Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Autonomous vehicle virtual verification plaftorm development

Jan Svoboda

Supervisor: Ing. Tomáš Haniš Ph.D.
Field of study: Cybernetics and Robotics
May 2021

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Svoboda Jan**
Personal ID number:	**474750**
Faculty / Institute:	**Faculty of Electrical Engineering**
Department / Institute:	**Department of Control Engineering**
Study program:	**Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Autonomous vehicle virtual verification platform development

Bachelor’s thesis title in Czech:

Vývoj verifikační platformy pro autonomní vozidla

Guidelines:

The development and verification process of autonomous vehicle is time costly and expensive. The virtual development and verification environment have potential to significantly cut development process related cost. The trajectory tracking algorithm for autonomous car will be designed and tested based on such virtual platform in this thesis. Following points will be addressed:
1. Get familiar with vehicle virtual testing platforms.
2. Implement suitable test scenario in selected software.
3. Implement reference trajectory tracking algorithm.
4. Verify the control algorithm using virtual testing scenarios.

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers

Name and workplace of bachelor’s thesis supervisor:

Ing. Tomáš Haniš, Ph.D., Department of Control Engineering, FEE

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

Date of bachelor’s thesis assignment: **20.01.2021** Deadline for bachelor thesis submission: ________

Assignment valid until:
by the end of summer semester 2021/2022

Haniš [illegible] [illegible]

Ing. Tomáš Haniš, Ph.D.	prof. Ing. Michael Šebek, DrSc.	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.

5.5.2021 — Date of assignment receipt

Svoboda — Student’s signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to everyone who gave me even for a small time his helping hand through my studies at Czech Technical University.

Thank you.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, 21. May 2021

Abstract

This thesis describes the implementation of a virtual environment and a trajectory tracking algorithm of the first autonomous vehicle of the eForce FEE Prague Formula team for the Formula student competition. It is mainly focused on two topics. The first topic is the implementation of a virtual environment in which control algorithm testing occurs. The Ansys VRXPERIENCE Driving Simulator tool is used for this implementation. The second topic focuses on designing a path tracking algorithm. For this purpose, the Stanley control law is used for lateral control and negative feedback control with a P regulator for longitudinal control.

Keywords: autonomous vehicle, formula student driverless, virtual environment, simulator, path tracking, lateral control, longitudinal control

Supervisor: Ing. Tomáš Haniš Ph.D.

Abstrakt

Tato práce se zabývá implementací virtuálního prostředí a algoritmu pro následování vytyčené trajektorie pro první autonomní vozidlo týmu eForce FEE Prague Formula team pro soutěž Formula student. Primárně se zaměřuje na dvě témata. Prvním tématem je implementace virtuálního prostředí, ve kterém se odehrává testování samotného řídícího algoritmu. Pro tento účel byl zvolen Ansys VRXPERIENCE Driving Simulator. Druhým tématem je design algoritmu pro sledování trajektorie. Pro příčné řízení je využit řídící zákon vyvinutý na Stanfordově univerzitě a pro podélné řízení je využita záporná zpětnovazební smyčka s P regulátorem.

Klíčová slova: autonomní vozidlo,formula student driverless,virtuální prostředí,simulátor,sledování trajektorie, příčné řízení,podélné řízení

Překlad názvu: Vývoj virtuálního verifikačního prostředí pro autonomní vozidla

Contents

1 Introduction **1**

1.1 Formula student SAE competition 1

1.2 eForce FEE Prague Formula team 2

1.3 Driverless formula eForce DV.01 . 2

1.3.1 DV.01 hardware 3

1.4 Formula Student Driverless disciplines 4

1.4.1 Skidpad 5

1.4.2 Acceleration 6

1.4.3 Autocross 6

1.4.4 Trackdrive and efficiency 6

2 Objectives of thesis **7**

3 Virtual testing platforms **9**

3.1 Research of automotive simulation tools 10

3.1.1 CarSim 10

3.1.2 dSpace - Automotive Simulation Models 10

3.1.3 rFpro 11

3.1.4 IPG CarMaker 12

3.2 ANSYS VRXPERIENCE Driving Simulator 13

3.3 Comparison of simulation tools . 13

4 Virtual reality implementation **15**

4.1 Virtual reality environment 15

4.2 Road profile 16

4.3 Road interconnection 17

4.4 Vehicle 18

4.4.1 Scenario 18

4.5 Vehicle Sensors 18

4.5.1 Sensors definition 19

4.5.2 Adding sensors to vehicle . . . 20

4.6 Simulink interface 20

4.7 Simulation 21

4.8 Vehicle control 21

4.8.1 Simulator inputs 21

4.8.2 Simulator outputs 22

5 Path tracking **23**

5.1 Path planning 23

5.2 Stanley control law 24

5.2.1 Lateral control 25

5.2.2 Lateral control improvement . 27

5.2.3 Longitudinal control 27

6 Experiments **29**

6.1 Advanced track 29

6.2 Route approach 31

6.3 Circle track 32

6.4 Star track 33

7 Results **35**

8 Conclusion **37**

A Bibliography **39**

B Project Specification **41**

Figures

1.1 eForce DV.01 3

1.2 Sensor layout 1,2 - Intel Realsense; 3 - Ouster OS1-64 LiDaR; 4 - Stereolab ZED 4

1.3 Skidpad track layout[7] 5

3.1 CarSim simulator[10] 10

3.2 dSpace simulator[11] 11

3.3 rFpro simulator[9] 12

3.4 IPG CarMaker simulator[12] . . . 13

4.1 DefaultWideMonotrack profile . . 17

4.2 Network generation 17

4.3 Sensor configuration screen 19

5.1 Kinematic model of the vehicle[5] 25

5.2 States of the vehicle[5] 26

5.3 Front wheels angle controller . . . 26

5.4 Speed controller 28

6.1 Advanced track experiment - trajectories 30

6.2 Advanced track experiment - crosstrack errors 30

6.3 Route approach trajectories 31

6.4 Route approach crossroad errors 32

6.5 Circle track experiment trajectories 33

6.6 Star track experiment trajectories 34

Tables

1.1 Point distribution amongst disciplines 4

3.1 Simulation tools comparison . . . 14

Chapter 1

Introduction

1.1 Formula student SAE competition

Formula student is the largest competition for technical students in the world. It was founded in 1981 in Texas, USA, as a competition in which students could apply their knowledge to a real world engineering design problem - the design of a race car. Since then, this competition has undergone great changes. In 1998, the first formula student race took place in Europe. Nowadays, there are dozens of races all around the world. Every race has its own control of rules but rules across all races are more or less the same with small variances. One of the most prestigious races in Europe is Formula Student Germany, which has built a reputation of benchmark across races. From the beginning Formula Student was purely a competition for vehicles with combustion engines but in 2010 due to the development in the automotive industry and probably the social opinion on fossil fuels, the competition was divided in two divisions - FSC and FSE, Formula Student Combustion and Formula Student Electric. In 2017, regarding the rise of autonomous driving vehicles, another division was announced - FSD, Formula Student Driverless. Nowadays, over 800 student teams compete across these three divisions. Some of the teams worth mentioning are KIT Karlsruhe, ETH Zurich, UAS Hamburg, TU Delft. In the future, another change is expected. It is already announced that in 2022 at FSG all categories will be merged together, and autonomous functionality will be demanded in every vehicle. However, FSG is the only race that has announced such statement and none of the other races e.g. FS Spain, FS Netherlands, has done this yet.

1.2 eForce FEE Prague Formula team

eForce FEE Prague Formula is a student team of the international Formula Student competition competing under the Faculty of Electrical Engineering of the Czech Technical University in Prague. The team was founded in 2010 under the banner of CTU CarTech, as the first Czech team to successfully build a formula with an electric drive. It has been the only team with this prerogative ever since then. This team has achieved many great results. One of the most significants is a 1st overall place at FS West in USA and FS North in Canada in 2016. Other honourable mentions are a 2nd Overall place at FS Italy in 2014 or a 1st overall place at FS Czech in 2018. The team consists of about 30 active team members who work every year on designing a new vehicle from scratch. The team is divided in working groups such as electrotechnical, mechanical, IT and project group. In summer 2019, with taking the fact of the creation of a driverless division into account, a driverless group has been formed under the supervisors Marek Szeles and Ondřej Šereda.

1.3 Driverless formula eForce DV.01

Eforce DV.01 is the name of the first autonomous eForce formula (DV stands for Driverless). To speed up the process, our team has decided not to build a new vehicle from scratch but to rebuild one of the cars from the previous seasons. From a few operational vehicles, FSE.07 was chosen. Not only monococcus but also engines, suspension, brake system, battery pack, and all electronics were reused in DV.01. However, changes had to be made to meet the rules, e.g., an emergency brake system and steering actuators were added. The vehicle was equipped with a set of sensors to enable vision and orientation in space and finally a CPU.

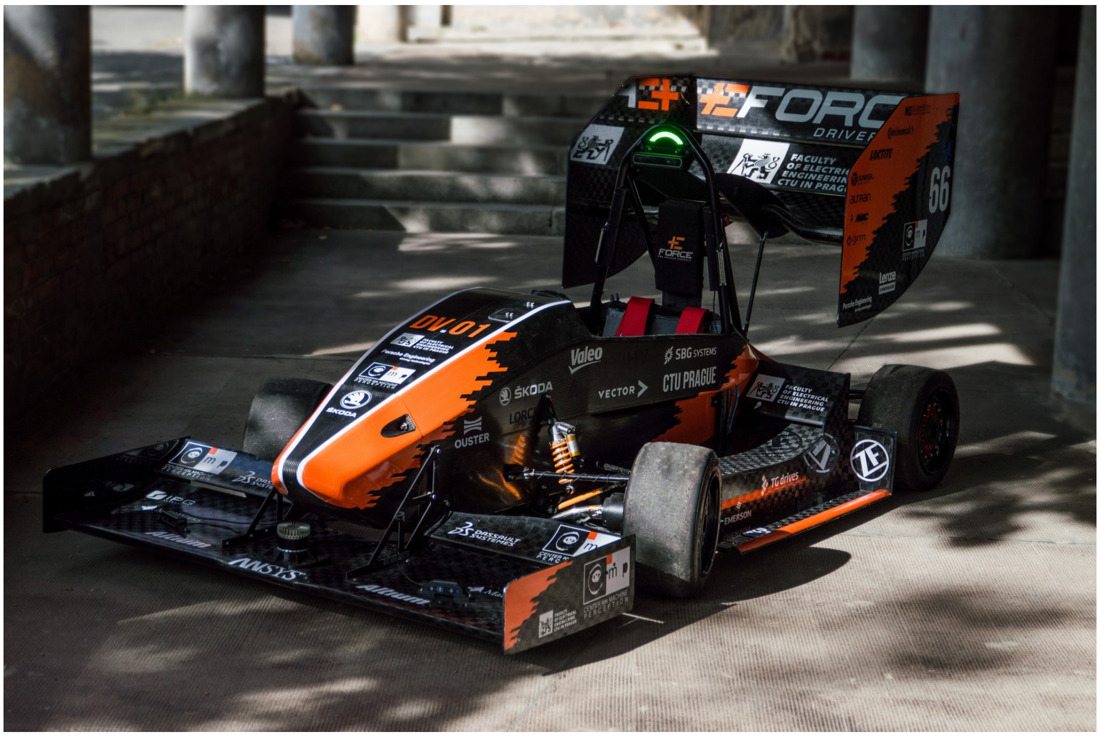


Figure 1.1: eForce DV.01

1.3.1 DV.01 hardware

As said, DV.01 has evolved from FSE.07 and it has inherited all its parameters. DV.01 has the weight of 202 kg with a 1.54 m wheelbase. Every wheel is powered by a separate engine placed inside the wheel. These engines are 8.6 kW with the weight of 3.5 kg and a maximal torque 16.2 Nm in front and 35.3 kW with the weight of 8 kg and a maximal torque 48 Nm in the rear wheels. This gives the vehicle the total power of 87.8 kW, but this parameter is limited to 80 kW because of the FS rules. The steering wheel is controlled by an electric power steering kit. Braking is managed by engines in the wheels by forcing them to exact torque. To enable perception, 3 stereocameras and LiDaR were added. There are 2 Intel Realsense and 1 Stereolab ZED stereocameras and Ouster OS1-64 LiDaR on the vehicle.

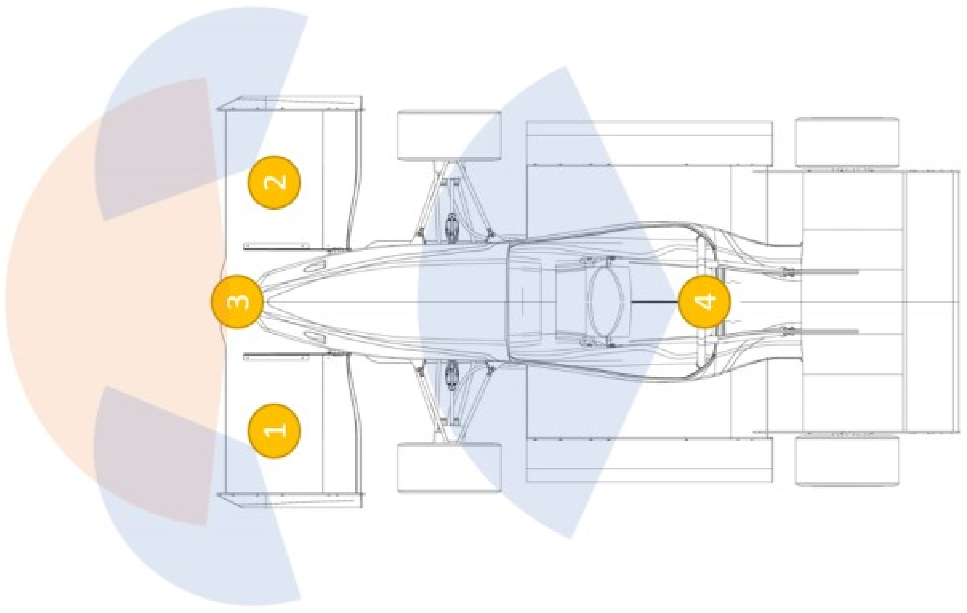


Figure 1.2: Sensor layout 1,2 - Intel Realsense; 3 - Ouster OS1-64 LiDaR; 4 - Stereolab ZED

1.4 Formula Student Driverless disciplines

Driverless vehicles are rated in two disciplines - static and dynamic. Both of these are divided to 4 subdisciplines. Engineering design report, cost report, business plan and energy efficiency in the static disciplines and acceleration, skid pad, autocross and endurance in the dynamic disciplines. Vehicles can get 1000 points at maximum, these points are distributed between disciplines in the manner described in the table below. In this thesis, we are developing

	Discipline	Maximum points
Statics	Bussines Plan	75
	Cost and Manufacturing	100
	Engineering design	300
Dynamics	Acceleration	75
	Skidpad	75
	Autocross	100
	Track Drive	200
	Overall	1000

Table 1.1: Point distribution amongst disciplines

a simulating platform for our autonomous formula, so the main goal is to increase the point gain in the dynamic disciplines. They will be introduced

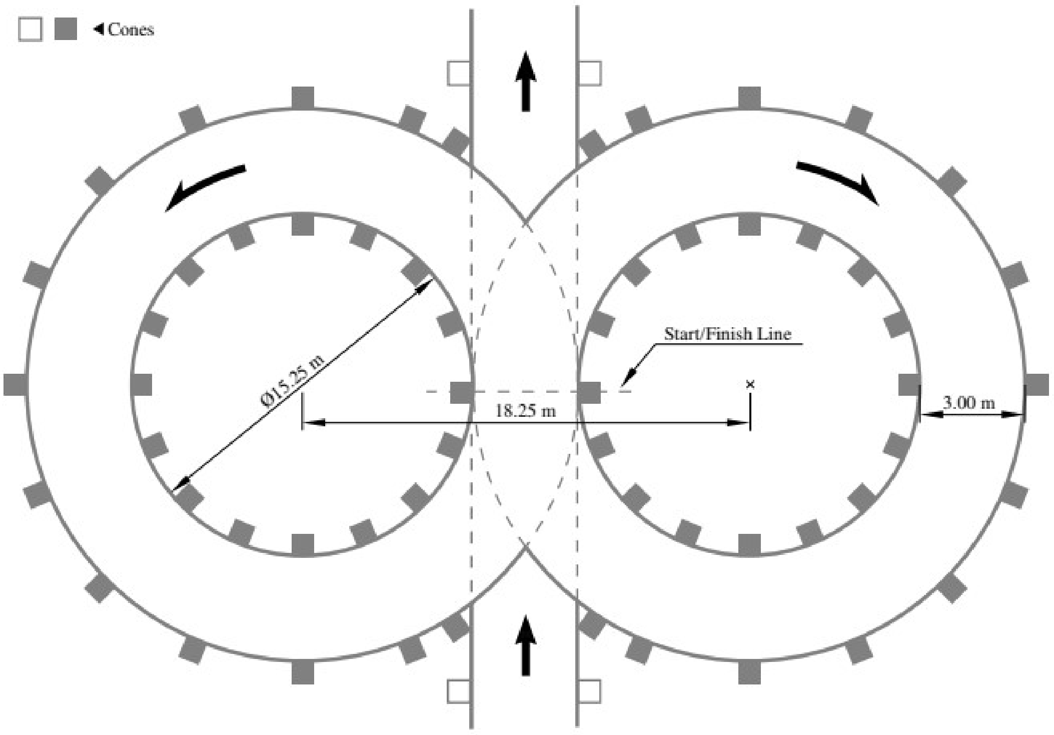


Figure 1.3: Skidpad track layout[7]

further in the following paragraphs

1.4.1 Skidpad

The skidpad track consists of two pairs of circles in a figure of an eight pattern. The distance between the circles is fixed to 18.25m, as the diameters of the inner (15.25m) and outer (21.25m) circle. The driving path is marked by 16 cones on the inside and 13 cones on the outside of each circle. The discipline consists of arriving perpendicularly to the skidpad track, then encircling each circle twice. The right circle is encircled first, then the left circle. Only the second encirclement of each circle is measured. After the completion of the skidpad discipline, the vehicle leaves the skidpad track in the same direction as it arrived. Finally, it must come to a full stop after 25m. Apparently, the goal is to get the best time possible. It should be mentioned that in the skidpad procedure as well as in every following procedure, teams get penalties for knocking down cones and noncompliance with other rules mentioned in [7].

1.4.2 Acceleration

The acceleration track is 75 metres long and at least 3 metres wide. Cones are placed along the track in intervals of 5 metres. The foremost part of the vehicle is staged at 0.30 m behind the starting line, the vehicle accelerates from a standing start. After the finish line, the vehicle must stop within 100 m inside the marked exit lane. The goal is to get the best time possible.

1.4.3 Autocross

The autocross discipline takes place on a handling track with the following parameters:

- Straights: No longer than 80 m.
- Constant turns: Up to 50 m in diameter.
- Hairpin turns: Minimum of 9 m outside diameter
- Slaloms: Cones in a straight line with 7.5 m to 12 m spacing.
- Minimum width: 3 m.

The length of the race is approximately 200m to 500m. The vehicle accelerates from a standing start and takes one lap aiming for the best time possible.

1.4.4 Trackdrive and efficiency

The trackdrive and efficiency discipline is almost the same as the autocross discipline. Apart from autocross, in trackdrive and efficiency discipline, the vehicle takes 10 laps. Energy consumption is measured and the goal is to get the best time possible with the lowest power consumption possible.

Chapter 2

Objectives of thesis

To achieve the best results possible, the whole car pipeline must be working at its best. This means hours and hours of testing. Since eForce is a student project with limited resources, this is not possible. Having a high-reliable tool for simulating the real environment is crucial. This thesis aims at implementing this tool. Several algorithms will be used. This includes a track generating algorithm that generates the positions of a set of cones marking the track, a path planning algorithm which calculates a center line from the detected cones, and an identification of the physical model of eForce DV.01. These were created by eForce team members and are already tested. Objectives for this project are the following:

1. Become familiar with virtual testing platforms
    - Search market for possible options
    - Choose best solution
2. Implement suitable test scenario in selected software
    - Design vehicle as similar to DV.01 as possible
    - Add sensors to vehicle
    - Create track
    - Automate track creation
3. Implement reference trajectory tracking algorithm
4. Verify the control algorithm using virtual testing scenarios

Chapter 3

Virtual testing platforms

While the automotive industry invests an immense amount of resources in autonomous driving, the development of the autonomous driving industry is conditioned by the development of reliable virtual testing platforms. In general, these platforms are developed to match the needs of single automotive manufactures, they are expensive, and it is hard to get access to such technologies. In this chapter, the research of these platforms is described and further choice of ANSYS VRXPERIENCE is explained.
A virtual testing platform must fulfill these requirements:

- Python and Matlab interface
- Physics engine
- Creating and editing scenarios
- Creating and editing vehicle
- Sensors used in the car must be supported
- Software in the loop simulation

in addition, other functionalities are welcome:

- User-friendly manipulation
- Free to use software

3.1 Research of automotive simulation tools

3.1.1 CarSim

CarSim[1] is a commercial software package that delivers accurate, detailed and efficient methods for simulating the performance of passenger vehicles. It also supports sensors but without any way of editing them. Furthermore, using sensors in CarSim is a condition of the ownership of an extended license.

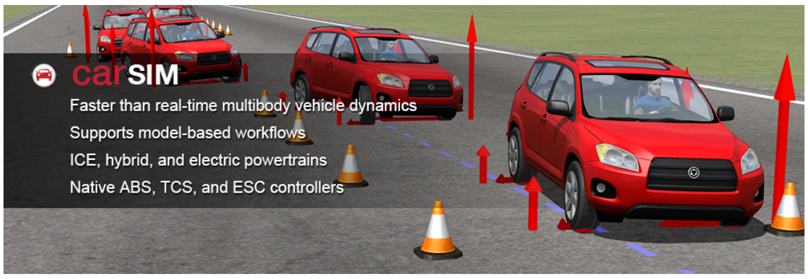

carSIM
Faster than real-time multibody vehicle dynamics
Supports model-based workflows
ICE, hybrid, and electric powertrains
Native ABS, TCS, and ESC controllers

Figure 3.1: CarSim simulator[10]

3.1.2 dSpace - Automotive Simulation Models

dSpace ASM[2] is a tool suite for simulating combustion engines, vehicle dynamics, electrical components, and the traffic environment. It has Simulink interface, also vehicle editing is at a great level. However, this software mainly focuses on vehicles with a combustion engine and regarding this fact, dSpace is not the best tool for this work.

[1]For more information see `www.carsim.com`
[2]For more information see `www.dspace.com`

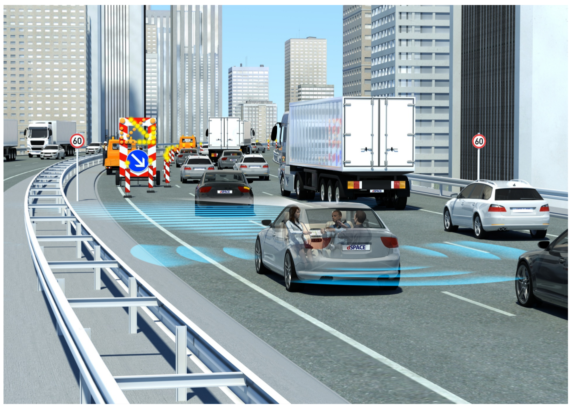


Figure 3.2: dSpace simulator[11]

3.1.3 rFpro

rFpro[3] provides driving simulation software, and Digital-Twins for the development and testing of autonomous vehicles, ADAS, and vehicle dynamics. It is solely focused on ground-based road vehicle simulation. rFpro was created in 2007 as a project within a Formula 1 team, where performance, speed of response and simulation of the fastest, most dynamic road vehicle on earth was all that mattered. The main disadvantage of this tool seems to be the absence of creating your own scenarios. Scenarios are, probably because of the top level of road quality and the fact that these roads are very close to reality, made mainly by rFpro and it looks like they cannot be edited. Also this product appears to be more premium thanwhat we could afford.

[3]For more information see `www.rfpro.com`

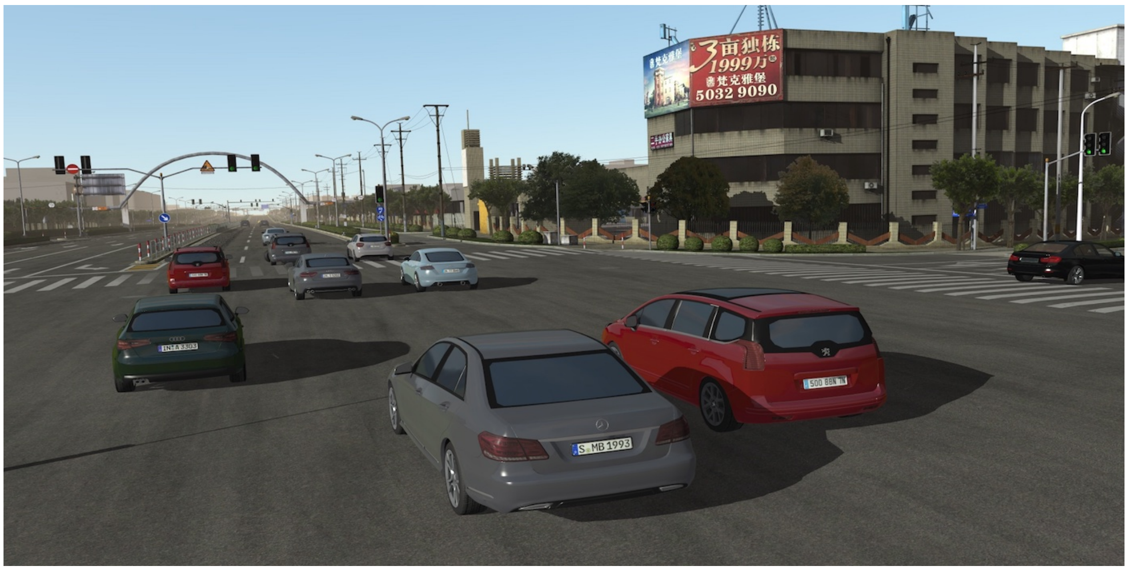


Figure 3.3: rFpro simulator[9]

3.1.4 IPG CarMaker

IPG CarMaker[4] is one of the best softwares for simulating car behaviour. It meets almost all of our requirements except for Python interface. IPG CarMaker includes a complete model environment for vehicle testing. Most of the student formula teams use IPG CarMaker as their main simulating tool which only underlines the usability of this tool. Despite that we decided not to choose IPG CarMaker and go for a new product on the market with great expectations.

[4]For more information see `https://ipg-automotive.com/products-services/simulation-software/carmaker/`

Figure 3.4: IPG CarMaker simulator[12]

3.2 ANSYS VRXPERIENCE Driving Simulator

The author's decision was to use ANSYS VRXPERIENCE Driving Simulator as the tool for our simulations. VRXPERIENCE is powered by SCaNeR by AV Simulation. It enables creating our own scenarios, creating and editing a vehicle. The software in the loop is available as well as the hardware in the loop and the driver in the loop. Matlab and Python interface is possible and it meets all of our requirements. It supports LiDaR, camera, which can be created, adjusted to match our sensors and put on objects in the simulation. The only disadvantage is the absence of INS, but we have found a pretty easy workaround, which will be presented further in the thesis. Python interface is used mainly for creating scenarios and Matlab interface is used for vehicle control and reading data from sensors.

3.3 Comparison of simulation tools

All of the simulation tools mentioned above were compared regarding requirements.

	CarSim	dSpace	rFpro	CarMaker	VRXPERIENCE
Python and Matlab interface	Yes	Yes	Yes	Partially	Yes
Physics engine	Yes	Yes	Yes	Yes	Yes
Creating and editing scenarios	Yes	Yes	No	Yes	Yes
Creating and editing vehicles	Yes	Yes	Yes	Yes	Yes
Sensor support	Yes, with license	Yes	Yes	Yes	Yes
Software in the loop	Yes	Yes	Yes	Yes	Yes

Table 3.1: Simulation tools comparison

It is shown that CarSim, dSpace, CarMaker and Vrxperience completely fulfill our requirements. The Ansys VXPERIENCE Driving Simulator was chosen between these because I wanted to explore the capabilities of a new software rather than using a proven solution. In addition, support from Ansys was promised.

Chapter 4

Virtual reality implementation

In this chapter, virtual reality implementation is described. Ansys VRXPERIENCE was chosen as our main tool. It is important to create a virtual reality environment, vehicle similar to eForce DV.01 with appropriate sensors, to create a track which consists of a set of cones and finally to create an interface between VRXPERIENCE and Simulink.

4.1 Virtual reality environment

The first thing that needs to be done is the creation of a virtual reality environment. In VRXPERIENCE virtual reality is called *Terrain*, so in the rest of the thesis virtual reality is referenced to as terrain. The VRXPERIENCE was developed mainly for the simulation of real-life traffic. It aims at creating road networks with realistic road intersections, traffic lights and pedestrians for autonomous vehicle development purposes. Therefore, editing the road profile gets us straightforward to creating real world roads with certain properties, such as the number of traffic lanes in the road, travel direction and so on. However, nothing from the above is needed for the purposes of this work. That is why in creating the terrain we somply focus on creating a vast concrete plain similar to a small airport or a big parking lot. The dimensions of this plain are 1000m x 500m. This is done by placing several 1km long roads abreast.

4.2 Road profile

VRXPERIENCE comes with 27 prebuilt road profiles. Most of them are however defined by traffic lanes and surrounded by grass at the sides. The only choice was to use 'DefaultMonotrack'. Because 'DefaultMonotrack' has a default width of 5 m, placing 100 roads would be unnecessarily difficult, change in width is used to simplify the process of terrain generation. 'DefaultMonotrack' was used to form 'DefaultWideMonotrack'. The road profile is defined by several properties where the most importants are width, ground and material, which are specified later. However a majority of the properties are used for controlling a vehicle in traffic mode, in which the car's behaviour is controlled by a set of rules, which vehicles obey. Since I do not use traffic mode and in the simulation there will be only one car controlled by Simulink, only these parameters are necessary to be set:

- Width
- Ground name
- Material name

Width should be set to the biggest value possible, which is 100m. Ground name is the type of ground that is represented by the track profile. VRXPERIENCE comes with 4 built-in ground types:

- Asphalt
- Concrete
- Cobblestone
- Grass

Even though ground types can be created or edited by setting parameters to different values, e.g., grip, roughness, etc., I decided to use asphalt as our road profile because it appears that creating several road profiles with different values would be unnecessary. Yet creating a set of ground types for each track on which the formula would race seems as a good opportunity to get better results. Material name only defines VISUAL parts of the road profile. It can be either a smooth color or it can be defined by texture. For the purposes of this thesis, I wanted to get a concrete look material 'macadam', it is a perfect match.

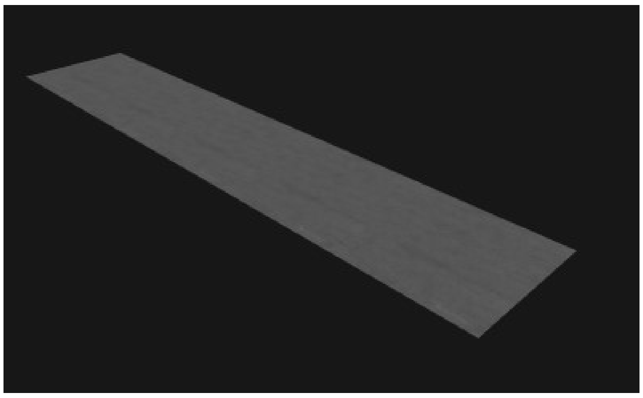
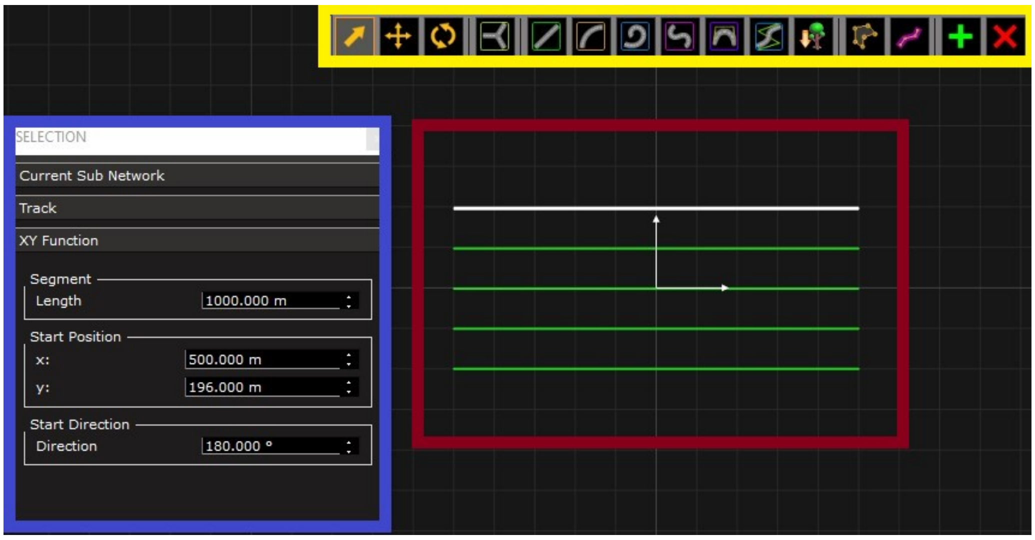


Figure 4.1: DefaultWideMonotrack profile

4.3 Road interconnection

Each road in VRXPERIENCE is represented as a curve in a road network. Every road is made easily by drawing a line using tools (highlighted yellow in figure 4.2). As said earlier, I want to generate just a vast plain with the proportions 1000m x 500m, and since I have created a road profile 100m wide, putting 5 roads side by side is needed. Drawing a line of an exact length in an exact position is almost impossible, exact values can be edited using the selection window in XY function section(highlighted blue in figure 4.2).



SELECTION
Current Sub Network
Track
XY Function

Segment	
Length	1000.000 m
Start Position	
x:	500.000 m
y:	196.000 m
Start Direction	
Direction	180.000 °

Figure 4.2: Network generation

When a network (highlighted in red in 4.2) is created, it is important to generate the terrain and export it to 3D Visual. This is done in 'Terrain ->

Terrain generation' and 'File -> Export 3D Visual'.

4.4 Vehicle

For a vehicle model generation, the mode 'Vehicle' is used. Vehicles are divided into different groups in VRXPERIENCE. Most of the vehicles are in *Simple* class which means that their physics is just a 6DOF object floating through space with some friction, and an internal force drives them forward. No further dynamics, vehicle tires nor suspension is implemented. Because my goal is to get as close to reality as possible, formula will be simulated as *Callas* vehicles. *Callas* is a French acronym for *Couplé A La Limite d'Adhérence au Sol* translated as "Coupled with the Limit of Adhesion to the Ground". These have a much wider range of possibilities. Due to this fact, I decided to redesign 'Callas' vehicles to resemble an eForce DV.01 car. Every part of the car can be edited in VRXPERIENCE. For this purpose, models of engine, aerodynamics, transmission and tires were incorporated.

4.4.1 Scenario

When the terrain and the vehicle are created, the next step is to generate a driving scenario. In 'Scenario' mode, I created a new scenario following the user manual, I selected the generated terrain and edited a scenario. For this thesis, it means adding a vehicle to the scenario, which is done by drag and drop from *Resources*. After that, track generation follows. Since there was no possibility shown to me on how to generate it automatically, this was done manually by drag and drop of cones from resources to scenario and then editing the position for each cone individually.

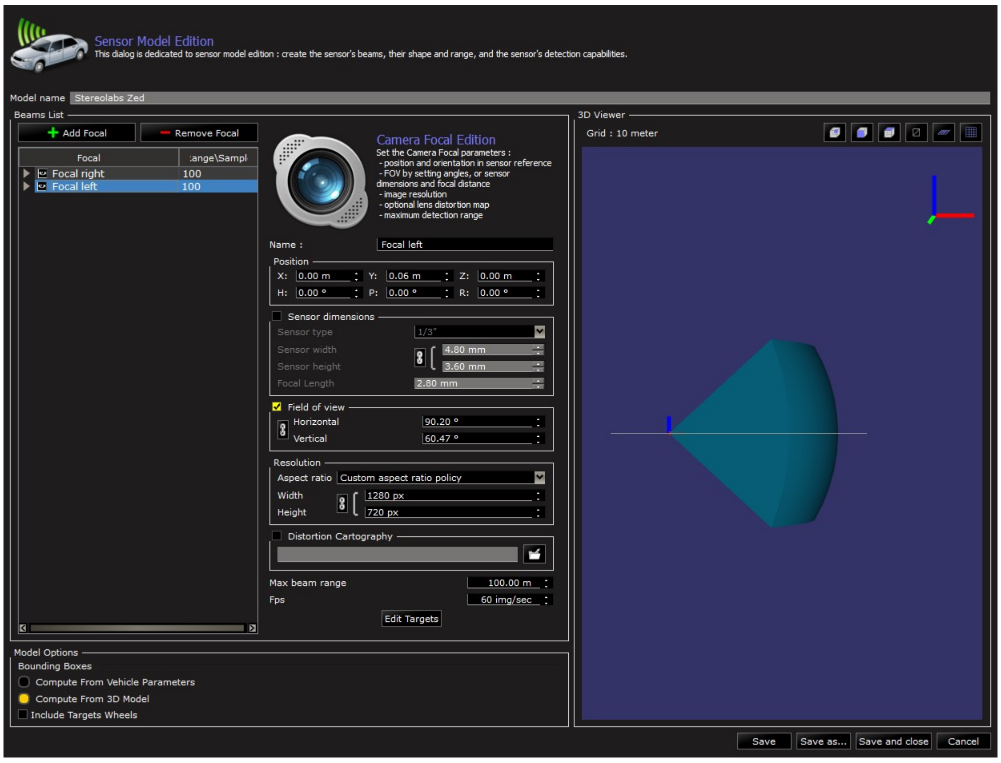
4.5 Vehicle Sensors

Sensors cannot be added to the vehicle right away in the 'Vehicle' mode during vehicle generation. As all sensors behave as part of the scenario and thus they need to be added during a scenario definition. A doubleclick on the selected vehicle opens the vehicle instance setup which includes Sensor configuration. Here, I created a sensor configuration. In this thesis, it includes creating two models of stereocameras, one model of LiDaR, and adding them to the sensor configuration.

4.5.1 Sensors definition

The biggest advantage of VRXPERIENCE is that it enables sensor generation. In generating a stereocamera system, it includes creating a set of lens with these parameters:

- Position of sensor reference frame
- Focal dimensions
- Field of view
- Resolution
- Distortion cartography
- Max beam range
- FPS


Sensor Model Edition
This dialog is dedicated to sensor model edition : create the sensor's beams, their shape and range, and the sensor's detection capabilities.

Model name: Stereolabs Zed

Beams List

Focal	:ange\Sampl
Focal right	100
Focal left	100

Camera Focal Edition
Set the Camera Focal parameters :
- position and orientation in sensor reference
- FOV by setting angles, or sensor dimensions and focal distance
- image resolution
- optional lens distortion map
- maximum detection range

Name : Focal left

Position
X: 0.00 m Y: 0.06 m Z: 0.00 m
H: 0.00 ° P: 0.00 ° R: 0.00 °

☐ Sensor dimensions
Sensor type: 1/3"
Sensor width: 4.80 mm
Sensor height: 3.60 mm
Focal Length: 2.80 mm

☑ Field of view
Horizontal: 90.20 °
Vertical: 60.47 °

Resolution
Aspect ratio: Custom aspect ratio policy
Width: 1280 px
Height: 720 px

☐ Distortion Cartography

Max beam range: 100.00 m
Fps: 60 img/sec

Model Options
Bounding Boxes
○ Compute From Vehicle Parameters
● Compute From 3D Model
☐ Include Targets Wheels

3D Viewer
Grid : 10 meter

Figure 4.3: Sensor configuration screen

This was done for both stereocameras. The next step was generating the model of LiDaR. Generating the LiDaR model is as easy as generating camera models. Parameters that can be edited are:

- Horizontal and vertical field
- Horizontal and vertical step
- Range
- FPS

The next step is adding sensors to a vehicle reference frame.

4.5.2 Adding sensors to vehicle

When sensor models are created, adding sensor models to the vehicle reference frame is necessary. As stated in figure 1.2 , each sensor has its defined position on the real vehicle. All sensors were added to the simulated vehicle at the same positions.

4.6 Simulink interface

To be able to control the vehicle, Simulink interface is implemented. Every Simulink model which is used in VRXPERIENCE, must be compiled. MinGW64 compiler is recommended. Additionally, the target compilation must be set. Steps for a well-working interface before starting are the following:

1. In Matlab use command mex -setup to configure mex to use MingGW64 C++ compiler.
2. Run setScanerPath.m and setupScaner.m
3. Open new Simulink model.
4. Press CTRL+R and in code-generation choose $grt_scanerapi.tlc$ as target, and C++ as language.

5. Check model.
6. Add Controller from ScanerAPI to model (remember to add ScanerAPI to path.

4.7 Simulation

When all of the above is done, the last step is to create a process in Simulation. In VRXPERIENCE start 'Simulation' and in 'Configuration' -> 'Configuration manager' add process named 'SIMULINK'. Name does not need to be exactly 'SIMULINK' but it is the most practical. Then in the Simulink model change Controller property process name to 'SIMULINK'. After this is done, select the vehicle that will be controlled and change its Driver process to 'SIMULINK'. After that, simply run the Simulink model and start Simulation. If everything was set right, a black square in a green circle will appear over SIMULINK process in VRXPERIENCE.

4.8 Vehicle control

As noted earlier, the Simulink interface is used for the vehicle control. In this thesis, as in a real vehicle, the control of the steering wheel and the accelerator and brake system is utilized. For this task a block in ScanerAPI is created. These blocks use direct communication with the shared memory of VRXPERIENCE.

4.8.1 Simulator inputs

For control of the steering rod, brakes, and the accelerator blocks that are used are named $'ModelCabin\_CabToModel\_Output'$ for the accelerator and the brake pedal control and $'ModelCabin\_CabToSteering\_Output'$ for the steering rod control. Both blocks have several inputs but only a few are important. For the steering rod control these are:

- IsTorqCommand(boolean) - if is true, torque which is also one of inputs is applied

- SteeringWheel(rad) - steering wheel angle

since vehicle steering is regulated to an exact position and does not use torque, IsTorqCommand is set to 0. For control of the accelerator and the brake pedal important inputs are:

- Accelerator - from 0 to 1 (1 is fully pressed) accelerator
- Brake(Newton) - force applied to brake
- IgnitionKey - ignition key position
- GearBoxAutoMode - gear box mode

IgnitionKey MUST be set to 2 which corresponds to the running engine and GearBoxAutoMode to 11 which corresponds to Gearbox in automode racing. More modes can be used but author prefers this choice as it appears to be the most reliable.

4.8.2 Simulator outputs

For the communication between the vehicle and the Simulink block named $'ModelCabin\_CabToModel\_Input'$. From this block user is able to get all information about the vehicle and its state. In this project, position, speed, acceleration in reference frame and yaw position, speed and acceleration is everything that is needed.

Chapter 5

Path tracking

5.1 Path planning

A path planning algorithm implemented by an eForce team member Matěj Zorek is used. This algorithm calculates the center line as a set of discrete points in the plain, taking into account the fact that at least 3 cones were detected. In this thesis I don't focus on the vision algorithm, so it is assumed that all cones are already detected. Algorithm is further described later in algorithm 1. Because the Stanley control law, which is described below, takes not only positions as input but also the yaw angle, yaw rate and vehicle speed, this algorithm needs to be upgraded. First, as the speed reference generator is not implemented yet the author has decided that the constant speed $v_{ref} = 5\frac{m}{s}$ will be used at all path points. Next the yaw angle and yaw rate reference signal generation was implemented. The yaw angle signal is calculated by the following equation:

$$\psi = atan2(\frac{b_y - a_y}{b_x - a_x}), \tag{5.1}$$

where $\vec{a} = [a_x; a_y]$ is the point for which the yaw angle is calculated and $\vec{b} = [b_x; b_y]$ is the next point in the trajectory. The yaw rate signal is calculated as

$$r = \frac{v_{ref}(\psi(i) - \psi(i-1))}{d} \tag{5.2}$$

where $\psi(i)$ is the yaw angle at a point for which the yaw rate is calculated, $\psi(i-1)$ yaw angle, and d is the distance between the two points.

Algorithm 1: Path planning algorithm

```
Result: Path
B ← set of points in 2D representing blue cones;
Y ← set of points in 2D representing yellow cones;
k ← 1;
Path(k) ← starting point;
while True do
    instructions;
    if $k = 1$ then
        b ← B(argmin(‖B - Path(k)‖));
        y ← Y(argmin(‖Y - Path(k)‖));
    else
        p ← line defined by normal vector(Path(k) - Path(k-1)) and
         point Path(k);
        $\rho$ ← half-plane degined by lin $p$ and direction of vector
         (Path(k) - Path(k-1));
        $\hat{B}$ ← B $\cap \rho$ ;
        $\hat{Y}$ ← Y $\cap \rho$ ;
        if $\hat{B} = \emptyset$ or $\hat{B} = \emptyset$ then
            break ;
        else
            b ← B(argmin(‖$\hat{B}$−Path(k)‖));
            y ← Y(argmin(‖$\hat{Y}$−Path(k)‖));
        end
    end
    Path(k+1) ← mean(b+y);
    k ← k+1 ;
end
```

5.2 Stanley control law

Due to easy implementation and its robustness, it was decided that the Stanley control law will be used. The Stanley control design has proved to be useful as it was implemented and tested on vehicle competing in the DARPA Grand Challenge 2005. This vehicle was the only one out of 40 competitors not to hit an obstacle or miss a gate, and had the fastest course completion time.[5]

5.2.1 Lateral control

Lateral control is used to control vehicle steering. It calculates the angle of the front wheels based on the following equation[5]:

$$\delta(t) = (\psi(t) - \psi_{ss}(t)) + \arctan(\frac{ke(t)}{k_{soft} + v(t)}) + \\ + k_{yaw}(r_{meas}(t) - r_{traj}(t)) + k_{steer}(\delta_{meas}(t - \delta t) - \delta meas(t)) \tag{5.3}$$

where $\delta(t)$ is the angle of the front wheels with respect to the vehicle,r_{traj} is the yaw rate for the trajectory, r_{meas} measured yaw rate, $\psi(t)$ the yaw angle, k_{yaw}, k_{steer}, and k_{soft} are tunable constants, and ψ_{ss} is a steady state yaw which can be found by the equation:

$$\psi_{ss} = \frac{mv(t)r_{traj}(t)}{C_y(1 + \frac{a}{b})} \tag{5.4}$$

where m is the weight of the vehicle, C_y is the lateral tire stiffness, a is the distance of the front axle from the vehicle center of gravity, and b is the distance from of the rear axle from the vehicle center of gravity.

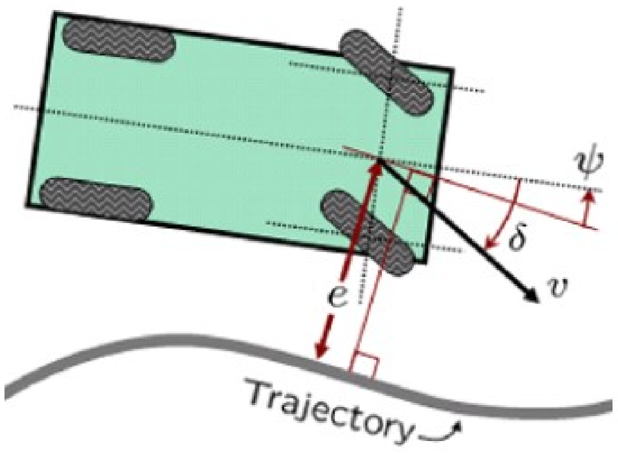


Figure 5.1: Kinematic model of the vehicle[5]

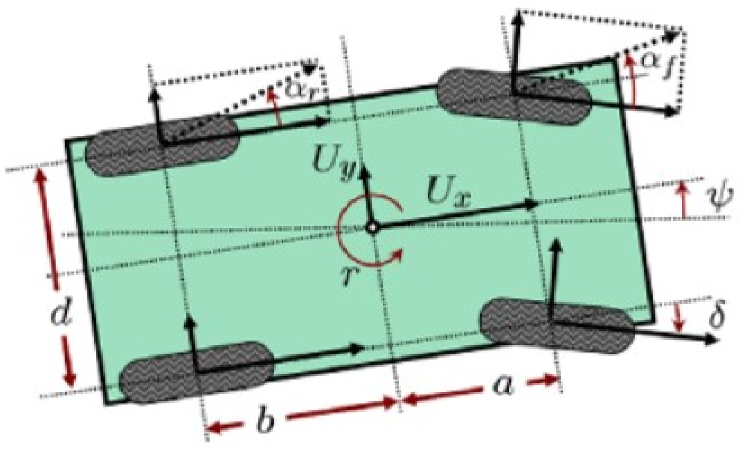


Figure 5.2: States of the vehicle[5]

This control rule implementation is taken from Automated driving toolbox in Matlab. Because the angle of the front wheels cannot be forced to an exact value and the only interaction with this angle is through the steering wheel, negative feedback control with P regulator was implemented. In the figure below results can be seen.

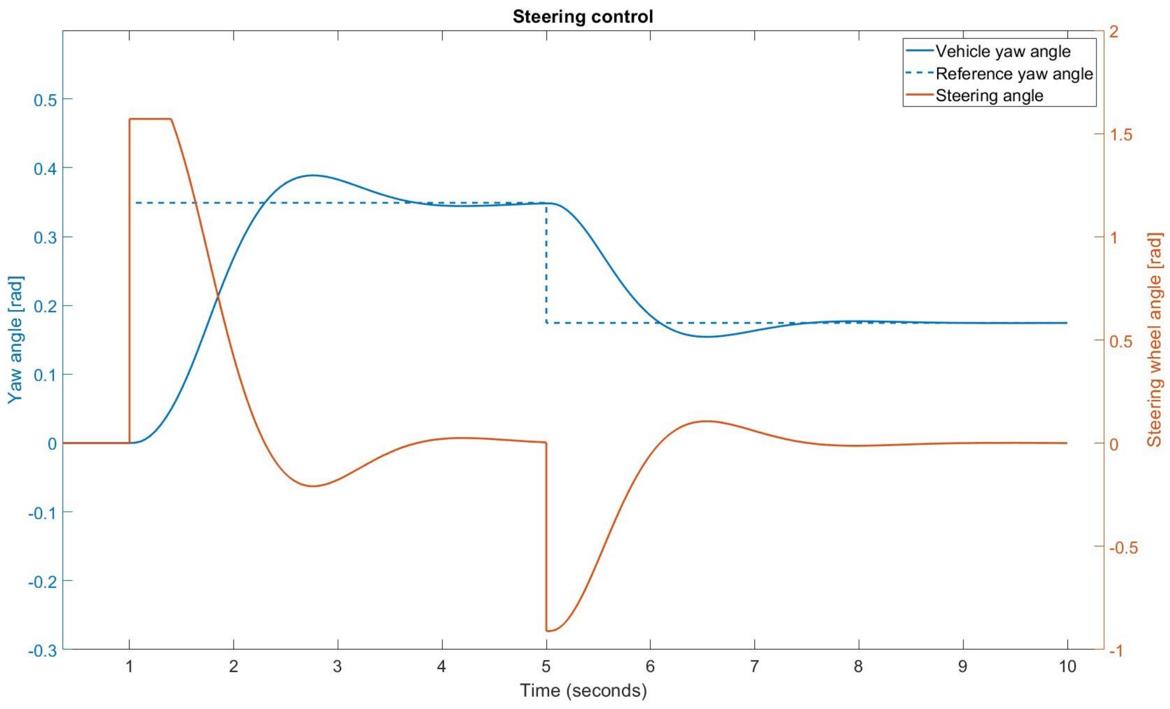


Figure 5.3: Front wheels angle controller

5.2.2 Lateral control improvement

It was found during testing that this algorithm could be simplified. $\psi_{ss} = \psi_{ref}$, and $k_{steer} = 0$ This was done and lateral control rule changes to:

$$\delta(t) = (\psi(t) - \psi_{ref}(t)) + arctan(\frac{ke(t)}{k_{soft} + v(t)} + k_{yaw}(r_{meas} - r_{traj})) \quad (5.5)$$

this control rule had better overall results in comparison to the Stanley lateral control. These results are further presented in the next chapter. This may be caused by either a low constant speed or a wrong choice of constants.

5.2.3 Longitudinal control

The longitudinal control follows simple rules:

$$a = \begin{cases} 0.1(v_{ref} - v_{meas}) & \text{if } 0 \leq v_{ref} - v_{meas} < 10 \\ 1 & \text{if } vref - v_{meas} \geq 10 \\ 0 & \text{if } vref - v_{meas} < 0 \end{cases} \quad (5.6)$$

$$b = \begin{cases} 0.2 & \text{if } \leq -0.1 \\ 0 & \text{if } vref - v_{meas} > -0.1 \end{cases} \quad (5.7)$$

where v_{ref} is speed reference and v_{meas} is measured speed, and a is press ratio of accelerator pedal and b is press ratio of brake pedal.

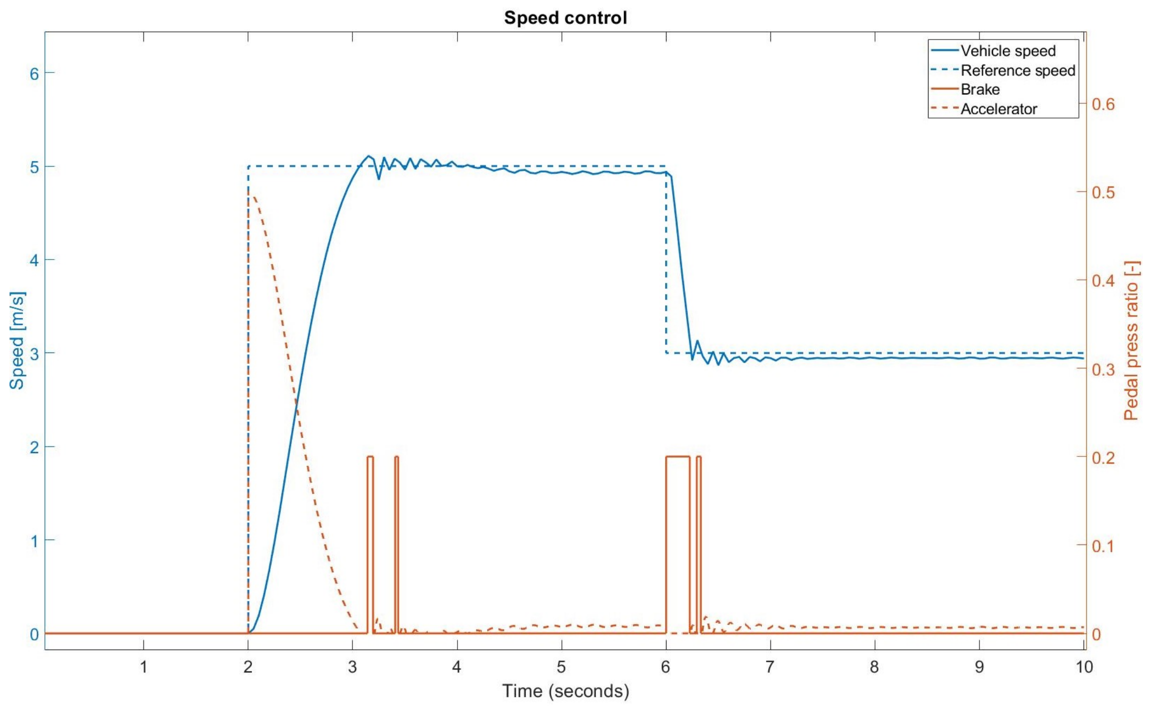


Figure 5.4: Speed controller

Chapter 6

Experiments

In this chapter results of path tracking algorithm are presented. Virtual reality implementation and path tracking algorithm were tested on different tracks. These experiments show, that even though I thought that Stanley control law was performing better if $k_{steer} = 0$ and $\psi_{ss} = \psi_{ref}$, it is only partially true. Surprisingly this weakness of perfomance was revealed on simple and monotonous tracks such as circle track. Regarding this fact, it seems that using Stanley control law (5.3) rather than law(5.5) is necessary.

6.1 Advanced track

First experiment took place on advanced track with challenging setup.

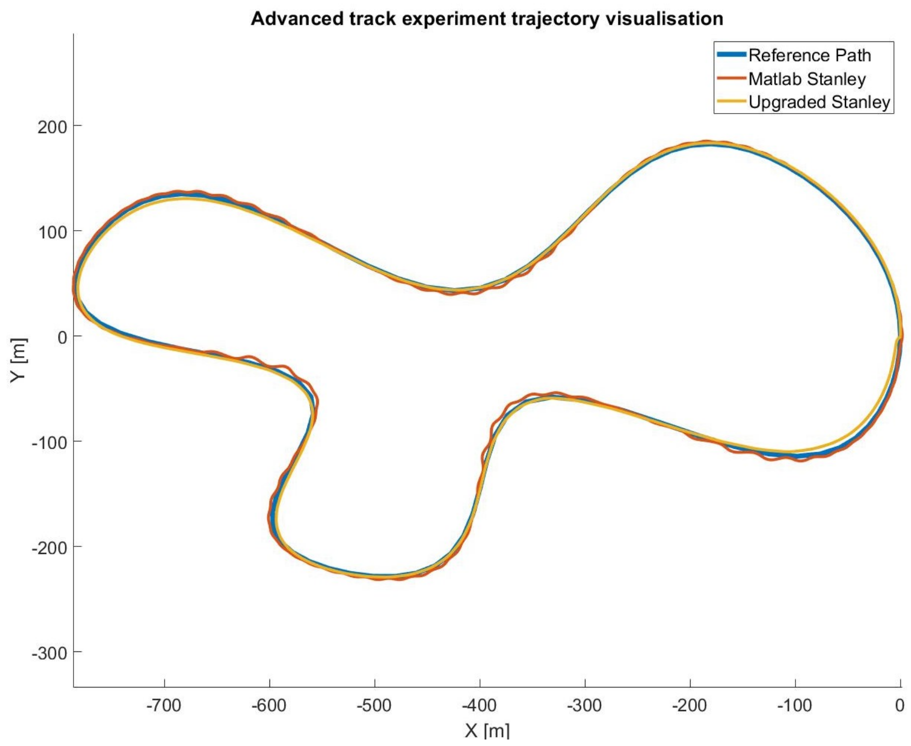


Figure 6.1: Advanced track experiment - trajectories

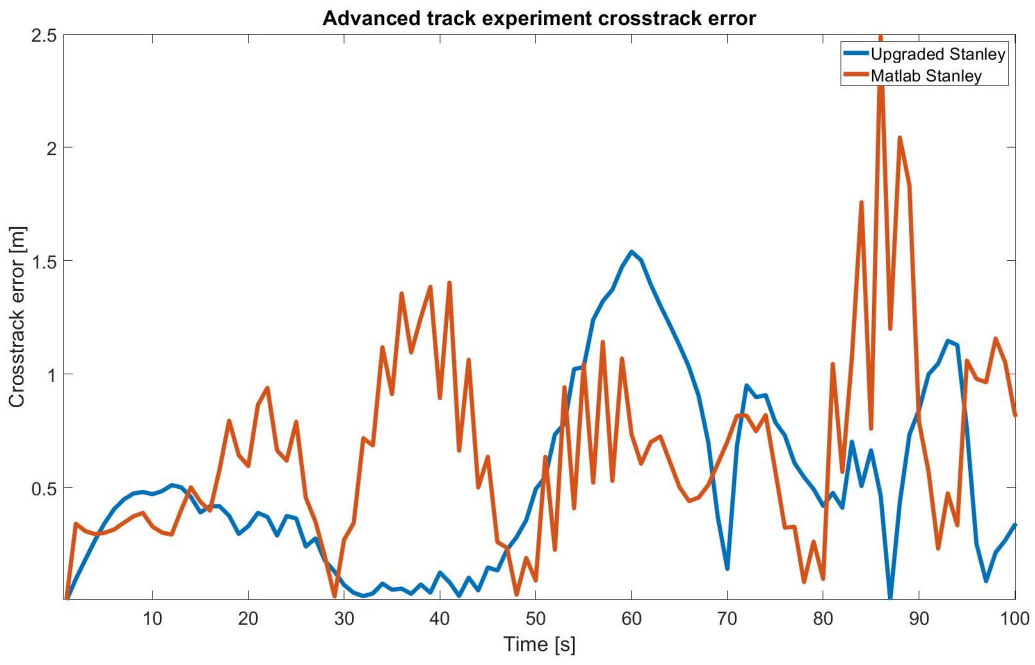


Figure 6.2: Advanced track experiment - crosstrack errors

In this experiment improved Stanley control was performing better.

6.2 Route approach

Another experiment tests approach to the trajectory in a case, when vehicle is not on the starting line. It is easily shown, that here the original Stanley control law 5.3 has better performance.

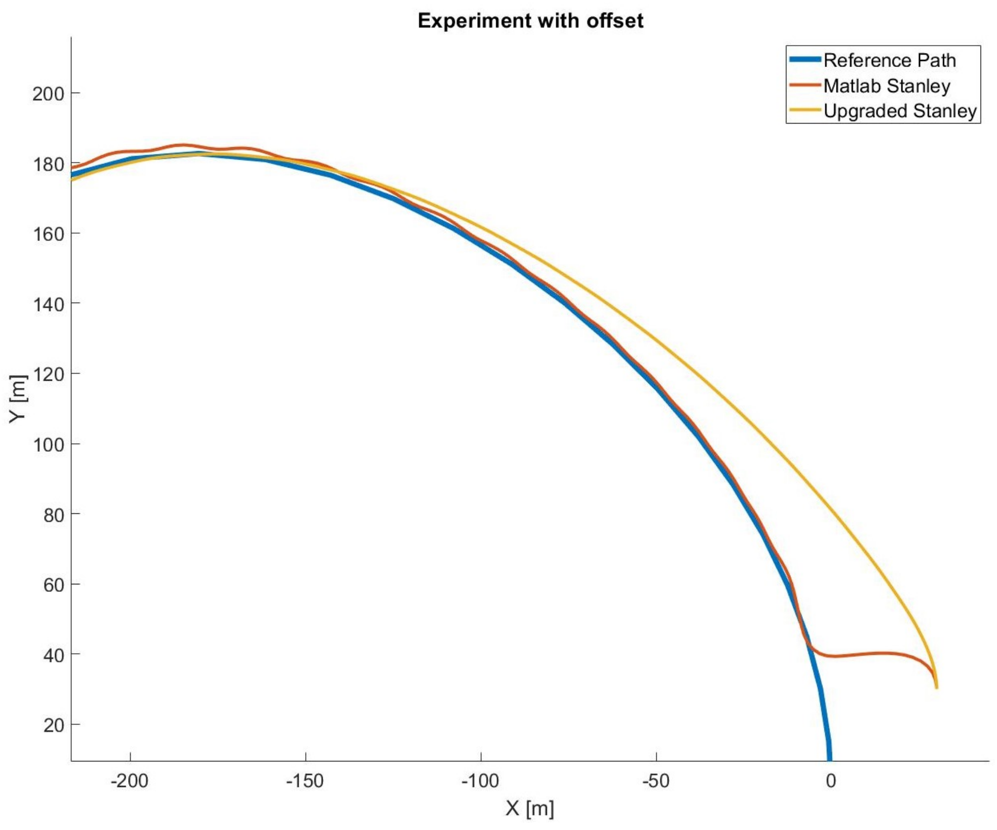


Figure 6.3: Route approach trajectories

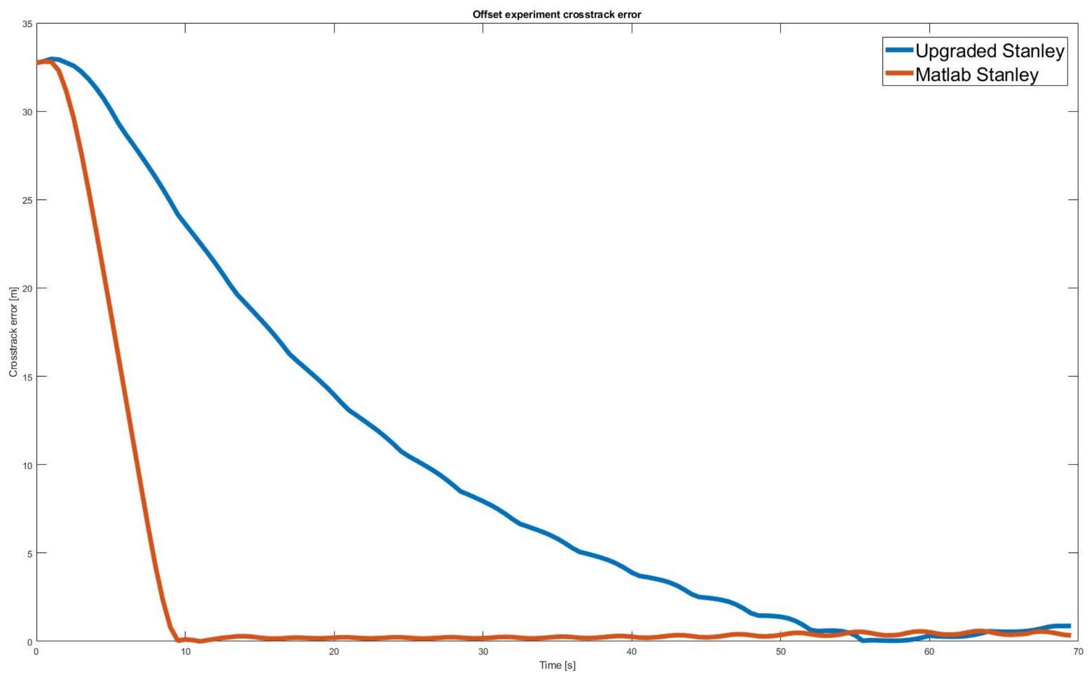


Figure 6.4: Route approach crossroad errors

6.3 Circle track

Second experiment takes place on track with circle profile with diameter of 300m. It is probably the most basic experiment. This experiment shows, that despite good performance on advanced track, Upgraded Stanley algorithm 5.5 is not best choice.

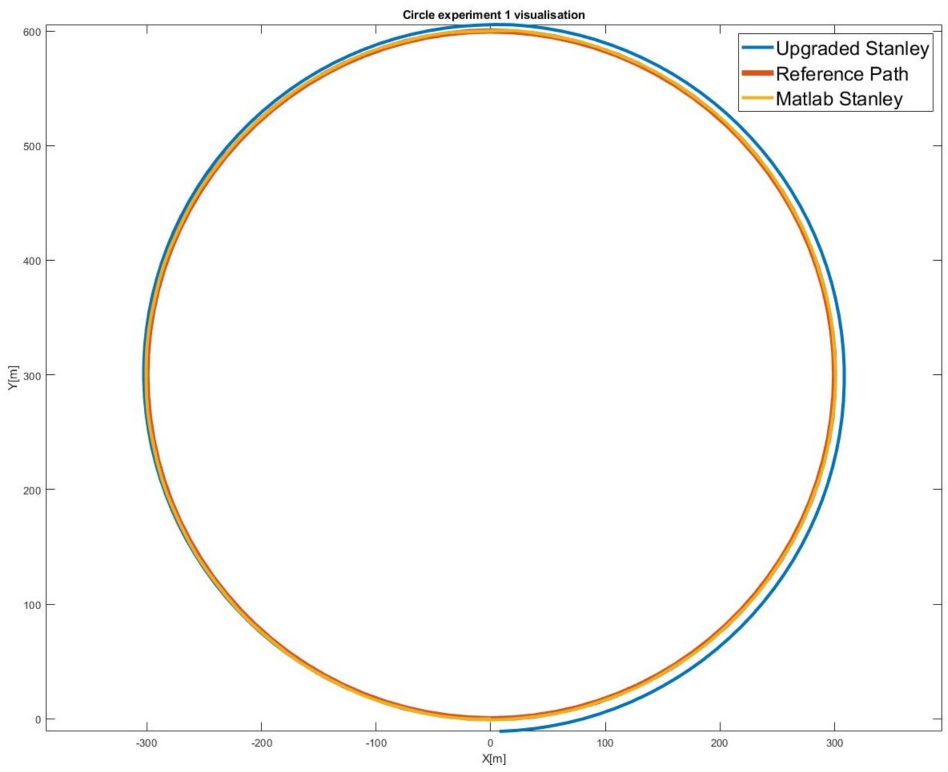


Figure 6.5: Circle track experiment trajectories

6.4 Star track

In experiment with track of a star shape concerns about performance of Upgraded Stanley algorithm 5.5 are confirmed. As this algorithm is not able to follow path which is unacceptable.

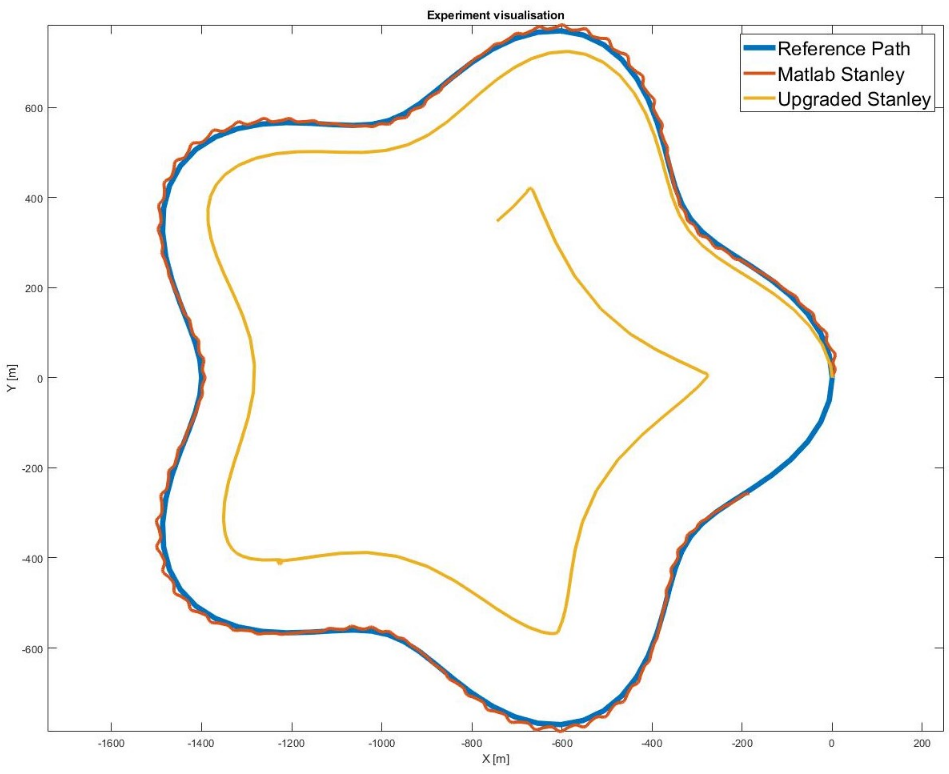


Figure 6.6: Star track experiment trajectories

Chapter 7

Results

- In chapter 3 I did research of possible solutions for implementing virtual reality scenarios and chose a new and challenging option - Ansys VRXPERIENCE
- In chapter 4 I successfully implemented a virtual reality scenario with a vehicle model similar to eForce DV.01. A large concrete plane was generated as well as a track marked with cones.
- In chapter 5 I got familiar with a path planning algorithm, and the Stanley control law. Also, an improvement of this path tracking algorithm was suggested.
- In chapter 6 I tested a virtual reality implementation, the Stanley control law and the suggested improvement to the algorithm. Unfortunately, this algorithm did not prove to be working and therefore will not be used in future work.

Chapter 8

Conclusion

In this thesis all of the objectives were met. However another work should be done in the future. Next step would definitely be to automatize the process of creating a track, because adding each cone manually is tedious. Also the vehicle dynamics should be more detailed, for example by implementing a suspension system, which was not included in this thesis. Finally, connecting the simulating tool to the DV.01 pipeline should be done to test all capabilities of Ansys VRXPERIENCE, and also the DV.01 pipeline itself.

Appendix A

Bibliography

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini *Vehicle dynamics*, Duisburg 2014

[2] Hans B. Pacejka *Tire and Vehicle dynamics*, The Netherlands 2012

[3] Robert Bosch GmbH *Bosch automotive handbook*, Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers [distributor], 2007.

[4] Marek Boháč *Design of Control System for an Autonomous Racecar*,Bachelor's thesis, Czech Technical University in Prague, 2020

[5] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, *Autonomous Automobile Trajectory Tracking for Off-Road Driving:Controller Design, Experimental Validation and Racing*, American Control Conferecence 2007, pp. 2296-2301, 2007

[6] Gillespie, T. D., *Fundamentals of Vehicle Dynamics*, Society of Automotive Engineers, Warrendale, PA 1992

[7] Formula Student Germany, *Formula Student Rules 2020* `https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf`

[8] ANSYS, Inc. *ANSYS VRXPERIENCE Driving Simulator powered by SCANeR™ 2021R1 - User's Guide*

[9] `www.rfpro.com`

[10] `www.carsim.com`

[11] `www.dspace.com`

[12] `www.ipg-automotive.com`

[13] `www.ansys.com`

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Svoboda Jan**	Personal ID number:	**474750**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor's thesis details

Bachelor's thesis title in English:

Autonomous vehicle virtual verification platform development

Bachelor's thesis title in Czech:

Vývoj verifikační platformy pro autonomní vozidla

Guidelines:

The development and verification process of autonomous vehicle is time costly and expensive. The virtual development and verification environment have potential to significantly cut development process related cost. The trajectory tracking algorithm for autonomous car will be designed and tested based on such virtual platform in this thesis. Following points will be addressed:

1. Get familiar with vehicle virtual testing platforms.
2. Implement suitable test scenario in selected software.
3. Implement reference trajectory tracking algorithm.
4. Verify the control algorithm using virtual testing scenarios.

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers

Name and workplace of bachelor's thesis supervisor:

Ing. Tomáš Haniš, Ph.D., Department of Control Engineering, FEE

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

Date of bachelor's thesis assignment: **20.01.2021** Deadline for bachelor thesis submission: ______

Assignment valid until:
by the end of summer semester 2021/2022

Haniš	[illegible]	[illegible]
Ing. Tomáš Haniš, Ph.D.	prof. Ing. Michael Šebek, DrSc.	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.

5.5.2021	Svoboda
Date of assignment receipt	Student's signature

© ČVUT v Praze, Design: ČVUT v Praze, VIC