Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Design of Control System for an Autonomous Racecar

Marek Boháč

Supervisor: doc. Ing. Martin Hromčík, Ph.D.
August 2020

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Boháč Marek** Personal ID number: **465967**

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:

Design of a Control System for an Autonomous Racecar

Bachelor's thesis title in Czech:

Návrh řídícího systému autonomního závodního vozidla

Guidelines:

The goal is to develop new functionalities for the upcoming auton omous racing car of the eForce team of the FEE CVUT, related to trajectory planning and vehicle control. Specific tasks follow below.
1) Create a mathematical model capturing key features of the kinematics and dynamics of racecar.
2) Create a simulation environment for design and verification of the control system.
3) Design a control system for longitudinal and lateral trajectory tracking.
4) Implement the control system on the racecar operating system.
5)Verify the controller experimentally.

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
[4] Franklin, Gene F.; Powell, J. David; Emami-Naeini, Abbas, Feedback Control of Dynamic Systems, Global Edition, Pearson Education Limited, 2019, ISBN: 9781292274522

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Martin Hromčík, Ph.D., Department of Control Engineering, FEE

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

Date of bachelor's thesis assignment: **31.01.2020** Deadline for bachelor thesis submission: **14.08.2020**

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

doc. Ing. Martin Hromčík, 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.

.	
Date of assignment receipt	Student's signature

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, doc. Ing. Martin Hromčík, Ph.D., for his advice while writing this thesis. Special thanks belong to Ing. Jan Čech, Ph.D. for his work for eForce Driverless as Faculty Advisor. Without his advice and guidance of the team, this thesis would not be possible. Also, my thanks go to Ing. Tomáš Haniš, Ph.D. for initial guidance on the topic and his advice and help with adapting car for RC car as the last step of this thesis.

Secondly, I would like to express my gratitude to all my previous lecturers and the Czech Technical University itself for helping me gain invaluable academic and practical experience. Last but not least, I would like to thank the eForce Driverless team for their hard work and collaboration with me on this project.

Declaration

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

Prague, August 13, 2020

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Praze, 13. srpna 2020

Abstract

This thesis describes the design, development, and implementation of the first autonomous racecar of team eForce Driverless for Formula Student competition. It is mainly focused on motion planning algorithms and trajectory tracking algorithms and related topics such as vehicle dynamics, but it describes other systems as well. The motion planning algorithm consists of a path planning algorithm and a speed reference generator. Trajectory tracking uses Stanley control laws for lateral control and PI regulator for longitudinal control. As part of the thesis, the simulation environment was designed as well. The solution designed in this thesis was then used in Formula Student Online competition and implemented on a smaller platform than the actual racecar to safely test its behavior.

Keywords: autonomous vehicle, autonomous racing, trajectory tracking, motion planning, vehicle dynamics, lateral control, longitudinal control, formula student, formula student driverless

Supervisor: doc. Ing. Martin Hromčík, Ph.D.

Abstract

V této práci se zabývám popisem návrhu, vývoje a implementace řídícho systému pro první autonomní závodní formuli týmu eForce Driverless pro soutěž Formula Student. Práce se zaměřuje zejména na plánování pohybu, řízení vůči planované trajektorii a s tím související témata jako je jízdní dynamika vozidel, nicméně popisuje i jiné části systému. Plánování pohybu spočívá v nalezení cesty a následném vygenerování reference rychlosti pro každý bod. Podélné řízení je poté zajištěno pomocí PI regulátoru a příčné řízení zajišťuje implementovaný řídící zákon vyvinutý na Stanfordské univerzitě pro DARPA Challenge a auto pojmenované Stanley.

Keywords: autonomní vozidlo, autonomní závodění, sledování trajektorie, plánování pohybu, dynamika vozidla, příčné řízení, podélné řízení, formula student, formula student driverless

Title translation: Návrh řídícího systému autonomního závodního vozidla

Contents

Figures

Tables

Chapter 1
Introduction

This thesis describes the development of a control system for the first fully autonomous racing car for the Formula Student team eForce Driverless. Formula Student competition is an international competition founded in 1981 by Society of Automotive Engineers (SAE). Since the foundation, the competition spread all over the world. It was first held in Europe in 1998. In 2017, a new vehicle class was added to the competition introducing Driverless Vehicle. In 2019, team eForce Driverless was founded as the second Formula Student (FS) team on the Faculty of Electrical Engineering. Later that year the biggest competition in Europe, Formula Student Germany (FSG), made a strategic announcement about merging all three classes (Combustion Vehicles, Electric Vehicles, and Driverless Vehicles) to a single class, meaning every team willing to attend FSG competition will need to be capable of autonomous racing otherwise there will be penalization. This announcement enhanced a need for an autonomous racecar capable of attending FSG competition on the Faculty of Electrical Engineering.

1.1 Platforms

For multiple reasons, the developed system was adapted for multiple platforms. The eForce driverless ancestor, eForce FEE Prague Formula, provided FSE.07 car, which is the 7th generation of formula racecar that has been built together with a knowledge the team had already gathered to speed up the development process. eForce Driverless can only focus on adapting the car and developing the autonomous system. The new car is named DV.01.

Unfortunately, due to the world pandemic in 2020, DV.01 was not finished on time, and algorithms could not be validated on the racecar. However, as the whole world adapted to the pandemic, so did Formula Student competition. Formula Student Online (FSO) was founded and developed its own simulator [3]. Furthermore, because the entire system is running on Robot Operating System (ROS), thus the entire autonomous system can be easily adapted and validated on another platform. The suitable platform was found in the Department of Control Engineering, RC car Losi Desert Buggy adapted for autonomous driving with the identical computational unit and primary camera.

Figure 1.1: FSE.07 car being rebuilt into DV.01 [1]

1.1.1 DV.01

The main control unit for the autonomous system is NVIDIA Jetson Xavier with ROS as a high-level framework [4]. NVIDIA Jetson Xavier is used solely for the autonomous system. Other systems (safety systems and fundamental systems) use their Microcontroller Unit (MCU), either adapted or developed. Additional two NVIDIA Jetson Nano computational units are used for higher computational performance.

The car is equipped with multiple sensors. The main sensors for the perception are three stereo-cameras - a Stereolab ZED and two Intel Realsense D435. The car is also equipped with an Ouster OS1-64 LiDAR. Stereolab ZED stereo-camera is mounted on the top of the main hoop, and two Intel Realsense D435 are mounted on sides of the front wing to provide a more detailed view of cones closer to the car and those in tight corners. The Ouster OS1-64 LiDAR is used for the more precise location of the cones used to mark the track. Figure 1.2. The odometry is provided by SBG systems Ellipse-N Inertial Navigation System (INS).

As stated in the rules [5] car must be equipped with Emergency Braking System (EBS) for safety reasons. EBS is a pneumatic system attached to the brake pedal. Two Lenze Schmidhauser DCU 60/60 motor controllers are used. Each controller has two channels for the two motors, thus allowing all-wheel drive. Controllers are connected via Controller Area Network (CAN bus). Controllers can receive a set-point of torque for each motor. The autonomous system will use the motor to break, but the car is also equipped with redundancy to the EBS. Three SAVOX SB-2230SG servos connected to the brake pedal. For steering, a universal power steering kit by Kartek was

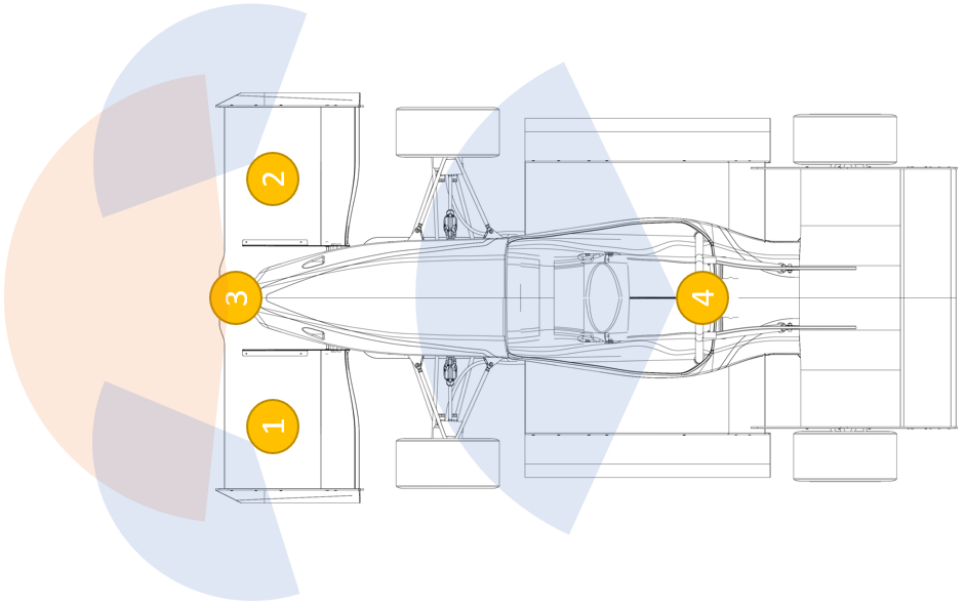


Figure 1.2: Sensors placement - Top view
1, 2 - Realsense; 3 - OS-64; 4 - ZED

used. Table 1.1

1.1.2 RC car

As well as DV.01, the RC car is equipped with NVIDIA Jetson Xavier. Only a Stereolab ZED camera is used on this platform, and it lacks LiDAR as well. Odometry is provided by the Global Navigation Satellite System (GNSS) receiver and inertial measurement unit (IMU) Navio2. Raspberry Pi 3 is used to control motors and steering. These are connected to the NVIDIA Jetson Xavier via serial communication. Table 1.2

1.2 Formula Student Driverless Challenges

In this thesis, I will focus on developing a system for the 2020 edition of FSG. The competition consists of four main dynamic events. In general, it can be stated that the track is marked with four types of cones - a blue, a yellow, an orange, and a big orange. Blue cones are used to mark the left side of the track; yellow cones are used to mark the right side of the track, orange (both small and big) are used to mark the start or the end of the track. Details about cones can be find in [6, p. 12-14]. FSO has almost the same rules as FSG. The difference is in the simplified state machine and only two but the most challenging dynamic events (autocross and track-drive).

There are four main dynamic events - skid-pad, acceleration, autocross, and track-drive. The skid-pad track consists of two pairs of concentric circles whose outer diameter is 21.25 meters, and the inner diameter is 15.25 meters.

Name	Manufacturer	Description
Jetson AGX Xavier	NVIDIA	The main computational unit
Jetson Nano	NVIDIA	Two additional computational units used to preprocess images
ZED Stereo Camera	Stereolabs	Main camera placed on the top of the car (mainhoop)
Realsense D435	Intel	Two additional cameras on outer sides of the front wing
OS1-64	Ouster	LiDAR placed in the middle of the front wing
Ellipse-N	SBG	Odometry sensor
DCU 60/60	Lenze Schmidhauser	Two dual channel motor-controllers
Universal Power Steering Kit	Kartek	Steering servo
SB-2230SG	SAVOX	Redundancy break for EBS, can be used by the autonomous system

Table 1.1: DV.01 hardware

The two of the pair are placed 18.25 m apart from each other. The car makes two turns on each side, starting on the right side. Only the second turn on each side is timed [5, p. 120-122]. The acceleration track is a straight line with a length of 75 meters from the starting line to the finish line [5, p. 122]. The autocross and the track-drive track layout are not known in advance. It is the hardest part of the driverless challenge. However, there are some constraints to the track. It is always closed-looped track consisting of straight sections with a maximum length of 80 m, constant turns up to the diameter of 50 m, or hairpin turns with minimum outer diameter of 9 m. One lap is approximately 200 to 500 m. Autocross and track-drive use the same track; the only difference is that autocross consists of one lap, and track-drive consists of ten laps.

Because the track of skid-pad and acceleration is known in advance, these are considered a minor issue. A defined path can be used, and the trajectory tracking algorithm can remain unchanged. Most of this thesis and efforts will be inserted into navigating trough an unknown environment as it is in the autocross and track-drive.

Figure 1.3: Adapted RC car Losi Desert Buggy [2]

Name	Manufacturer	Description
Jetson AGX Xavier	NVIDIA	The main computational unit
ZED Stereo Camera	Stereolabs	Main camera placed on the top of the car
Raspberry Pi 3	Raspberry Pi	Motor controller and base for Navio2
Navio2	Emlid	Odometry sensor

Table 1.2: RC car hardware

Chapter 2

Objectives

Based on the problem definition in the previous chapter, these objectives arise.

The main goal for the first system developed is the capability to finish the race and stability on all tracks. With respect to the results in previous seasons, the hardest challenge for most teams is to finish the race. This thesis, neither the team efforts are focused on the development of a highly optimized system. Main goals can be summarized as

- Implementation of the motion planning algorithm
  - Implementation of path planning algorithm
  - Development of speed reference generator
- Design and implementation of the trajectory tracking algorithm
  - Design and implementation of lateral controller
  - Design and implementation of longitudinal controller
- Development of the testing framework in ROS
  - Vehicle dynamics simulation
  - Perception simulation
  - Steering mechanism simulation
  - All simulations must include noise in its output
- Verification of the system using ROS framework using simulation
  - Verify all simulation parts individually
  - Verify trajectory tracking using full simulation framework
- Adaptation of the system for FSO competition
  - Adaptation of the system for the FSDS
  - Adaptation to the rules changes
  - Validation of the system architecture using FSDS

- Adaptation of the system for RC car and validate it under real-world conditions
  - Adaptation of the system to the HW of the RC car
  - Validation of the system with experiments in the real world

Chapter 3

Vehicle dynamics

Vehicle dynamics is a well-described problem that has been studied for many decades. Multiple mathematical models were derived throughout the years, varying in complexity and fidelity. Multiple simulators with high fidelity models already exist as well.

In this thesis, nonlinear single-track model with 3 degrees of freedom (DOF) is used [7]. Although more complex models with higher fidelity exist and can be implemented for the sake of this thesis more complex model is not necessary. Also, a more complex model requires higher computational performance or specialized software, which is contradictory to the objective of creating simple real-time simulation inside of the ROS framework.

Tire dynamics will be modeled using simplified Pacejka Magic Formula [8].

3.1 Nonlinear single-track model

A classical single track model is used. Car has two single axes. Front axis has distance from the Center of Gravity (CoG) l_f, respectively l_r is used for the rear axis. Only front axis can be steered by steering angle of δ. Motion of the vehicle is considered planar and the vehicle is assumed to be rigid body. The vehicle has a mass m and a moment of inertia I_z and it is represented as a single point in its CoG. Both tires on single axis are assumed to act same and are virtually represented by tire in the center of axis. The aligning torque is neglected. This 3 DOF model is expressed in Newton-Euler equations as follows.

$$F_x = F_{x,r} + F_{y,f}\cos\delta - F_{y,f}\sin\delta - F_{x,aero} \tag{3.1}$$

$$F_y = F_{y,r} + F_{y,f}\cos\delta + F_{x,f}\sin\delta \tag{3.2}$$

$$M_z = l_f F_{x,f}\sin\delta + l_f F_{y,f}\cos\delta - l_r F_{y,r} \tag{3.3}$$

$F_{x/y,f/r}$ is force acting on the f front or r rear axis in direction of axis x or y with respect to the tire coordinate frame. These forces are transformed using steering angle δ to the car coordinate frame resulting in combined forces F_x and F_y forces acting on the CoG of the car in direction of axis x and y respectively with respect to the car coordinate frame. M_z represent torque

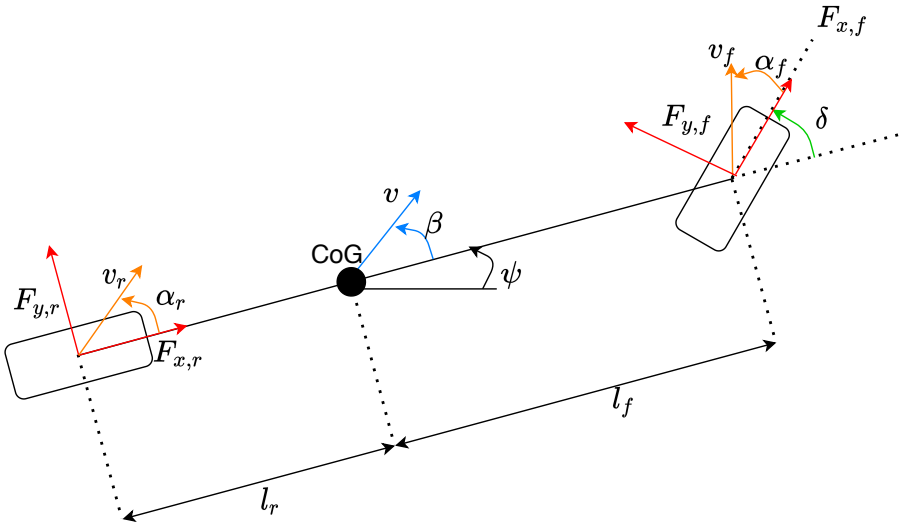


Figure 3.1: Single track model coordinates

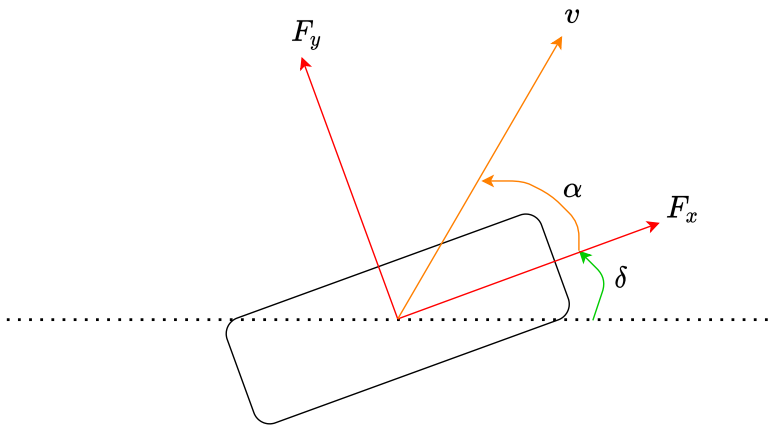


Figure 3.2: Tire model

about z-axis. $F_{x,aero}$ is aerodynamic drag of the vehicle.

$$F_{x,aero} = \rho A_{ref} C_D v^2 \tag{3.4}$$

ρ is the air density, A_{ref} is the aerodynamic area, C_D is the drag coefficient and v is the velocity of the vehicle.

The equations are then further transformed to following state-space

$$\ddot{x} = \frac{F_x}{m} \tag{3.5}$$

$$\ddot{y} = \frac{F_y}{m} \tag{3.6}$$

$$\ddot{\psi} = \frac{M_z}{I} \tag{3.7}$$

$$\dot{\omega_f} = \frac{\tau_f - RF_{x,f}}{J} \tag{3.8}$$

$$\dot{\omega_r} = \frac{\tau_r - RF_{x,r}}{J} \tag{3.9}$$

where τ_f and τ_r is torque on front and rear wheel respectively. Additional two states are added to represent state of each wheel. These are used to calculate slip ratio. Vehicle slip angle β is calculated as follows

$$\beta = \arctan \frac{\dot{y}}{\dot{x}} \tag{3.10}$$

Magnitude of vehicle's velocity and velocity with respect to the world coordinates

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{3.11}$$

$$\dot{X} = v \cos(\psi + \beta) \tag{3.12}$$

$$\dot{Y} = v \sin(\psi + \beta) \tag{3.13}$$

3.2 Tire modeling

Mathematical description of the force interaction between the surface and the tire is a major challenge in vehicle modeling. Due to the elasticity of the tire, a mathematical description is very complicated. Hans B. Pacejka was studying the tire dynamics his entire life. As a result, he invented the Pacejka magic formula [8]. The formula is based on pure empirical methods and has no connection to the actual physics of the tire. However, it is reasonably accurate and commonly used in complex high-fidelity models as well as in the games industry for its low processing time.

There are two models differing in the number of parameters used to describe the tire. In this thesis, tires are represented by four parameters A, B, C and D. Pacejka magic formula in this case is

$$F(F_z, \alpha) = F_z D \sin\{C \tan^{-1}\left[B\alpha - E(B\alpha - \tan^{-1} B\alpha)\right]\} \tag{3.14}$$

where α is slip parameter and F_z is downforce acting on the tire. The base formula is same for both longitudinal and lateral dynamics but parameters for each of them differ.

Slip parameter for longitudinal variant is called slip ratio and it is calculated as follows

$$\lambda = \frac{\omega R - v_x}{|v_x|} \tag{3.15}$$

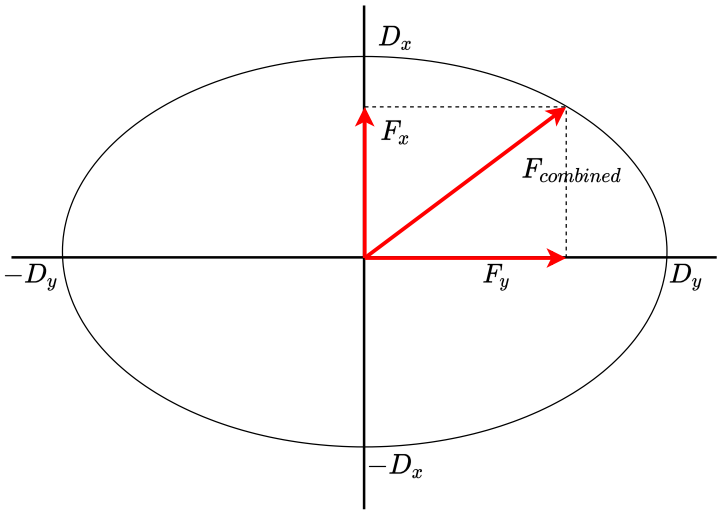


Figure 3.3: Friction ellipse for normalized forces

where ω represents angular speed of wheel, v_x is velocity of tire in direction of axis x with respect to the tire coordinate frame and R is wheel radius. The calculation of the wheel slip ratio is a major issue with Pacejka magic formula when it comes to lower speeds. As seen in the equation, slip ratio has a singular value of $v_x = 0$ thus being unreliable for low speeds. This is avoided using another method to calculate slip ratio which is applied for velocities near zero. For velocities $v < \epsilon$ following is applied

$$\lambda = \frac{2(\omega R - v_x)}{\epsilon + \frac{v_x^2}{\epsilon}} \tag{3.16}$$

The slip parameter for lateral variant is called slip angle and it is calculated as follows

$$\alpha = \delta - \tan^{-1} \frac{v \sin\beta + l_f \dot{\psi}}{v \cos\beta} \tag{3.17}$$

Since rear axis is not steerable, first element of the sum is always zero for calculation of the slip angle of the rear tire. There is no singularity in this case thus this equation is valid all the time.

3.3 Friction circle

The friction circle, also referenced as Kamm's circle or Friction ellipse or circle of forces, is an ellipse expressing maximum tire traction. The magnitude of combined forces of traction is limited by the vertical force acting on the tire.

$$F_z = mg + \rho A_{ref} C_L v^2 \tag{3.18}$$

$$F = \sqrt{\frac{F_x^2}{D_x^2} + \frac{F_y^2}{D_y^2}} \leq \mu F_z \tag{3.19}$$

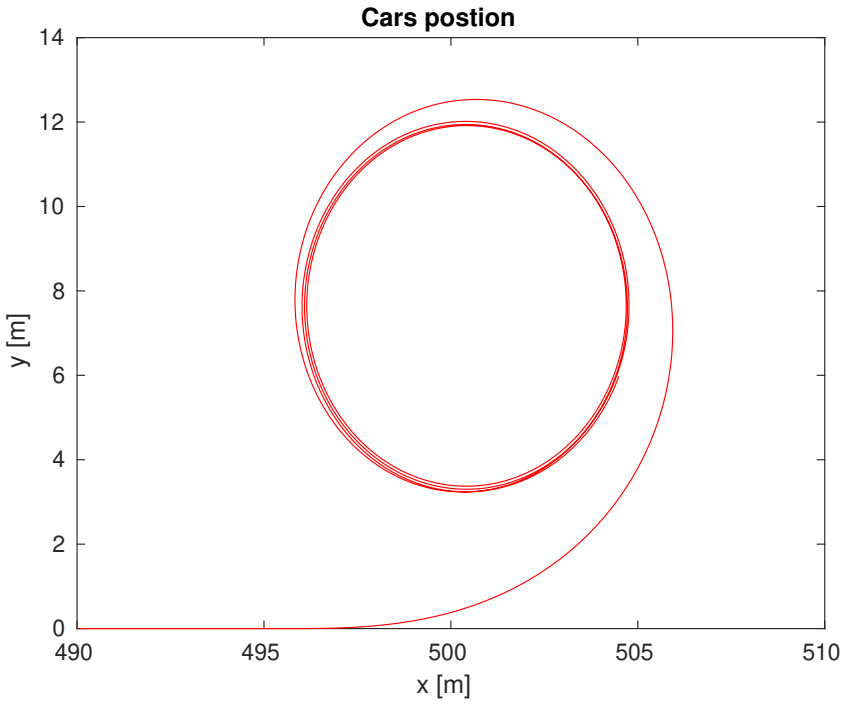


Figure 3.4: Car position with respect to global coordinates

μ is friction coefficient for tire-road interaction, D is a parameter from Pacejka magic formula. Vertical force is calculated using vehicle mass and aerodynamic lift.

When combined force

$$F = \sqrt{\frac{F_x^2}{D_x^2} + \frac{F_y^2}{D_y^2}} \tag{3.20}$$

exceeds friction circle, longitudinal force and lateral force must be scaled to match actual tire performance. Forces are scaled using algorithm described in [9].

3.4 Model verification

To verify simulation, a simple experiment is proposed. The steady car receives a step of torque command. The command is constant for a few seconds, and then torque request is reset. After a few seconds, step command is sent to the steering. Results are in Figures 3.4, 3.5 and 3.6

Based on these results, it is verified that the car behaves as expected.

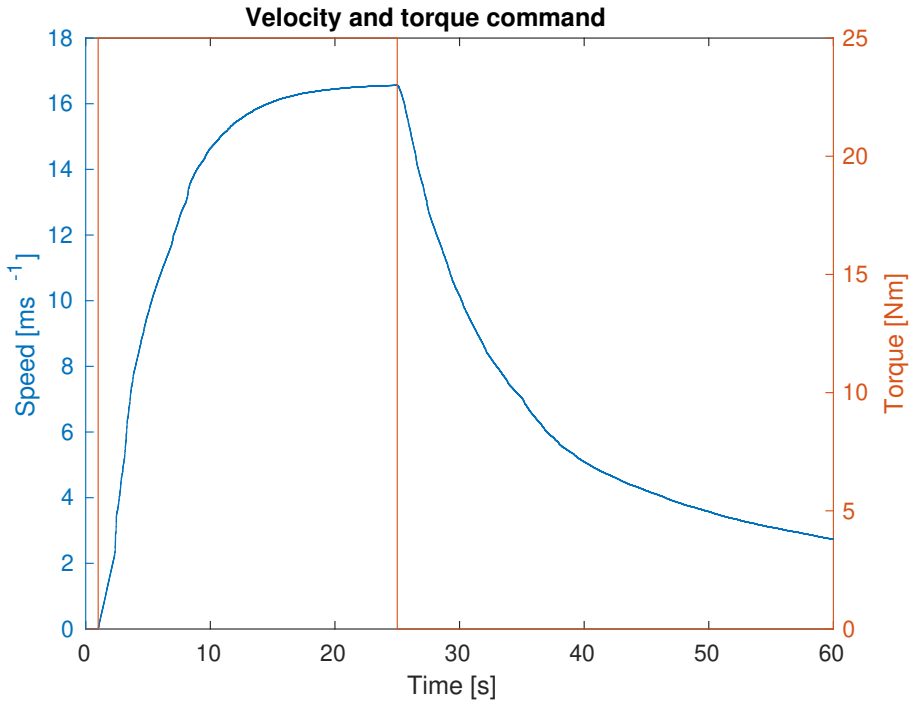


Figure 3.5: Torque command for each wheel and velocity response

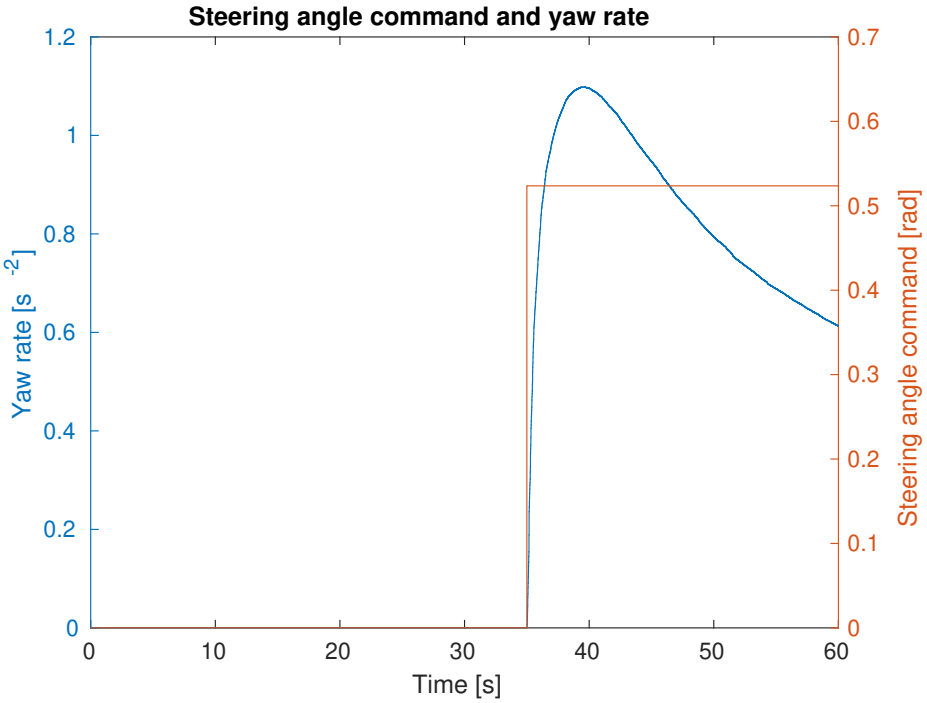


Figure 3.6: Steering step command and yaw rate response

Chapter 4

Motion Planning

The motion planning algorithm is used to generate path and speed reference for the trajectory tracking algorithm. As stated in 1.2 some of the tracks are known in advance. These are not to be planned by this algorithm, but the trajectory will be precomputed and using related topics published by the motion planning node. ROS allows this diversity inside the motion planning node without causing trouble to the rest of the pipeline. Also in 1.2 were specified constraints to the unknown tracks [5]. For the clarity, let me summarize the constraints here again It is always closed-looped track consisting of straight sections with a maximum length of 80 meters, constant turns up to a diameter of 50 meters, or hairpin turns with a minimum outer diameter of 9 meters. One lap is approximately 200 to 500 meters. Autocross and track-drive use the same track; the only difference is that autocross consists of one lap, and track-drive consists of ten laps.

- The track is a closed-loop
- The length of one lap is 200 meters to 500 meters
- Straight segments are no longer than 80 meters
- Constant turns have a diameter up to 50 meters
- Hairpin turns have a minimum outside diameter of 9 meters
- Minimum track width is 3 meters
- Right side is marked with yellow cones
- Left side is marked with blue cones

Many algorithms were designed and described in different papers, but we have decided to develop our own algorithm. The algorithm was developed with respect to these objectives.

- The path must always be found on tracks defined by the rules and summarized at the beginning of this chapter
- The path must be found based on cones coordinates

- The number of cones on each side can differ
- The path must be found on any number of cones (full track or just one frame)
- The speed reference must ensure that the car is able to track the path precisely

The current implementation of the motion planning does not generate race track in terms of the fastest way through the track. The motion planning has two steps. First, it finds an approximation of the center-line, thus creating path. Second, it assigns reference speed to each waypoint base on the dynamics of the vehicle creating trajectory. In the ROS, the motion planning algorithm is implemented in motion_plannning node. It receives the autonomous mission and status, detected cones in the car-fixed coordinates, and the information about the pose of the vehicle at the moment when the cones were captured. It publishes trajectory expressed as waypoints and speed and heading reference at each of them in the global coordinates.

4.1 Path planning

In this section, the path planning algorithm is described. The algorithm was not developed by the author of the thesis, but it must be introduced to let the reader understand the system. Also, implementation in the ROS was made by the author as well as some adjustments to the algorithm itself. The algorithm is written in 1

It can be seen that the algorithm can be run properly only when at least one blue cone and one yellow cone are detected. This was later found an issue when tested on real data using only the primary camera (ZED). In tight turns, the algorithm was never activated due to the limited field of view (FOV) of the primary camera, which was capturing only one side of the track. To improve the path planning behavior and to enable it to find the path in these tight corners when only one side of the cones is visible, we took advantage of the track constraints. Virtual cones were added to the other side of the track in the distance of 3 m to the paired cone.

In order to correctly add cones, at least three cones of one color were required. When only yellow cones were seen, virtual cones were added to the left side. When only blue cones were seen, virtual cones were added to the right side. The first virtual cone was added in the direction of a vector perpendicular to the vector from the first actual cone to the second actual cone. The second cone was added in the direction of the sum of the vector from the second actual cone to the first and third. The third cone was added similarly to the first cone in the direction of the perpendicular vector from the second actual cone to the third one. While using these virtual cones, the path planning algorithm successfully created the path, and the car was able to navigate through the corner to the point where it detected both colors again.

Algorithm 1: Path planning algorithm

Result: Path
$B \leftarrow$ set of points in 2D representing blue cones;
$Y \leftarrow$ set of yellow in 2D representing yellow cones;
$k \leftarrow 1$;
$Path(k) \leftarrow$ starting point;
while *True* **do**
  if $k = 1$ **then**
    $b \leftarrow B\,(argmin(\ B - Path(k)\ ))$;
    $y \leftarrow Y\,(argmin(\ Y - Path(k)\ ))$;
  else
    $p \leftarrow$ line defined by normal vector $(Path(k) - Path(k-1))$ and point $Path(k)$;
    $\rho \leftarrow$ half-plane defined by line p and direction of vector $(Path(k) - Path(k-1))$;
    $\hat{B} \leftarrow B \cap \rho$;
    $\hat{Y} \leftarrow Y \cap \rho$;
    if $\hat{B} =$ *or* $\hat{Y} =$ **then**
      break;
    else
      $b \leftarrow B\left(argmin\left(\left\|\hat{B} - Path(k)\right\|\right)\right)$;
      $y \leftarrow Y\left(argmin\left(\left\|\hat{Y} - Path(k)\right\|\right)\right)$;
    end
  end
  $Path(k+1) \leftarrow mean(b + y)$;
  $k \leftarrow k + 1$;
end

Another improvement discovered during the testing is keeping only the longest path. The trajectory is only generated when it has more waypoints than few last iterations had. The trajectory tracking does not delete its reference until it receives a new one. Because the trajectory is in the global coordinate system, the car will navigate on the last sent trajectory. Because of the limited FOV, car has a better view of the corner before it enters the corner. Keeping the longer trajectory improves speed reference generation and its tracking. Another effect is that the car does not receive slightly different path too often. Due to the imperfection of the detection algorithms, even when the car is in a steady-state, the reference trajectory differs on every frame.

When the path was successfully determined, it was generated only on one frame, and it had at least four waypoints, it is interpolated using third-order B-spline to smoothen curvature and create more points that improved behavior of the speed reference generator and trajectory tracking algorithm.

4.2 Speed reference generator

An existing speed reference generator, as well as slight changes to it, are presented in this chapter. The speed reference has to be generated for every point of the path. The challenge is to maximize speed while keeping the car stable and safely navigating through the track. In this thesis, I utilize the algorithm from [10]. The generator makes two passes through the path. The first pass starts at the end of the path. It determines speed for every previous point as the minimum of maximum cornering speed and the maximum speed for the car to safely decelerate to the current point. The second pass goes through the path in a forward direction, and it can only lower speeds from the previous pass. At each point, it is determined if the speed in the following point is kept or it is lowered so that the car can safely accelerate to that point. This method is maximizing tire traction by reaching the limit of Kamm's circle at each point but not exceeding it. Backward pass algorithm can be found in 2 and forward pass algorithm can be found in 3.

Algorithm 2: Speed reference generator - backward pass

Data: $\kappa(k), v_{lim}, \Delta s(k)$
Result: $v_{bwd}(k)$
$k \leftarrow \text{length}(N)$;
$v_{bwd}(k) \leftarrow \min(\frac{1}{\sqrt{m\kappa}}, v_{lim})$;
while $k > 1$ **do**
  $v_{max}(k-1) \leftarrow \min(\frac{1}{\sqrt{m\kappa}}, v_{lim})$;
  $a_{lim} \leftarrow [v_{max}(k-1)^2 - v_{bwd}(k)^2]/[2\Delta s(k)]$;
  $a(k-1) \leftarrow \min(-h(k(\kappa), v_{bwd}(k), -1), a_{lim}$;
  $v_{bwd}(k-1) \leftarrow \sqrt{v_{bwd}(k)^2 + 2a(k-1)\Delta s}$;
  $k \leftarrow k+1$;
end

Algorithm 3: Speed reference generator - forward pass

Data: $\kappa(k), v_{bwd}, \Delta s(k)$
Result: $v_{fwd}(k)$
$k \leftarrow 1$;
$v_{fwd}(k) \leftarrow v_{bwd}(k)$;
while $k > 1$ **do**
  $a_{lim} \leftarrow [v_{bwd}(k+1)^2 - v_{bwd}(k)^2]/[2\Delta s(k)]$;
  $a(k) \leftarrow \min(h(k(\kappa), v_{bwd}(k), 1), a_{lim}$;
  $v_{fwd}(k+1) \leftarrow \sqrt{v_{fwd}(k)^2 + 2a(k-1)\Delta s}$;
  $k \leftarrow k+1$;
end

The function $h(v, \kappa, d)$ for calculating maximum acceleration respectively

deceleration is given by

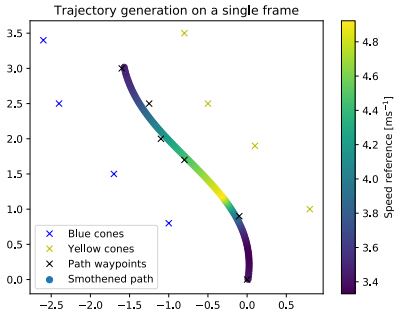
$$h(\kappa, v, d) = \begin{cases} \frac{1}{m}\left[-F_{dis} + \frac{1}{D_x} min\left(\sqrt{\mu^2 F_z^2(v) - \frac{F_y^2(v,\kappa)}{D_y^2}}, F_{acc,max}\right)\right], & \text{if } d = +1 \\ \frac{1}{m}\left[-F_{dis} - \frac{1}{D_y} min\left(\sqrt{\mu^2 F_z^2(v) - \frac{F_y^2(v,\kappa)}{D_y^2}}, F_{dec,max}\right)\right], & \text{if } d = -1 \end{cases} \tag{4.1}$$

This adaptation makes full advantage of friction ellipse as described in 3 instead of simple circle. D_x and D_y constants are usually greater than one for racecars and simplifying this would neglect effect of racing tires. It also models limitation of braking force which improves behaviour in tight corners. Another advantage is in lowering overall acceleration and deceleration thus giving maneuver capability for used feedback control as it will always react later and will tend to brake harder than planned.

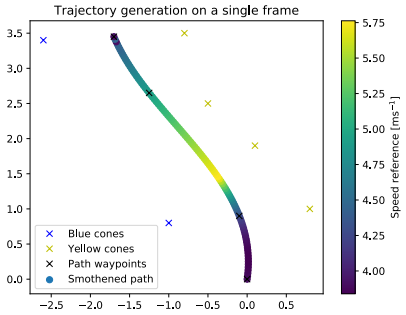
The speed reference generator calculates speed accordingly to the friction coefficient μ. Team eForce FEE Prague Formula already designed a system that calculates the friction coefficient during the race based on data from IMU and information about RPM of the wheels. An autonomous vehicle gave an excellent opportunity for this system to further improve its performance. Most of the autonomous vehicles are equipped with a camera as it is in the case of eForce Driverless. Using deep-learning and other image processing techniques, a newly developed algorithm can be based on images from camera identifying segments of the road such as wet tarmac with the different friction coefficient and providing this information to the speed reference generator to further improve its reliability.

4.3 Simulation scenarios

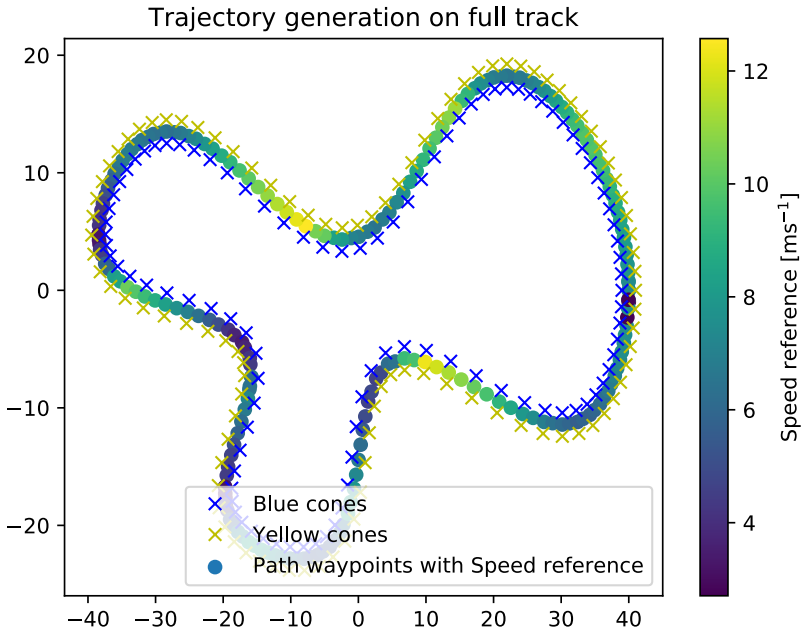
The motion planning was validated on simulated scenarios. These scenarios were designed to validate that system achieved specified objectives in the beginning. Both single frame detection and full-track were tested. Also single frame detection with missing cones was tested. Results are in Figure 4.1.



(a) : Single frame without missing cones



(b) : Single frame with missing cones



(c) : Full track

Figure 4.1: Motion planning algortihm results under different conditions

Chapter 5

Trajectory Tracking

Trajectory tracking or control system is divided into two parts according to the axis in which the system acts. These categories are longitudinal and lateral control. Already existing algorithms are implemented for both. Lateral control is a geometrically based law acting as feedback control. Longitudinal control is done with a simple feedback PI regulator.

Controllers are implemented as two individual nodes, thus two individual processes. Both subscribe to the trajectory and the pose of the car. Their output topic and message types differ in each application. It can be found in 6 where full graph is shown or is described in chapters related to platforms used (7, 8 and 9). In general, lateral controller outputs steering angle set-point and longitudinal controller outputs desired torque, which is then distributed evenly over all motors.

Both of these controllers will be part of the future development to be converted to more complex laws for improved performance. Although, as defined in chapter 2, it is not a subject to this season development nor this thesis. Both lateral and longitudinal control is implemented with respect to objectives defined in chapter 2.

5.1 Lateral control

In 2005 Standford university won DARPA Grand Challenge with its autonomous car named Stanley. As the only car in the competition, Stanley was able to pass all gates in all runs. The car was controlled based on newly developed lateral control laws described in [11]. It is a geometric control law, and its stability was proven by the authors in [11]. Full control law follows

$$\delta = \left(e_h - k_{ss} v^2 \kappa\right) + \arctan \frac{k e_c}{k_{soft+v}} + k_{d,yaw} \left(\dot{\psi} - v\kappa\right) \tag{5.1}$$

$$e_h = \psi - \psi_{ref} \tag{5.2}$$

$$\tag{5.3}$$

where δ is the steering angle, ψ is the vehicle's yaw angle (heading), κ is the curvature of the trajectory at the reference point, v is the vehicle speed and

e_c is the crosstrack error. Constants k, k_{soft} and $k_{d,yaw}$ are tunable. Constant k_{ss} is derived from simplified tire model as

$$k_{ss} = \frac{m}{C_y(1 + \frac{l_f}{l_r})} \tag{5.4}$$

The first member of the sum performs heading control. Direct input of the heading error steers the car parallel to the curvature while member $k_{ss}v^2\kappa$ improves performance on curvy tracks by setting non zero yaw reference to lead the vehicle more towards the center of the curve. The second member performs offset control. When cross-track error is high, arctan function saturates at $\pm\frac{\pi}{2}$. When the cross-track error is small solution is approximately $e_c(t) = e^{-kt}$ resulting in exponential converging to the trajectory. The last member of the sum is yaw damper. It is a simple feedback yaw damper that compensates for the diminishing damping effect of the tires at high speeds.

The control law was proven stable by the authors in the original paper [11], and it was proven as well in our simulation on different platforms, which will be described in the following chapters. In future development, a non-linear controller might be tested, but Stanley control laws met objectives for this season and are considered a strong baseline for future development.

5.2 Longitudinal control

For longitudinal control, feedback PI regulator was implemented. Control law is

$$torque = k_p\dot{e} + \frac{2}{3}k_i e \tag{5.5}$$

$$\dot{e} = v_{ref} - v \tag{5.6}$$

where k_p and k_i is tunable.

The DV.01 is an electric vehicle, thus allowing regenerative braking. Negative torque requests can then be used directly to the motors causing motors to break the car and regenerating power at the same time. However, it must be taken into account that we are rebuilding the FSE.07 car, which was not build to withstand full-time braking using its powertrain. The powertrain cooling system is able to keep the powertrain cool when used only for an acceleration. For safety reasons, the car would not be pushed to its limits lowering cooling requirements, but at the same time lowering speed will result in lower airflow, thus lower cooling efficiency. If it is necessary, the longitudinal controller can be adapted such that torque reference will be normalized and then transformed into torque command if torque reference is positive (acceleration) else it will be translated into a braking command for the hydraulic brake system.

Neither Anti-lock braking system (ABS) nor Traction control system (TCS) was designed or implemented. The speed reference generator calculates speed reference with respect to the tire traction capacity thus neglecting the need for these systems when a longitudinal controller is tuned optimally. Because

the controller is designed only as feedback controller, in some situations ABS or TCS might be useful. However, this need is further lowered by user-specified maximal (minimal) force artificially lowering friction ellipse, thus giving maneuvering capability for the longitudinal controller.

Chapter 6

System architecture

In this chapter, the autonomous system architecture is derived. The base autonomous system consists of multiple subsystems, and to fulfill its mission, it must have access to the peripherals. ROS was found to be the best choice for our purpose.

ROS [4] is a framework and set of libraries and state-of-the-art algorithms created to ease robot development. The implementation of the ROS is a set of so-called nodes and topics. Each node is a process which can subscribe or publish data as a message to a topic. Receiving a message from the topic is acting as an interrupt activating callback with the message as an argument. A message is not a single value or an array. A message can consist of multiple arrays or single values of multiple datatypes. ROS node can be written in C++ or Python. All packages implemented are based on ROS 1 (commonly referenced as ROS).

The base topology of the system, as used in car DV.01, can be found in 6.1. It is simplified ROS graph. Each package is represented as a box, and topics are simplified as arrows. Each package can consist of multiple nodes, and each arrow may represent multiple topics. Arrow also represents the direction of the communication.

Autonomous state machine implements the state machine described in [5, p. 92-93], which is subscribed by most nodes, and it is superior to the car operation. The state machine can accesses peripherals via CAN. It accepts and sends system critical signals. This node will not be further described in this thesis.

The cone detector publishes messages with cones' positions with respect to car-fixed coordinates at the moment of capturing it. For further processing, the message also contains the pose of the car when cones where captured. In the current implementation, only the camera image is processed. The cone detector was not developed by the author of the thesis. However, for real-world testing, another platform was used, and for this purpose, the image processing algorithm must have been adapted by the author. Details about this are in 9.

The motion planning subscribes cone positions published by the cone detector. Immediately upon receiving a message, it calculates a path and generates a speed reference for each point. The generated path is an approximation

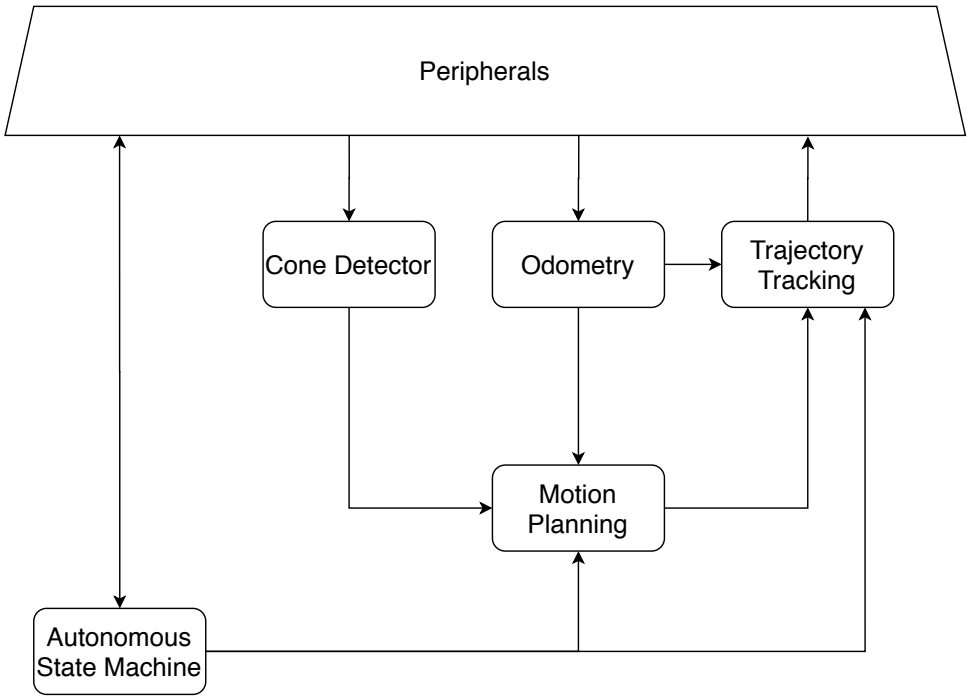


Figure 6.1: Base system architecture as used in DV.01

of the center-line. The path generating algorithm was not developed by the author of the thesis. ROS implementation, extrapolation, and speed reference generator were created by the author. The node and related algorithms were described in 4. The node publishes generated path and speed reference with respect to the global coordinates. When path planning runs on a single frame, transformation to the global coordinates is done with respect to the pose at the moment of capturing the image.

The trajectory tracking subscribes to the trajectory and odometry topic. Signals to steering mechanism are calculated based on the Stanley control laws. The second output of this node is a torque set-point for each motor generated by a PI controller. The development and functionalities of this node are described in 5.

Chapter 7

ROS based simulator

ROS framework was used to implement a vehicle simulator to validate motion planning and trajectory tracking algorithm before deployed to the real car. It is common and necessary to do these tests in a simulator for two main reasons. First, when testing under real-world conditions, there is a probability of an accident, possibly damaging property or hurting people. The second reason is pure economics. It is less costly to run a simulation. However, the real-world cannot be simulated precisely, and tests under real-world conditions are still required before final deployment. High level architecture is in the Figure 7.1.

For purposes of this thesis, the single-track model described in 3 was implemented as a single node in ROS simulating physics. It is assumed that DV.01 steering will not reach the set-point immediately. To implement this imperfection, steering simulation is implemented as a first-order system in another node. Another issue withstands when trying to simulate dynamic events with an unknown track. For this reason, a vision node is implemented to provide information about visible cones.

7.1 Steering simulation

The steering simulation implements following law

$$\dot{\delta} = k(\delta_{ref} - \delta) \tag{7.1}$$

where δ is the steering angle, δ_{ref} is set-point and k is estimated gain but it is to be identified to much actual system when it is finished.

Gaussian noise is added to the final steering angle and sent to the other topics to simulate real-world inaccuracy in the measurements of the steering angle.

7.2 Vision simulation

In this section, I will describe the vision simulation. In order to simulate vision, I do not use any visual input to the cone detection algorithms. Based on the knowledge of the car pose and information about the location of

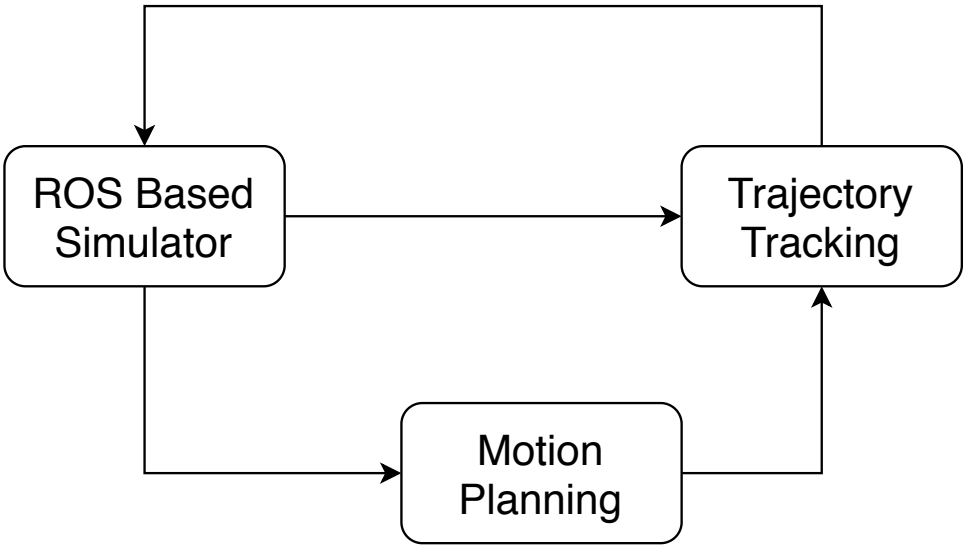


Figure 7.1: ROS based simulator high-level architecture

all cones, simply cones in predefined range and field of view are considered detected and output.

To model the imperfection of the real system, noise is added to both x and y coordinates of the detected cones. Furthermore, real-world detections are less precise when detecting objects further from the car. For this reason, noise is scaled based on distance from the car.

This node could be further improved by implementing false positive and false negative detections. For the sake of the simplicity of the simulator environment, these are currently neglected because the character of false detections is more complicated than just simple Gaussian noise. After further investigation, the node can be modified to this behavior as a part of future work.

In the future development, it can be used to simulate multiple sensors with different accuracy for testing of the Simultaneous Localization and Mapping (SLAM) and other mapping functionalities.

7.3 Vehicle simulation

The last part of the simulator is the vehicle dynamics simulation itself. The simulator is based on the model introduced in 3. As in the previous part of the simulation, Gaussian noise is added to the output of the node to model the sensor's imperfection. The output of the model is the pose of the car (x coordinate, y coordinate, and heading), speed and yaw rate with respect to the global coordinates.

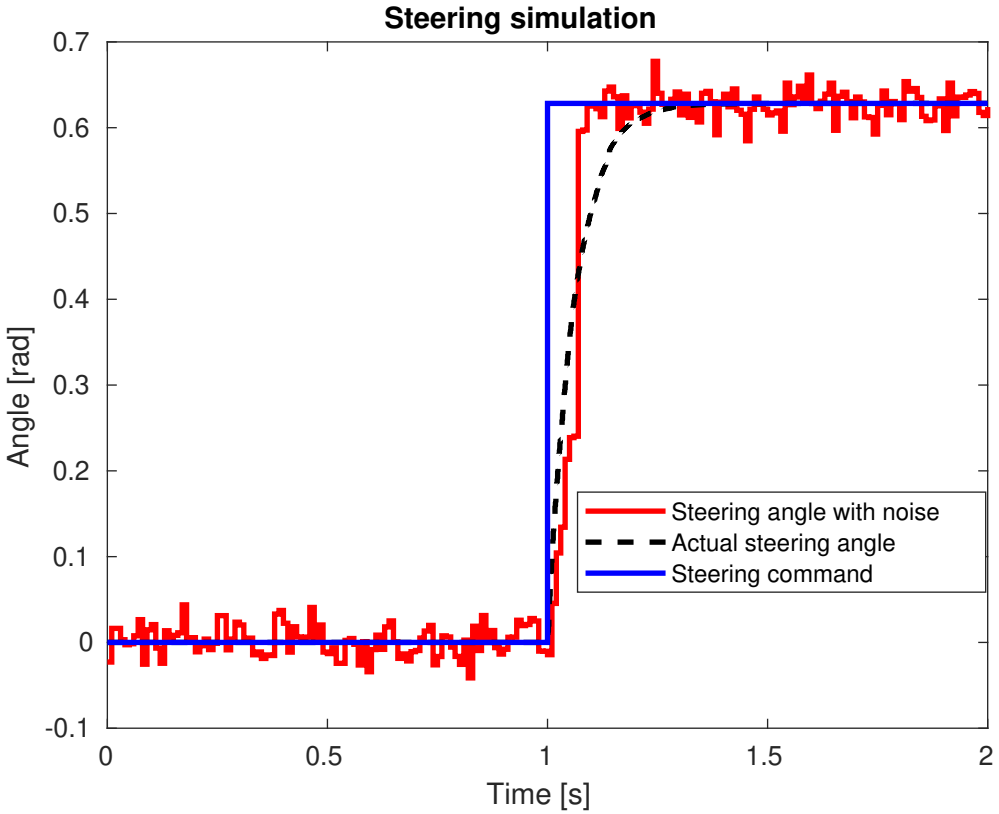


Figure 7.2: Steering simulation

7.4 Experiment with track

In this section, I will focus on the autocross track. All nodes from the previous section are used in order to simulate vehicle and sensors behavior. During the autocross mission, the car will drive in the track based on a single frame. Although by proving system working in autocross mission, the system is proven to be working in a track-drive mission as well. Both missions use the same track constraints, and the only difference is the number of laps. When SLAM [12] is implemented, teams can gain the advantage of the knowledge of the track from the first track. This is not implemented in the DV.01 autonomous system yet. However, algorithms in 4 are not dependent on the length of the track, and it is assumed they will work on full track as well. This was proven in FSO, and it is described further in this thesis in 8.

The motion planning and trajectory tracking were validated on track in Fig. 7.3. The start and finish of the track are at the point $(0, 0)$, and the car has the initial heading of zero. It is in the positive direction of the x-axis. In the figure is visible ground truth for the cones positions and of the car position. All simulations were running according to the description in this chapter, so none of them had access to information plotted in the figure. The noise was added to the odometry and cones detections. Even after this car

was able to successfully navigate through the track.

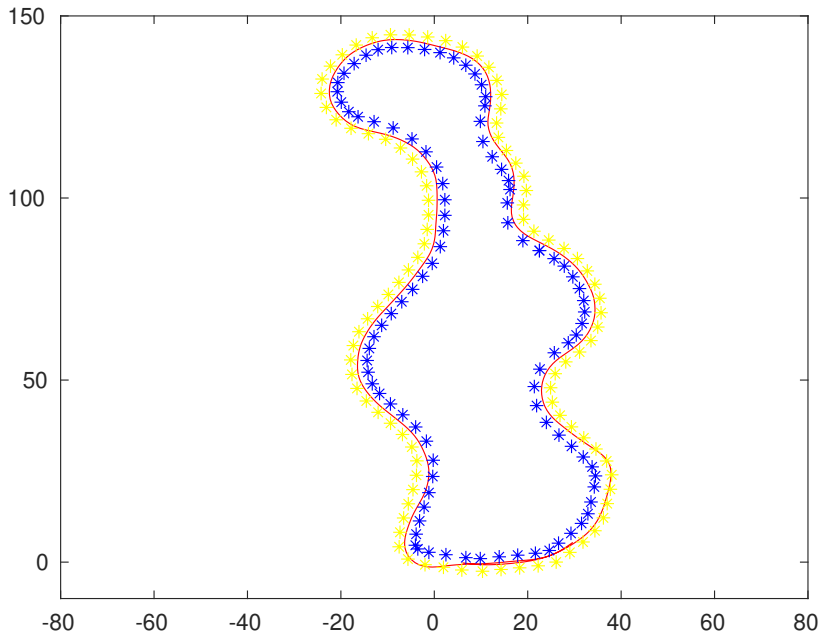


Figure 7.3: Autocross track and logged position of the car

However, it can be seen that it slightly overshot the last corner. The hairpin corner has the smallest diameter allowed in the rules. Usually, in the tight corners, the track gets wider, which would improve performance. For simplicity of the track generation and to reach the best performance at any time, the generated track has a constant track width. Relatively speaking, with respect to the speed the car was able to navigate through the track, this overshoot is okay.

As seen in Fig. 7.4, car exceeded the minimum average speed on the track. The minimum average speed is defined in the rules as $4\,\mathrm{m\,s^{-1}}$.

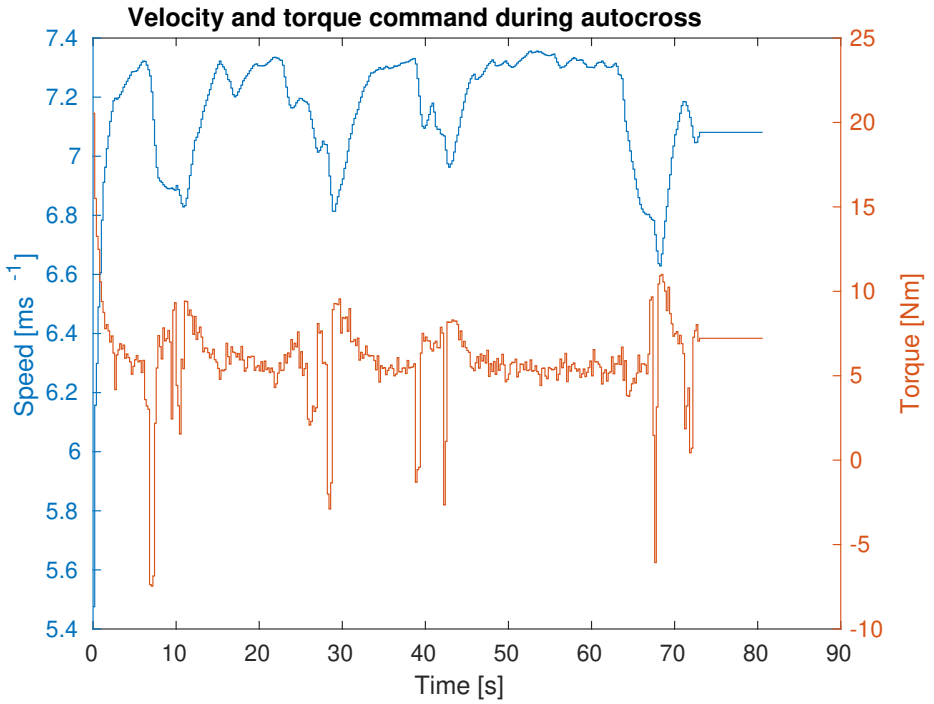


Figure 7.4: Car speed and torque command

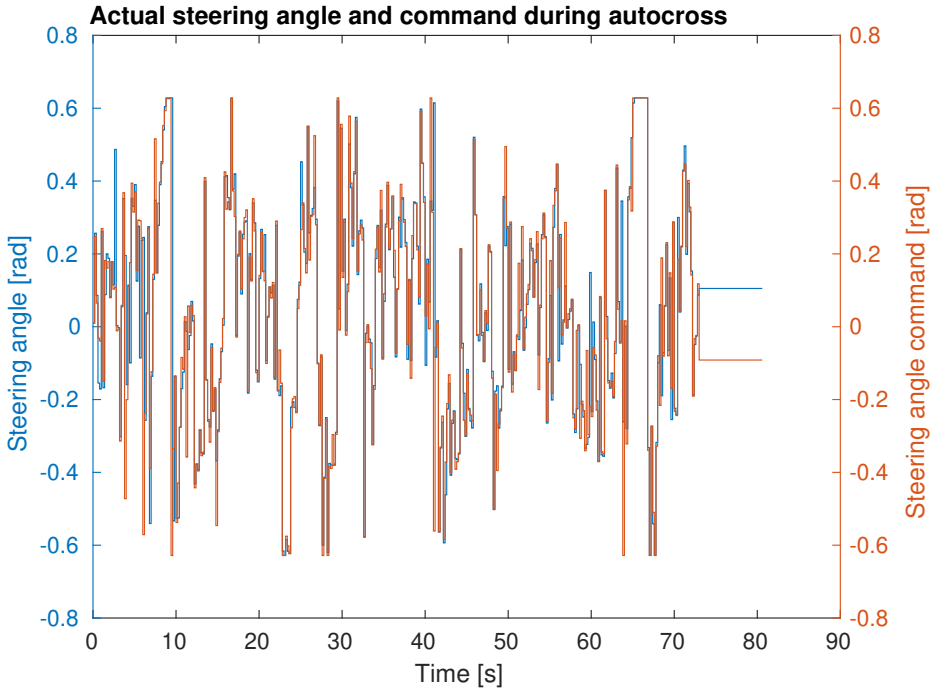


Figure 7.5: Car steering angle and command

Chapter 8

Formula Student Driverless Simulator

A concept of FSO was introduced in March 2020 as a reaction to world pandemic and cancellation of competitions all over the world. The goal was to keep FS racing spirit alive even when teams could not meet and compete physically. As platform for Driverless competition, team of students from Delft University of Technology (TU Delft) and Massachusetts Institute of Technology (MIT) developed FSDS [3].

The organizers of the competition decided to use AirSim by Microsoft Corporation [13], which is a simulator built on Unreal Engine 4 (UE4) [14]. Rules for the Virtual Driverless Event were derived from FSG rules [15]. The Virtual Driverless Event consists of only track-drive and autocross as the most challenging dynamic events in the competition. Main constraints to the tracks, as presented in 1.2 remained unchanged.

8.1 System adaptation

Each of the teams could have selected multiple cameras and LiDARs. Both numbers and parameters of these sensors were limited by the rules. Team eForce Driverless decided to use one camera on top of the car and one LiDAR in the middle of the front wing.

Unfortunately, competitors could not upload their own parameters for vehicles; thus, the only one provided by the organizers was used. Moreover, not all of the parameters were shared with the competitors. As most critical parameters were identified mass of the vehicle, the friction coefficient, the maximum acceleration and deceleration as these are used in speed reference generator 4. Out of these, only the mass of the vehicle was shared with the teams. Because only limited outputs from the simulator were provided - odometry and visual data, other parameters used by the speed reference generator were simplified or neglected. The aerodynamics of the car was neglected, and D_x and D_y parameters were simplified to one simplifying friction ellipse into friction circle.

Adjustments had to be also made to the trajectory tracking algorithm. The minor one was changing the maximum steering angle of the car. All other coefficients used by the lateral controller had to be tuned experimentally in accordance with FSDS. Quite unrealistic behavior was spotted in the speed

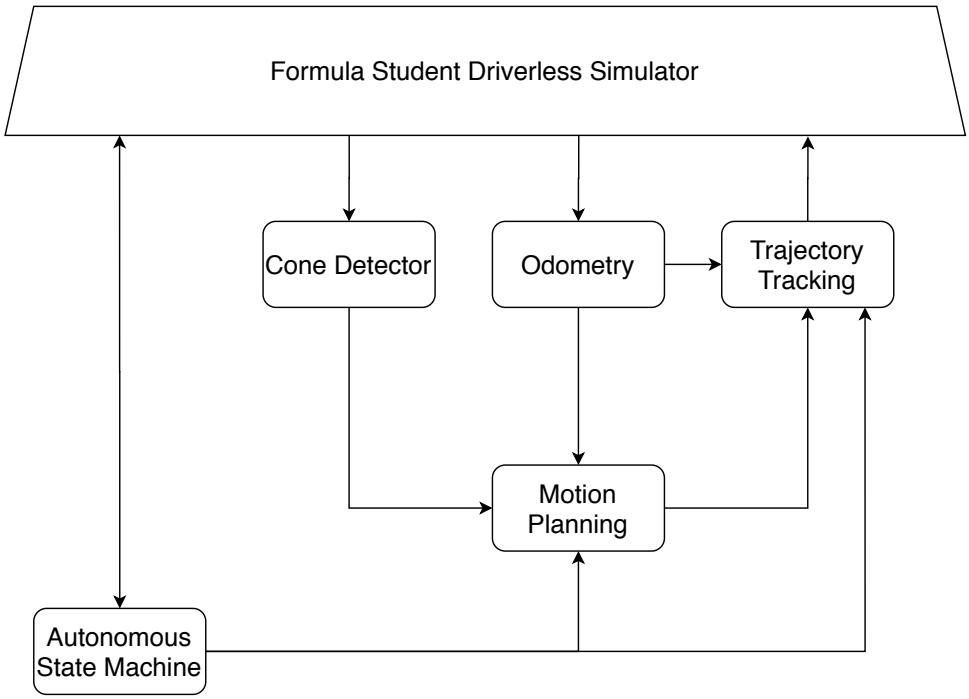


Figure 8.1: FSDS high-level architecture

and accuracy of the steering mechanism. After further investigation, it was found that the steering mechanism reacts almost immediately to set-point without overshooting it. This potentially improves the performance of the lateral controller. Another minor change was normalizing the output into range $(-1, 1)$.

The longitudinal controller had to be tuned experimentally for the FSDS. Same as for lateral controller, requested output was in range $(-1, 1)$ where negative values were used as normalized breaking, and positive were used as torque command.

The perception was also changed. The LiDAR was used instead of the camera as the main source for its more accurate detections. SLAM was not implemented before the final submission of the code for the simulator, so only one sensor was used. In order to improve performance on the track-drive, a simplified method was used. It was found that odometry had near-zero error. For the track-drive mission logged positions from the first lap were used as a path in remaining laps instead of detecting cones and planning path on every single frame. The speed reference generator remained unchanged.

This solution is not ideal, although it was easy to implement and it improved results significantly. In the future, this should be done using SLAM algorithm and sensor fusion.

Fully adjusted pipeline for the FSO competition in the ROS framework can be found in the figure 8.1.

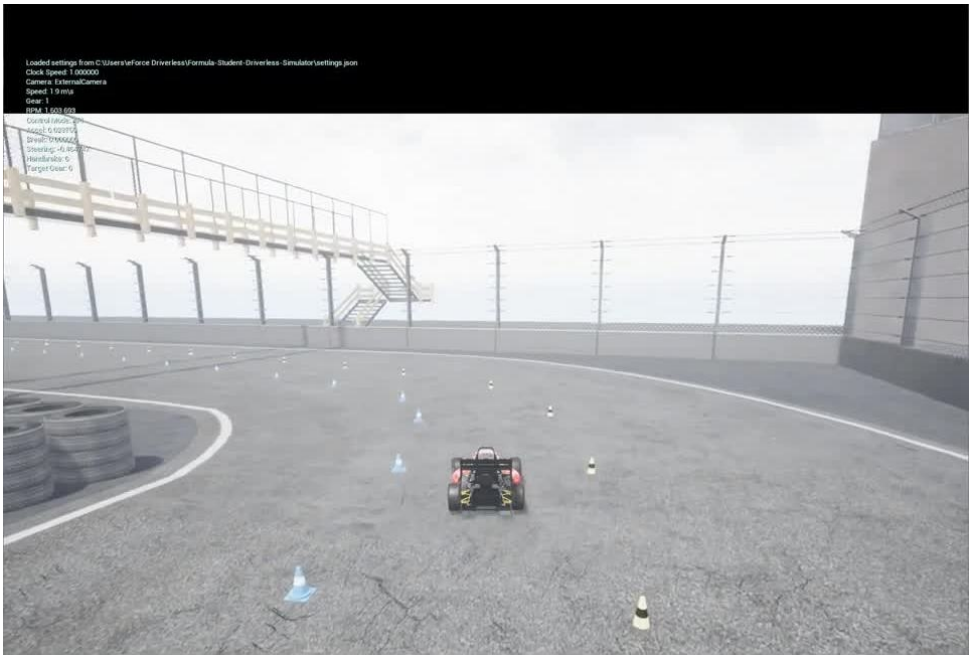


Figure 8.2: FSDS simulator

8.2 Results

There were three competition days. Each day the difficulty of the track increased as well as points awarded. The team was able to finish the autocross mission every competition day. But due to the bug in the state machine, we did not finish the track-drive mission because the car was unable to transition to a different mission. Last day, the team finished all missions, and it was the only team able to finish track-drive that day.

Example of the system can be found in Figures 8.2 and 8.3. The first one captures the simulator itself. The second one captures visualization using Rviz tool of what car sees and plans.

All days were live-streamed on the official FSO YouTube channel and can be still accessed [16].

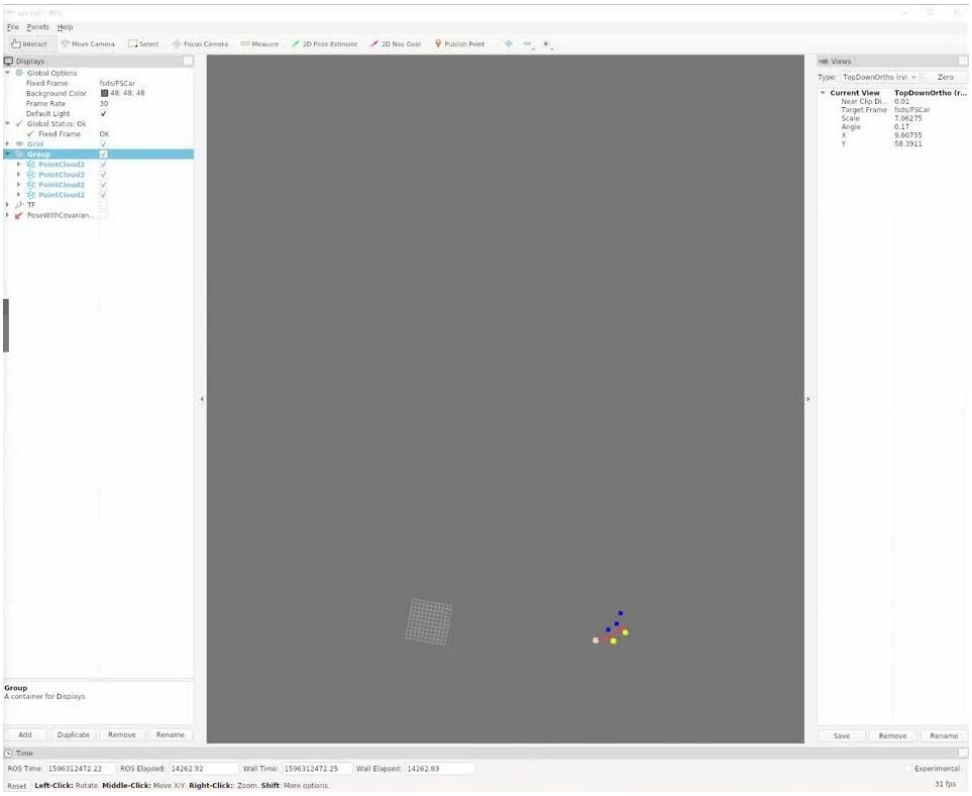


Figure 8.3: Rviz visualization

Chapter 9

Radio-controlled Car

As a final platform used to validate the presented system and mainly control algorithms, RC car was used. The ROS framework had to be adapted to this platform again. The goal was to prove current system architecture working with a focus on motion planning and trajectory tracking algorithm, so only minor changes to the architecture were done.

9.1 System adaptation

The system in this application lacks an autonomous state machine because it is redundant. CAN bus interface is replaced by interface for the RC car which does not use CAN bus. For motion planning and trajectory tracking, physical constants and tunable coefficients had to be changed. These were done with respect to [2] where these were identified and tuned in simulation.

In the base architecture, cone detection is done using cameras and LIDAR. The RC car is equipped only with one of the three cameras and has no LIDAR. ZED camera, which is installed on both vehicles, is not at the same spot; thus, the base system could not be used. Two significant changes had to be made. The first was to gather images from the track from the new position of the camera, label them, and train Neural Network (NN) using these pictures.

9.1.1 Cone detection

Cone detection is done with the YOLOv3 [17]. YOLOv3 offers a large number of objects it detects. These are redundant in this specific use case, and desired categories (blue, yellow, orange, and big orange cones) that are desired are not part of the YOLOv3 output. As a standard procedure, the YOLOv3 was changed, so it outputs only desired categories and then trained using a shared database of labeled cones [18]. The approximate size of the training set is 8000 pictures. This NN is trained again using pictures obtained with RC car to adapt to a new position, which is not represented in FSOCO database [18].

Over 350 images were captured and then manually labeled using OpenLabeling open-source labeler as labeling environment [19]. An example of the environment is in Fig. 9.1.

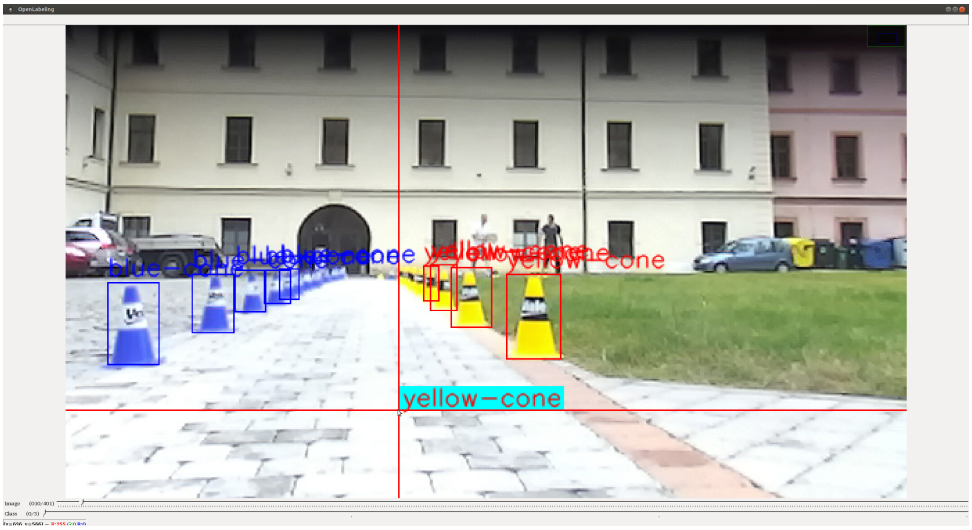


Figure 9.1: OpenLabeling open-source labeler

When NN is running, it outputs bounding boxes of the cones in the image. The cone coordinates in the image are transformed into the car-fixed coordinates using homography. Homography project points from the image plane to the x-y plane with respect to the car-fixed coordinates [20]. Cone's coordinates in the picture are obtained as the middle of the bottom line of the bounding box. Because this point does not represent the center of the cone, it makes this method slightly inaccurate.

Homography matrix is calculated as non-trivial solution h of

$$\mathrm{Ph} = 0 \tag{9.1}$$

where h is vector of 9 elements then transformed to the 3x3 H matrix and P is $2n$x9 matrix from n corresponding points. P is a stack of following 2x9 matrices

$$\begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & ux & vx & x \\ 0 & 0 & 0 & -u & -v & -1 & uy & vy & y \end{bmatrix} \tag{9.2}$$

where (u, v) are coordinates of the cone in the picture and (x, y) are coordinates in the x-y plane with respect to the car-fixed coordinates. To solve homography, matrix P is decomposed using Singular Value Decomposition (SVD). h is vector corresponding to the smallest singular value.

9.1.2 Motion planning and trajectory tracking

Both had to be adapted in terms of vehicle dynamics and tunable parameters. Motion planning algorithm was validated and performed as expected base on results in ROS based simulator and FSDS. Unfortunately, problems with refactoring codes for odometry and controls of motors and steering were not resolved on time thus trajectory tracking could not be validated.

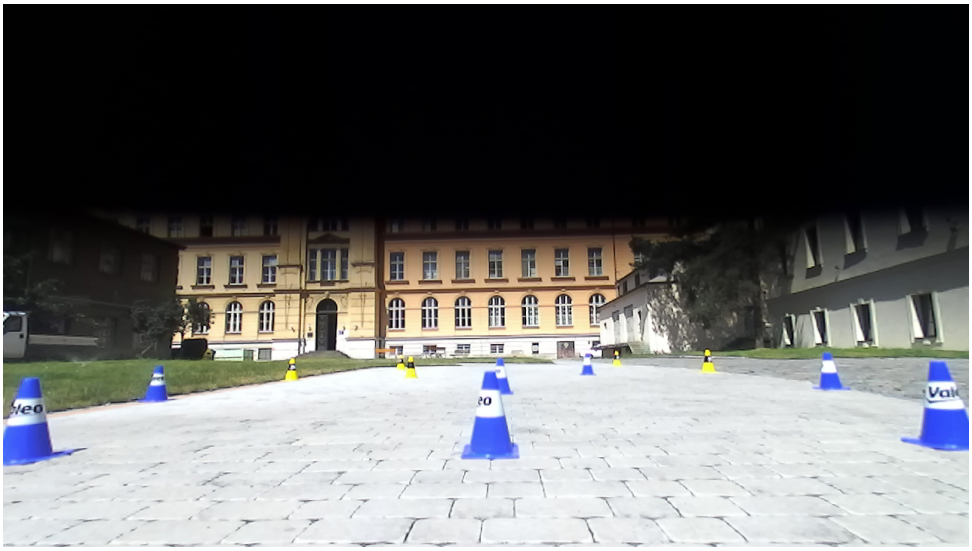


Figure 9.2: Setup and picture used to obtain coordinates in picture

9.2 Experiments

As stated in previous section, system was not fully refactored to the RC car on time. However, cone detection and motion planning were validated while the car was being controlled manually. Example of such output could be seen in Fig. 9.3. Based on these result, the system was proven working. After further work will be done on this platform, the trajectory tracking will be validated as well.

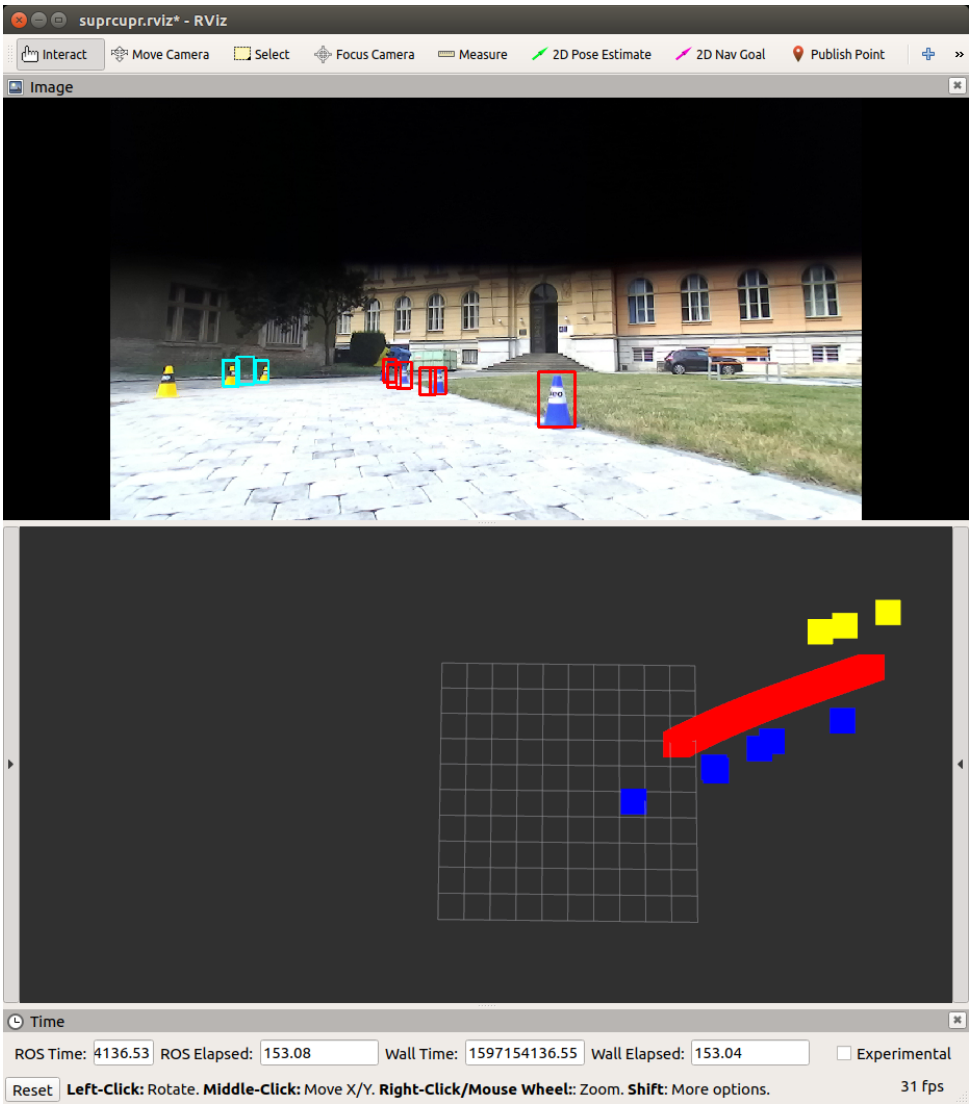


Figure 9.3: Motion planning validation

Chapter 10

Results

As part of this thesis in accordance with objectives stated in chapter 2, multiple parts of the system were derived, developed, or implemented. The following results were done by the author. The author of the thesis:

- as a founding member, core member, and leader for six months period of eForce Driverless collaborated on the definition of the system architecture as described in this thesis.
- developed ROS based simulator for motion planning and trajectory tracking validation using vehicle dynamics described in Chapter 3. The simulator, in addition to vehicle dynamics simulation, features simple steering and vision simulation. The simulator outputs all data with specified noise.
- collaborated on development and implementation of the motion planning as described in Chapter 4 with the following roles
  - The author refactored path planning algorithm written as standalone python code by the other team member to the ROS framework
  - The author improved path planning algorithm by implementing cones filling method, for a situation when only one side of the cones is visible, and by implementing the smoothening process
  - The author developed the speed reference generator
- designed, implemented, and empirically tuned full trajectory tracking algorithm, both lateral and longitudinal control, as described in Chapter 5.
- adapted and empirically tuned trajectory tracking algorithm used in FSO competition on FSDS.
- in collaboration with faculty members adapted the system for RC car and conducted experiments with RC car with the following roles:
  - The author conducted experiments and adapted a car's perception.

- The author is actively working on the tuning of the trajectory tracking.
- The author found no need to change the motion planning algorithm.

Chapter 11

Conclusions

In this thesis, I have described the architecture of the autonomous system for the DV.01 racecar. I have implemented motion planning and trajectory tracking algorithms. Due to multiple reasons, this system could not be tested on the racecar in real-world conditions. Although, as a part of this work was developed ROS based simulator to verify motion planning and trajectory tracking algorithms.

The system was, at the same time, adapted to two platforms where it was successfully validated and proved working. The first platform was done in a more complex simulator. The second platform was the RC car, and algorithms were tested in real-world conditions. Despite all the challenges, there were no major issues discovered during extensive testing.

Despite all the work that was done until now, there is a lot of future work that has to be done. Now, work will focus on finishing the rebuilding of the FSE.07 car to DV.01 and the deployment of the final system that was described and proven working in this thesis. After the deployed system is extensively tested to ensure stability and safe operations under all conditions, work can focus on improving algorithms or implementing more complex ones while taking advantage of the ROS framework, thus keeping this proven architecture.

Acronyms

ABS Anti-lock braking system. 22, 23

CAN bus Controller Area Network. 2, 37

CoG Center of Gravity. 9, 49

DOF degrees of freedom. 9

EBS Emergency Braking System. 2, 4

FOV field of view. 16, 17

FS Formula Student. 1, 33

FSDS Formula Student Driverless Simulator. viii, 7, 33, 34, 38, 41

FSG Formula Student Germany. 1, 3, 33

FSO Formula Student Online. 1, 3, 7, 29, 33–35, 41

GNSS Global Navigation Satellite System. 3

IMU inertial measurement unit. 3, 19

INS Inertial Navigation System. 2

MCU Microcontroller Unit. 2

MIT Massachusetts Institute of Technology. 33

NN Neural Network. 37, 38

RC Radio-controlled. viii, 1, 3, 5, 8, 37, 39, 41, 43

ROS Robot Operating System. 1, 2, 7, 9, 15, 16, 25–27, 34, 37, 38, 41, 43

SAE Society of Automotive Engineers. 1

SLAM Simultaneous Localization and Mapping. 28, 29, 34

SVD Singular Value Decomposition. 38

TCS Traction control system. 22, 23

TU Delft Delft University of Technology. 33

UE4 Unreal Engine 4. 33

Bibliography

[1] "eforce fee prague formula." https://eforce.cvut.cz/. Accessed on August 13, 2020.

[2] D. Filyo, "Control laws for autonomous racing," Master's thesis, Czech Technical University in Prague, 5 2020.

[3] FS-online, "Formula student driverless simulator." https://github.com/FS-Online/Formula-Student-Driverless-Simulator/, 2020. Accessed on August 13, 2020.

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[5] Formula Student Germany, *Formula Student Rules 2020*, 13/09/2019. https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf.

[6] Formula Student Germany, *FSG Competition Handbook 2020*, 23/01/2020. https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FSG20_Competition_Handbook_v1.0.pdf.

[7] D. Schramm, *Vehicle dynamics : modeling and simulation*. Heidelberg: Springer, 2014.

[8] H. B. Pacejka, *Tyre and vehicle dynamics*. Oxford: Butterworth-Heinemann, 2006.

[9] D. Efremov, T. Haniš, and M. Hromčík, "Introduction of driving envelope and full-time-full-authority control for vehicle stabilization systems," in *2019 22nd International Conference on Process Control (PC19)*, pp. 173–178, 2019.

[10] J. Filip, "Trajectory tracking for autonomous vehicles," Master's thesis, Czech Technical University in Prague, 6 2018.

[11] 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," in *2007 American Control Conference*, pp. 2296–2301, 2007.

[12] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *IJCAI*, pp. 1151–1156, 2003.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, pp. 621–635, Springer, 2018.

[14] A. Sanders, *An introduction to Unreal engine 4*. CRC Press, 2016.

[15] F. Online, "Virtual driverless event rules." `https://fso-srv.sze.hu/wp-content/uploads/2020/06/FS-Online-2020-Virtual-Driverless-Event-Rules_vol2.pdf`. Accessed on August 13, 2020.

[16] "Formula student online youtube channel." `https://www.youtube.com/channel/UCGZ9jX9aeGjeTYlsyWI5zBg`, note = Accessed on August 13, 2020.

[17] A. F. Joseph Redmon, "Yolov3: An incremental improvement," tech. rep., University of Washington, 2018.

[18] "Formula student objects in context." `https://github.com/ddavid/fsoco`, note = Accessed on August 13, 2020.

[19] J. Cartucho, R. Ventura, and M. Veloso, "Robust object recognition through symbiotic deep learning in mobile robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2336–2341, 2018.

[20] E. Dubrofsky, "Homography estimation," Master's thesis, The University OF British Columbia, 5 2009.

[21] M. László, "Flight control solutions applied for improving vehicle dynamics," Master's thesis, Czech Technical University in Prague, 2 2019.

Appendix A

DV.01 car parameters

This section summarizes DV.01 parameters used for vehicle modeling.

Description	Notation	Value
Vehicle mass	m	220 kg
Distance from the CoG to the front axle	l_f	1.286 m
Distance from the CoG to the rear axle	l_r	1.230 m
Moment of inertia	I_z	120 kg m^2
Aerodynamic reference area	A_{ref}	1.14 m^2
Drag coefficient	C_D	1.3
Lift coefficient	C_L	3.5

Table A.1: DV.01 model parameters

The mass is distributed between axles with front to rear ration 0.5. Aerodynamic lift is distributed evenly across axles.

Description	Value
Maximum rotates per minute of the front motor	1294
Maximum rotates per minute of the rear motor	1490
Maximum torque of the front motor	322 N m
Maximum torque of the rear motor	113 N m
Front motor to front wheel ratio	6.95
Rear motor to rear wheel ratio	6.71

Table A.2: DV.01 drivetrain limitation

For the tire modeling same parameters as in [21] were used.

Description	Notation	Value
Lateral stiffness factor	B_y	0.184
Lateral shape factor	C_y	1.45
Lateral peak factor	D_y	1.4
Lateral curvature factor	E_y	-0.3
Longitudinal stiffness factor	B_x	0.165
Longitudinal shape factor	C_x	1.4
Longitudinal peak factor	D_x	1.4
Longitudinal curvature factor	E_x	-1
Friction coefficient	μ	0.8
Tire radius	R	0.2 m

Table A.3: DV.01 tire parameters

Appendix B

ROS Simulator Parameters

Parameters for controllers and speed reference related parameters. For the dynamics of the car, DV.01 parameters are used.

Description	Notation	Value
Yaw damper coefficient for lateral control	$k_{d,yaw}$	1.5
Cross-track error coefficient for lateral control	k_d	15
Cross-track error softening coefficient for lateral control	k_{soft}	0.5
Proportional coefficient for longitudinal control	k_p	8
Integral coefficient for longitudinal control	k_i	0.05
Maximum speed for autocross		$8\,\mathrm{m\,s^{-1}}$
Maximum speed		$25\,\mathrm{m\,s^{-1}}$
Maximum longitudinal for speed planning	$F_{acc,max}$	880 N
Maximum longitudinal for speed planning	$F_{dec,max}$	660 N
Maximum steering angle	δ_{max}	$\frac{\pi}{4}$

Table B.1: ROS simulator parameters for trajectory tracking and motion planning

Standard deviation of output parameters from the simulator. Mean of the noise is always 0.

Noise parameter	Value
Standard deviation for position of the car (coordinates)	0.05 m
Standard deviation for heading of the car	1°
Standard deviation for linear velocity of the car	$5\,\mathrm{m\,s^{-1}}$
Standard deviation for angular velocity of the car	$1^{\circ}\ \mathrm{s^{-1}}$
Standard deviation for steering angle of the car	$1^{\circ}\ \mathrm{s^{-1}}$
Standard deviation for position of the cones	0.03 m

Table B.2: Noise standard deviations used in simulation

Appendix C

FSDS Parameters

Parameters for controllers and speed reference related parameters.

Description	Notation	Value
Yaw damper coefficient for lateral control	$k_{d,yaw}$	0.2
Cross-track error coefficient for lateral control	k_d	1.4
Cross-track error softening coefficient for lateral control	k_{soft}	0.2
Proportional coefficient for longitudinal control	k_p	0.3
Integral coefficient for longitudinal control	k_i	0.001
Maximum speed for autocross		$5\,\mathrm{m\,s^{-1}}$
Maximum speed		$25\,\mathrm{m\,s^{-1}}$
Maximum longitudinal for speed planning	$F_{acc,max}$	110 N
Maximum longitudinal for speed planning	$F_{dec,max}$	44 N
Maximum steering angle	δ_{max}	$\frac{\pi}{5}$

Table C.1: FSDS parameters for trajectory tracking and motion planning

Parameters to calculate dynamics of the car. Only parameters used by trajectory tracking and motion planning are described as others are not used.

Description	**Notation**	**Value**
Vehicle mass	m	110 kg
Tire-road friction coefficient	μ	0.2
Lateral peak factor	D_y	1
Longitudinal peak factor	D_y	1
Aerodynamic reference area	A_{ref}	1.14 m^2
Drag coefficient	C_D	1.3
Lift coefficient	C_L	3.5

Appendix D

RC Car Experiments

Experiment results for homography matrix calculation.

Real world points		Image points	
x	**y**	**u**	**v**
2	1.78	30	600
2	0	645	595
2	-1.67	1260	580
4.5	3	203	524
4.5	0	660	515
4.5	-2.8	1100	510
7.38	3	384	500
7.38	1.25	543	494
7.38	-1.1	776	491
7.38	-2.8	936	489
10.38	2.1	528	483
10.38	-2.25	814	478

Table D.1: World-image correspondences

Appendix E

Content of the CD

File/Directory	Description
bp_bohacm11.pdf	Text of the thesis
src/motion_planning/	Base directory of motion planning ROS package
src/trajectory_tracking/	Base directory of trajectory tracking ROS package
src/motion_planning/	Base directory of simulation ROS package
competition.mp4	Video from FSO competition

Table E.1: CD Content