Bachelor Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Simulation of Automated Cayenne Porsche in the CARLA Simulator

Miroslav Matějček

Supervisor: Ing. Jiří Vlasák
Supervisor–specialist: Ing. Michal Sojka, Ph.D.
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2023

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to my supervisor Ing. Jiří Vlasák, for his exceptional patience, particularly during the revisions of this text. I am immensely grateful for his guidance and valuable feedback throughout the entire process.

I would also like to extend my gratitude to Ing. Michal Sojka, Ph.D., for his valuable professional advice and insights. His contributions greatly enriched the research and development of this work.

Special thanks to my family, girlfriend, and friends for their endless support throughout not only my study but my whole life.

Last but not least, I would like to mention the open-source community for their continuous efforts in sharing solutions to a wide range of issues, extending beyond the scope of the CARLA simulator.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses

In Prague, 26. May 2023

Abstract

The automotive industry deals with the advances in the automation of driving. Automation promises improvement in safety, provided that all systems are functioning properly. This thesis explores the usage of simulation for the purpose of Automated Driving System testing. Namely, the thesis presents the simulation of the Porsche Cayenne (equipped with sensors and actuators for automated driving) in the CARLA simulator. Based on the comparison of the real-world and simulation experiments, the conclusion is that the presented simulation environment is suitable for Automated Driving System testing.

Keywords: Automated Driving, CARLA Simulator, ROS, Porsche JUPITER, Livox Horizon

Supervisor: Ing. Jiří Vlasák
CIIRC A-517,
Jugoslávských partyzánů 1580/3,
Praha 6

Abstrakt

Automobilový průmysl se potýká s pokroky v automatizaci řízení. Automatizace slibuje zvýšení bezpečnosti za předpokladu, že všechny systémy fungují bezproblémově. Tato práce zkoumá využití simulací za účelem testování systémů automatizovaného řízení. Konkrétně představuje simulaci Porsche Cayenne (vybaveného senzory a aktuátory pro automarizované řízení) v simulátoru CARLA. Na základě porovnání výsledků experimentů provedených s reálným vozidlem a v simulaci usuzuje, že představené simulační prostředí je adekvátní k testování systémů automatizovaného řízení.

Klíčová slova: Automatizované řízení, Simulátor CARLA, ROS, Porsche JUPITER, Lidar Livox

Překlad názvu: Simulace automatizovaného Porsche Cayenne v simulátoru CARLA

Contents

Figures

Listings

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Matějček Miroslav**	Personal ID number:	**498854**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Simulation of Automated Cayenne Porsche in the CARLA Simulator

Bachelor’s thesis title in Czech:

Simulace automatizovaného Porsche Cayenne v simulátoru CARLA

Guidelines:

Preparing the simulation environment with the CARLA simulator for experiments with the Porsche Cayenne equipped with sensors and actuators for automated driving.
1. Familiarize yourself with the CARLA simulator and project JUPITER -- Porsche Cayenne prepared for experiments with automated driving.
2. Create a virtual model of the vehicle JUPITER in the CARLA simulator, including sensors (especially Livox LiDAR), and prepare an interface to the vehicle that resembles the real vehicle as much as possible. Use the ROS 2 framework.
3. Create simple applications in ROS 2 to control the simulated vehicle, e.g. emergency braking, adaptive cruise control, lane keeping.
4. In collaboration with the supervisor, run and test the developed applications in the vehicle JUPITER. Compare the behavior of the control algorithms in the simulation and in the real vehicle. Based on the experimental results, improve the simulation to resemble the real vehicle as much as possible.
5. Document the results accurately.

Bibliography / sources:

[1] “J3016_202104: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International.” Accessed August 2, 2022. https://www.sae.org/standards/content/j3016_202104/.
[2] Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. “CARLA: An Open Urban Driving Simulator.” In Proceedings of the 1st Annual Conference on Robot Learning, 1–16, 2017.
[3] Riedmaier, Stefan, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. “Survey on Scenario-Based Safety Assessment of Automated Vehicles.” IEEE Access 8 (2020): 87456–77. https://doi.org/10.1109/ACCESS.2020.2993730.
[4] Weissensteiner, Patrick, Georg Stettinger, Johannes Rumetshofer, and Daniel Watzenig. “Virtual Validation of an Automated Lane-Keeping System with an Extended Operational Design Domain.” Electronics 11, no. 1 (January 2022): 72. https://doi.org/10.3390/electronics11010072.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jiří Vlasák Department of Control Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **23.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jiří Vlasák	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Chapter 1

Introduction

Automated driving has become an increasingly significant topic in the automotive industry. Nowadays, it is common for new cars to have some driving automation features like parking assistants, Adaptive Cruise Control, emergency braking or Automated Lane-Keeping Systems, and companies are continuously trying to develop higher levels of Automated Driving Systems [1]. To ensure safety, these systems must undergo thorough testing before being deployed in traffic [2]. Due to the challenges associated with real-world testing, there is a growing trend to conduct as much testing as possible in simulations [3], [4].

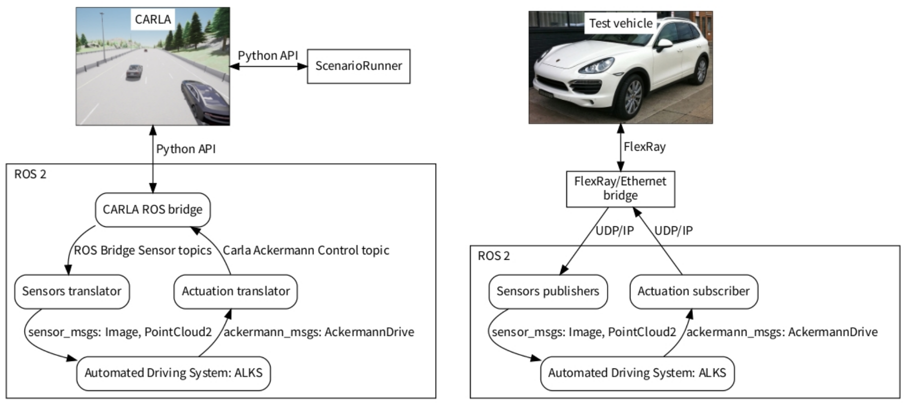


Figure 1.1: Comparison of architecture used in a simulator (left) and in the real vehicle (right). The figure is taken from [4].

The advantage of a simulator is the possibility of testing under various traffic conditions and weather, regardless of the time of day or season. Testing in a simulator also allows testing of software that is still under development, and its safe operation in a real environment cannot be guaranteed. Detecting software errors during the simulation can prevent high property damage or even injury that could happen in real-world tests.

This thesis describes the utilization of the CARLA simulator [5] to create the simulation of the Porsche Cayenne equipped with sensors and actuators for autonomous driving, namely the camera, lidar and GNSS sensor, and compares the simulation with the real vehicle.

This requires incorporating the model of the Porsche Cayenne into the CARLA simulator, integrating the Livox Horizon lidar into the CARLA simulator, creating a simulation with added Porsche Cayenne and sensors appropriately attached to it, and ensuring communication between the simulator and automated driving system. Subsequently, make experiments in the simulator and compare them with corresponding experiments done in the real car.

1.1 Background

This section contains a description of used terms and related projects.

Unreal Engine

A comprehensive gaming engine for 3D graphics developed by Epic Games, Inc [6]. It includes an integrated editor, the Unreal Editor, that allows the addition of custom models into its projects.

CARLA

Car Learning to Act [5], [7] is an open-source urban driving simulator created under Unreal Engine 4 that is being developed to test, train and validate autonomous driving systems.

ROS

Robot Operating System in an open-source set of software tools and packages for developing robots that enable easy communication between individual components. One of these tools is Robot Visualizer (RViz) that enables the visualisation of data from ROS, for example, data generated by lidars.

CARLA-ROS Bridge

A collection of ROS packages that allows CARLA to be controlled using ROS [8]. The packages can for example spawn objects, control simulated vehicles or launch prepared scenarios.

Project JUPITER

Joint User Personalized Integrated Testing and Engineering Resource is a project of Porsche Engineering Services that established an automotive platform based on the ROS system for researching automated driving. The part of the project is three vehicles Porsche Cayenne equipped with sensors. One of them is currently being used in research at the Czech Institute of Informatics, Robotics, and Cybernetics in Prague.

Livox

A company that is creating industrial lidars used in automated mobility. Livox Horizon lidar equipped on the Porsche Cayenne is made by their company.

Chapter 2

Adding Porsche Cayenne into CARLA

This chapter describes how to add a model of the Porsche Cayenne along with Livox lidar to the CARLA simulator [9] version 0.9.13 on Ubuntu 20.04.

Adding custom assets like vehicle or sensor models requires the CARLA simulator to be compiled from the source code. Because CARLA is based on Unreal Engine 4 [6], custom models must be specified in *.uasset* format, which can be created in the Unreal Engine editor.

The following sections summarise the process of adding assets to the CARLA simulator. Particularly, Section 2.1 describes the compilation of the CARLA simulator from the source code, adding a model of the Porsche Cayenne is summarised in Section 2.2 and Livox lidar in Section 2.3, and Section 2.4 describes how the CARLA standalone package with custom assets is created.

2.1 Building CARLA from Source

The CARLA simulator is based on Unreal Engine 4 (UE4). So, to compile CARLA, UE4 must be installed first. UE4 is very demanding on storage and performance. At least 100 GB of free disk space and 6 GB of GPU are required. The engine often uses over 20 GB of RAM, so it is recommended to expand the swap memory sufficiently.

To obtain the source code of the Unreal Engine, it is necessary to have a GitHub account and an Epic Games account and to have these accounts linked together.

When the prerequisites are satisfied, the rest of the installation of the UE4 is straightforward.

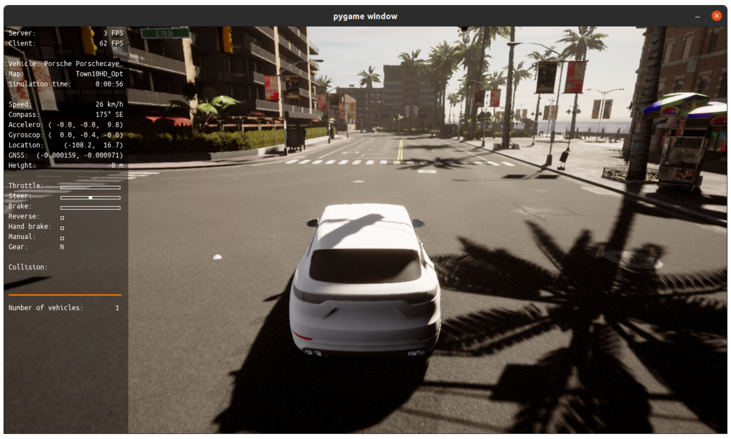
The compilation of CARLA requires another 30 GB of memory, and the other prerequisites are the same as for the installation of UE4. The installation should then happen without complications. However, it may happen that the compilation of *PythonAPI* fails due to downloading *libxerxes*. In such a case, it is necessary to modify line 431 of the **`Util/BuildTools/Setup.sh`** file to:

```
XERCESC_REPO=https://archive.apache.org/dist/xerces/c/3/
    sources/xerces-c-${XERCESC_VERSION}.tar.gz
```

Finally, when the engine starts, the individual assets are still being compiled. Therefore, it is recommended to leave the program running for a few hours until the assets are compiled.

2.2 Adding Porsche Cayenne into CARLA

Adding a custom vehicle model requires the CARLA simulator to be compiled from the source code. There are strict requirements for the 3D model to be added to CARLA. Although there are two tutorials in official documentation on how to add a vehicle, it is easier to reuse already created models from the simulator. Such an approach to creating the model is demonstrated in the tutorial from Haowei Zhang[1]. The Porsche Cayenne added to the CARLA is shown in Figure 2.1.



```
Server:                3 FPS
Client:               62 FPS

Vehicle: Porsche Porschecaye…
Map:             Town10HD_Opt
Simulation time:      0:00:56

Speed:               26 km/h
Compass:             175° SE
Accelero: ( -0.0, -0.0,  9.8)
Gyroscop: (  0.0, -0.4, -0.0)
Location:      (-108.2,  16.7)
GNSS:  (-0.000159, -0.000971)
Height:                  -0 m

Throttle:
Steer:
Brake:
Reverse:
Hand brake:
Manual:
Gear:        N

Collision:

Number of vehicles:         1
```

Figure 2.1: Added model of Porsche Cayenne in CARLA.

It takes some time to grasp creating assets for Unreal Engine. It can take several attempts to add a vehicle properly. The rest of the section provides suggestions that can help with this.

- Tesla 3D model can be exported from *Content Browser* in Unreal Engine. Path to it from CARLA directory is **`Unreal/CarlaUE4/Content/Carla/Static/Vehicles/4Wheeled/Tesla/SM_TeslaM3_v2.uasset`** as demonstrated in Figure 2.2.

[1]Video tutorial about adding a vehicle into CARLA - `https://www.youtube.com/watch?v=0F3ugwkISGk`

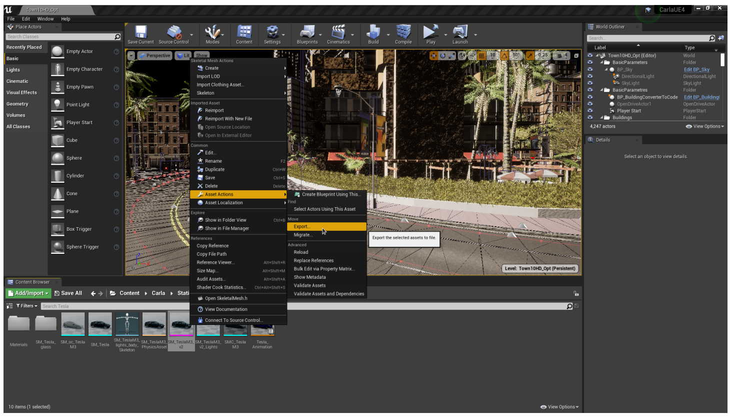

Asset Actions
Export...
Export the selected assets to file
Level: Town10HD_Opt (Persistent)

Figure 2.2: Exporting 3D model from CARLA.

- If your model is composed of multiple parts, the shortcut in Blender for merging them is alt+p.
- After importing the model into UE4, click on *save all* to save all assests like materials, that will not be modified.
- When modifying the physics asset, change the *Physics Type* to *Kinematic* only for the wheels, the body must remain *Default*.
- The *Collision Mesh* for the wheels' blueprint should be *Wheel shape* despite older sources saying *Cylinder*.

The following are interesting sources about content creation for CARLA.

- Videotutorial of adding vehicle into CARLA [10]
  `https://www.youtube.com/watch?v=0F3ugwkISGk`
- GitHub issue where people discuss they problems about adding a vehicle into CARLA [11]
  `https://github.com/carla-simulator/carla/issues/2738`
- Official documentation about adding an vehicle into CARLA
  `https://carla.readthedocs.io/en/latest/tuto_A_add_vehicle/`
- Official documentation about creating an vehicle model
  `https://carla.readthedocs.io/en/latest/tuto_content_authoring_vehicles/`
- Videotutorial about adding a vehicle into Unreal Engine [12]
  `https://www.youtube.com/watch?v=0F3ugwkISGk`

2.3 Adding Livox Lidar into CARLA

The Livox lidar differs from the default lidar in CARLA. The main difference is in the shape of the trajectory of the rays. The Livox lidar's trajectory is complex; the default lidar's trajectory from CARLA is circular.

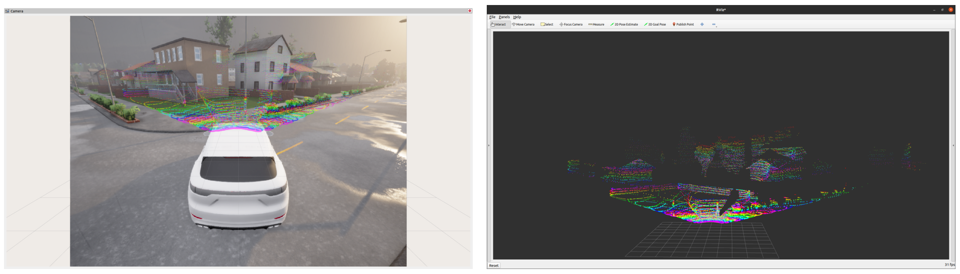


Figure 2.3: Livox horizon lidar in CARLA.

Adding a custom sensor into the CARLA simulator requires changes in its source code. This section describes the changes needed to the CARLA source code to add Livox Horizon lidar.

Adding a sensor to CARLA involves reverse engineering how the sensor works, writing the sensor actor, the serializer and the corresponding data class. Changes in the CARLA-ROS bridge are needed to control the sensor through ROS.

The source code of the Livox lidar for the CARLA simulator has already been published as an open-source project [13]. The rest of this section describes how to merge this project into the CARLA source code.

The Livox lidar source code is meant to be only moved into proper directories and be recompiled along with the rest of the CARLA source code. However, because the project is done for an older version of the CARLA simulator, the files cannot be just replaced. It is necessary to merge parts of the Livox lidar code into the CARLA source files. We have published an updated version of the CARLA simulator that includes the code of Livox lidar[2]. We also published an updated CARLA-ROS bridge[3].

Figure 2.3 shows a visualisation of the added Livox lidar after recompilation of the simulator.

2.4 Exporting a Package

CARLA standalone package needs to be created to run the CARLA with custom assets without Unreal Engine.

Creating the CARLA standalone package can be done by the CARLA build system, as can be seen in Listing 2.1. The process takes several hours and produces approximately 14 GB of CARLA assets, including executables, maps, vehicles, sensors and Python API.

```
make package ARGS="--packages=Carla"
```

Listing 2.1: Exporting own CARLA package.

[2]Carla with Livox lidar - https://github.com/matejm42/carla/tree/livox

[3]CARLA-ROS bridge with Livox lidar https://github.com/matejm42/ros-bridge/tree/0.9.13-livox

To run the CARLA simulator from an exported package with Nvidia drivers, the environmental variable has to be exported before launching CARLA:

```
export VK_ICD_FILENAMES="/usr/share/vulkan/icd.d/
nvidia_icd.json"  && ./CarlaUE4.sh
```

Listing 2.2: Launching CARLA.

Chapter 3

Simulating Porsche Cayenne in CARLA

This chapter describes simulations of the Porsche Cayenne (ADS) in the CARLA simulator. Particularly with ADS for parking developed with the cooperation between the Czech Technical University in Prague and Porsche Engineering Services.

To perform the simulation, it is necessary to: 1) Spawn Porsche Cayenne with sensors appropriately attached to it. 2) Create ROS nodes that translate status information from the CARLA simulator to ROS messages for the ADS. 3) Create ROS nodes that control the simulated vehicle according to ROS messages from ADS. 4) Launch the ROS node with the ADS function.

(1) - (3) relates to CARLA-ROS bridge [14], as covered in Section 3.1. Sections 3.2 and 3.3 deal with (4). In Section 3.2, the simulation environment is verified on a simple example of a slalom between cones. Finally, the simulation environment is used with an automated parking system for the JUPITER vehicle in Section 3.3.

3.1 Simulation Environment

This section describes, how to set up a simulator with Porsche Cayenne with needed sensors attached and how to translate messages between CARLA and ADS. All the communication between the developed ADS and the JUPITER vehicle is done through the ROS framework. CARLA-ROS bridge is used to communicate with the CARLA simulator over ROS.

Because everything is controlled by ROS, a ROS launchfile can be created to launch all parts of the system: packages from the CARLA-ROS bridge to communicate with the CARLA simulator and nodes responsible for translating the messages between the CARLA-ROS bridge and ADS. It also allows passing parameters to all nodes and defining their default values. The values can be also set by exporting the corresponding environment variable, which is also suitable when using a Docker environment. Parameters of the CARLA-ROS bridge allow to choose a map for the simulation or an IP address of the simulator since it can be run on a remote server. Another parameters belongs to the CARLA Spawn Objects package that allows spawning vehicles into the simulation.

CARLA Spawn Object takes *objects definition file* in JavaScript Object

Notation (JSON) format as a parameter, where objects to spawn are defined. It contains the type of the vehicle to spawn, and a set of sensors with their parameters and relative positions to the vehicle. Besides a vehicle and sensors from CARLA, also pseudosensors can be defined there. They are part of the ROS bridge and turn on publishing extra topics with information about the simulation or vehicle. They are not necessary for the simulation, because the state of the vehicle is described in summary message *status*, but they enable the usage of some other ROS packages like CARLA Manual Control or some visualisation features in RViz.

For translating the messages between the CARLA-ROS bridge and ADS nodes called *translators* are used. They listen for the messages from CARLA and send the same information in the format expected by ADS. An example of this translator is in listing 3.1.

The camera translator listens for messages from the camera sensor on the simulated vehicle. The ADS expect the same standard image format as comes from the CARLA-ROS bridge, so only the name of the topic and header is changed and the same message is published.

The front and rear lidar translators work the same as the translator for the camera because data from lidars are also in a standard format.

The tricky part is handling coordinate systems and headings because both the CARLA-ROS bridge and JUPITER car use multiple different systems. All used data came from the CARLA-ROS bridge, but it is good to point out that the CARLA simulator has the left-handed coordinate system, to be compatible with the unreal engine. This system CARLA-ROS bridge changes to right-handed, which is standard for ROS.

Translator for GNSS data listens for GNSS sensor in CARLA simulator. Its message is in a standard format and GNSS data are mapped 1:1 between the simulator and ADS.

$$GNSS_{\text{ADS}} = GNSS_{\text{CARLA}}$$

The translator for heading listens for a summary message from the simulator *vehicle status*. Both the simulator and ADS use quaternion format to store heading, but the angle has to be changed. Heading expected by ADS has a different positive direction, than heading that is part of simulated vehicle status. Furthermore, the sensor expected by ADS gives zero value, when the car is heading north, the simulated when heading east.

$$H_{\text{ADS}} = -1 \cdot H_{\text{CARLA}} + \pi/2$$

The translators handling messages with information about the state of the vehicle listen for a summary message from the simulator *vehicle status*. This message is split into topics in the proprietary format of messages from a bus in the Porsche Cayenne. They contain information about speed of the vehicle, speed of rear wheels, stand still flag and the relative heading of the vehicle (*gierwinkle*). The gierwinkle has the same direction as the heading form the CARLA simulator, but it is shifted by the starting position and stored in radians.

$$G = H_{\text{CARLA}} - H_{\text{START}}$$

The system used for describing the parking path and visualisation has the origin in starting position and the x-axis in the direction H_{START}. The distance from the origin is computed in meters from GNSS and transformed

$$\begin{bmatrix} \sin(H_{\text{START}}) & \cos(H_{\text{START}}) \\ -\sin(H_{\text{START}}) & \cos(H_{\text{START}}) \end{bmatrix} \cdot \begin{bmatrix} moved\_east \\ moved\_north \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

The differences in coordinate systems are shown in Figures 3.1 and 3.2. The easiest way to visualise them in CARLA is to use the CARLA Manual Control package, which displays information about the vehicle organized in the status bar as shown in Figure 3.3. Part of this information is also a cardinal direction, but that is wrong. It can be seen from Figure 3.1 that the displayed cardinal direction does not correspond to GNSS data. Moreover, the east can not be to the left from the north. We pushed an issue informing about the problem and merge request suggesting fixes [15].

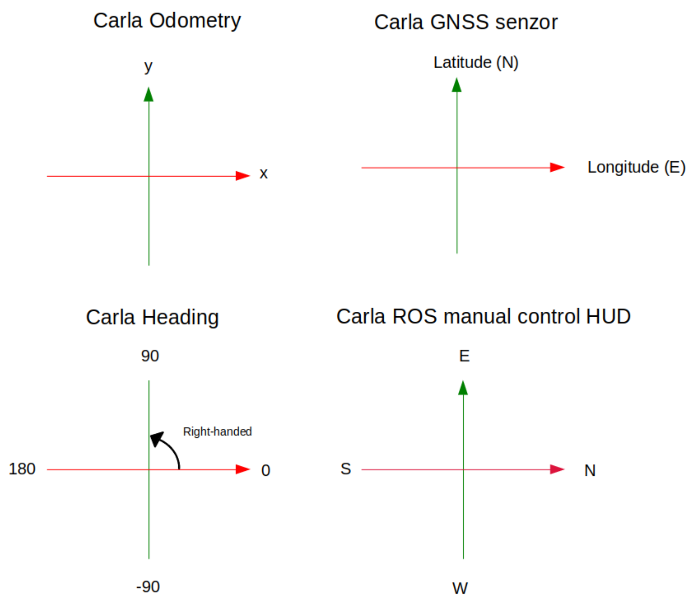


Figure 3.1: Coordinate systems of CARLA under ROS.

There is only one translator, that translates communication from ADS to the simulator. It listens for a proprietary control message, which contains desired acceleration, steering angle and direction (Front, Back, Stay). The acceleration is integrated to get speed and sent with steering angle as *ackermann control message* to CARLA Ackermann Control, which is a node that through PID controller computes throttle and steering to control the simulated vehicle in CARLA.

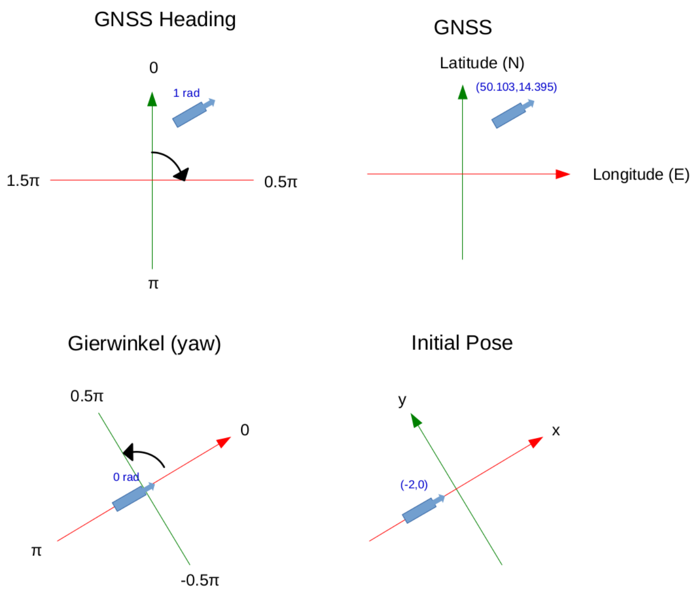
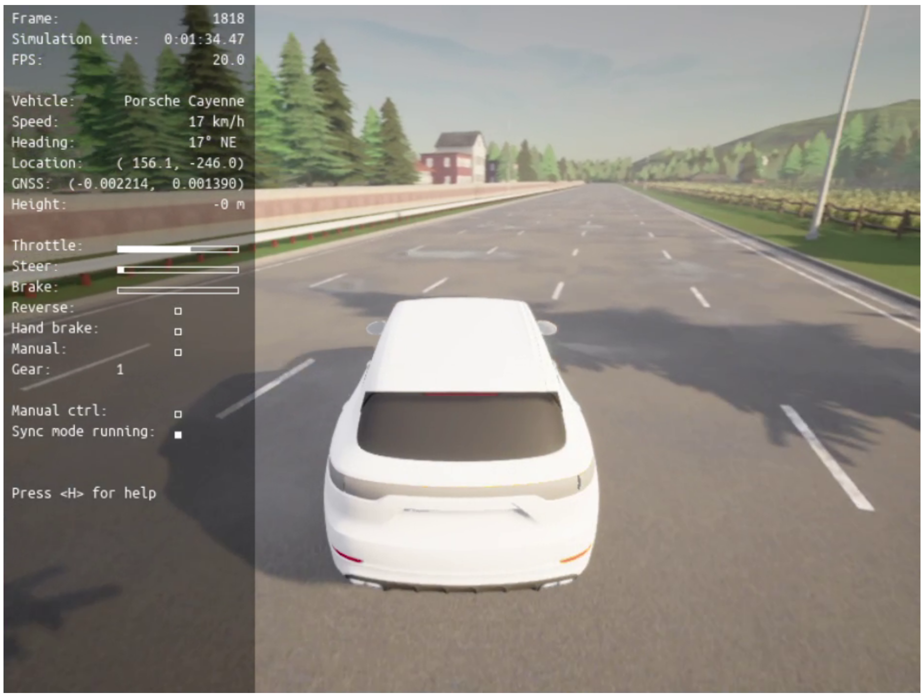


Figure 3.2: Coordinate systems of ADS.



```
Frame:                    1818
Simulation time:    0:01:34.47
FPS:                      20.0

Vehicle:       Porsche Cayenne
Speed:                 17 km/h
Heading:               17° NE
Location:     ( 156.1, -246.0)
GNSS:  (-0.002214,  0.001390)
Height:                   -0 m

Throttle:
Steer:
Brake:
Reverse:             □
Hand brake:          □
Manual:              □
Gear:         1

Manual ctrl:         □
Sync mode running:   ■


Press <H> for help
```

Figure 3.3: HUD in Manual Control.

```
import rclpy
from rclpy.node import Node
import sensor_msgs.msg as msgs
QS = 10  # Quality of service profile

class ExampleTranslator(Node):
    def __init__(self):
        super().__init__('example_translator')
        self.subscription = self.create_subscription(
            msgs.Image, 'carla/ego_vehicle/camera/image',
    self.translate,  QS)
        self.publisher = self.create_publisher(
            msgs.Image, 'jupiters_camera/image', QS)

    def translate(self, img):
        new_img = copy.deepcopy(img)
        new_img.header.frame_id = "cameras_id"
        self.publisher.publish(new_img)


if __name__ == '__main__':
    rclpy.init()
    example_translator = ExampleTranslator()
    try:
        rclpy.spin(example_translator)
    except KeyboardInterrupt:
        example_translator.get_logger().info(f'Ending the
    node.')
    finally:
        example_translator.destroy_node()
        rclpy.shutdown()
```

Listing 3.1: Example of simple translator.

3.2 Slalom

The functionality of written translator nodes and the simulation environment is tested by the simple ADS that drives Porsche Cayenne between the cones in the CARLA simulator. On a five-line road in CARLA was spawned a vehicle with sensors and the translators were launched using the launch file as described in Section 3.1. Then 4 cones were spawned via *Python API* as ROS Bridge can spawn only vehicles and sensors. Simple ADS, launched as another ROS node, then navigated the vehicle between the cones. The ADS communicates only with translators representing Porsche Cayenne.

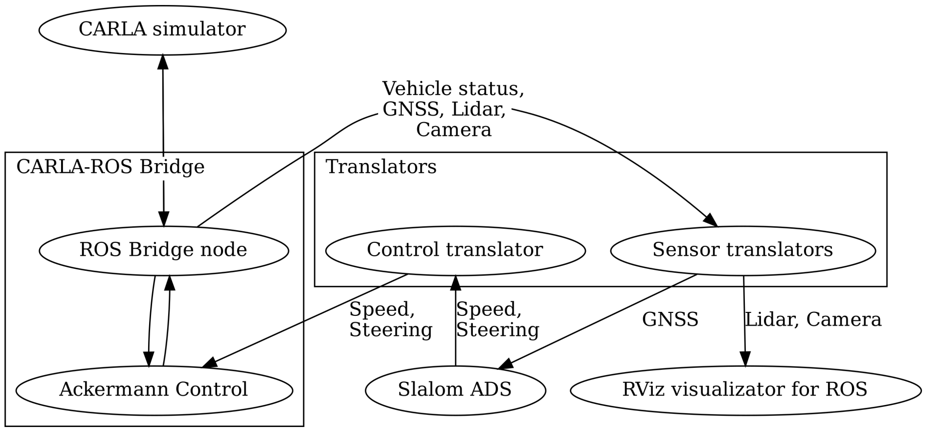
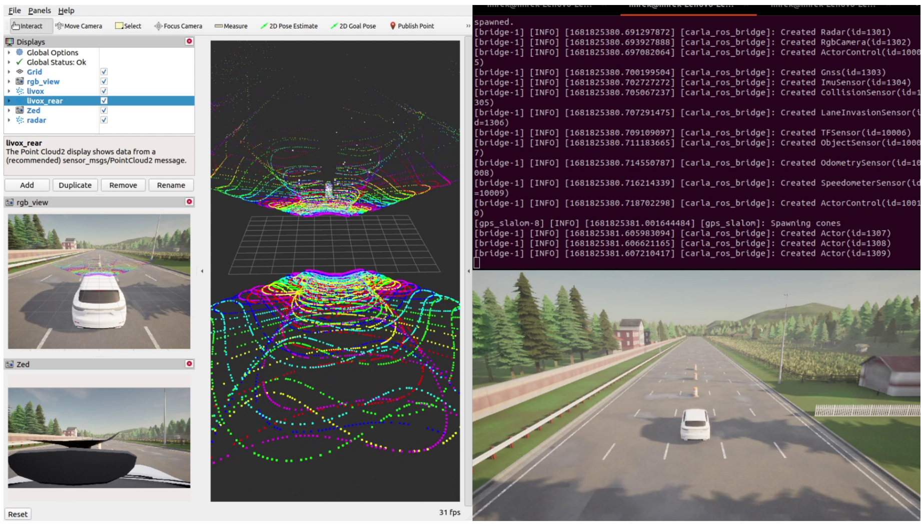


Figure 3.4: Communication between nodes in simple slalom.

The data flow of this test can be seen in Figure 3.4. Data from CARLA are transferred into ROS by the CARLA-ROS bridge. Sensor translators listen for vehicle status and data from sensors and translate them into the format expected by the ADS. Lidar and camera can be visualised in RViz though they are not used by the Slalom ADS. The visualisation is shown in Figure 3.5.

The ADS uses GNSS to determine speed and steering of the vehicle to navigate it between the cones. Control translator sends desired speed and steering angle to CARLA Ackermann Control package. Ackermann Control package uses a PID regulator to compute throttle, brake and steering that are, through ROS Bridge, used back in the CARLA simulator to control the simulated Porsche Cayenne.



```
spawned.
[bridge-1] [INFO] [1681825380.691297872] [carla_ros_bridge]: Created Radar(id=1301)
[bridge-1] [INFO] [1681825380.693927888] [carla_ros_bridge]: Created RgbCamera(id=1302)
[bridge-1] [INFO] [1681825380.697082064] [carla_ros_bridge]: Created ActorControl(id=1000
5)
[bridge-1] [INFO] [1681825380.700199504] [carla_ros_bridge]: Created Gnss(id=1303)
[bridge-1] [INFO] [1681825380.702727272] [carla_ros_bridge]: Created ImuSensor(id=1304)
[bridge-1] [INFO] [1681825380.705067237] [carla_ros_bridge]: Created CollisionSensor(id=1
305)
[bridge-1] [INFO] [1681825380.707291475] [carla_ros_bridge]: Created LaneInvasionSensor(i
d=1306)
[bridge-1] [INFO] [1681825380.709109097] [carla_ros_bridge]: Created TFSensor(id=10006)
[bridge-1] [INFO] [1681825380.711183665] [carla_ros_bridge]: Created ObjectSensor(id=1000
7)
[bridge-1] [INFO] [1681825380.714550787] [carla_ros_bridge]: Created OdometrySensor(id=10
008)
[bridge-1] [INFO] [1681825380.716214339] [carla_ros_bridge]: Created SpeedometerSensor(id
=10009)
[bridge-1] [INFO] [1681825380.718702298] [carla_ros_bridge]: Created ActorControl(id=1001
0)
[gps_slalom-8] [INFO] [1681825381.001644484] [gps_slalom]: Spawning cones
[bridge-1] [INFO] [1681825381.605983094] [carla_ros_bridge]: Created Actor(id=1307)
[bridge-1] [INFO] [1681825381.606621165] [carla_ros_bridge]: Created Actor(id=1308)
[bridge-1] [INFO] [1681825381.607210417] [carla_ros_bridge]: Created Actor(id=1309)
```

Figure 3.5: Slalom simulation.

3.3 Parking

This section describes the testing of ADS written for the JUPITER vehicle in the simulator. We were provided with an automated parking system for the JUPITER vehicle that has already been tested on the real vehicle on multiple scenarios. We also got files with parking paths and logs from performed experiments. These logs are compared with our simulation in Chapter 4.

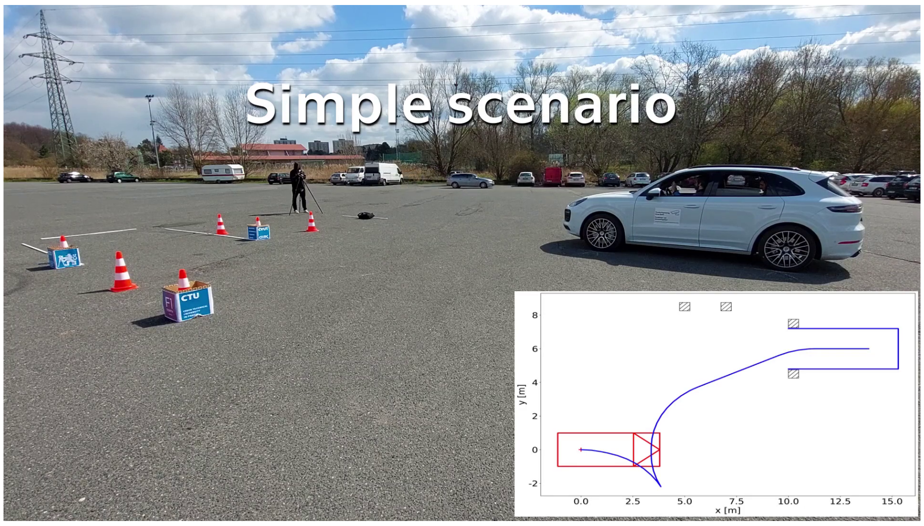


Figure 3.6: Simple scenario with the JUPITER vehicle.

This system takes a precalculated path, a parameter for its P regulator and starting position. Then it uses information from the *FlexRay* bus and GNSS sensor, which also returns a heading of the vehicle, to navigate the vehicle through the desired path. It also sends real-time markers for visualisation of the position and logs all the data from the experiment.

The simulated Porsche Cayenne was spawned on a five-line road, and translators were launched. The automated parking system was able to control the simulated vehicle according to the given paths.

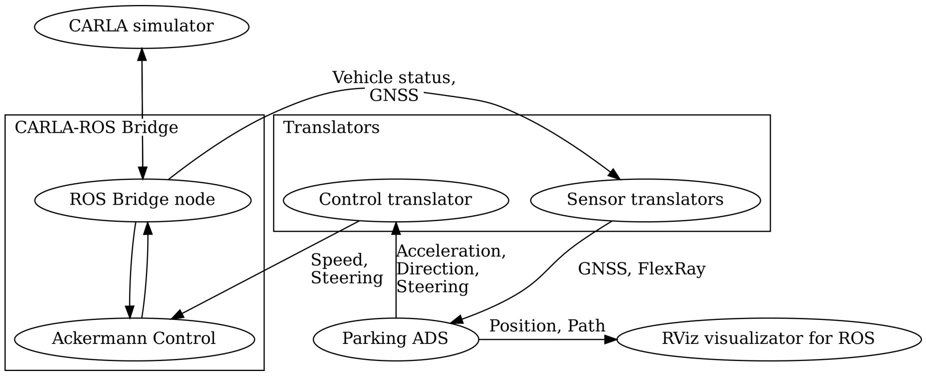


Figure 3.7: Communication with parking ADS.

The data flow of this test can be seen in Figure 3.7. Data from CARLA are transferred into ROS by the CARLA-ROS bridge. Sensor translators listen for vehicle status and data from the GNSS sensor, change the sensor's topic and publish proprietary messages about the state of the vehicle.

The parking ADS uses this data to compute the vehicle's desired direction, acceleration and steering angle. The control translator uses them, computes desired speed and sends it with a steering angle to the CARLA Ackermann Control package. Ackermann Control package uses a PID regulator to compute throttle, brake and steering that are, through ROS Bridge, used back in the CARLA simulator to control the simulated Porsche Cayenne.

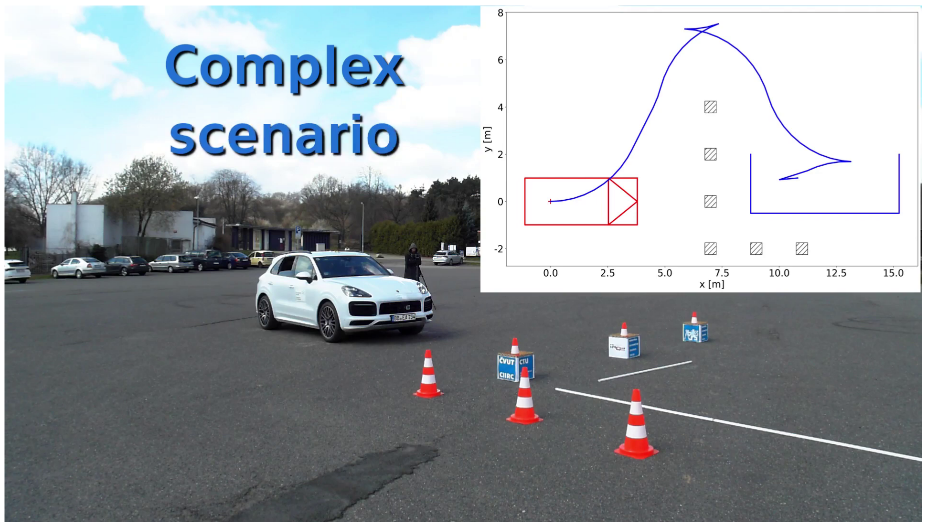


Figure 3.8: Complex scenario with the JUPITER vehicle.

Chapter 4

Comparing Real Vehicle with the Simulation

This chapter compares the results of experiments in the simulation with the experiments on the real vehicle. In Section 4.1, Livox lidar is compared to the default lidar sensor available in CARLA and to the simulated Livox lidar. In Section 4.2, the results of the experiments with ADS for parking are presented.

4.1 Lidar

This section briefly compares the Livox lidar installed in the real vehicle, the default lidar sensor of CARLA, and the Livox lidar added to the CARLA simulator as described in Section 2.3.

The parameters of CARLA lidar were set according to the Livox manual:

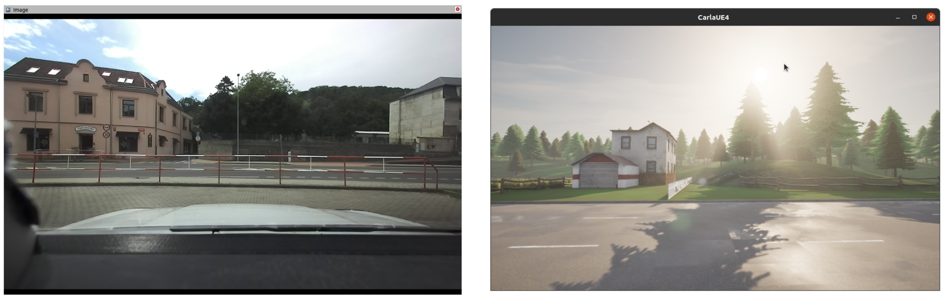
Horizontal field of view	81.7°
Vertical field of view	25.1°
Detection range	260 m
Points per second	240 000

Table 4.1: Livox specification.

The real Livox lidar was recorded using rosbag, so the ROS data from rosbag and CARLA-ROS bridge can be compared.

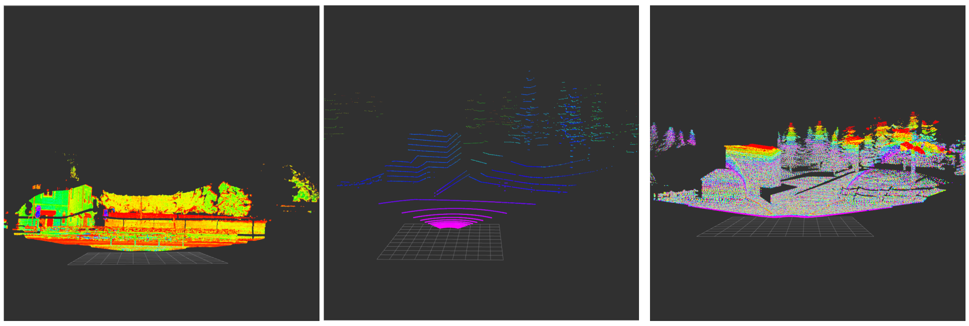
The rosbag has 31 s and 3345 lidar data messages. This means that the lidar sends data every 0.0093 s. To simulate this, the time between simulation ticks had to be lowered. It is achieved by setting parameter *fixed_delta_seconds* when starting *CARLA-ROS bridge*. The data were visualised by the tool RViz.

Figure 4.2 shows how lidars can scan the environment in 5 s. The CARLA lidar, shown in the middle, covers it the least because it always scans the same points, so a big time window for scanning does not reveal more different points. To reveal more points, more lasers can be set for this lidar. Simulated and real Livox scanned detailed images of the environment in front of the car. The simulated Livox seems to cover more distanced objects better than the real one and has a wider range of intensity on near points than the real one.



(a) : Real **(b) :** CARLA

Figure 4.1: Scanned environment.



(a) : Real Livox lidar **(b) :** CARLA Ray cast lidar **(c) :** Simulated Livox lidar

Figure 4.2: Lidar data collected in 5 s.

The shorter time period and three individual data messages for every Lidar were visualised in figures 4.3 and 4.4. The lidar from CARLA is rotating and scans only one angular section at a time. Livox lidars scan wider surroundings through some pattern.

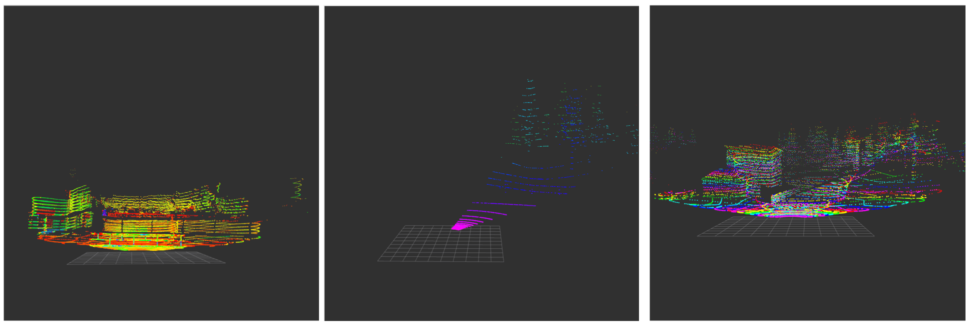

(a) : Real Livox lidar **(b) :** CARLA Ray cast lidar **(c) :** Simulated Livox lidar

Figure 4.3: Lidar data collected in 0.1 s.

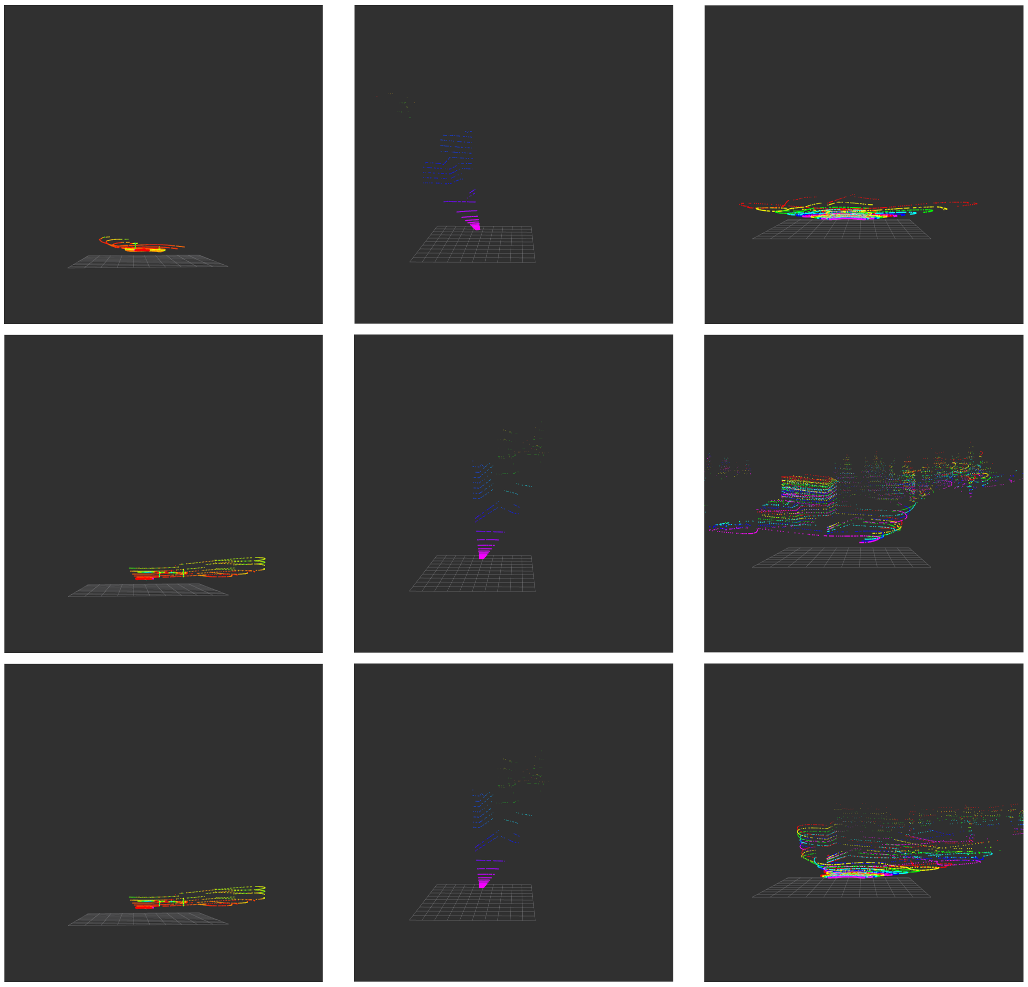


Figure 4.4: Visualisation data from 3 consecutive messages from used lidars.

4.2 Parking

Recorded data used in this next section comes from the experiments conducted on two parking scenarios: Simple and Complex. These data were recorded during experiments with the automated parking system on the JUPITER vehicle. The same experiments were done using the simulated car in the CARLA simulator, and data from these experiments can be compared.

The ADS was launched with the same settings as during the experiments on the real car. Parameters for Ackermann drive described in Section 3.1 were found experimentally and set to $p = 0.05, i = 0, d = 0$.

On the graphs 4.5 and 4.6 can be seen that in both experiments, the simulated vehicle does the expected manoeuvres to follow the desired path. It drove a trajectory with the same length, changed the direction when desired, and turned in the direction of the desired angle.

The real car had problems with a sharp turn in the Simple scenario, but the Complex scenario drove almost perfectly.

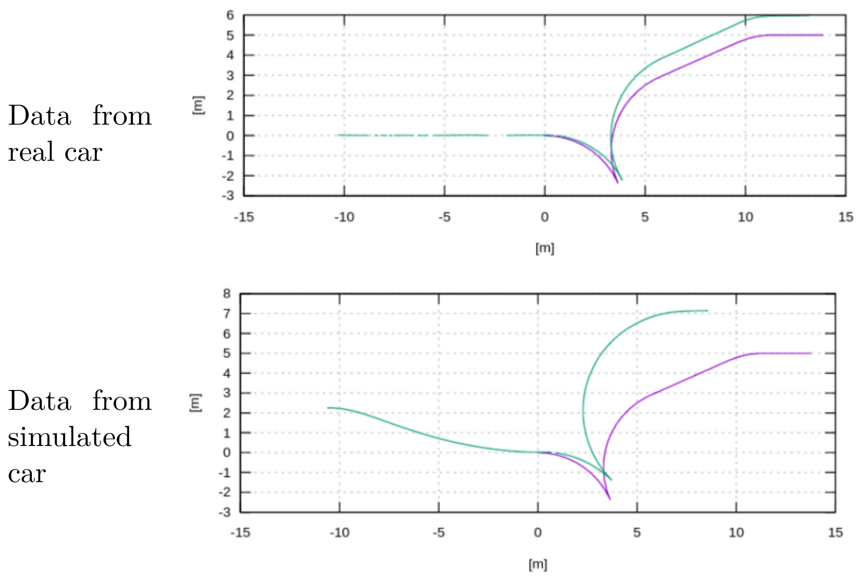


Figure 4.5: GNSS data from Simple scenario.

Data from real car

Data from simulated car

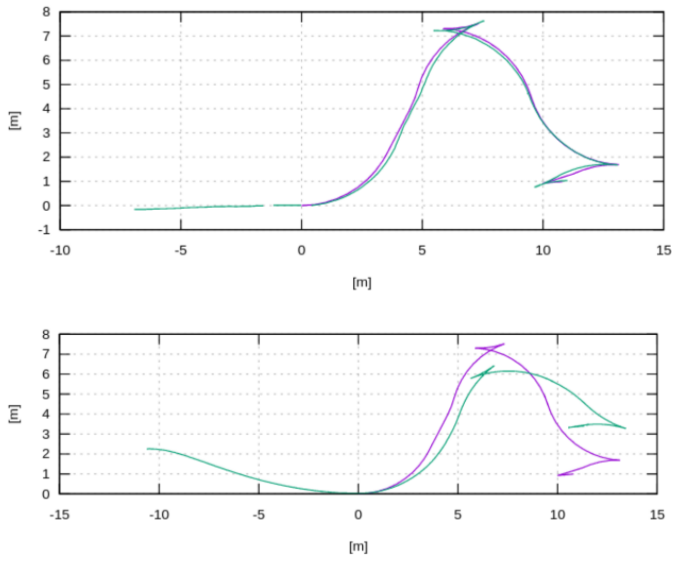


Figure 4.6: GNSS data from Complex scenario.

Figures 4.7 and 4.8 display the relative heading of the car to the starting point in degrees. From the figures, it can be seen that the simulated car turns slower than desired. One can see that the heading is not always improving to the desired but often remains constant. This is because the vehicle's heading changes only when the vehicle is moving. These constant parts correspond to parts with low speed on Figures 4.9 and 4.10.

Data from real car

Data from simulated car

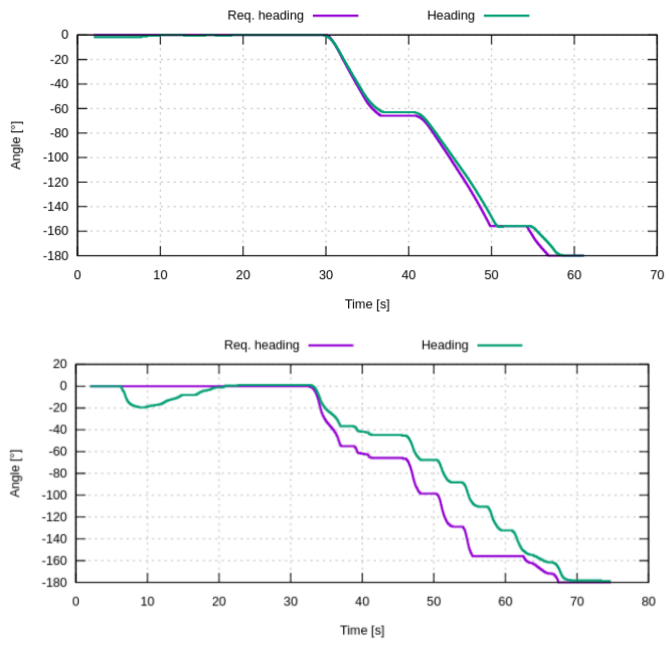


Figure 4.7: Headings in Simple scenario.

Data from real car

Data from simulated car

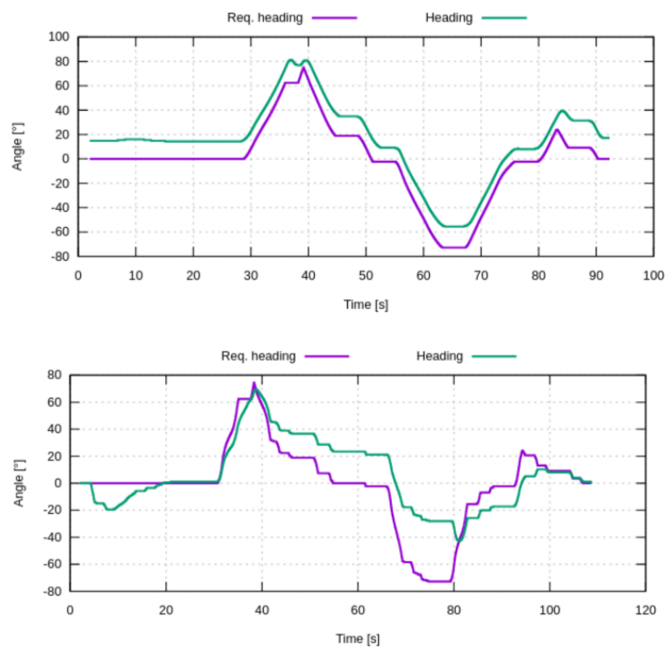


Figure 4.8: Headings in Complex scenario.

Figures 4.9 and 4.10 show speed in km/h, and also an acceleration in m/s^2. They show that the real car reacts to acceleration and breaking messages with some delay. On the other hand, the reaction delay of the simulated vehicle is shorter. Differences in delay and intensity of reaction are projected into the oscillation amplitude.

All the data together are available in Figures 4.11 and 4.12 for better context.

Data from real car

Data from simulated car

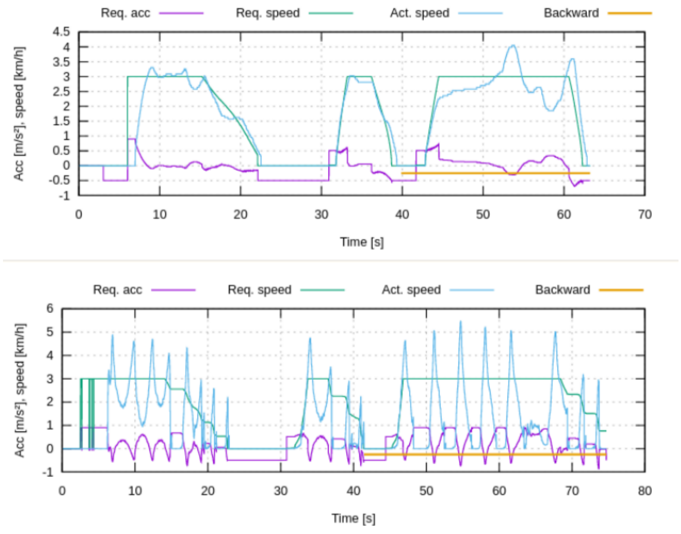


Figure 4.9: Speed and acceleration in Simple scenario.

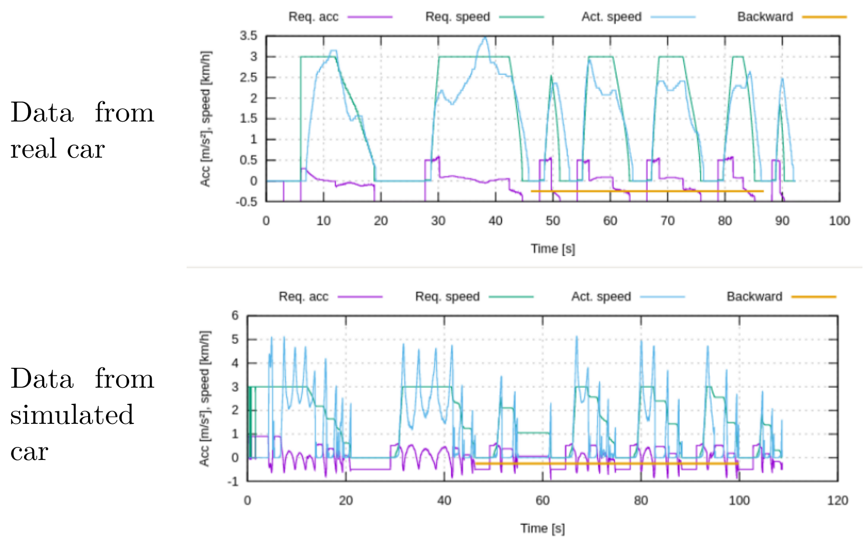


Figure 4.10: Speed and acceleration in Complex scenario.

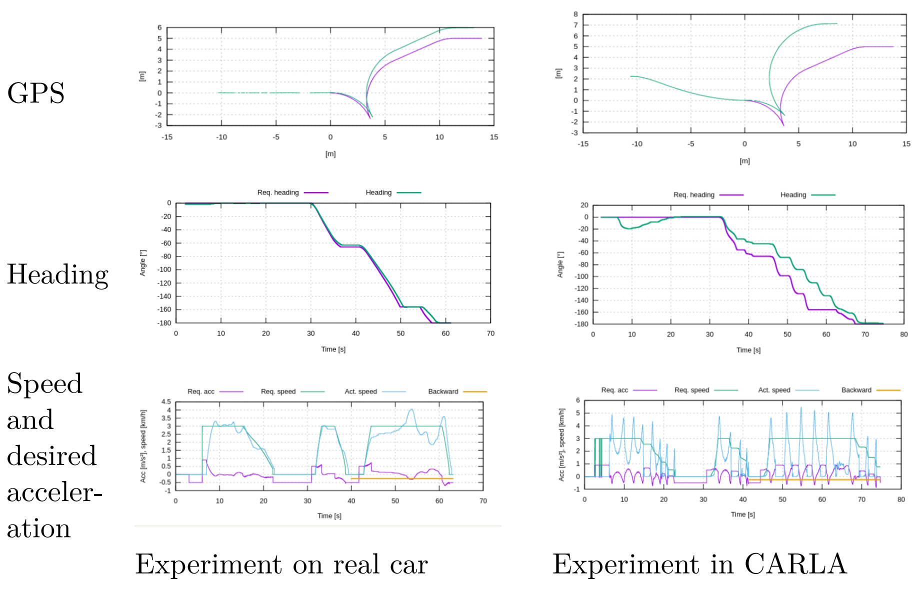


Figure 4.11: Experiments on simple scenario.

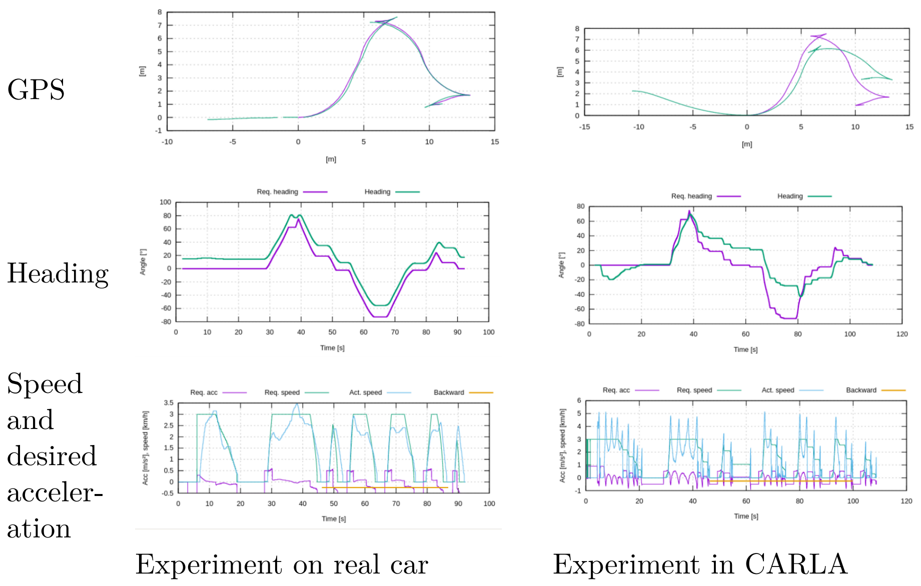


Figure 4.12: Experiments on complex scenario.

Chapter 5

Conclusion

This thesis presents the simulation environment for the JUPITER vehicle to test the Automated Driving Systems. The simulation environment is based on the state-of-the-art CARLA simulator extended with the custom model of a Porsche Cayenne and Livox lidar sensor.

The simulation environment includes ROS nodes translating ROS messages between the Automated Driving System and the CARLA Simulator because the subsystems of the JUPITER vehicle communicate via ROS.

The last chapter describes the differences between the results of experiments with the JUPITER vehicle and the simulation. The visual comparison of the lidar measurements shows that added model of lidar simulates the real Livox lidar better than the default lidar of the CARLA simulator. Moreover, the last chapter compares automated parking with a real vehicle and in the simulation.

The results from the experiment correspond to the expectations. It is possible to test the functionality of Automated Driving Systems developed for the JUPITER vehicle in the simulation environment presented in this thesis.

Based on the experiments, the direction for future work is to improve the physical parameters of the vehicle model and adjust the configuration of the ROS nodes translating ROS messages between the Automated Driving System and the CARLA Simulator. These changes would improve the simulation accuracy.

Bibliography

[1] *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.* [Online]. Available: https://doi.org/10.4271/j3016_202104.

[2] O. Krejčí, “Test environment for automated lane keeping system verification”, Bachelor’s Thesis, CTU, 2022.

[3] P. Weissensteiner, G. Stettinger, J. Rumetshofer, and D. Watzenig, “Virtual validation of an automated lane-keeping system with an extended operational design domain”, *Electronics*, vol. 11, no. 1, p. 72, Dec. 2021. [Online]. Available: https://doi.org/10.3390/electronics11010072.

[4] J. Vlasak, M. Sojka, and Z. Hanzalek, “Simulation environment for validation of automated lane-keeping system”, *International Conference on Advances in Vehicular Systems, Technologies and Applications*, Mar. 2023. [Online]. Available: https://www.thinkmind.org/articles/vehicular_2023_1_50_30013.pdf.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator”, in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[6] Epic Games, *Unreal engine*, version 4.26, Jan. 30, 2023. [Online]. Available: https://www.unrealengine.com.

[7] S. Malik, M. A. Khan, and H. El-Sayed, “CARLA: Car learning to act — an inside out”, *Procedia Computer Science*, vol. 198, pp. 742–749, 2022. DOI: 10.1016/j.procs.2021.12.316. [Online]. Available: https://doi.org/10.1016/j.procs.2021.12.316.

[8] E. Barbour and K. McFall, “Autonomous vehicle simulation using open source software carla”, *Journal of the UAB Early Career Technical Conference*, vol. 18, pp. 50–57, Nov. 2019. [Online]. Available: https://facultyweb.kennesaw.edu/kmcfall/2019ECTC01.pdf.

[9] CARLA Team. “Carla documentation”. (), [Online]. Available: https://carla.readthedocs.io/ (visited on 11/2022).

[10] H. Zhang. “How to add a vehicle/truck in carla using unreal engine editor 4 + blender for beginners”. (Mar. 2021), [Online]. Available: https://www.youtube.com/watch?v=0F3ugwkISGk (visited on 05/2023).

[11] MingooJ. “How do i add new car? #2738”. (Apr. 2020), [Online]. Available: https://github.com/carla-simulator/carla/issues/2738 (visited on 11/2022).

[12] pinkpocketTV. “Unreal engine 4 tutorial | drivable cars and vehicle physics”. (Dec. 2020), [Online]. Available: https://www.youtube.com/watch?v=_ZBQg7293v0 (visited on 11/2022).

[13] IKAROS93. “Livox laser simulation for carla”. (Jul. 2022), [Online]. Available: https://github.com/IKAROS93/Livox_laser_simulation_for_CARLA (visited on 03/2023).

[14] CARLA Team. “Ros bridge documentation”. (), [Online]. Available: https://carla.readthedocs.io/projects/ros-bridge/en/latest/ (visited on 11/2022).

[15] M. Matejcek. “Carla ros bridge manual control hud cardinal direction mismatch gnss #676”. (May 2023), [Online]. Available: https://github.com/carla-simulator/ros-bridge/issues/676 (visited on 05/2023).

Appendix A

List of attachments

source_code.zip

source_code.zip is an archive of the source code of ROS packages with translators, launchfiles and configuration files. They should be used with the CARLA-ROS bridge.

They are meant to be compiled together with packages containing interfaces of proprietary messages of ADS for the JUPITER vehicle. These interfaces are not attached due to licensing.

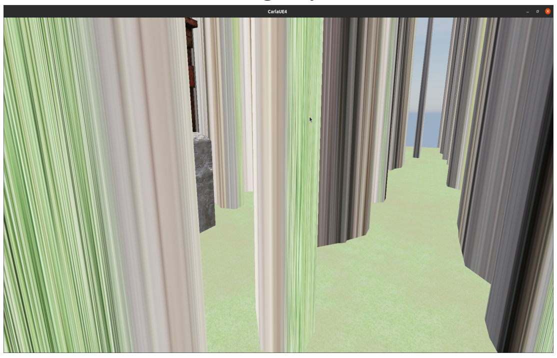
Appendix B

CARLA Is Awesome

```
Collision with 0 (impulse 57003.43427431811)
```

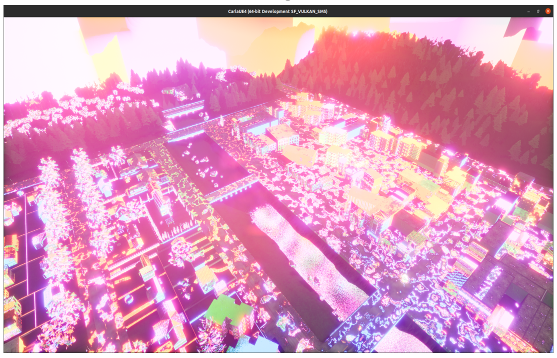
Carla will get you down

To the ground



Be the ground

But also high



Very high

Bye