have direct meanings:

- c_0 represents the curvature;
- c_1 stand for the curvature gradient of the line at the origin.

When entering a camera on the vehicle, the equation (1) becomes:

$$g(x) = \frac{C_1}{6}x^3 + \frac{C_0}{2}x^2 - \psi x - y$$

The difference between the line in co-moving coordinate system and that one referred to the moving of the camera on the vehicle, gives the lateral deviation between the centerline and one of the lateral shift:

$$f(x) - g(x) = \psi x - y = \text{lateral deviation 1}$$

The camera is able to read up to four lines and it is possible to reproduce the same procedure for the other lane limits.

By calculating in the same way the lateral deviation 2, which is the distance from the left line of the lane, it is possible to obtain le lane width:

$$\text{Lane width} = \text{lateral deviation 1} - \text{lateral deviation 2}$$

The Yaw Angle of the vehicle respect to the co-moving reference frame is computable by deriving $f(x)$ for x.

The expression of this angle then is computed with the following equation:

$$\Psi_c(x) = \text{atan}\left(\frac{df(x)}{dx}\right)$$

With:

$$\frac{df(x)}{dx} = \frac{C_1}{18}x^2 + \frac{C_0}{4}x$$

It is possible to estimate the vehicle angle with respect to a stationary ground reference system by considering the integration of the yaw rate measured by the yaw rate sensors with which the car can be equipped with.

The ground reference system can be considered as a system pointed in the initial position of the vehicle reference frame, with the axes X and Y parallel to the initial layout of the axes of the same reference frame. Then it is possible to compute the Yaw angle in the global reference frame by integrating in time the Yaw Rate r:

$$\psi = \int_0^t r\, dt$$

Finally, the Yaw Angle of the co-moving reference frame referred to the global one is the difference between the vehicle absolute yaw angle and its orientation compared to the co-moving one:

$$\Psi_{cm} = \psi - \psi_c$$

6.3 Software X transmission data to an external environment

In order to obtain all the data needed to create a scientific memory by making more simulation environments communicate with each other, it is essential to position the sensors into the reference vehicle, defined as "Ego Car".

The camera sensor filtered all the desired signals and send it in the inner protocol of the Software X, in a default port number.

To translate these signals into input to be controlled with the model implemented in Matlab/Simulink must proceed by decoding the information, translating the code written text into a clear language for every external simulation environment.

In the Software X, all the data read by the sensors are collected in a series of categories that give the name to the myriad of packages in which they are grouped, such as the "Road Position package", the "Road Marks package" and others, as can be seen in the following figure:

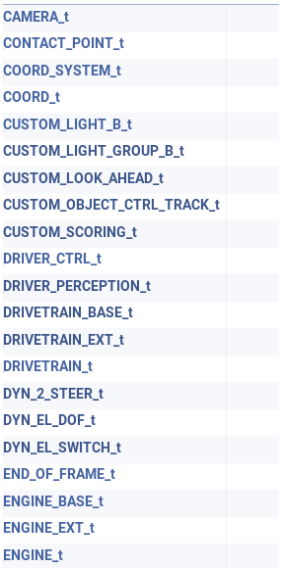


Figure 51 – Software X data packages

The packages chosen to be used must be read from the port where they were sent and then decoded, through the C-programming language.
A file.c called “reader” is created and it is structured as follows:

- Definition of the name of the package to be read
  (for example: DRIVER_CRTL_t).
- Definition of the total size of the considered package, making the sum of the size of each individual parameter in bytes (for example: 20 *4 bytes).
- Establish the channel in which to show the number of total elements that the sensor is seeing during each moment of the real-time simulation (for example: channel 9).
- Establish the channel to which the decode block of the objects seen in the previous channel, can be connected (for example: channel 10).

Some parts of this code are represented in the figure 52, for DRIVER_CTRL_t and for other packages too:

```
//    _CONTACT_POINT_t
// port 5 holds the number of valid elements in port 6
// port 6 holds up to 20 contact point infos (each with 56 bytes, i.e. 14 * 4 bytes)
ssSetOutputPortWidth(    S, 5, 1 );
ssSetOutputPortDataType( S, 5, SS_UINT32 );
ssSetOutputPortWidth(    S, 6, 20 * 14 );
ssSetOutputPortDataType( S, 6, SS_UINT32 );

//    _ENVIRONMENT_t
// port 7 holds the number of valid elements in port 8
// port 8 holds up to 1 environment state info (each with 48 bytes, i.e. 12 * 4 bytes)
ssSetOutputPortWidth(    S, 7, 1 );
ssSetOutputPortDataType( S, 7, SS_UINT32 );
ssSetOutputPortWidth(    S, 8, 1 * 12 );
ssSetOutputPortDataType( S, 8, SS_UINT32 );

//    _DRIVER_CTRL_t
// port 9 holds the number of valid elements in port 10
// port 10 holds up to 1 driver control infos (each with 80 bytes, i.e. 20 * 4 bytes)
ssSetOutputPortWidth(    S, 9, 1 );
ssSetOutputPortDataType( S, 9, SS_UINT32 );
ssSetOutputPortWidth(    S, 10, 1 * 20 );
ssSetOutputPortDataType( S, 10, SS_UINT32 );
```

Figure 52 – reader.c code

Then, the reader.c file will be compiled in Matlab/Simulink environment and the S-function included in the transmitting information model from Software X to an external simulation software is created.

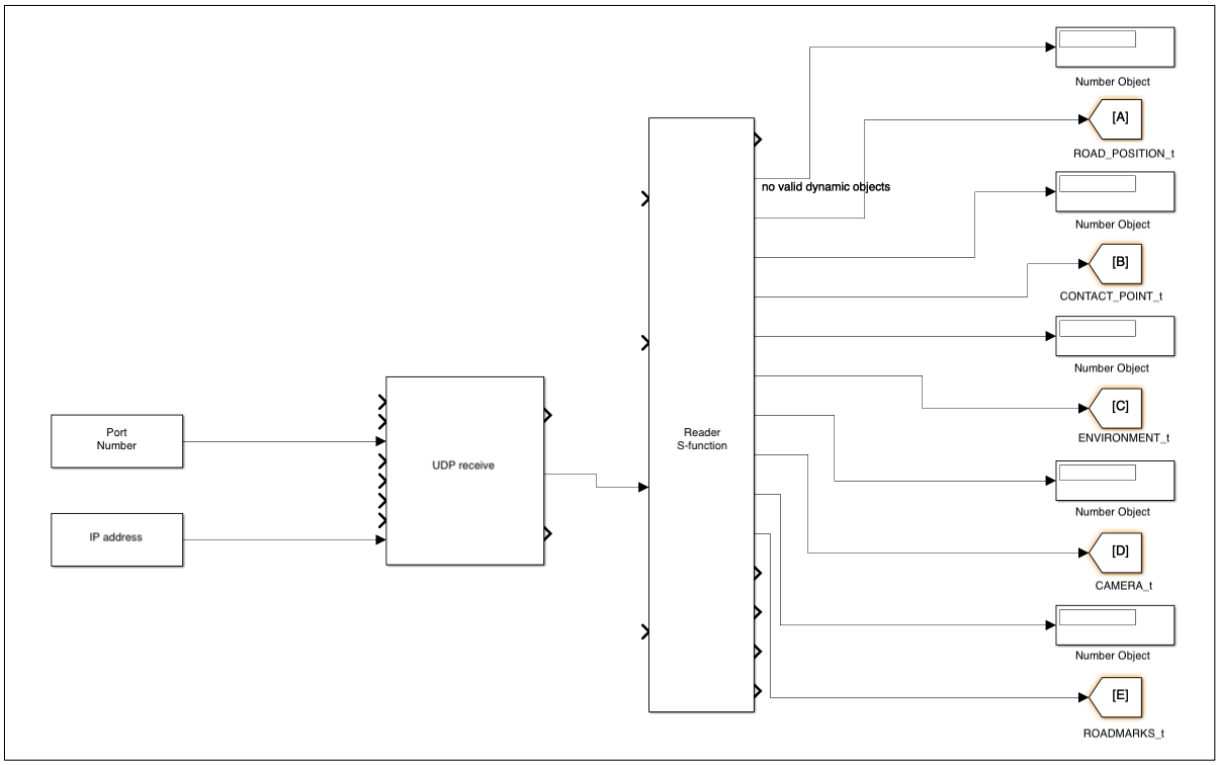


Figure 53 – Transmission data from software x model

Each packet contains a data structure, that is a set of parameters to be decoded and in which the type and the bytes size, of each one, are specified.
For example, in the following image there is the DRIVER_CTRL_t reference structure with all the parameters read by the camera and organized to be decode and used by an external environment.

Data Fields

uint32_t	playerId
float	steeringWheel
float	steeringSpeed
float	throttlePedal
float	brakePedal
float	clutchPedal
float	accelTgt
float	steeringTgt
double	curvatureTgt
float	steeringTorque
float	engineTorqueTgt
float	speedTgt
uint8_t	gear

Figure 54 – DRIVER_CRTL_t structure

A second file.c will be created and it will be called " Decode_<structure_name>". The information to be introduced in the code are:

- Insertion of both the package structure and the size of each parameter that you want to get in output.

- Definition of the output type of the data packet that you want to decode.
- Each output must be given a name by creating a data map.

```
// configure output ports
if ( !ssSetNumOutputPorts( S, 20 ) )
    return;

ssSetOutputPortWidth(    S, 0, 1 );            // ID
ssSetOutputPortDataType( S, 0, SS_UINT32 );

ssSetOutputPortWidth(    S, 1, 1 );
ssSetOutputPortDataType( S, 1, SS_SINGLE );     // steeringWheel

ssSetOutputPortWidth(    S, 2, 1 );
ssSetOutputPortDataType( S, 2, SS_SINGLE );     // steeringSpeed

ssSetOutputPortWidth(    S, 3, 1 );
ssSetOutputPortDataType( S, 3, SS_SINGLE );    // trotthlePedal

ssSetOutputPortWidth(    S, 4, 1 );
ssSetOutputPortDataType( S, 4, SS_SINGLE );       // brakePedal

ssSetOutputPortWidth(    S, 5, 1);
ssSetOutputPortDataType( S, 5, SS_SINGLE );        // clutchPedal

ssSetOutputPortWidth(    S, 6, 1);
ssSetOutputPortDataType( S, 6, SS_SINGLE );       // accelTgt

ssSetOutputPortWidth(    S, 7, 1);
ssSetOutputPortDataType( S, 7, SS_SINGLE );        // steeringTgt

ssSetOutputPortWidth(    S, 8, 1 );
ssSetOutputPortDataType( S, 8, SS_DOUBLE );       // curvatureTgt

ssSetOutputPortWidth(    S, 9, 1 );
ssSetOutputPortDataType( S, 9, SS_SINGLE );        // steeringTorque
```

Figure 55 – DRIVER_CTRL_t package structure and parameters size

```
tatic void mdlOutputs(SimStruct *S, int_T tid)

    _DRIVER_CTRL_t* inData       = (   _DRIVER_CTRL_t*) ssGetInputPortSignal( S, 0 );
  unsigned int*  outDataId        = (unsigned int*)    ssGetOutputPortSignal( S, 0 );
  float* outDataSteeringWheel  = (float*)   ssGetOutputPortSignal( S, 1 );
  float* outDataSteeringSpeed      = (float*)   ssGetOutputPortSignal( S, 2 );
  float* outDataTrhottlePedal      = (float*)   ssGetOutputPortSignal( S, 3 );
  float*outDataBrakePedal  = (float*)  ssGetOutputPortSignal( S, 4 );
  float*       outDataClutchPedal  = (float*) ssGetOutputPortSignal( S, 5 );
  float*        outDataAccelTgt     = (float*)  ssGetOutputPortSignal( S, 6 );
  float*       outDataSteeringTgt   = (float*) ssGetOutputPortSignal( S, 7 );
 double*        outDataCurvatureTgt  = (double*)  ssGetOutputPortSignal( S, 8 );
  float*       outDataSteeringTorque   = (float*) ssGetOutputPortSignal( S, 9 );
  float*        outDataEngineTorqueTgt  = (float*)  ssGetOutputPortSignal( S, 10 );
  float*        outDataSpeedTgt   = (float*)  ssGetOutputPortSignal( S, 11 );
```

Figure 56 – Data type

```
if ( !inData )
    return;

// let's map the data
*outDataId       = inData->playerId;
*outDataSteeringWheel = inData->steeringWheel;
*outDataSteeringSpeed     = inData->steeringSpeed;
*outDataTrhottlePedal = inData->throttlePedal;
*outDataBrakePedal      = inData->brakePedal;
*outDataClutchPedal = inData->clutchPedal;
*outDataAccelTgt    = inData->accelTgt;
*outDataSteeringTgt  = inData->steeringTgt;
*outDataCurvatureTgt       = inData->curvatureTgt;
*outDataSteeringTorque = inData->steeringTorque;
*outDataSpeedTgt     = inData->speedTgt;
```

Figure 57 – Map of the data

Again, the file.c is compiled into Matlab environment and it is possible to get an S-function for decoding the package in question.

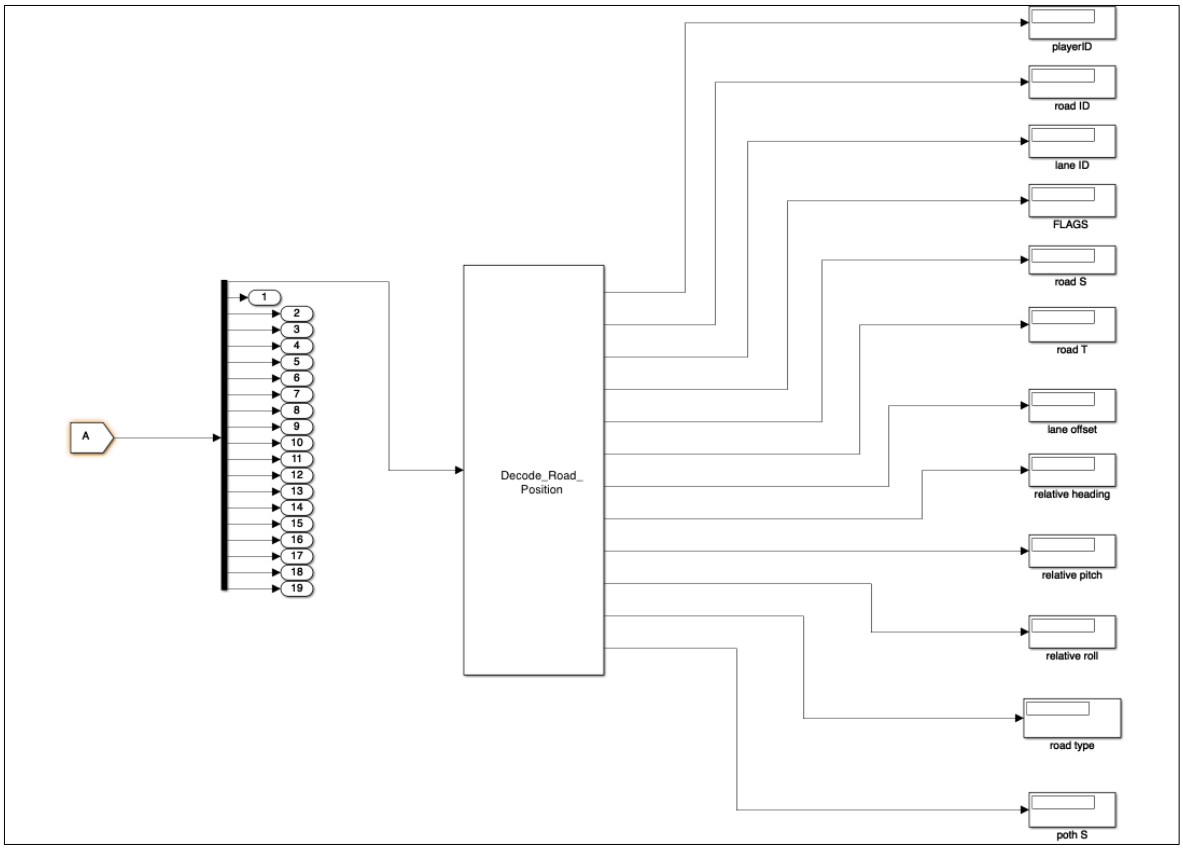


Figure 58 – Decode block example

During the simulation, if the sensor sees more than one element of the same type it is possible to create a decoding block to read the parameters for each of them; for this reason in the Figure 53 in the "Number objects" display is possible to read the number of the element that the camera is reading in a certain moment and in the Figure 58 there is the possibility to connect other 19 Decode_Road_position blocks.

The mathematical approach of Lane-Keeping has been analyzed to understand the parameters to be checked for the correct operation of the maneuver implemented on the two scenarios.

In the chapter 5, the two variables, on which the control was done, were the lateral deviation and the yaw error; among the data read from the camera inserted on the Ego Car inside the Software X, these two are located inside the structure ROAD_POSITION_t and are defined as LANE-OFFSET and RELATIVE HEADING.

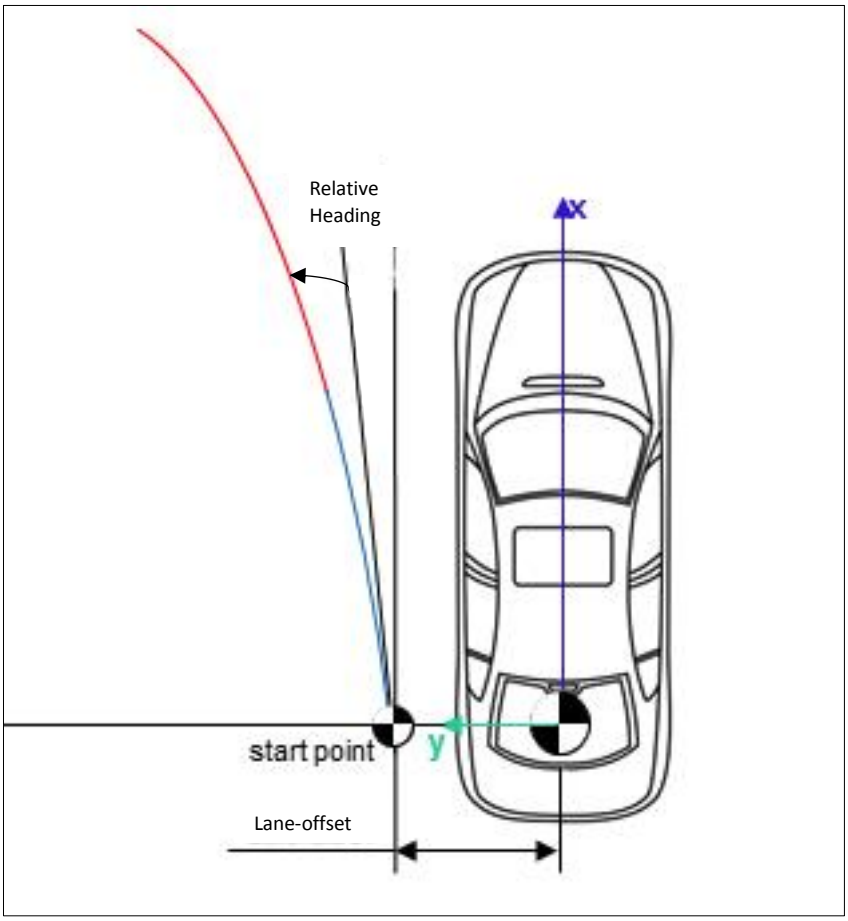


Figure 59 -parameters to be checked

During the simulation, the lane-offset and the relative heading are read by the camera instant by instant and are decoded through the Road Position block, which is a block inside the control model control made in Linux environment.

To connect in real-time these values with the two PID controllers described in the chapter 5 and to close the loop, both the lane-offset and the relative heading are sent in Windows environment through a calibrated UDP communication send block.

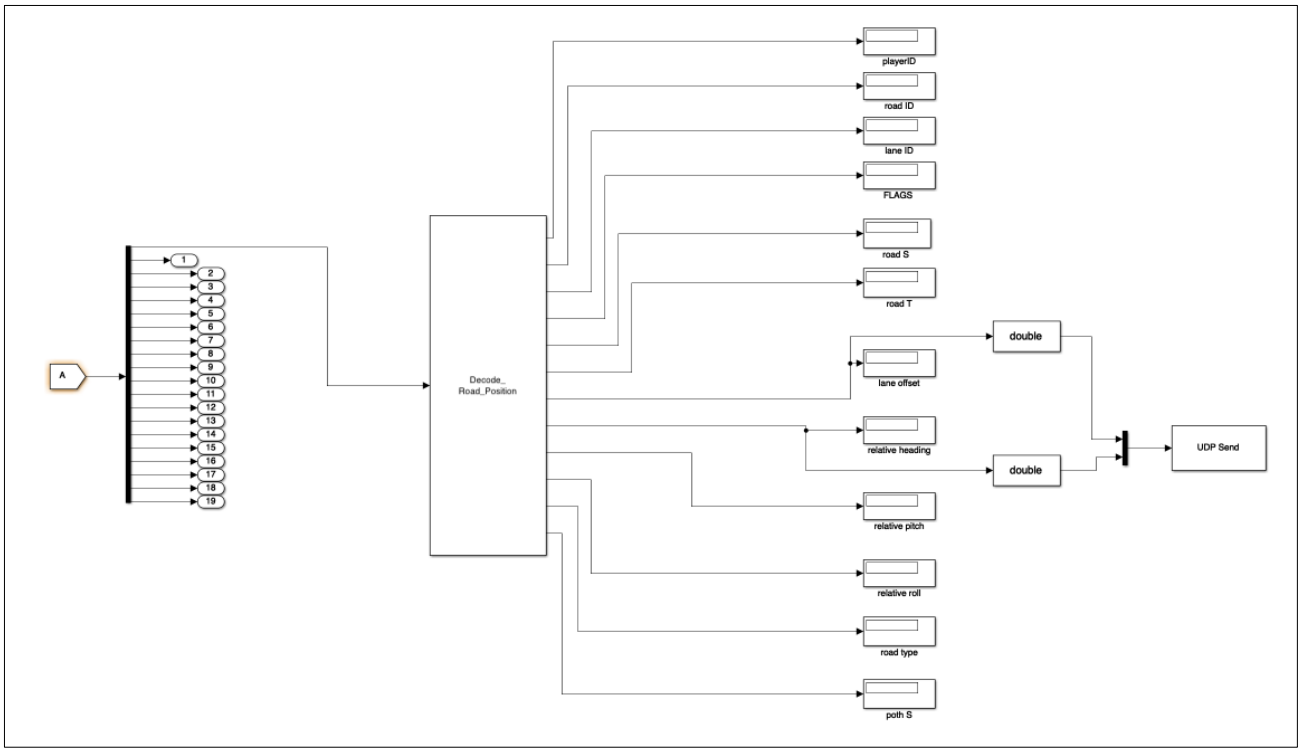


Figure 60 – Sending data to WINDOWS environment

Then, the desired data are received by a UDP receive block and are checked by the model shown in figure 61, calibrating the PID controller based on the speed in which the Ego Car moves too.

In the urban scenario, the car can go a maximum of 50 km/h, however in the path that includes a traffic light crossing and a roundabout it is preferred to maintain a speed of 25-30 km/h.

In the extra-urban scenario, the Ego car adjusts its speed based on the signs it meets, keeping in a range from 90 km/h to 60 km/h in the section reduced for work in progress.

For this reason, it is necessary to implement two equally structured models to control the parameters mentioned above, with a different calibration of the PID controllers based on the Ego car's speed.
The control on a faster vehicle should be much more restrictive than in the urban case, where the car reaches a maximum of 30 km/h.
The Software-in-the-Loop is thus realized and schematized in Matlab/Simulink environment.

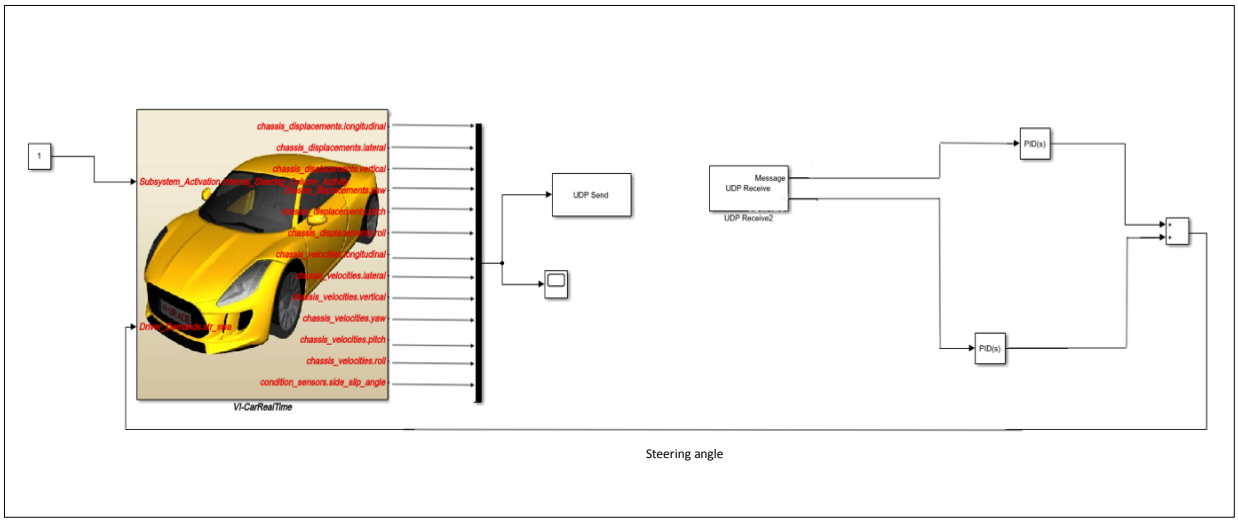


Figure 61 – Lane-Keeping control

Starting the simulation on the Software X, the camera instantly reads the parameters of lane-offset and relative heading on the selected scenario and sends them in the Simulink model of figure 61; here, these parameters are corrected by the PID controllers, which tends to reduce to zero these errors, generating a steering angle that will be the input of the vehicle model made in CarRealTime and will allow to correct its trajectory in the base scenario on which it moves.

At that point, the communication among the three simulation environments of CarRealTime, Software X and Matlab/Simulink are complete, and the loop is concluded. Through this Software-in-the-Loop will be possible to test different dangerous traffic situations that can be faced in the two types of scenarios chosen and created in this thesis activity.

7. CONCLUSIONS

The aim of the following thesis work was to implement an autonomous driving maneuver, realized in Matlab/Simulink environment, on two different types of scenario, created through the Software X, which are the subject of interest of three of the most important projects on ADAS at European level.
The two kinds of scenarios were created through a state-of-the-art simulation software, used specifically for the generation of 3D content, the simulation of complex traffic scenarios or that of simplified sensors or based on interaction with the real environment.
First, both an urban and an extra-urban scenario have been designed that are true to the reality.
Urban scenario was characterized by the connections among a traffic lights intersection, a straight road and an urban roundabout with tracks sized following the current decree of the Ministry of Infrastructure.
The same procedure was adopted for the design of the extra-urban scenario, created by a three-lane curved road, followed by a roadway restriction for work in progress and a two-lane curved road.
The surrounding environment has been created with the appropriate characteristics in the case of urban and extra-urban scenario, trying to remain as faithful as possible to reality.
Then, it was important to focus on the universal study of the maneuver to be carried out, which in our case is that of Lane-Keeping.
Both the mathematical approach and the parameters to be analyzed for the control of the custom vehicle, made in CarRealTime, must be studied to achieve the Lane-Keeping of an objective trajectory, using the calibration of the PID controllers.
Finally, all the three simulation environments (Software X, CarRealTime and Matlab/Simulink) were connected to each other, through an UDP communication, and it was possible to create a serious of videos in which the vehicle generated in CarRealTime is perfectly able to maintain any lane set, both in the urban and extra-urban scenario and to undertake both the traffic light intersection and the urban roundabout, even varying the speed levels, thanks to the control of logic implemented in Matlab/Simulink environment described in chapter 5.

7.1 Further development

Previously it has been said that simulation software for the implementation of autonomous driving, such as Software X, are used to test any dangerous situation that in reality is difficult to reproduce.
For this reason, control logics can be developed by populating the understanding logics with new cases.

An improvement of such a project could be to add any obstacles in the way the car travels, such as pedestrians, animals or other vehicles and also analyze an emergency braking or an emergency overtaking. It is also possible to create new interesting kinds of rural, motorway, residential scenarios following the characteristics described in the European projects with different kinds of the surrounding scenario.

It would also be interesting to analyze these virtual tests on a physical static or dynamic simulator and understand if the maneuver can be performed by a driver on the road. The question of whether or not the results obtained in the simulator are applicable to real world driving was the object of study years ago. It was proven that there is a correlation between the number and types of driving error committed by a driver on simulator and on road [16], as well as between the scores on a driving simulator assessment and crash involvement at fault within five years from the simulative session [17]. Then it is possible to say that the behavior of a driver on simulator can be at least compared to that of a driver on road. It was yet discussed that results related to vehicle dynamics evaluation can be well linked to the real-world analysis, since the vehicle dynamic model is built on the basis of a real car on the market.

Appendix 1

The linear regulators most used in the industrial field are certainly the PID, or Proportional, Integral and Derivative; they are systems in negative feedback widely used in control systems. The reasons for their success are varied:

- their use allows to control satisfactorily a wide range of processes.
- for their simplicity, PID regulators can be made with the most varied technologies: mechanical, pneumatic, hydraulic, analog and digital electronics.

Traditionally, the structure of PIDs is introduced on the basis of empirical considerations, according to which the control variable u should be generated as the sum of three contributions:

- The first, of intuitive meaning, is proportional to the error and, present between the reference signal w and the output variable y of the system under control.
- the second is proportional to the integral of e (that is to say, its mean value) and is required to require that the error cancel out asymptotically against additional reference signals or constant disturbances.
- The third is proportional to the derivative of and and aims to try to "anticipate" the trend of the error in the future moments: if for example the derivative of the error is positive, as well as the gain of the system, it is appropriate to increase u to cause an increase of y and therefore a decrease of e.

The law of control, that is the link between e and u, is, therefore:

$$u(t) = K_P \cdot e(t) + K_I \int_{t0}^{t} e(\tau)d\tau + K_D \frac{de(t)}{dt}$$

where K_P, K_I and K_D are positive or zero constants (assuming the process gain is positive). The coefficient K_P is called the coefficient of proportional action, while K_I and K_D are respectively the coefficient of integral action and the coefficient of derivative action.

Applying the Laplace transform to the previous equation with t0 = 0, it is immediately deduced that they are described by the transfer function:

$$R_{PID}(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

where the term proportional ($R_{\_P}=K_P S$), integral ($R_{\_I}(s)= K_I/s$) and derivative ($R_{\_D}(s) = K_D s$) are present in the previous block diagram.

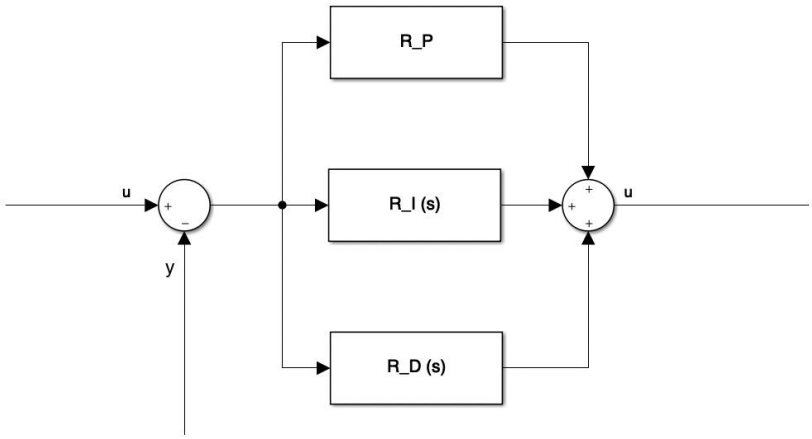


Figure 62 - Block diagram of an ideal PID

Bibliography

[1] Vellinga, Nynke E. Automated driving and its challenges to international traffic law: which way to go?

[2] Lexus advanced safety research vehicle (2013). URL: Tecnologia - Lexus Advanced Safety Research Vehicle - MotorBox.

[3] Self-driving car. URL: Self-driving car - Wikipedia.

[4] Guida autonoma: le auto più avanzate sul mercato (2019). URL: https://www.sicurauto.it/news/guida-autonoma-le-auto-piu-avanzate-sul-mercato/;

[5] Auto a guida autonoma in UE: dalla fantascienza alla realtà | Attualità | Parlamento europeo (europa.eu).

[6] Grant Agreement-826653-NewControl.pdf.

[7] Automation as Driver Companion: Findings of AutoMate Project. Andrea Castellano, Massimo Fossanetti, Elisa Landini, Fabio Tango, Roberto Montanari.

[8] http://www.safespot-eu.org/objectives.html.

[9] Emmanuel Genty (2017). "Physical and Virtual simulation, a winning combination for autonomous driving ". Group.renault.com.

[10] VTDUserManualRevO.pdf.

[11] https://www.mit.gov.it/.

[12] https://www.segnaleticatemporanea.it/Norme.

[13] Genta, G. (1996) Meccanica dell'autoveicolo. Torino: Levrotto&Bella.

[14] Genta G., Morello L. - The Automotive Chassis- Vol.2- System design.

[15] Toro O., Becsi T., Aradi S. (2016)- Design of Lane Keeping Algorithm for Autonomous vehicles.

[16] Shechtman, Orit, Classen, Sherrilene, Awadzi, Kezia, Mann, William (2009) Comparison of Driving Errors Between On-the-Road and Sim- ulated Driving Assessment: A Validation Study.

[17] Hoffman L., & McDowd, J. M. (2010). Simulator driving performance predicts accident reports five years later. Psychology and Aging.