Bachelor's thesis

Algorithm to Complete the First Lap for Autonomous Student Formula

Dmytro Khursenko

Supervisor: Ing. Jan Čech, Ph.D.

Faculty of electrical engineering

Department of cybernetics

May 26, 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Khursenko Dmytro** Personal ID number: **498996**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Algorithm to Complete the First Lap for Autonomous Student Formula

Bachelor’s thesis title in Czech:

Algoritmus pro průjezd prvním kolem závodu autonomní studentské formule

Guidelines:

One of the disciplines of the autonomous student formula race is to drive, as quickly as possible, through a multi-lap course laid out by traffic cones. The first lap is the hardest in the sense that the track is not known. Design an algorithm to determine the immediate trajectory in the vehicle coordinate system and find the optimal velocity along that trajectory, taking into account the physical limits of vehicle dynamics.
Assume a detector providing cone detections in the vehicle coordinate system. Since the track is usually very narrow, assume the path of the vehicle to follow a centerline between the cones.
Compare the algorithm to the conservative baseline approach, where the formula drives at a small constant speed in which it always stays on track. Perform either simulations, or optionally experiments on a real vehicle.

Bibliography / sources:

[1] Nitin R. Kapania. Trajectory planning and control of an autonomous race vehicle. PhD Thesis. Stanford University, 2016.
[2] Alexander Liniger. Path Planning and Control for Autonomous Racing. PhD Thesis. ETH Zurich, 2018.
[3] Adam Slomoi. Path Planning and Control in an Autonomous Formula Student Vehicle. Technical Report. Monash University, 2018.
[4] Michal Horacek, Finding the Fastest Trajectory for Autonomous Student Formula. Bachelor's Thesis, Czech Technical University, FEE, 2022.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Čech, Ph.D. Visual Recognition Group FEE

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

Date of bachelor’s thesis assignment: **06.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

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

III. Assignment receipt

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

. _____ Date of assignment receipt

_____ Student's signature

Declaration

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

Prague, May 26, 2023

Dmytro Khursenko
Signature

Acknowledgement

I would like to thank my supervisor Ing. Jan Čech, Ph.D for his invaluable guidance, support, and expertise throughout the duration of this thesis. His insightful feedback, unwavering encouragement, and dedication have been instrumental in shaping and refining my work. I am truly grateful for his patience, mentorship, and commitment to my academic and personal growth.

I would also like to thank all members of eForce FEE Prague Formula, who have been instrumental in the successful completion of this thesis. Their collective efforts, collaboration, and support have been invaluable throughout this research endeavor. Especially, I want to thank Michal Horáček for his help whenever I needed it, Roman Šip for developing eForce simulator, Ondřej Kuban for implementing physics in the simulator, and other members of the driverless section.

Lastly, I would like to express my sincere gratitude to Czech Technical University in Prague for providing me with an exceptional resources throughout my educational journey. The quality of education, the dedicated faculty members, and the wide range of opportunities offered by the university have played a crucial role in shaping my knowledge and skills. I extend my heartfelt thanks to CTU in Prague for helping Ukrainian refugees and students, including my close people and me, during the war in Ukraine.

In conclusion, I would like to thank my parents for giving me a chance to study and live in the Czech Republic, for their constant love, encouragement, and unwavering belief in my abilities. Their support throughout my studies has been invaluable, and I am truly grateful for their guidance and sacrifices. Their unwavering faith in me has been a source of inspiration and motivation, and I am forever grateful for their unwavering support.

Abstract

The thesis focuses on local planning algorithms that allow an autonomous formula to complete the first lap of the Formula Student Driverless competition. Local planning includes local path planning and speed profile generation algorithms. The primary objective of local planning is to efficiently complete the first lap of an unknown track, aiming for the fastest possible time. The path planning produces a smooth path based on real-time sensor inputs and uses a center-line approach that prioritizes reliability over path optimality. The algorithm was tested with different levels of perception accuracy, showing its robustness against potential inaccuracies. The local speed planning generates a speed profile taking into account path curvature and vehicle dynamics. Speed planning ensures that the vehicle drives within its maximum safe speed limits, eventually leading to faster completion of the lap. The proposed algorithms were evaluated through experiments in a simulator, considering various scenarios and tracks. The results demonstrate the successful navigation of the vehicle in the first lap, achieving competitive lap times.

Keywords: Local planning, Path planning, Speed profile, Autonomous driving, Formula Student Driverless

Abstrakt

Tato práce se zaměřuje na algoritmy pro lokální plánování, které umožňují autonomní formuli dokončit první kolo soutěže Formula Student Driverless. Lokální plánování zahrnuje algoritmy lokálního plánování trasy a generování rychlostního profilu. Hlavním cílem lokálního plánování je efektivně dokončit první kolo na neznámé trati za co nejkratší čas. Plánování trasy produkuje hladkou trasu na základě vstupů z senzorů v reálném čase a využívá přístupu založeného na středové linii, který klade důraz na spolehlivost před optimálností trasy. Algoritmus byl testován s různými úrovněmi přesnosti detektoru, což ukazuje jeho odolnost vůči potenciálním nepřesnostem. Lokální plánovač rychlostí generuje rychlostní profil s ohledem na křivost trasy a dynamiku vozidla. Plánovač rychlosí zajišťuje, že vozidlo jezdí v rámci svých maximálních bezpečných rychlostních limitů, což v konečném důsledku vede k rychlejšímu dokončení kola. Navrhované algoritmy byly vyhodnoceny prostřednictvím experimentů v simulátoru, přičemž byly zohledněny různé scénáře a různé trati. Výsledky demonstrují úspěšnou navigaci vozidla v prvním kole a dosažení konkurenčních časů na kolo.

Klíčová slova: Lokální plánovaní, Plánování trasy, Rychlostní profil, Autonomní řízení, Formula Student Driverless

Table of Contents

1 Introduction **1**
1.1 Thesis Structure . . . 1
1.2 Formula Student . . . 2
1.3 Autocross Event . . . 3
1.4 Related Work . . . 4
1.5 DV.01 . . . 6
1.6 Thesis Contribution . . . 7

2 Algorithm for the first lap **9**
2.1 Overview . . . 9
2.2 Local Path Planning . . . 11
2.2.1 Path Planning . . . 12
2.2.2 Additional Cones . . . 14
2.2.3 Sort and Filter Cones . . . 17
2.2.4 Next Center Point . . . 19
2.2.5 Cones in Front of the Car . . . 19
2.2.6 Path Smoothing . . . 21
2.3 Local Speed Planning . . . 22
2.3.1 Circle of Forces . . . 22
2.3.2 Path Curvature . . . 24
2.3.3 Safe Speed . . . 24
2.3.4 Speed Profile . . . 25

3 Experiments **30**
3.1 Data . . . 30
3.2 Simulator . . . 32
3.3 Path Planning Tests . . . 34
3.4 Speed Planning Tests . . . 38

4 Conclusion **41**
4.1 Achieved Results . . . 42
4.2 Future Work . . . 42

A Contents of the Attachment **43**

5 References **44**

Chapter 1
Introduction

Many automotive and tech companies are investing in research and development in autonomous cars because these cars have the potential to improve safety on the roads by reducing the risk of human error. Autonomous cars are capable of driving without human intervention using advanced technologies, machine learning algorithms, cameras, radars, and other sensors. The development of autonomous vehicles has led to a new field - autonomous car races. Especially in the context of formula student competition where it becomes an exciting field for innovation and learning. The objective of these competitions is to design and build a fully autonomous race car capable of completing a lap on a given track as quickly as possible while adhering to specific rules and constraints. These races help to push the boundaries of autonomous systems in vehicles. The most well-known autonomous car competitions are the DARPA Grand Challenge, the Roborace, the F1Tenth and Formula Student Driverless.

The thesis presents an algorithm that allows a driverless formula to complete the first lap of the Formula Student Driverless competition. The main challenge for a formula in the first lap is to drive on an unknown track. Based on this challenge, we need to have the robust and reliable algorithms. The algorithm for the first lap involves perception, local planning and control nodes. This thesis mainly focuses on local planning and propose local path planning and speed profile algorithms. The first goal is to develop the robust path planning algorithm that does not result in going off the track. The second goal is to design a speed profile algorithm that plans a speed taking into account the physical limits of vehicle dynamics. Then the developed algorithms are tested in a simulator.

Section 1.1
Thesis Structure

The structure of this thesis is organized as follows. Section 1 is an introduction that provides a motivation and overview of the research topic. In Section 1.2 we introduce the Formula Student competition. Section 1.3 presents the autocross event which is a dynamic discipline in the Formula Student competition that focuses on completing one lap. Then in Section 1.4 we provide a related work about local planning that includes a path planning and speed profile. Section 1.5 introduces the concept of DV.01, an autonomous student formula developed by eForce FEE Prague Formula which is a student team from the Czech Republic. The proposed algorithms in this thesis are designed for DV.01. In Section 1.6 we outline the thesis contribution.

Section 2 is the the core of the thesis, focusing on the developed algorithms to complete the first lap. Section 2.1 gives an overview of the first lap algorithms, providing a high-level explanation of its functionality and objectives. Then we introduce the local planning algorithms, such as local path planning in Section 2.2 and local speed profile in Section 2.3. The proposed algorithms are developed for DV.01. Both algorithms are described and explained in details, including auxiliary methods. The main algorithms – path planning and speed profile – are presented in Section 2.2.1 and Section 2.3.4 respectively.

Section 3 is a validation section where we test the proposed local algorithms on real-world and synthetic tracks. Section 3.1 describes the data for testing and analysis of developed algorithms. The dataset includes tracks from previous formula student competitions, tracks from real testing conducted by eForce and synthetically generated tracks. Then in Section 3.2 the eForce simulator is introduced to conduct the path planning and speed profile experiments. Evaluation of the robustness of the developed path planning algorithm is presented in Section 3.3. Section 3.4 demonstrates the results of the speed profile tests, evaluating the proposed algorithm's ability to plan suitable speeds during the first lap.

The thesis conclusion is presented in Section 4 and provides a summary of the thesis, highlighting the achieved results and their significance. It discusses the effectiveness of the developed algorithm in successfully completing the first lap of the autocross or trackdrive event. The conclusion also discusses potential areas for future work and improvements.

Section A presents the content of attachment. The attachment contains the videos of the conducted experiments in a simulator.

Section 1.2

Formula Student

Formula Student is an international student engineering competition. The competition challenges students to design, build and race electric or combustion single-seat race cars. One of the most prestigious competitions is Formula Student Germany see Figure 1.1. In 2017, Formula Student Germany introduced a Driverless class where the teams compete with their autonomous vehicles.

Figure 1.1: Photo of all teams at Formula Student Germany 2022. The photo is taken from FSG web page [1].

The first team from the Czech Republic that successfully built an autonomous electric formula is eForce FEE Prague Formula. The team under the Faculty of Electrical Engineering of the Czech Technical University in Prague. In 2019 the team introduced the first driverless formula in the Czech Republic — DV.01 see Figure 1.4. The author of the thesis joined the team in 2021. So far, the best overall achievements for DV.01 are 3rd places at FS Czech in 2021 and 2022.

The Formula Student Driverless competition includes both static and dynamic events. The static events typically involve the car's design and business aspects of the team, while the dynamic events evaluate the car's racing and autonomous performance. During the dynamic events, the car drives on a track marked by traffic cones of different colors see Figure 1.2. The big orange cones with two white stripes denote the start and end of the track. Exit and entry lanes are marked with a small orange cone with a single white stripe. The small blue and yellow cones indicate the track's left and right sides.

Figure 1.2: The traffic cones used in dynamic events. The picture is taken from [2].

There are four dynamic events: acceleration, skidpad, autocross, and trackdrive. The acceleration event evaluates the formula's acceleration performance on a straight track of 75 meters long. The skidpad event tests the formula's ability to navigate a figure-eight-shaped track while maintaining a high speed. The last two events are quite similar: the objective for both is to complete the event as quickly as possible without knocking over any cones or going off the track. The main difference is that the autocross has only one lap, while the trackdrive has 10 laps. The track for both events is the same. In the thesis, we will develop the algorithm for autocross and for the first lap of trackdrive. After completion of the first lap of trackdrive, the car computes the optimal path and optimal speed profile for the last nine laps using algorithms described in [3].

Section 1.3
Autocross Event

The autocross layout is a closed loop circuit built according to the following guidelines from [4]:

- Straights: No longer than 80 m.
- Constant Turns: up to 50 m diameter.
- Hairpin Turns: Minimum of 9 m outside diameter (of the turn).

- Miscellaneous: Chicanes, multiple turns, decreasing radius turns, etc.
- The lap length is approximately 200 m to 500 m.

In autocross event, the track is unknown for the formula before it starts driving. The constraints for the track layout for autocross are defined in FSG rules [4] and FSG handbook [2]. The track must have a minimum width of 3 m, with blue cones placed on the right side of the car and yellow cones on the left side. The distance between the cones of the same color must not exceed 5 m. The start position is 6 m in front of the start line. After the successful completion of one lap, the car is required to come to a full stop within 30 m behind the finish line. Figure 1.3 outlines certain limitations on the cone placement for the autocross event. The distance between yellow or blue cones must not exceed 5 m. The width of the track must not be less than 3 m at any point. Four big orange cones with two white stripes indicate the start and finish line. The start position is 6 m before the start line. After the run, the vehicle must come to a complete stop within 30 m behind the finish line. Additionally, there are some dynamic events penalties: 2 seconds penalty is applied if the cone has been knocked over, and 10 seconds penalty is applied if the vehicle had all four wheels outside the track boundary as indicated by edge marking.

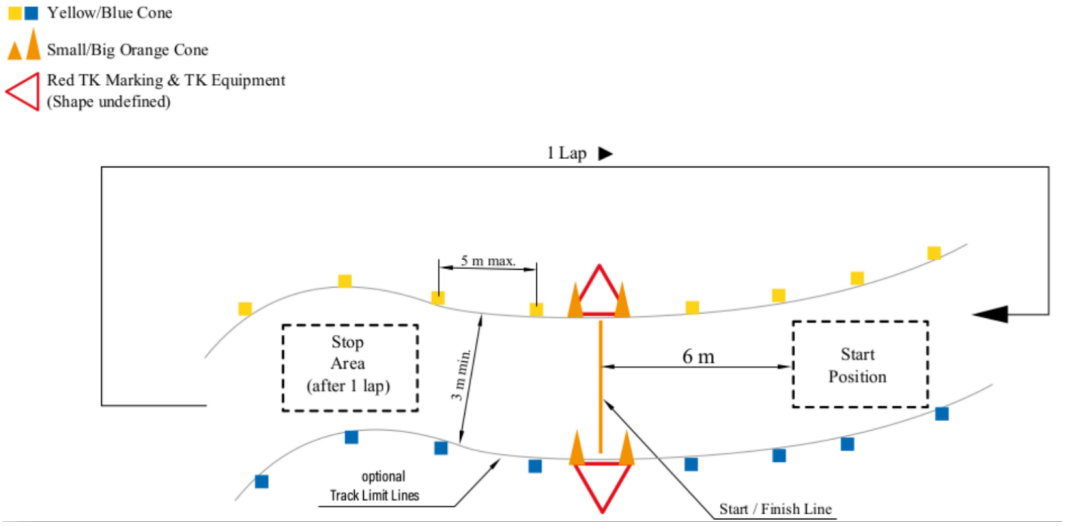


Figure 1.3: Limitations on the location of the cones in autocross event. The picture is taken from [2] and was modified for autocross specifications.

Section 1.4
Related Work

The problem of planning is fundamental and essential in robotics. It refers to the process of generating a sequence of actions to achieve a desired goal. Planning plays a critical role in enabling robots to perform complex tasks and navigate the robot in dynamic environments.

Planning can be divided into two parts: local and global planning. Global planning focuses on long-term decision-making and determining the overall trajectory from the starting point to the goal. On the other hand, local planning deals with short-term decision-making and handling immediate dynamic changes in the environment.

Global planning typically considers a global map or model of the entire environment with obstacles. Global planning algorithms aim to generate an optimal path while ensuring collision avoidance. Several algorithms have been developed for global path planning in robotics. A* is a widely used algorithm for finding the shortest path in a graph or grid-based environment [5]. Dijkstra's algorithm is another popular algorithm for finding the shortest path between nodes in a graph [6]. Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT) are sampling-based algorithms that construct road maps or trees to plan a path [7] [8]. These global planning algorithms provide efficient and optimized paths while considering obstacles and other constraints.

Local planning is responsible for navigating in the dynamic environment. The local planning algorithms ensure safe navigation in dynamic environments by considering real-time sensor information. The Dynamic Window Approach (DWA) is a local planning algorithm that searches through the robot's velocity and steering space to find a feasible and collision-free trajectory [9]. Another algorithm is an artificial potential field approach (APF) based on the concept of representing the robot's environment as a potential field [10]. The attractive forces guide the robot towards the goal and repulsive forces repel it from obstacles. The resulting path is planned through the regions of the least resistance in the potential field.

The thesis also focuses on speed planning for an autonomous car. Speed planning is responsible for finding the most suitable speed plan ensuring smooth and efficient motion while considering the road conditions, traffic, and vehicle dynamics. One research proposes a speed profile generation taking into account track curvature, and vehicle dynamics [11]. Another approach is using reinforcement learning (RL) techniques to generate a speed profile for autonomous vehicles [12]. The RL-based approaches mainly use the ability of RL algorithms to learn optimal policies through trial and error. The RL agent interacts with a racing simulator, receiving observations of the current state (position, speed, track geometry) and selecting actions (speed commands) to minimize the lap time.

In formula student driverless competition the teams use different approaches for local path planning: graph search algorithm with Delaunay triangulation [13], artificial potential fields [14], rapidly-exploring random trees [15] and others.

Horáček's research [3] on the optimal trajectory for autonomous student formula shows that the advantages of the optimal race line are not significant compared to the center line approach for before known track. The narrowness of the track results in a small difference between the optimal path and the center line approach. In some sections of the track, the optimal race line is planned very close to the track boundaries. Consequently, inaccuracies in other algorithms may result in knocking down the cones, applying a 2 seconds penalty per cone to the lap time. These inaccuracies can occur from miscalculations of cone positions in the perception node or in the path-tracking algorithm in the control node.

Considering the local planning, it is better to plan a reliable rather than optimal path to prevent knocking down the cones. As a result from Horáček's study [3], in this thesis we introduce the local path planning algorithm described in Section 2.2 based on the center line approach, minimizing the possibility of knocking down the cones.

The local speed profile proposed in Section 2.3 is based on the algorithm from [11]. The algorithm from [11] is responsible for global speed planning, but we adjust this algorithm for local speed planning.

Section 1.5

DV.01

The DV.01 is the first autonomous electric student formula in the Czech Republic. It was developed in 2019 and is based on the mechanical design of the FSE.07 monopost. The electrical system was completely reworked to allow autonomous driving. An autonomous system is the main feature of DV.01 that is responsible for observing the environment using sensors, planning a path, and sending the proper commands to the car's actuators to follow a planned path and maintain a planned speed. The DV.01 uses various sensors, including the Ouster OS1 LiDAR, a Stereolabs ZED camera, and an SBG Systems Ellipse2-D inertial navigation system (INS), to observe environment and collect data. Figure 1.4 shows the DV.01 with the sensors.

Figure 1.4: Photo of the DV.01 with the sensors.

The autonomous system includes the algorithms that are responsible for the driverless abilities of DV.01. The autonomous pipeline consists of separate modules. For communication between the modules, eForce members implemented in Python programming language the system based on the Robot Operating System (ROS). A diagram of the autonomous pipeline with modules and their connections is shown in Figure 1.5.

The autonomous pipeline 1.5 starts with the data input provided by the sensors. RGB images from the camera are processed by the convolutional neural network YOLOv3 introduced in [16]. The neural network detects the traffic cones shown in an image. When the cones are detected, the cone pixels are subsequently projected into the car coordinate system via a homography mapping between the image and the car. This image cone detector algorithm is described in [17]. Another option to detect the cones is using a LiDAR detector presented in [18]. Nevertheless, for now the LiDAR is only used for camera calibration. The inertial navigation system

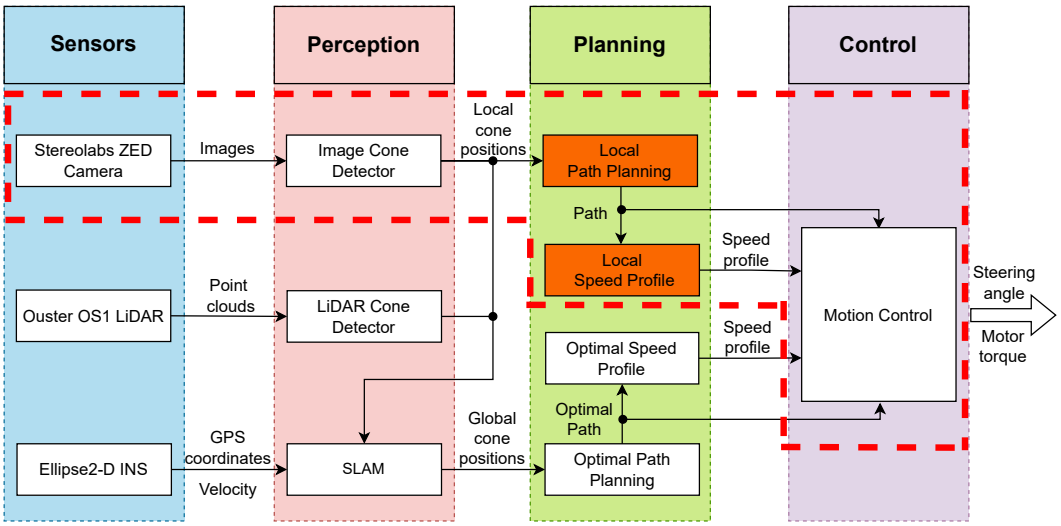


Figure 1.5: Diagram of the autonomous pipeline. Delimited area by a rad dashed line is responsible for the first lap. The highlighted orange modules in planning are introduced in this thesis. Local path planning and local speed profile are presented in Section 2.2 and Section 2.3.

(INS) provides the formula's position, orientation, and speed.

For the first lap, we use an image cone detector introduced in [17] that generates the cones positions in the vehicle coordinate system. The vehicle coordinate system is a two-dimensional space, where the x-axis is a forward/backward direction and the y-axis is a left/right direction in which both axes are measured in meters. After, we use these cone positions for the local path planning introduced in Section 2.2 to generate a center line trajectory between the blue and yellow cones. The planned trajectory is sent to the speed profile described in Section 2.3 and the motion control algorithm presented in [19]. Based on the planned trajectory, the speed profile provides a speed plan to motion controller. In motion controller, we have a nonlinear lateral regulator based on Stanley algorithm from [20] to transform the planned path into a sequence of steering angles, and a longitudinal regulator to calculate motor torque based on the planned speed profile.

For optimal lap planning, we use simultaneous localization and mapping algorithm (SLAM) from [21]. The output of SLAM is, the absolute cone positions in the world coordinates, used for optimal path planning introduced in [3]. The optimal path is sent to the optimal speed profile and motion control. The optimal speed profile generates an optimal speed plan and sends it to motion control. The final step in a pipeline shown in Figure 1.5 is to send the steering angle and motor torque commands via the controller area network (CAN) bus to the appropriate electronic control units that execute the commands.

Section 1.6
Thesis Contribution

The thesis describes an algorithm for an autonomous student formula to successfully complete the first lap of a Formula Student Driverless competition. The algorithm involves perception from [17], local planning, and control algorithms from [19] to

navigate the car on the unknown track.

The thesis makes the following contributions to the field of autonomous racing. It focuses on the planning for the first lap, also known as a local planing. The local planing consist of two algorithms: local path planning introduced in Section 2.2 and speed profile generation presented in Section 2.3. A local path planning algorithm is designed to generate a smooth and feasible path for the autonomous vehicle during the first lap. A local speed profile algorithm calculates the optimal speed for the planned path to maximize the car's performance, taking into account the vehicle's dynamics, path curvature, and maximum acceleration and deceleration limits of the car. The proposed algorithms were tested and validated using a simulator. The results achieved in this research can further drive improvements in autonomous vehicle performance and contribute to the future development of autonomous systems.

Chapter 2
Algorithm for the first lap

This chapter describes the algorithm to complete the first lap for the autonomous student formula. In Section 2.1 we give an overview of the algorithm for the first lap that involves perception, local planning and control nodes. The primary objective of this thesis is to present local planning that consists of two developed algorithms – local path planning described in Section 2.2 and local speed profile proposed in Section 2.3. Both methods have the main algorithms presented in Section 2.2.1, Section 2.3.4, and auxiliary algorithms presented in the remaining sections.

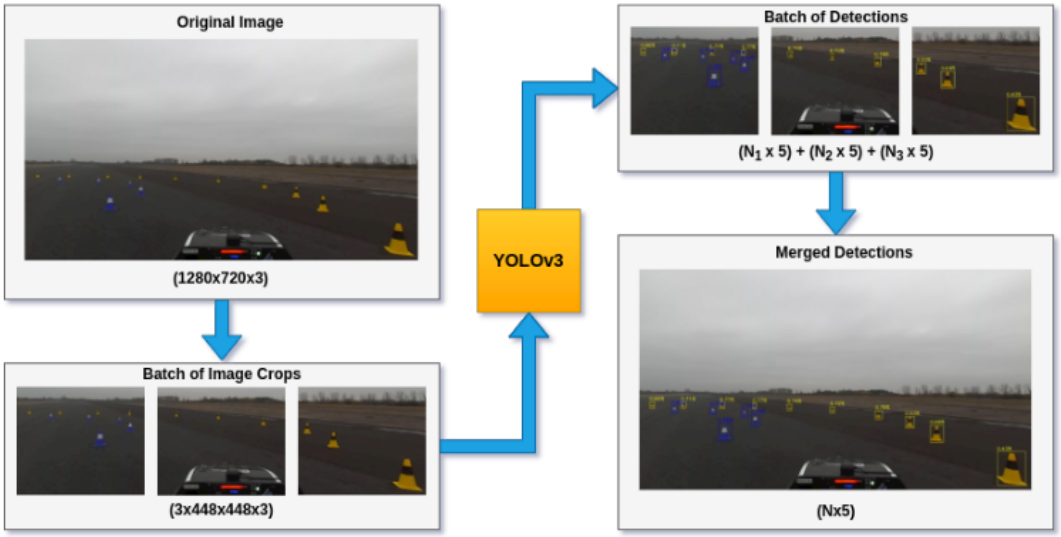
Section 2.1
Overview

The algorithm for the first lap is designed for autocross event, discussed in Section 1.3, and for the first lap of trackdrive event. It involves processing an image from the camera in the perception node, calculating a path and speed profile in the planning node, and generating steering, acceleration, or braking commands in the control node. The algorithm is responsible for driving on the unknown track and successfully completing a lap with the goal of minimizing the lap time.

The area marked by a red dashed line in Figure 1.5 is a part of the autonomous pipeline responsible for the first lap. The autonomous car uses a stereo camera as the main sensor for the first lap. The RGB images from the camera are sent to the image cone detector proposed in [17]. It has three steps: cone detection, cone center estimation, and cone localization. Consequently, the image cone detector detects the cones on the image, then the cone centers are estimated from its bounding boxes, and finally, the homography matrix is used to compute the positions of the cones relative to the vehicle coordinate system. The perception process using an image cone detector from [17] is shown in Figure 2.1. It is necessary to mention that the image cone detector was fully developed by Šíp [17]. Thus, it is not the author's work, and the included figures are taken from [17] for better problem understanding, because the output of the image cone detector is used for local path planning presented in Section 2.2.

The next step in the autonomous pipeline is local planning. The thesis focuses on planning for the first lap, also known as local planning. In this thesis for autonomous student formula, we refer to local planning as the process of path planning and speed profile generation In Section 2.2 we introduce local path planning. Local path planning takes the cone positions in the vehicle coordinate system from the image cone detector and produces a smooth center-line path between blue and yellow cones. Then, the local speed planning introduced in Section 2.3 plans a speed profile for a given path. A speed profile is calculated taking into account vehicle dynamics and limits of the car.

The last part of the pipeline is the control algorithm. We use the motion control algorithm that was introduced in [20]. It calculates the appropriate steering angle and motor torque based on the path and speed profile generated by the planning node. The control algorithm is not the author's work and was implemented by a former eForce member – Hynek Zamazal. Finally, the resulting output of the control algorithm is transmitted to the electronic control units through the CAN bus.



(a) Cone detection process

(b) Cone center estimation process

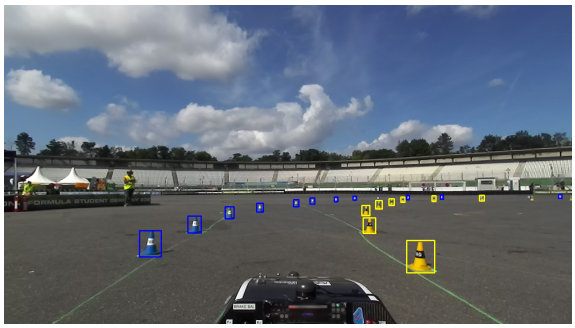
(c) Cone localization process

Figure 2.1: Perception process using image cone detector proposed in [17]. Firstly, the image is split into several sub–crops, on each sub–crop the cones are detected using YOLOv3 detection model. The next step is to merge back the detections into the original image. Then, the cone centers are estimated from its bounding boxes. Finally, the cone centers are projected into the car coordinate system using the homography matrix, creating a local map of the scene. All pictures are taken from [17].

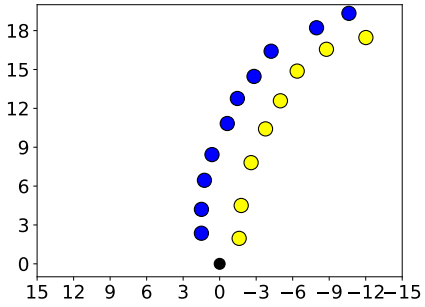
Section 2.2

Local Path Planning

In this section, we introduce a local path planning algorithm. Local path planning, also known as reactive path planning, is a type of path planning algorithm that focuses on short-term planning and plans a path in a dynamic environment. In our case, the environment is the unknown track for the autocross event discussed in Section 1.3. Local path planning produces a path for a given frame. A frame refers to an individual image. It represents a snapshot of the visual information captured by a camera at a specific point in time. The cone detector algorithm processes each frame to detect the cones. As a result, the local path planning needs to generate a path for each frame in order to enable the car to navigate through the dynamic environment. An example of a frame is shown in Figure 2.2.



(a) The camera image with detected cones. Detected cones are enclosed within bounding boxes of their corresponding colors.



(b) The plot with the positions of detected cones in the car coordinate system. It is a visual representation of input for local path planning. Blue and yellow cones are visualized as a point with the corresponding color. The black dot represents the car position.

Figure 2.2: A frame visualization at FS Germany 2021 with detected cones and their positions in the car coordinate system. In Figure 2.2b the axes of the car coordinate systems were swapped for better cone visualization, aligning them with the corresponding camera image. Figure 2.3 is an example showing the path planning with the original axes.

According to research conducted by Horáček [3] on the optimal trajectory for autonomous student formula, the benefits of following the optimal race line are not significant compared to the center line approach. The narrowness of the track contributes to a small difference between the optimal global path and the center-line approach. In certain sections of the track, the optimal race line may be planned very close to the track boundaries. Hence, inaccuracies in cone positions in perception or inaccuracies in the path-tracking algorithm can lead to knocking down the cones. As a consequence, each knocked-down cone results in a 2 seconds penalty added to the lap time. Taking into account this factor, we prioritize reliability rather than path optimality for local path planning. As a result, in the next section, we propose a local path planning algorithm based on the center-line approach that aims to minimize the chance to knock down the cones.

Subsection 2.2.1

Path Planning

In this section, we introduce a path planning algorithm described in Algorithm 1 that generates a smooth center-line path between blue and yellow cones. This path planning is an improved version of the old algorithm developed by eForce alumni Matěj Zorek. The old algorithm was the first version of path planning and was used for local and global planning. But it had problems with local path planning and the author of the thesis reworked the algorithm by adding more functionality such as using big orange cones see Section 2.2.2, filtering the cones see Section 2.2.3, checking the center placement see Section 2.2.4, and smoothing the path see Section 2.2.6. As discussed earlier in Section 2.2, the main objective of path planning is to produce a reliable path in the first lap. The planned path is not optimal, but the primary goal is to plan a path that does not lead to going off the track.

The algorithm is iterative and plans a smooth path for the given frame. An example of a frame is shown in Figure 2.2. Assume the track that satisfies the constraints for the autocross event described in Section 1.3. The input is the cone positions from the image cone detector. The positions of the cones are in the vehicle coordinate system, a two-dimensional space. The output of the algorithm is a smooth path. The path $(\boldsymbol{p_i})_{i=0}^{P}$ of length P is a sequence of path points $\boldsymbol{p_i} = (x_i, y_i)$ in the vehicle coordinate system. A path point is a center point between blue and yellow cones.

Suppose the cone detector correctly detected at least two cones, without counting big orange cones. Let $Blue$, $Yellow$, and $BigOrange$ be the positions of detected blue, yellow, and big orange cones. Let $Path$ $(\boldsymbol{p_i})_{i=0}^{P}$ be the center line or original path and $SmoothPath$ $(\boldsymbol{s_i})_{i=0}^{P}$ be a smooth path. Suppose the origin of the vehicle coordinate system is the initial position of the car $\boldsymbol{p_0} = (0, 0)$. The positions of the cones are fixed and the future position of the car changes and is represented as a path point $\boldsymbol{p_i}$ at iteration i. In the first iteration set i to 0 and append $(0, 0)$ to $Path$. Then if it is possible, extend $Blue$ and $Yellow$ cones with additional cones see Section 2.2.2. In the case when $Blue$ or $Yellow$ cones do not contain any cone, fill in missing cones using Algorithm 2. If $BigOrange$ cones were detected, assign them to $Blue$ or $Yellow$ cones using Algorithm 3. The next step is to sort and filter $Blue$ and $Yellow$ cones. The sorting and filtering process is described in Section 2.2.3. After sorting and filtering $Blue$ and $Yellow$ cones, assign them to $\hat{B}$ and $\hat{Y}$. Repeat the next block of procedures until $\hat{B}$ or $\hat{Y}$ is not empty: calculate the next center point see Section 2.2.4, compute the cones in front of the last added path point see Section 2.2.5 and assign them to $\hat{B}$, $\hat{Y}$, increment i. The final procedure is smoothing the $Path$ described in Section 2.2.6. The output of the path planning algorithm is a smooth path that is provided to the local speed profile described in Section 2.3 and the control algorithm. The example of the planned path for the start position is shown in Figure 2.3 and Figure 2.4 shows a planned path for the frame from Figure 2.2. In Section 3.3 we test and evaluate the robustness of the proposed algorithm – Algorithm 1.

Algorithm 1 Local path planning algorithm

Input: Blue, yellow and big orange cone positions
Output: *SmoothPath*

Add *(0, 0)* to *Path*
If there are no *Blue* or *Yellow* cones, fill missing *Blue* or *Yellow* cones
Assign *BigOrange* cones to *Blue* or *Yellow* according to the closest cone
Sort *Blue* and *Yellow* cones
Filter *Blue* and *Yellow* cones
$\hat{B} \leftarrow$ *Blue*, $\hat{Y} \leftarrow$ *Yellow*
while $\hat{B}$ is not empty and $\hat{Y}$ is not empty **do**
    Find the closest b, y cones from $\hat{B}$, $\hat{Y}$ to the last point of *Path*
    Calculate *center* between b and y
    Check correctness of the *center* placement
    Add *center* to *Path*
    $\hat{B} \leftarrow$ Compute *Blue* cones in front of the last added path point
    $\hat{Y} \leftarrow$ Compute *Yellow* cones in front of the last added path point
end while

Compute *SmoothPath* using *Path*
return *SmoothPath*

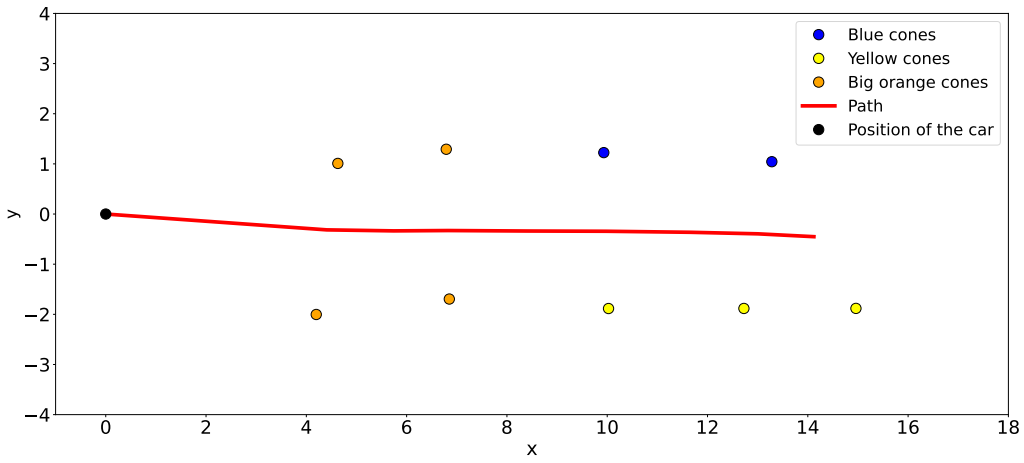
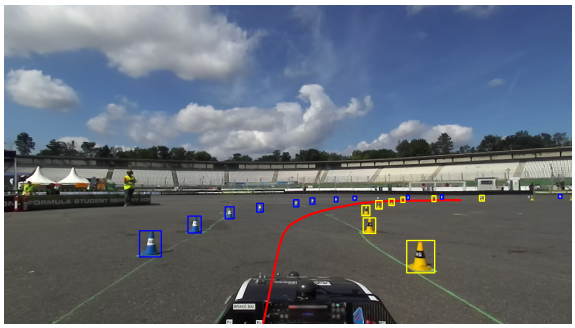
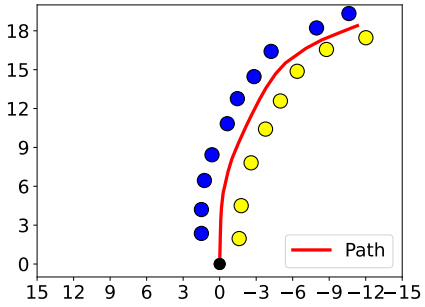


Figure 2.3: Illustration of the planned path using Algorithm 1 for the start position at autocross event see Figure 1.3. The start position is 6 meters before the start line. The start line locates between the big orange cones.



(a) Camera image with detected cones and the planned path shown as a red line. The path was firstly planned in the car coordinate system and after projected to the camera image using homography [17].



(b) Plot of blue and yellow cone positions with the planned path in the car coordinate system. It is a visual representation of the output of local path planning.

Figure 2.4: Visualization of the planned path for the frame from Figure 2.2. The frame was captured on FS Germany 2021 track.

Subsection 2.2.2

Additional Cones

In this section, we discuss the situation when the camera angle is not wide enough to capture inner cones, and how to use the big orange cones for path planning, because the path planning algorithm uses only blue and yellow cones to calculate a center between cones.

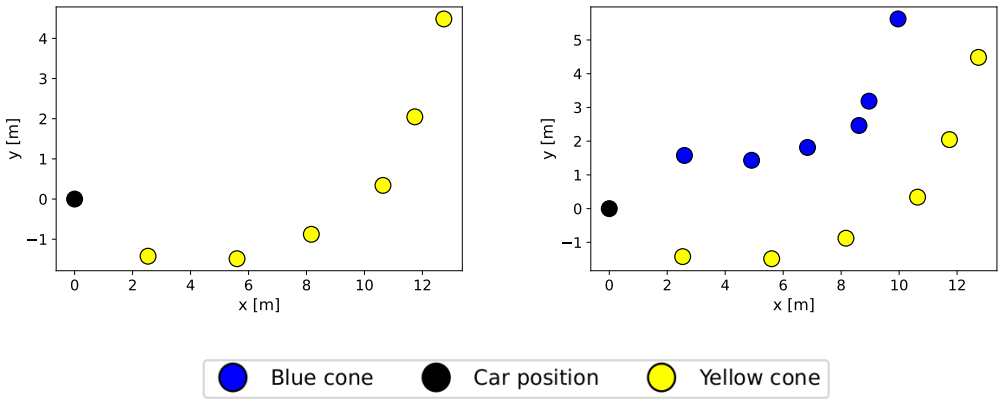


Figure 2.5: The situation with missing inner cones. On the left subfigure, the blue cones are missing. The right subfigure shows filled blue cones using Algorithm 2.

It is common that the track has at least one sharp turn and at some moment the camera does not capture the inner cones see Figure 2.5. To deal with situation we propose Algorithm 2 that describes how to fill missing cones. The objective of the algorithm is to fill in missing cones based on knowing the positions of cones with the opposite color. Fill missing cone function takes blue, yellow cones and track width. The first step is to detect what cones we have and what cones we need to fill. Then we sort available cones using Algorithm 4. Let $Full$ be the sorted cones and $ToFill$

be the missing cones. After we iterate through $Full$ cones. Take the current cone point $\vec{v_0}$ and the next sorted cone point $\vec{v_1}$. We get $\vec{v}$ by computing a vector $\vec{v_0} - \vec{v_1}$ and normalizing it. Only for the last iteration, $\vec{v_0}$ is the previous sorted cone point and $\vec{v_1}$ is the current cone point. To compute $\vec{w}$ use the following instructions: if we need to fill in blue cones, rotate $\vec{v}$ by 90°; to fill in yellow cones, rotate $\vec{v}$ by 270°. Then, add the current cone point and $\vec{w}$ multiplied by track width to get a new cone point. Append the new point to $ToFill$. Increment the iteration step. Finally, return the missing cones. Figure 2.6 shows the steps of Algorithm 2.

Algorithm 2 Fill missing cones

Input: Blue, yellow cones and filling cone distance
Output: Filled blue and yellow cones

1: **function** FillMissingCones(B, Y, $TrackWidth$)
2: **if** B is empty **then**
3: $FullYellow$=true
4: **else**
5: $FullYellow$=false
6: **end if**
7: **if** $FullYellow$ is true **then**
8: $Full \leftarrow SortCones(Y)$, $ToFill \leftarrow Blue$
9: **else**
10: $Full \leftarrow SortCones(B)$, $ToFill \leftarrow Yellow$
11: **end if**
12: $N \leftarrow$ size of $Full$
13: **for** $i \leftarrow 0$ to N **do**
14: **if** $i = N - 1$ **then**
15: $FromIdx \leftarrow N - 2$, $ToIdx \leftarrow N - 1$
16: **else**
17: $FromIdx \leftarrow idx$, $ToIdx \leftarrow idx + 1$
18: **end if**
19: $\vec{v_0} \leftarrow Full[FromIdx]$, $\vec{v_1} \leftarrow Full[ToIdx]$
20: $\vec{v} \leftarrow \dfrac{\vec{v_1} - \vec{v_0}}{\|\vec{v_1} - \vec{v_0}\|}$
21: **if** $FullYellow$ is true **then**
22: $\vec{w} \leftarrow \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \vec{v}$ ▷ 90° rotation
23: **else**
24: $\vec{w} \leftarrow \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \vec{v}$ ▷ 270° rotation
25: **end if**
26: $NewCone \leftarrow Full[i] + TrackWidth \cdot \vec{w}$
27: Add $NewCone$ to $ToFill$
28: **end for**
29: **if** $FullYellow$ is true **then**
30: **return** $ToFill$, $Full$
31: **else**:
32: **return** $Full$, $ToFill$
33: **end if**
34: **end function**

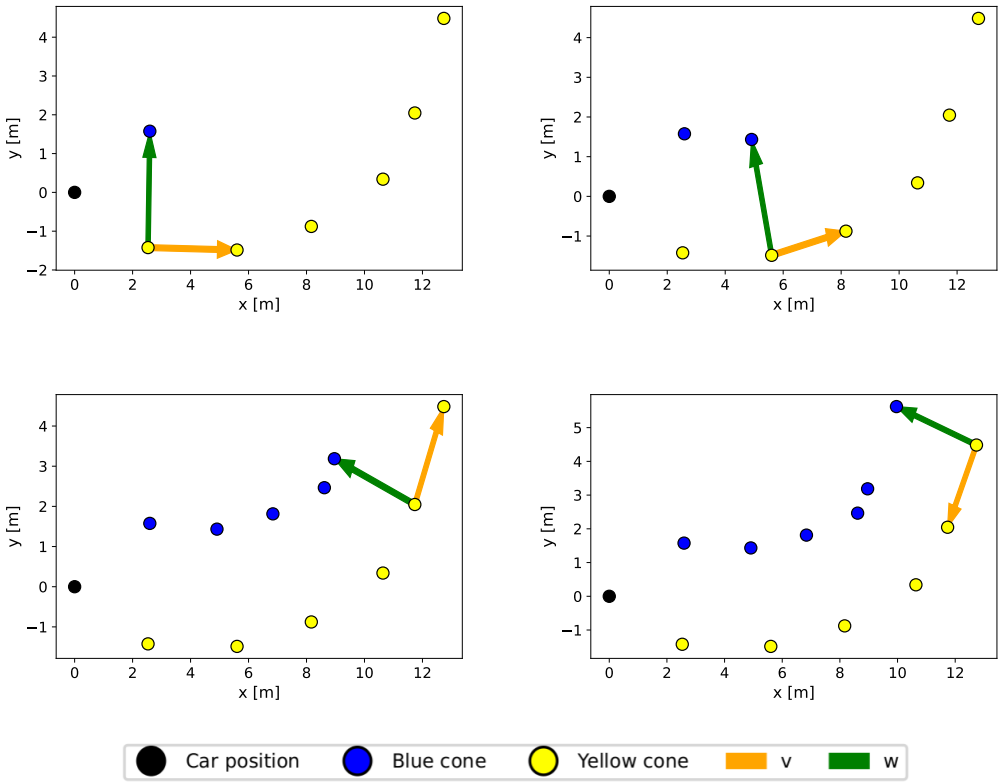


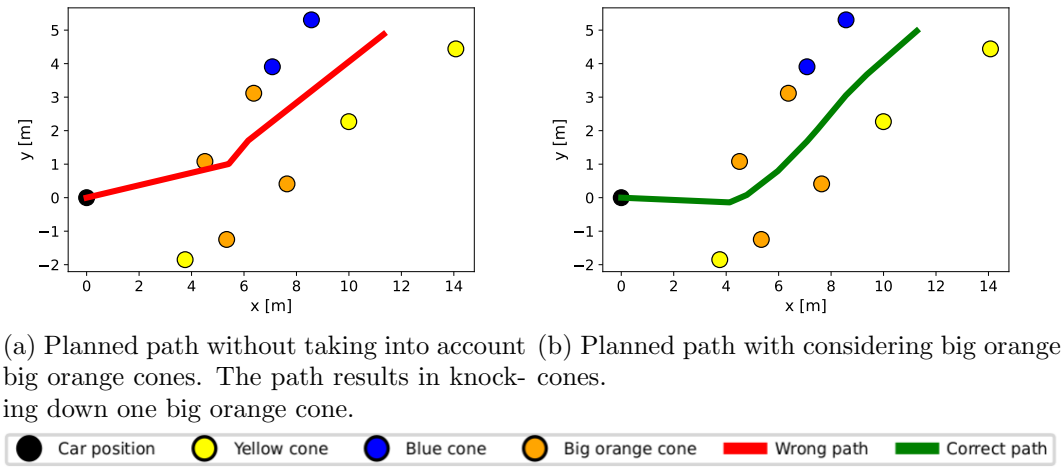
Figure 2.6: Process of filling missed inner cones. The first, second, pre-final, and final steps are illustrated in the subfigures. The angle between $\vec{v}$ and $\vec{w}$ is 90°.

The next situation we focus on is the start or the end positions for autocross. The start/finish line is determined by big orange cones see Figure 1.3. Based on team experience from the previous races, there is a possibility that the car can knock down some of the big orange cones because the old version of path planning did not count with them see Figure 2.8. Also, for better center-line planning at the start see Figure 2.3, it is good to take into account big orange cones. We propose Algorithm 3 for planning center points between big orange cones. The idea is to assign big orange cones to blue or yellow cones for the next steps in path planning computation, where the big orange cones are used as blue or yellow cones.

Algorithm 3 Assign big orange cones

Input: Blue, yellow and big orange cones
Output: Extended blue and yellow cones

```
1: function AssignOrangeCones(Blue, Yellow, BigOrange)
2:     for all o ∈ BigOrange do
3:         if Any b ∈ Blue is closer to o than any y ∈ Yellow then
4:             Add o to Blue
5:         else
6:             Add o to Yellow
7:         end if
8:     end for
9:     return Blue, Yellow
10: end function
```



(a) Planned path without taking into account big orange cones. The path results in knocking down one big orange cone.

(b) Planned path with considering big orange cones.

Figure 2.7: An example of the importance to take into account the big orange cones for the path planning.

Subsection 2.2.3

Sort and Filter Cones

The sorting process is important for correctly determining track boundaries. We sort separately blue and yellow cones identifying the next closest cone of each color iteratively see Algorithm 4. The closest cone to the initial position of the car is always the first cone in the sorted list.

Algorithm 4 Sort cones

Input: Cones
Output: Sorted Cones

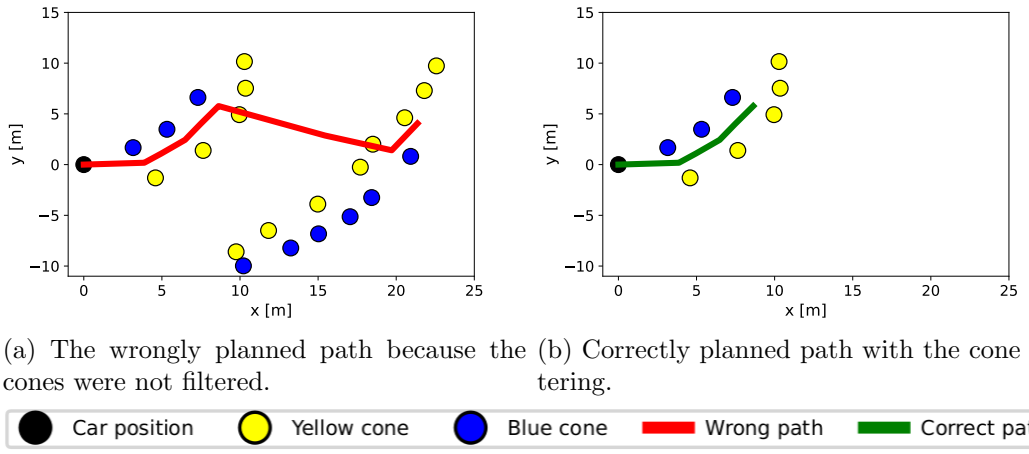
```
1: function SortCones(Cones)
2:     SortedCones is empty
3:     cone ← the closest cone from Cones to the car's initial position (0,0)
4:     Add cone to SortedCones and delete cone from Cones
5:     while Cones is not empty do
6:         Find the closest cone from Cones to the last cone from SortedCones
7:         Add cone to SortedCones and delete cone from Cones
8:     end while
9:     return SortedCones
10: end function
```

The next important procedure is to filter the cones. There are some parts of the track when a car can observe another section of the track and it can lead to the wrongly planned path and as a result, the car can go off the track. To prevent this situation we propose Algorithm 5 that solves this problem by filtering or deleting the redundant cones from another section of the track.

Algorithm 5 Filter cones

Input: Sorted cones
Output: Filtered cones

```
1: function FilterCones(Cones)
2:     Distances ← Compute distances among Cones
3:     FilteredCones is empty
4:     MaxDistBetweenCones ← 5  ▷ the max distance is defined in the rules [4]
5:     N ← size of Cones
6:     for i ← 0 to N do
7:         if Distances[i] > MaxDistBetweenCones then
8:             break
9:         end if
10:        Add Cones[i] to FilteredCones
11:    end for
12:    return FilteredCones
13: end function
```



(a) The wrongly planned path because the cones were not filtered.

(b) Correctly planned path with the cone filtering.

Figure 2.8: Illustration of the case when it is necessary to filter the cones, otherwise the path is planned out of track boundaries.

Subsection 2.2.4

Next Center Point

For computing the next center point we need to find the closest blue and yellow cone from $\hat{B}$, $\hat{Y}$ – the cones that are in front of the car see Section 2.2.5. Then we compute the center between the blue and yellow cones. The next step is to check the center placement. If the next center point is out of the car's steering range, this center is ignored and the path-planning process is stopped. And if the center point is reachable by the car's steering, it is appended to the path.

Subsection 2.2.5

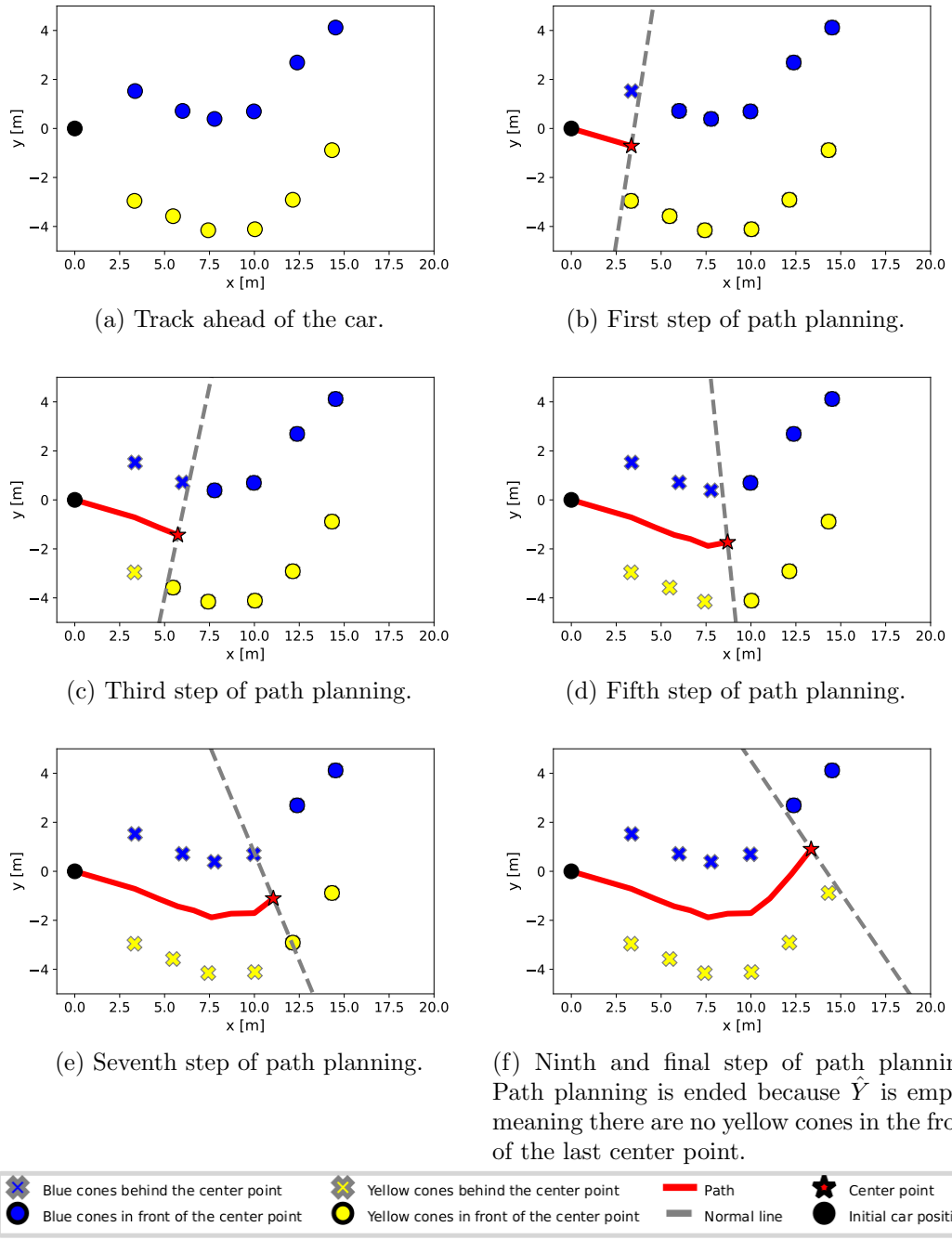
Cones in Front of the Car

After the center was appended, we need to identify the cones that are behind and in front of the car at a current point. As mentioned earlier, the center or path point is a future position of the car at specific path planning step. Algorithm 6 calculates the normal line (with line equation $y = kx + c$) to the vector of the last two points from path at the last path point. Then, based on the car's direction $\hat{B}$ and $\hat{Y}$ are computed. When $\hat{B}$ or $\hat{Y}$ is empty, the path planning is ended.

Algorithm 6 Compute cones in front of the car

Input: Blue, yellow cones and path
Output: Blue and yellow cones in front of the car

1: **function** ConesAheadOfTheCar($Blue$, $Yellow$, $Path$)
2: Find parameters k, c of normal line to $\boldsymbol{p_i} - \boldsymbol{p_{i-1}}$ at point $\boldsymbol{p_i} \in Path$.
3: $\hat{B} = \{\vec{b} = (x, y) \in Blue : sign(y - k \cdot x - c) = sign(-k)\}$
4: $\hat{Y} = \{\vec{y} = (x, y) \in Yellow : sign(y - k \cdot x - c) = sign(-k)\}$
5: **return** $\hat{B}$, $\hat{Y}$
6: **end function**



(a) Track ahead of the car.

(b) First step of path planning.

(c) Third step of path planning.

(d) Fifth step of path planning.

(e) Seventh step of path planning.

(f) Ninth and final step of path planning. Path planning is ended because $\hat{Y}$ is empty, meaning there are no yellow cones in the front of the last center point.

Figure 2.9: Example of path planning process.

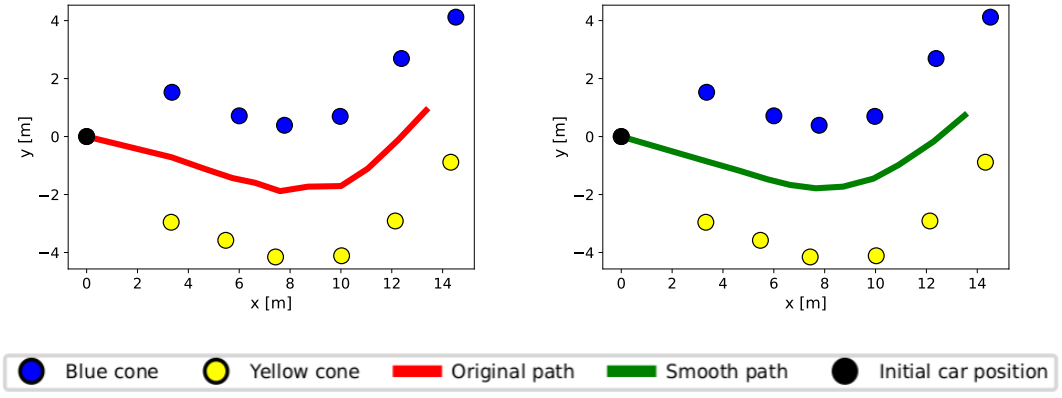
The path planning process is shown in Figure 2.9. At this moment we obtained the center-line path for the car. Then, we use a smoothing function to find a more continuous line for the speed profile stage. The speed profile algorithm benefits from lower angles between the adjacent segments of the path as it allows a more gradual speed response. We formulate this problem in Section 2.3.4 as optimization of a scalar function with respect to the initial center-line estimate.

Subsection 2.2.6

Path Smoothing

For the speed profile described in Section 2.3, it is better to have a smooth path, meaning to have a small path curvature or lower angles between the adjacent segments of the path. Equation (1) ensures the smooth path by minimizing the function, where $\|p_i - x_i\|$ is the distance between the waypoint x_i and corresponding path point p_i. This ensures that the new path stays close to the original path - a center line. The next term represents the curvature and serves to minimize the angle between two consecutive line segments in the path. This is accomplished by minimizing the dot product of the vectors representing the segments. The smaller angles mean a smoother path. The scalar β is a parameter that trades off these two terms.

$$\underset{x_1,\ldots,x_N}{\arg\min} \sum_i \|p_i - x_i\| - \beta \cdot \sum_N \frac{(x_{n+1} - x_n) \cdot (x_n - x_{n-1})}{|(x_{n+1} - x_n)| \cdot |(x_n - x_{n-1})|} \tag{1}$$



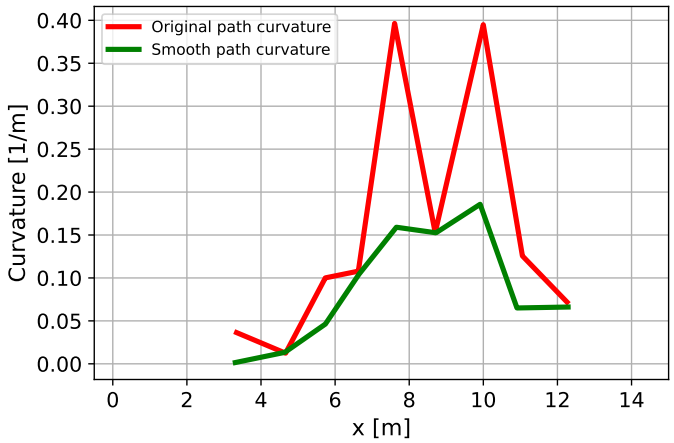


Figure 2.10: Path smoothing application on the planned path from Figure 2.9 and comparison of the original and smooth path curvatures at the planned path points.

Figure 2.10 shows the result of smoothing a path. Path smoothing is the last step in the local path planning described in Section 2.2. Finally, the smooth path is sent to the speed profile and control algorithms.

Section 2.3

Local Speed Planning

To win the autocross event it is not enough to drive only without knocking down any cones. The car should complete the first lap in the shortest possible time. Always driving at a constant high is a wrong approach because the car does go off the track in the turn due to the centrifugal force during turning. Therefore, it is important to adjust the speed based on the section of the track, in other words, based on the planned path curvature. Our motivation for the speed profile is the following: the car should speed up and go fast on the straights, and slow down before the turns to avoid going off the track or knocking down any cones.

In Section 2.3.4, we introduce an algorithm that computes a speed profile taking into account physical laws and the car's limits. Physical laws and the car's limits are discussed in Section 2.3.1 and Section 2.3.3. The speed profile computes a plan of target speed based on the path that is generated by the path planning algorithm described in Section 2.2.1. To determine if the path is smooth or contains a turn, we need to compute a path curvature, explained in Section 2.3.2.

Subsection 2.3.1

Circle of Forces

Vehicle motion is a complex process and nonlinear physical laws are applied in the real world that are hard to model. In this thesis, the model of the car is simplified. The car is represented as a single tire. The effect of the track bank angle and the track quality is neglected. The only part of the car that interacts with the track surface are tires. To stay on the track while maintaining a high speed, we need to consider that tires have a finite grip on the road surface. Excessive lateral and longitudinal forces cause the tires to lose traction with the surface and potentially result in skidding or sliding and in the worst case the car goes off the track.

The circle of forces, known as the traction circle or friction circle, describes the dynamic interaction between a vehicle's tire and the road surface. Equation (2) is a mathematical representation of the circle of forces, and Figure 2.11 is a visual representation. The friction circle points that the vector sum of the longitudinal force and lateral force of the tire is less than or equal to the product of the normal force and the friction coefficient.

$$F_x^2 + F_y^2 \leq (\mu F_z)^2, \tag{2}$$

where

F_x = longitudinal force,
F_y = lateral force,
μ = friction coefficient,
F_z = normal force.

Basically Equation (2) describes the driving limits of a car. A car is in contact with the road surface due to a normal force F_z, and the maximum force that the tires can transfer to the road surface is μF_z. The friction coefficient μ characterizes a tire-road interaction that is affected by several factors, including road conditions, environmental factors like temperature, wetness, and tire-related factors such as

the type of tire, its tread pattern, material, size, inflation pressure, temperature. Nevertheless, for our purpose, it is convenient to consider a constant value for μ that corresponds to driving on dry asphalt. Previous studies [22] suggest that selecting $\mu = 0.75$ is a reasonable and practical option. The longitudinal force F_x is responsible for acceleration or braking, while the lateral force F_y is responsible for left or right turning. In order for the car to perform stable driving, the lateral and longitudinal forces should remain within the limits defined by the friction circle see Equation (2).

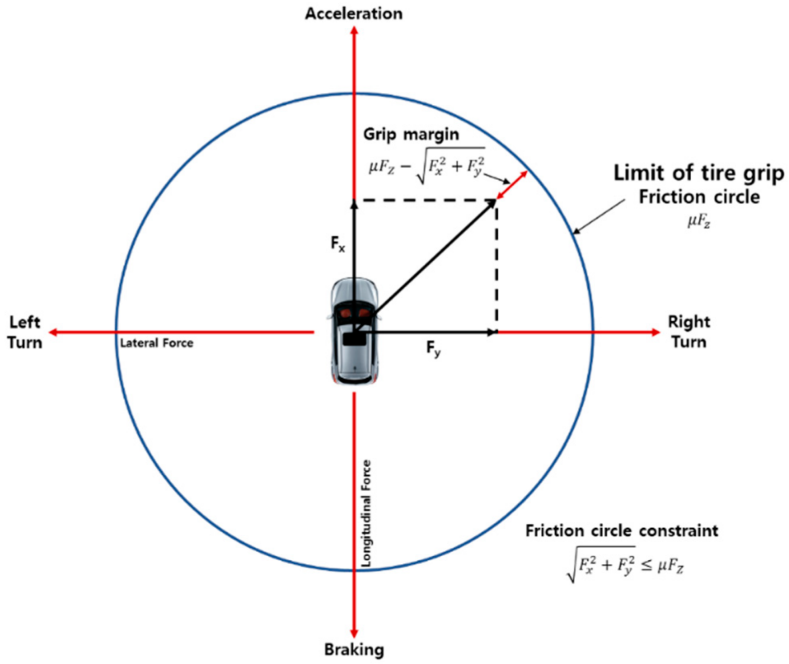


Figure 2.11: Circle of forces, also known as friction circle. The figure is taken from [23]

The available normal force F_z is primarily composed of the gravitational term that remains constant. However, when the vehicle is in motion, the normal force increases due to the aerodynamic downforce generated by the aerodynamic package. Our formula DV.01, see Figure 1.4, has an aerodynamic package designed to generate 977 N of downforce when the vehicle reaches its designated target speed of 16 m/s [24]. However, the target speed of 16 m/s is not always achievable by our autonomous formula. Horáček analyzed the importance of downforce for our formula [3] and shows that under average conditions for our formula, driving at a small constant speed, the aerodynamic downforce is approximately 313 N. This value corresponds to around one-third of the originally planned downforce provided by the aerodynamic package and roughly one-seventh of the gravitational term. Nevertheless, the aerodynamic package has the additional consequence of presenting aerodynamic drag in longitudinal force F_x [25], which acts as a resistance that slows down the vehicle. As a result, the advantages gained from the increased downforce are relatively reduced by aerodynamic drag. We decided to neglect aerodynamic effects, but we consider any aerodynamic improvements in the car's driving capabilities as a safety margin for stable driving.

Subsection 2.3.2

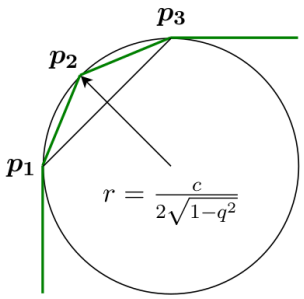
Path Curvature

For the first step of the speed profile, it is required to compute path curvature. Curvature is the scalar value by which a curve deviates from a straight line. We use a historical definition of curvature, where the curvature of a differentiable curve was defined through the osculating circle, which is the circle that best approximates the curve at a point. The curvature is the inverse of the radius of the osculating circle.

The path produced by Algorithm 1 is a sequence of points $(\boldsymbol{p_i})_{i=0}^{P}$ in the car's coordinate system. Thus, we use an algorithm to compute a curvature based on the osculating circle. Algorithm 7 from [3] computes a curvature of a circle inscribed to three points. Meaning the curvature is zero when the three points lie on the same segment, otherwise, the curvature is positive.

Algorithm 7 Curvature of a circle inscribed to three points. The algorithm is taken from [3]

Input: Points $\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$
Output: Curvature k of the inscribed circle



1: **function** ComputeCurvature($\boldsymbol{p_1}, \boldsymbol{p_2}, \boldsymbol{p_3}$)
2: $a = ||\boldsymbol{p_2} - \boldsymbol{p_1}||$
3: $b = ||\boldsymbol{p_3} - \boldsymbol{p_2}||$
4: $c = ||\boldsymbol{p_1} - \boldsymbol{p_3}||$
5: $q = \frac{a^2+b^2-c^2}{2ab}$
6: $k = \frac{2\sqrt{1-q^2}}{c}$
7: **return** k
8: **end function**

Subsection 2.3.3

Safe Speed

A speed profile is used to determine the most efficient way to complete a lap in the shortest lap time. In Section 2.3.4 we introduce a speed profile that is based on algorithm from [11]. The algorithm in [11] is used given a known track, and for DV.01 Horáček implemented this algorithm [3]. For local planning, the car does not known the entire track, but observes the track using sensors approximately up to 15–20 meters. Because of this constraint, we do some modifications to speed profile described in [11].

The first change is adding initial speed v_{init}. The speed profile proposed in Section 2.3.4 computes the speed in each frame. The initial speed refers to the speed at which the car is driving at a specific moment.

The next modification is the safe speed. Because in local environment, the car observes only a section of the track. Thus, when calculating the speed profile we need to take into account that the car does not know the next section of the track. We assume the worst case where the next section of the track is a sharp turn, also known as a hairpin turn. According to FSG rules [4] the outside diameter of hairpin turn is minimum 9 m. Thus, radius of the turn is minimum 4.5 m. We need to compute the safe speed for the turn, because the car must stay on the track while

turning. We refer the safe speed to a maximum cornering speed. To compute the maximum cornering speed, we start with the equation for centrifugal force F_c and equate it to the maximum lateral force F_{max} that the tires can generate.

Equation (3) is used to compute the centrifugal force, where m is a mass of the car, v is the maximum cornering speed and r is a radius of the turn.

$$F_c = \frac{\mathrm{m}v^2}{r} \tag{3}$$

To find the maximum lateral force that the tires can generate, we use Equation (4), where μ is the coefficient of friction and N is the normal force that is applied on the tires. Discussing earlier we neglect the aerodynamic downforce, therefore the normal force is equal to the weight of the vehicle see Equation (5), where g is the gravity acceleration. Substituting the expression for F_z from Equation (5) into Equation (4) we get Equation (6).

$$F_{\max} = \mu F_z \tag{4}$$

$$F_z = mg \tag{5}$$

$$F_{\max} = \mu mg \tag{6}$$

By equating the maximum lateral force F_{max} and the centrifugal force F_c see Equation (7), we get Equation (8)

$$F_c = F_{\max} \tag{7}$$

$$\frac{\mathrm{m}v^2}{r} = \mu mg \tag{8}$$

Equation (9) is derived from rearranging and simplifying Equation (8) to solve for the maximum cornering speed v.

$$v^2 = \mu gr \tag{9}$$

Finally, the formula for the maximum cornering speed is expressed in Equation (10):

$$v = \sqrt{\mu gr} \tag{10}$$

To compute the safe speed for DV.01 we use the values from Table 1: $\mu = 0.75$, $g = 9.8\,\mathrm{m/s^2}$ and $r = 4.5$ m. Substituting to Equation (10), we get the safe speed v_{safe} equals to $5.75\,\mathrm{m/s}$. The safe speed is always on the last index of the final speed profile.

Subsection 2.3.4

Speed Profile

In this section, we introduce the speed profile algorithm. The implementation is based on the description in [11]. Input for the algorithm is a path $(\boldsymbol{p_i})_{i=0}^{P}$ of length P, a sequence of path points in the car's coordinate system, produced by Algorithm 1. The speed planning algorithm consists of three passes, where a speed profile U_x is a plan of target speed. The first pass guarantees that the formula drives within the lateral and longitudinal limits, meaning the tire grip does always stay in the friction circle, also known as circle of forces, described in Section 2.3.1. The second pass

Parameter	Symbol	Value	Unit
Friction coefficient	μ	0.75	—
Formula mass	m	212	kg
Coefficient of lift	C_L	-3.82	—
Coefficient of drag	C_D	1.49	—
Reference aerodynamic area	A_{ref}	1.19	m^2
Maximum acceleration	a_{max}	2	m/s^2
Maximum deceleration	a_{min}	-4	m/s^2

Table 1: Physical parameters of DV.01 for the speed profile. Properties of the aerodynamic package come from an unpublished engineering design document [24].

adjusts the speed profile after the first pass to acceleration limits. The third pass is responsible for achievable breaking and as an input takes the speed profile after the second pass. The output of the speed profile, the sequence of speeds, is the result of the third pass.

The first pass of the speed profile is described in Algorithm 8. It determines the maximum allowable speed based on the constraints of the friction circle see Equation (2) with zero longitudinal force, and after simplification we get Equation (11). In circular motion, an object experiences an centrifugal force ensuring its continued movement along a curved trajectory. In our case, the lateral force F_y experienced by the car is equal to the centrifugal force expressed in Equation (3). The normal force F_z can be replaced with mg, because we neglect the downforce. Thus, we can rewrite Equation (11) to derive Equation (12). Since the radius of circle is the inverse value of the curvature, meaning $r = \frac{1}{k}$, we can substitute it into the Equation (12) and simplifying it we get Equation (13). Then in Equation (13) we express the speed V and obtain Equation (14) to calculate the speed in the first pass.

$$F_y = \mu F_z \tag{11}$$

$$\frac{\mathrm{m}v^2}{r} = \mu m g \tag{12}$$

$$v^2 k = \mu g \tag{13}$$

$$v = \sqrt{\frac{\mu g}{k}} \tag{14}$$

Algorithm 8 First pass (maximum speed limited by friction circle)

Input: Path $(\boldsymbol{p_i})_{i=0}^{P}$, initial speed v_{init} and safe speed v_{safe}
Output: Speed profile U'_x limited by initial speed, safe speed and friction ellipse

$U'_x[0] \leftarrow v_{init}$
for $s = 1$ **to** P-1 **do**
  $k = ComputeCurvature(\boldsymbol{p_{s-1}}, \boldsymbol{p_s}, \boldsymbol{p_{s+1}})$ ▷ use Algorithm 7
  $U'_x[s] = \sqrt{\frac{\mu g}{k}}$ ▷ use Equation (14)
end for
$U'_x[P-1] \leftarrow v_{safe}$

To determine the speed for the second pass, we use the equations of motion with constant acceleration: Equation (15) and Equation (16), where v is a final speed, v_0 is an initial speed, a is an acceleration and s is a displacement.

$$v = v_0 + at \tag{15}$$

$$s = v_0 t + \frac{1}{2} a t^2 \tag{16}$$

Express t in Equation (15):

$$t = \frac{v - v_0}{a} \tag{17}$$

Substituting Equation (17) into Equation (16), we get:

$$\begin{aligned} s &= v_0 \left(\frac{v - v_0}{a} \right) + \frac{1}{2} a \left(\frac{v - v_0}{a} \right)^2 \\ &= \frac{v_0 (v - v_0)}{a} + \frac{1}{2} \frac{(v - v_0)^2}{a} \end{aligned}$$

Eliminating the denominators in the equation above, we obtain:

$$2as = 2v_0(v - v_0) + (v - v_0)^2 \tag{18}$$

After simplification of Equation (18), Equation (19) is derived:

$$2as = v^2 - v_0^2 \tag{19}$$

The final step is to express the final speed v in Equation (19) and get Equation (20) for the second pass computation:

$$v = \sqrt{v_0^2 + 2as} \tag{20}$$

Algorithm 9 provides the description of the second pass. In the second pass the speed of a given point is determined by the speed of the previous point and the speed reached with the maximum acceleration a_{max} from the previous point. An important part of the second pass is that at every point, the value of $U_x[s]$ is compared to the corresponding value from the first pass and the lowest value is taken. After the second pass, the speed profile is achievable in terms of friction circle constraints and acceleration feasibility.

Algorithm 9 Second pass (acceleration)

Input: Speed profile after the first pass U_x, path $(\boldsymbol{p_i})_{i=0}^{P}$
Output: Speed profile U'_x limited by acceleration capabilities
  for $s = 1$ **to** P **do**
    $c = 2a_{max}||\boldsymbol{p_s} - \boldsymbol{p_{s-1}}||$
    $U'_x[s] = \min\left(U_x[s], \sqrt{U_x[s-1]^2 + c}\right)$ ▷ use Equation (20)
  end for

Algorithm 10 or the third pass ensures the feasibility of the braking. It is similar to the second pass, but in the third pass, we use a little bit different equation of motion - Equation (21), where a is the deceleration. Expressing v from Equation (21), we get Equation (22). During the third pass, we start from the last point on the

path and proceed in reverse, adjusting the speed at each point. The adjusted speed is determined by selecting the lower value between the speed from the second pass at the current point and the speed adjusted to the maximum deceleration at the following point, resulting in the final speed profile.

$$v^2 = v_0^2 - 2as \tag{21}$$

$$v = \sqrt{v_0^2 - 2as} \tag{22}$$

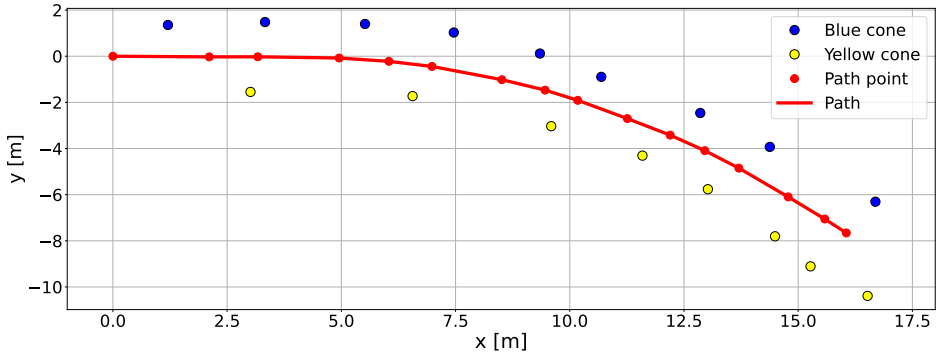
Algorithm 10 Third pass (braking)

Input: Speed profile U_x, path $(\boldsymbol{p_i})_{i=0}^{P}$
Output: Speed profile U'_x limited by braking capabilities
  for $s = P - 1$ **to** 1 **do**
    $c = 2a_{min}||\boldsymbol{p_s} - \boldsymbol{p_{s-1}}||$
    $U'_x[s-1] = \min\left(U_x[s-1], \sqrt{U_x[s]^2 - c}\right)$
  end for

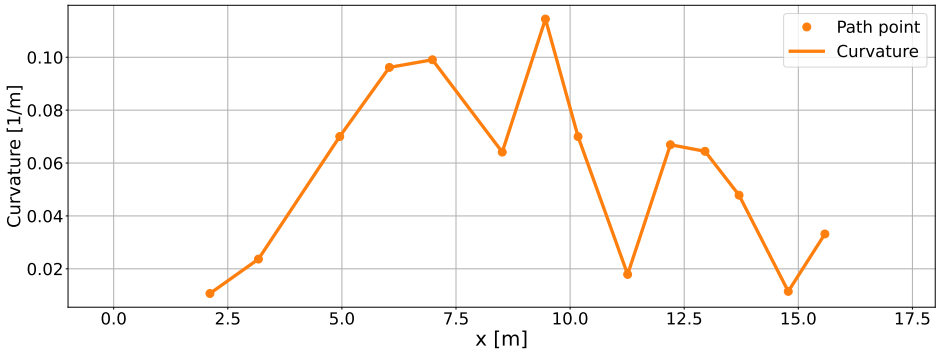
Figure 2.12 shows a speed profile computation for the computed path in Figure 2.12a. The path is a sequence of path point, where a path point was computed as a center between some blue and yellow cone. Figure 2.12b illustrates the computed path curvature. The curvature was computed in each path point using Algorithm 7. The curvature line shown in Figure 2.12b is not monotonic, because to compute a curvature at given point we use only three path points that is placed quite close to each other.

Figure 2.12c shows three speed profile passes, starting with an initial speed of 0 m/s. The speed gradually increases until reaching the penultimate point, as the last point is set to the safe speed v_{safe} computed in Section 2.3.3. Consequently, at the penultimate point, the car must drive at a slower speed compared to the computed speed during the second pass.

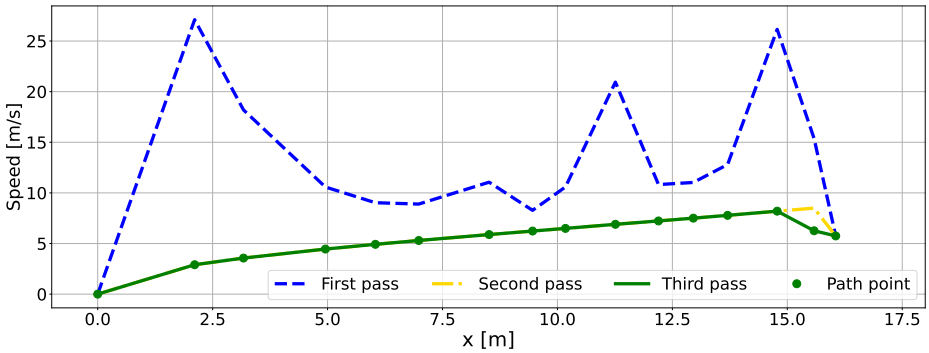
Figure 2.12d also shows three speed profile passes, but starting with an initial speed of 15 m/s. It can be observed that from the starting point to the path point where x is around 9 m, the speed is monotonically decreasing, because the car is driving through a turn, thus it should decrease the speed. From the path point where x is around 9 m until the path point with x equals around 15 m, the speed monotonically increases as the turn ends and the track enters a straight section. The speed continues to increase until the penultimate point, after which it decreases to maintain the safe speed discussed earlier.



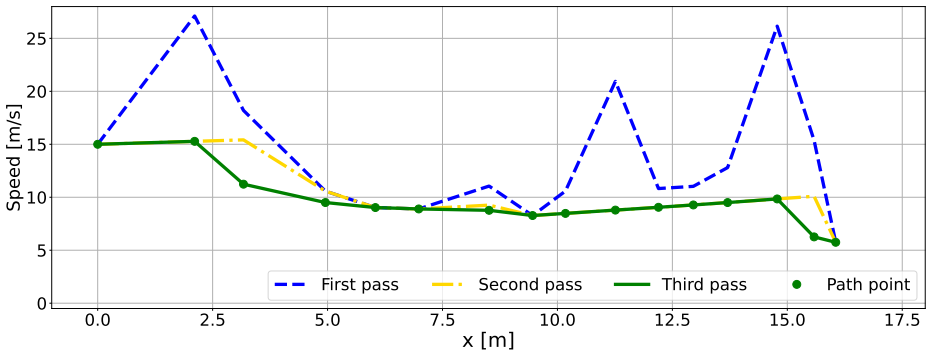
(a) The path points on a given path are used to calculate the speed profile.



(b) Path curvature of the given path from Figure 2.12a.



(c) Speed profile with $v_{init} = 0$ m/s.



(d) Speed profile with $v_{init} = 15$ m/s.

Figure 2.12: Computation of the speed profile described in Section 2.3.4 with 0 m/s and 15 m/s using parameters from Table 1. The third pass is a final speed profile.

Chapter 3
Experiments

In this section we validate the proposed path and speed planning algorithms from Section 2.2 and Section 2.3. Firstly, in Section 3.1 we discuss the dataset of tracks that we use for experiments. Then in Section 3.2 we describe a testing environment – eForce simulator. After we conduct path planning test in Section 3.3, where we mainly test a path planning robustness on different tracks. Section 3.4 presents the speed planning tests. We compare three different speed profiles: the constant 5 m/s profile (a conservative baseline approach), local speed profile introduced in Section 2.3 and optimal speed profile presented in [3]. The proposed algorithms were implemented in Python programming language.

Section 3.1
Data

The only data we need for path planning and speed profile experiments are the cone positions, in the car's coordinate system, received from the cone detector described in [17]. To ensure that the proposed algorithms, local path planning and speed profile, are robust and reliable, we use different sources for the tracks.

The first track source is the tracks from real Formula Student competitions. Specifically, FS Germany 2021, FS Italy 2022, and FS Czech 2022 tracks. We logged data from these competitions to be able to reproduce these runs in the future, and we use them in Section 3.3 to test the proposed path planning algorithm described in Section 2.2.1. We apply the algorithms for each frame by replaying the entire run. Then, we display the results in the camera image and the plot of cone positions in the car's coordinate system. An illustration of this visualization is shown in Figure 2.2, where on the left side there is a camera image with detected blue and yellow cones. The detected cones have bounding boxes of the corresponding cone color. On the right side, we can see the plot with detected cone positions in the car's coordinate system.

The next track source is real tracks from the team testing conducted in real-world conditions. One of the tracks from the team testing is Autocross 1. We have a log of the entire run, but we do not have the global map of the track. Thus, to test algorithms on this track we use the same method as for Formula Student tracks described earlier. Another track from team testing is shown in Figure 3.1 and we will refer to this track as Autocross 2. The track was reconstructed using SLAM [21] after completion of the first lap. We evaluate the path planning on this track in a simulator environment 3.2.

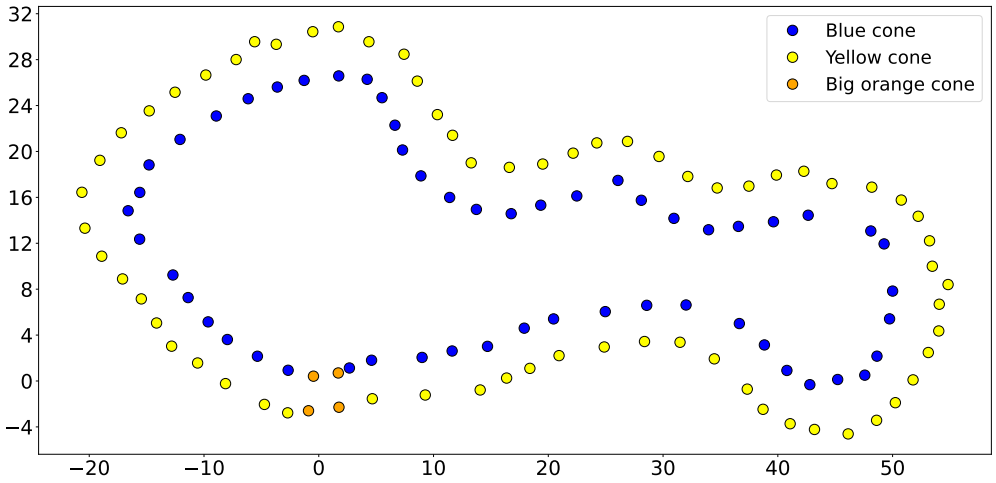


Figure 3.1: Visualization of Autocross 2 track. The track map was generated using SLAM [21]. The starting position is located in front of a group of big orange cones. The run direction is counterclockwise.

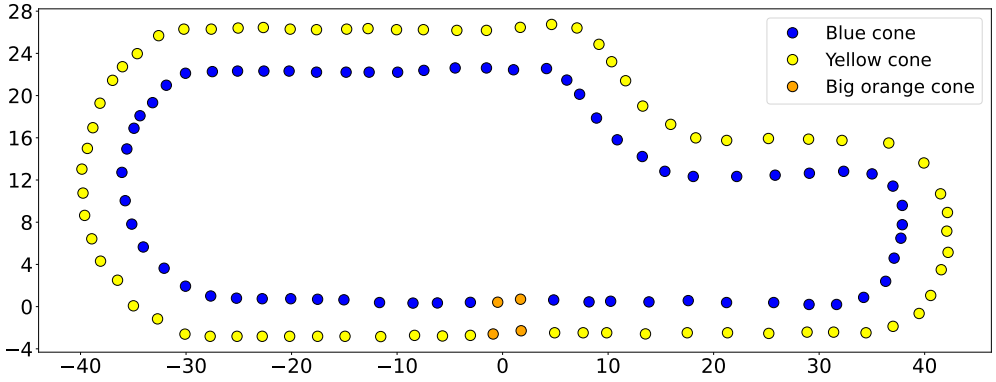


Figure 3.2: The layout of Autocross 3 track. The track layout was generated artificially using the graphic track editor described in Section 3.1. The direction of the run is counterclockwise.

The final source of tracks is synthetic or artificially created tracks. There are two ways how to generate them: eForce track generator or graphic track editor. The track generator assigns a predetermined number of points on a circle with a parametrizable radius and then changes their position using several Gaussian functions. The graphic track editor is used to generate a new track or to modify an existing track by adding or removing cones with a simple mouse click. These tools were developed by former eForce members – Tomáš Roun and Michal Horáček. An example of an artificial track called Autocross 3 is shown in Figure 3.2. This track

we mainly use to test the speed profile algorithm described in Section 2.3 in the simulator environment 3.2.

Section 3.2
Simulator

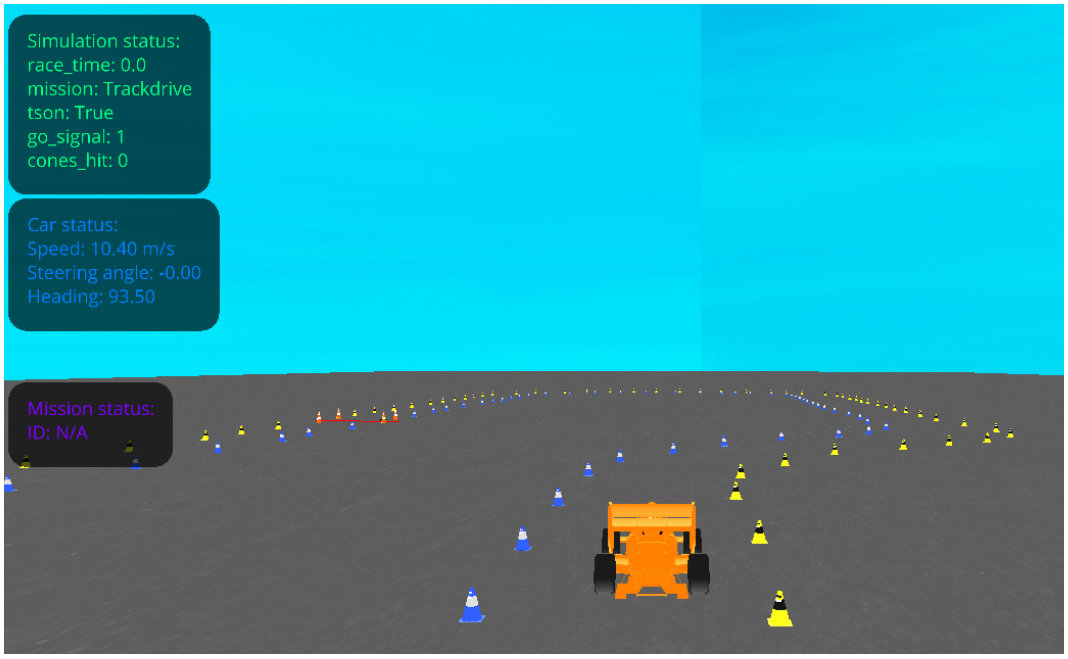


Figure 3.3: Screenshot of eForce Formula Student Simulator.

A simulator is an important tool for testing, validating, and improving the algorithms for DV.01 1.4. For team testing conducted in real-world conditions, it is required the participation of at least five team members. Because of organizational and time limitations, it is not possible to have more than a few testing events per racing season. Thus, the team members developed eForce Formula Student Simulator to test the developing algorithms. The simulator was started to develop at the end of 2022 and is still in the process of development. The simulator is not an author's work, the main contributor of simulator is Roman Šíp.

It simulates the outside world and the whole functionality of DV.01, including autonomous pipeline 1.5 see Figure 3.4. The simulator is end-to-end, meaning it provides a complete simulation of the car's controller area network (CAN), enabling seamless communication with the autonomous system (AS). Hence, during the simulation, the autonomous system is unable to distinguish whether it is operating in the car or in the simulator. The camera image is the only component that is not simulated in the simulator. Simulating realistic vision is challenging and computationally intensive. Instead, the simulator provides the AS with the relative positions of traffic cones to the car which are the output of the vision node in real life. By emulating vision in this manner, it becomes feasible to run the simulator on a personal laptop, making testing and development more convenient.

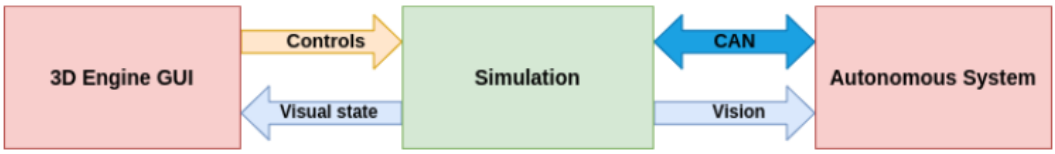


Figure 3.4: Simulator schema

The simulator consists of two distinct processes: the simulation process and the 3D engine visualization process. The schema of the simulator is shown in Figure 3.4. The simulation process manages the simulation state and performs all the necessary calculations, while the 3D engine visualization process receives the state of the simulation such as the car's coordinates, heading, wheels angle, and renders them accordingly. This design allows the main simulation loop to be very fast and the graphical engine runs at 60 FPS.

The simulator is developed using Python3 programming language, and the graphical user interface (GUI) uses a game library Ursina. Communication between the different processes of the simulator is implemented using ZeroMQ. To simulate CAN buses, we create virtual CAN ports and send and receive messages using the pycan library.

Advanced physics has not been implemented yet, but the simulator uses a kinematic model and a single track model [19] to mathematically represent the dynamics of the vehicle. The kinematic model is a simplified representation of a vehicle that considers only its position and orientation in space. The single-track model is a more advanced vehicle model that takes into account additional factors such as tire dynamics, lateral forces, and vehicle parameters. When the car is driving slowly, the speed is lower than 0.5 m/s, the kinematic model is used. Otherwise, the single-track model is used. While the existing physics implementation may not be fully realistic, it is sufficient for our purpose of testing and validation.

Section 3.3

Path Planning Tests

As mentioned in Section 3.1 to test path planning we use real Formula Student tracks 2.2 and the tracks obtained from team testing see Figure 3.1. For the Formula Student tracks and Autocross 1 we do not use the global map of a track, but instead, we apply path planning on the local environment for each frame by replaying the entire run. To test path planning on Autocross 2 track, we use a simulator described in Section 3.2. The algorithm is applied to each frame during the whole simulation.

Figure 2.4 and Figure 3.5 illustrate the visual representations of the path planning tests performed on the Formula Student track and in the simulator. Additionally, Figure 3.6 shows the path followed during the first lap in the simulator test.

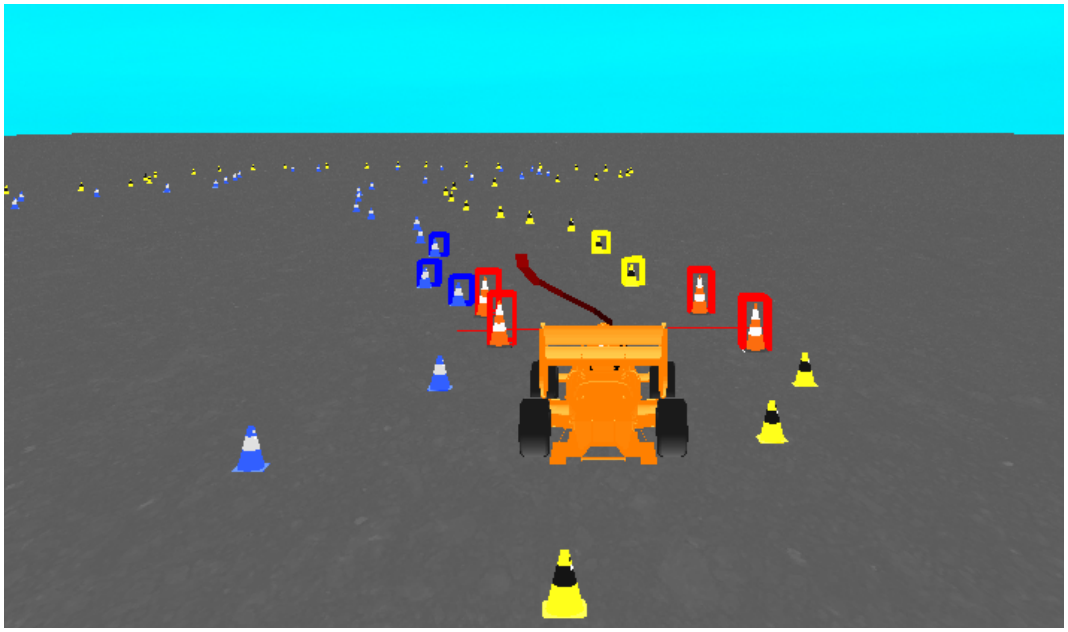


Figure 3.5: Illustration of the path planning test performed in the simulator 3.2 on Autocross 2 track 3.1. The car is situated at the starting position and begins its motion. Detected cones are visually represented by 3D bounding boxes, and the planned path is shown as a dark red line.

Firstly, we tested path planning accuracy on Formula student and real testing tracks. The results can be found in Table 2. Measuring the accuracy of path planning is significant for analyzing the robustness of the algorithm. For path planning tests we refer to path planning accuracy as an evaluation metric that measures the number of frames where the path was correctly planned in relation to the total number of frames. In the case of a wrongly planned path, it may lead to going off the track and ending the run. We consider a path that is planned to go beyond the track boundaries as a wrongly planned path.

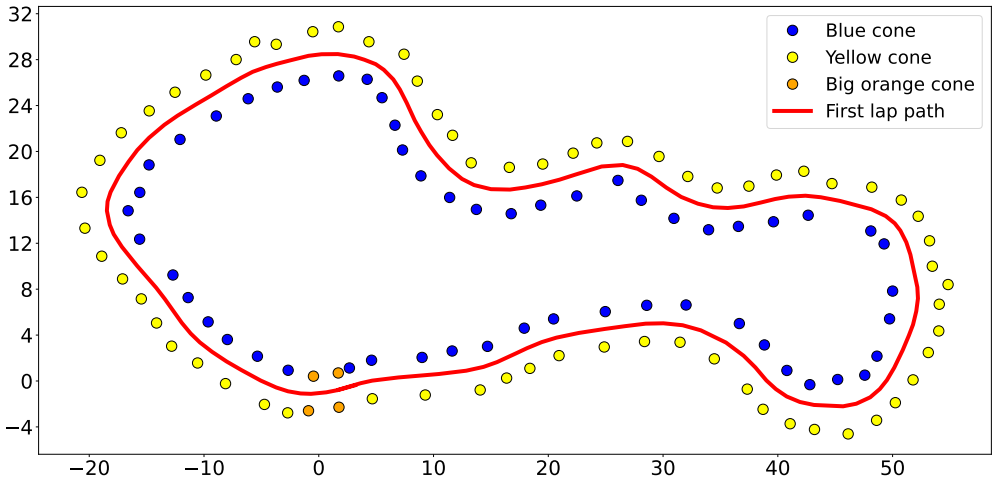


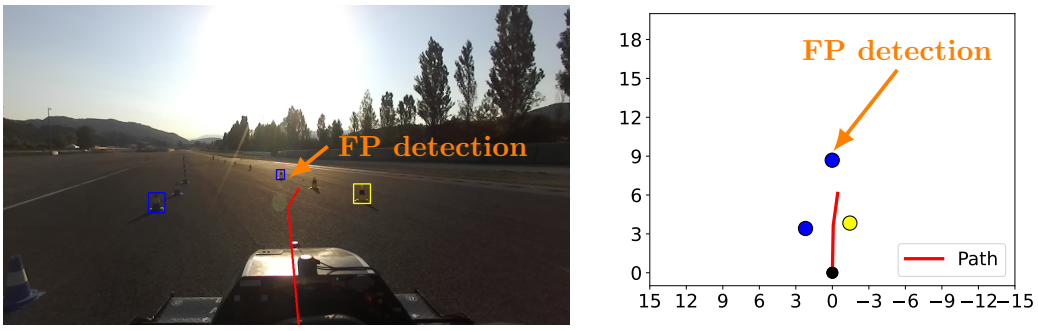
Figure 3.6: The global map of Autocross 2 track with the cone positions and the path followed during the first lap in the simulator.

Track name	Accuracy	Path length [m]	Min dist to failed point [m]
FS Germany 2021	0.99	13.4	7.4
FS Italy 2022	0.99	10.6	6.5
FS Czech 2022	0.98	7.3	4.4
Autocross 1	0.99	9.9	5.3
Autocross 2	0.99	11.3	5.1
Autocross 2 with noise	0.96	10.2	4.0
Average	0.98	10.4	5.5

Table 2: Evaluation of path planning accuracy on Formula Student tracks and real test tracks. The path length refers to the average length of the planned paths across all frames. The minimum distance to the failed point is the distance from the car to the closest point when the path was wrongly planned, meaning the planned path was outside of track boundaries.

Inaccurate cone detection is one of the factors contributing to a wrongly planned path. It can occur due to either the absence of cone detection or the false positive (FP) cone detection. False positive detection refers to the situation when a cone is mistakenly identified by cone detector as a cone of a different color than its actual color. The illustration of false positive detection that resulted in a wrongly planned path is shown in Figure 3.7. In the next frame, the cone detections are correct and the path is recalculated. The replanned path remains within the track boundaries see Figure 3.8. In Table 2 there is also included the minimum distance to the failed point. This metric provides an understanding of potential situations when the car might go off the track. The path tracking algorithm [20] uses a look-ahead distance of two meters. In other words, the car is driving to the point on the path that is three meters away from the car's position. From the Table 2 we can conclude that for these tracks the car always remains on the track. To the real tracks in Table 2 we tested Autocross 2 track with cone detector inaccuracies in simulator. This means that the cone detector wrongly detected or didn't detect some cones. The wrong

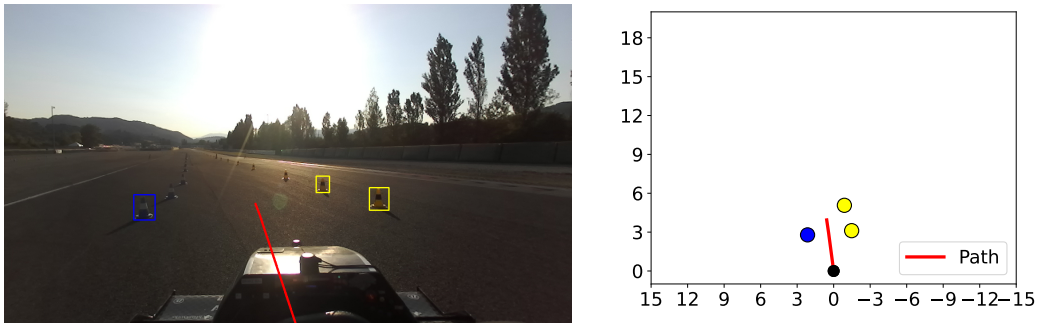
detection was based on the distance to the cone using a linear perturbation function, meaning the probability of wrongly detected cone that was placed 5 meters away was 0.05, for a cone placed 10 meters away the probability was 0.1.



(a) The camera image illustrates a case of false positive detection. The car was facing towards the sun and a strong sunlight led to the cone detection of the opposite color, which resulted in a wrongly planned path.

(b) Plot of positions of the cones including false positive detected cone that led to wrongly calculated path in car's coordinate system.

Figure 3.7: Example of false positive cone detection that led to a wrongly planed path at FS Italy 2022 competiton.



(a) The next taken camera image after the image with false positive detection.

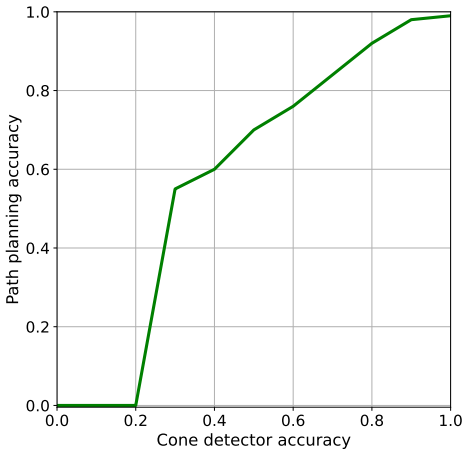
(b) Plot with the positions of correctly detected cones and recalculated path after the frame with false positive detection.

Figure 3.8: The next captured frame after the frame with false positive detection see Figure 3.7 at FS Italy 2022. At this frame the cone detection is correct and the path is recalculated and stays within track boundaries.

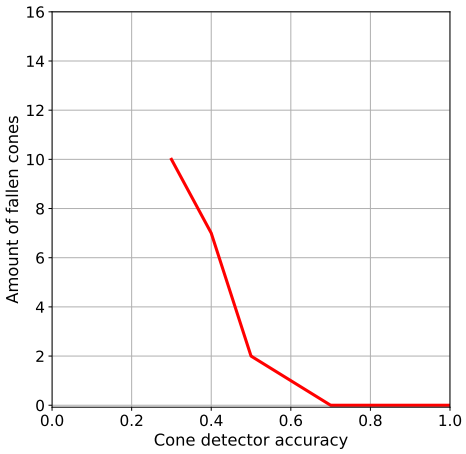
The next experiment was to test path planning accuracy for different cone detector accuracy. The cone detector accuracy represents the ratio of correct frames, where all cones were correctly predicted, to the total number of frames. We test it on Autocross 2 track see Figure 3.1 in simulator described in Section 3.2. The outcomes of this experiment are presented in Table 3. Furthermore, we illustrate the accuracy of path planning and cone detection and the number of fallen cones see Figure 3.9.

Cone detector accuracy	Path planning accuracy	Fallen cones	Finished lap
1.0	0.99	0	Yes
0.9	0.98	0	Yes
0.8	0.92	0	Yes
0.7	0.84	0	Yes
0.6	0.76	1	Yes
0.5	0.70	2	Yes
0.4	0.60	7	Yes
0.3	0.55	10	Yes
0.2	0.0	-	No
0.1	0.0	-	No
0.0	0.0	-	No

Table 3: Evaluation of path planning accuracy vs cone detector accuracy in the simulator.



(a) Relation between the accuracy of path planning and cone detection. When the accuracy of cone detection is less than 0.2, the car went off the track and failed to complete the run.



(b) Comparison of the number of fallen cones and cone detector accuracy. When the accuracy of cone detection was below 0.2, the car went off the track.

Figure 3.9: The figure shows the correlation among of path planning accuracy, cone detector accuracy and amount of fallen cones in the presence of noise added to the cone detector during the Autocross 2 track test. When the cone detector accuracy was higher than 0.7, the car successfully completed the first lap and did not knock over any cones.

Summarizing the results discussed earlier, the proposed path planning algorithm in Section 2.2.1 demonstrates robustness against cone detector inaccuracies. Although the path was wrongly planned in some frames, it was recalculated correctly in the next frames, preventing the car from going off the track. Based on the results, we can conclude that the algorithm was capable of completing the first lap every run for cone detector accuracy higher than 0.2. Moreover, for cone detector accuracy above 0.7, the car was able to complete the first lap without fallen cones.

Section 3.4

Speed Planning Tests

In this section, we test the local speed profile described in Section 2.3 in the simulator from Section 3.2. Speed profile tests are designed to evaluate the proposed local speed profile algorithm against two baselines: (1) conservative 5 m/s constant speed profile and (2) globally optimal speed profile presented in [3]. The constant 5 m/s speed profile means the formula drives during the entire run at a constant speed of 5 m/s and always stays on track. The local speed profile generates a speed plan for the given path that was produced by a local path planning described in Section 2.2. The local speed profile is used to compute a speed plan for the first lap, which means that the entire track is unknown to the car. In contrast, the optimal or global speed profile generates a speed plan for a track that is already known. Typically, the optimal speed profile is computed after the completion of the first lap.

Speed profile	Lap time[s]	Fallen cones	Time imp.[s]	Relative imp.
Constant 5 m/s	84.26	0	0	1
Local	48.97	0	35.29	1.72
Optimal (global)	42.29	0	41.97	1.99

Table 4: Results of different speed profiles.

All three profiles were tested in eForce simulator discussed in Section 3.2 on Autocross 2 track see Figure 3.1. The results are presented in Table 4.

The first approach examined was the constant speed profile set at 5 m/s. This served as the baseline reference for comparison. The lap time achieved with this constant speed was 84.26 seconds, without fallen cones. The 5 m/s constant speed ensures the formula always stay within the handling limits and does not go off the track, but resulted in a big lap time.

The second approach involved local speed planning, which aimed to dynamically adjust the speed profile based on the given track section. The lap time significantly improved to 48.97 seconds, with no fallen cones. This resulted in a considerable time improvement of 35.29 seconds compared to the constant speed profile. The relative improvement factor, calculated as the ratio of the local speed profile lap time to the constant speed profile lap time, is 1.72. These results demonstrate the effectiveness of local planning in optimizing the speed profile and reducing a lap time for the first lap on unknown track.

The third approach explored optimal (global) planning, which considered a broader scope of information and factors to determine the speed profile. This approach further enhanced the lap time to 42.29 seconds, with no fallen cones. The time improvement achieved was 41.97 seconds compared to the constant speed profile, yielding a relative improvement factor of 1.99. The globally optimal speed profile algorithm knows the entire track a priori and thus it achieved the lowest time.

Figure 3.10 illustrates more detailed speed planning for the run. Figure 3.10a shows the followed path by the car for the first lap. Based on a path curvature see Figure 3.10b, the local and optimal speed profiles generated the speed plans in Figure 3.10c. The constant speed approach of 5 m/s did not take into account the path curvature. In Figure 3.10c we can see that the optimal speed plan is smooth, while the local speed plan is rocky, because the speed profile was planned for every

frame, thus the speed in the next frame was always recalculated.

Figure 3.11 shows the executed speed plan for track in Figure 3.10a. It can be observed that on the straight sections, the path appears more yellowish, indicating higher speeds, while in the turns, the path color appears more bluish, indicating lower speeds were reached.

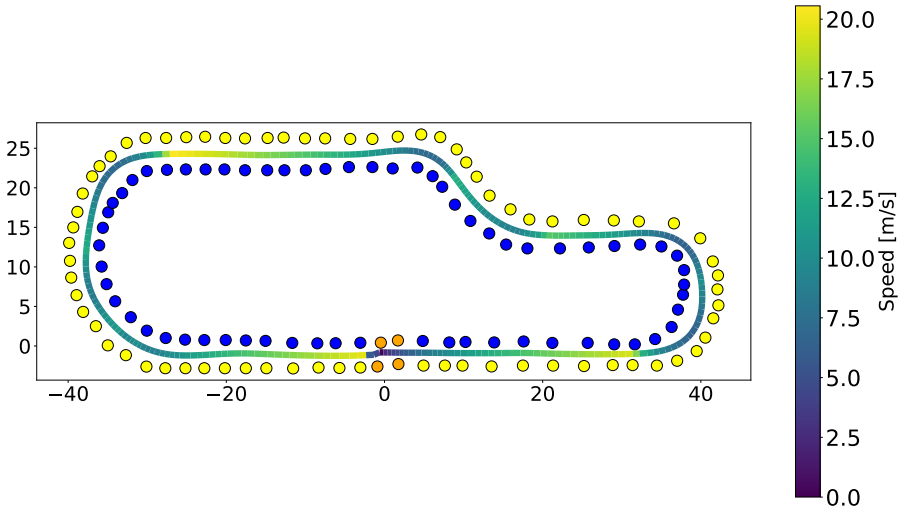
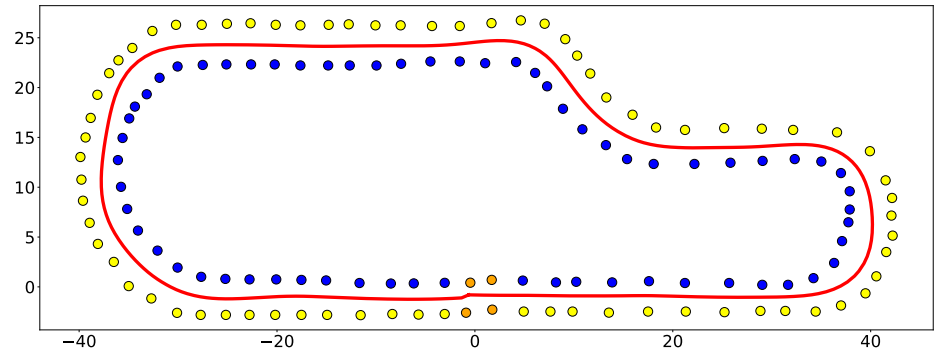
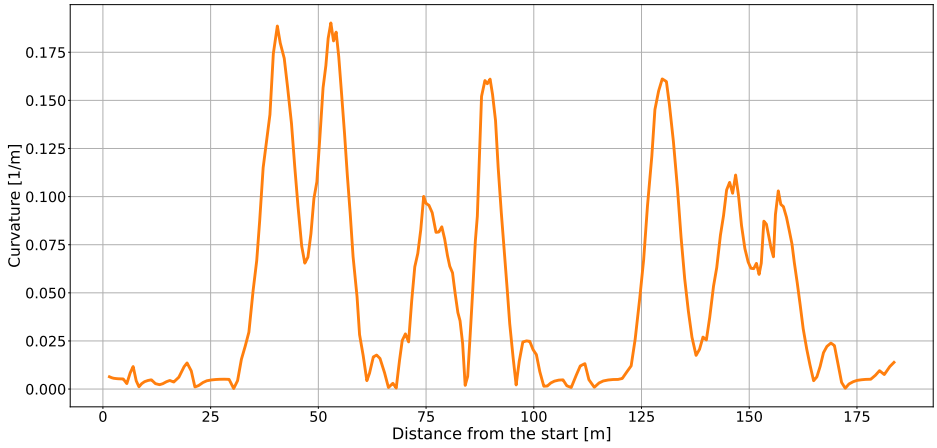


Figure 3.11: Simulation of driving on Autocross 2 track using speed profile described in Section 2.3.

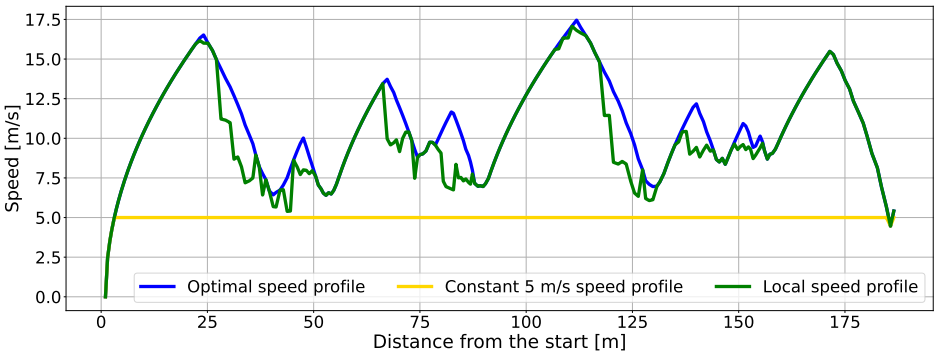
Summing up the speed profile tests, the proposed local algorithm significantly outperforms the conservative 5 m/s constant speed baseline. Moreover, the time achieved by our local algorithm is not much higher than the time of the globally optimal algorithm despite the track was not known. We further locally performed tests on a curved track, evaluating both the local and globally optimal speed profiles. The time achieved by our local algorithm closely matched the globally optimal algorithm time.



(a) Illustration of the used track for speed profile test and the followed path during the run.



(b) Figure shows the curvature of the followed path. We can observe that the higher curvatures corresponds to the turning section on the track.



(c) Illustration of the different speed profiles.

Figure 3.10: Comparison of optimal, constant and local speed profiles tested in the simulator. The optimal speed profile is taken from [3]. The local speed profile is described in Section 2.3. The optimal algorithm knows the track before the run, while the local algorithm does not know the whole track and plans the speed for the section of the track that is in front of the car.

Chapter 4
Conclusion

This thesis proposed a local planning algorithm, which is a part of the algorithm to complete the first lap for autonomous student formula. The presented algorithm focuses on local path planning and speed profile generation, aiming to navigate the autonomous formula efficiently on the unknown track. Both of path and speed planning algorithms have been developed and evaluated in this study.

In Section 1 we introduced the problem of local planning for autonomous student formula. We described the Formula Student competition, its rules and dynamic events. The proposed local planning algorithm was designed for autocross event and for the first lap of trackdrive. Then we reviewed the related work in planning for autonomous vehicles and student formulas, it was found that various approaches have been proposed, including global and local planning algorithms. The local planning algorithms, play a crucial role in determining the path and speed profile for the car in dynamic environments. The proposed local planning was mainly designed for DV.01 – the first autonomous electric student formula in the Czech Republic, developed by eForce FEE Prague Formula, the team under the Faculty of Electrical Engineering of the Czech Technical University in Prague.

Section 2 presented the algorithm for the first lap. It consist of perception, planning and control nodes. We focused on the planning node and introduced the local path planning in Section 2.2 and the speed planning in Section 2.3. The designed path planning produces a path based on real-time sensor inputs. It uses a center-line approach that prioritizes reliability over path optimality. This decision was made on the understanding that the narrowness of the track and the potential inaccuracies in perception and control algorithms can result in knocking down the cones, leading to applying penalties to the lap time. After the generation of a center line, we smooth it to find a more continuous line for the speed profile that benefits from lower path curvature. The introduced local speed planning takes into account track geometry, vehicle dynamics, its acceleration and deceleration limitations, but neglect aerodynamic downforce and tire modeling. By adhering to these constraints, the algorithm ensures that the car drives within its maximum safe speed limits resulting in faster completion of the lap.

The proposed algorithms were validated in Section 3. The algorithms were evaluated through experiments and simulations in eForce simulator, considering various scenarios and different tracks. The results demonstrated the successful navigation of the car in the first lap. The local path planning algorithm showed the robustness against cone detector inaccuracies, while the local speed profile demonstrated a significant improvement compared to the conservative baseline approach, when the formula drives with a constant speed of 5 m/s. Read Section 4.1 to know more about the achieved results.

In conclusion, the thesis offers a valuable solution for local path planning and speed profile generation in autonomous racing. It contributes to the local planning algorithms designed for autonomous student formulas, and contributes to the advancement of autonomous vehicle technology. Further research can be conducted to explore additional optimizations and improvements to the proposed algorithms.

Section 4.1

Achieved Results

The validation process involved extensive testing in a simulator developed by Force team see Section 3.2.

The path planning tests show that the proposed algorithm in Section 2.2 is robust. The average robustness of the local path planning is 0.98 on different tracks, including Formula Student tracks, real tracks from the team testing and synthetically generated tracks, see Table 2. The proposed algorithm is robust against cone detector inaccuracies: with a cone detector accuracy of 0.3 or higher, the car successfully completed the first lap, even though some cones have been knocked down. However, when the cone detector accuracy reached 0.7 or higher, the formula completed the first lap without knocking down any cones see Table 3.

The speed planning tests demonstrate that the local speed planning presented in Section 2.3 has better results than the conservative baseline approach of driving constantly 5 m/s that was used by eForce for previous competitions. The results indicate that the local speed profile outperforms the baseline approach by reducing the lap time by 35 seconds, resulting in a relative improvement of 1.72 see Table 4. This improvement is visually presented in Figure 3.10, which provides a detailed illustration of the conducted test and includes a comparison with the optimal speed profile.

The achieved results shows a significant step forward in the development of DV.01 and hold promising prospects for achieving top positions in upcoming Formula Student competitions.

Section 4.2

Future Work

In the future, there are several potential directions for further development and improvement of the proposed algorithms.

As a result of mechanical issues encountered with DV.01, all proposed algorithms were evaluated and tested in a simulated environment. Consequently, a critical aspect of future work involves testing the algorithms on the real car in real-world conditions to assess.

The future work for speed profile involves complex vehicle and tire modeling, taking into account the aerodynamic downforce.

The subsequent area of future work concentrate around the upcoming 2023 race season, where the eForce team is introducing the FSE.12, a new electric formula that integrates both piloted and driverless capabilities. After conducting real-world testing of the proposed algorithms on DV.01, the focus will shift towards integrating and adapting these algorithms specifically for the FSE.12 formula. The eForce FEE Prague Formula team is going to compete in four Formula Student competitions during the 2023 race season, namely FS Switzerland, FS Italy, FS Czech, and FS Germany. These events will provide valuable opportunities to showcase and further refine the algorithms in the context of competitive racing scenarios.

Chapter A
Contents of the Attachment

After the consultations and discussions with the supervisor and eForce team members, it has been decided that the thesis code will not be included in the attachment. This decision is driven by the competitive nature of Formula Student. Sharing the code openly gives other Formula Student teams access to it, thereby compromising our competitiveness in the upcoming race season.

The attachment contains the videos with different tests see Figure A.1.

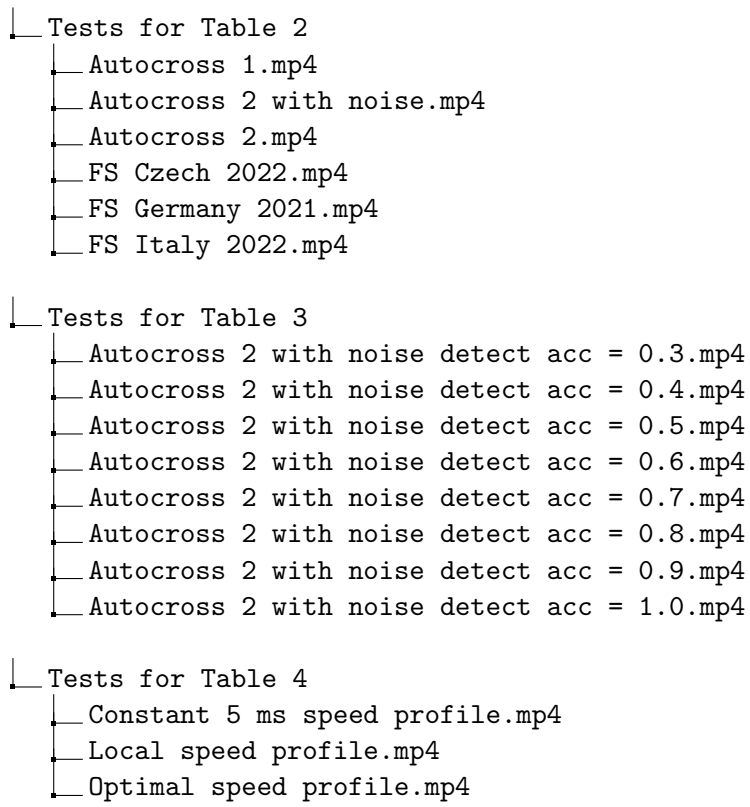


Figure A.1: Structure of attachment.

Chapter 5
References

[1] Formula student germany web page. https://www.formulastudent.de/pr/pictures/. Accessed: 2023-05-24.

[2] Formula Student Germany. Fsg competition handbook 2023. https://www.formulastudent.de/fileadmin/user_upload/all/2023/important_docs/FSG23_Competition_Handbook_v1.0.pdf, 2023.

[3] Michal Horáček. Finding the fastest trajectory for autonomous student formula. Bachelor's thesis, Czech Technical University in Prague, 2022.

[4] Formula Student Germany. Formula student rules 2023. https://www.formulastudent.de/fileadmin/user_upload/all/2023/rules/FS-Rules_2023_v1.1.pdf, 2023.

[5] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[6] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[7] Lydia E Kavraki, Pavel Svestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[8] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Department, Iowa State University, 1998.

[9] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine*, 4(1):23–33, 1997.

[10] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[11] Nitin R. Kapania. *Trajectory planning and control of an autonomous race vehicle*. PhD thesis, Stanford University, 2016.

[12] Dhruv Shah, Kyle Stachowicz, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. FastRLAP: A system for learning high-speed driving via deep RL and autonomous practicing. In *ICRA2023 Workshop on Pretraining for Robotics (PT4R)*, 2023.

[13] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Bharadwaj Gosala, Mehak Gupta, Ramya Sivanesan, Ankit Dhall, Eugenio Chisari, Napat Karnchanachari, Sonja Brits, Manuel Dangel, Inkyu Sa, Renaud Dubé, Abel Gawel, Mark Pfeiffer, Alexander Liniger, John Lygeros,

and Roland Siegwart. AMZ driverless: The full autonomous racing system. *CoRR*, abs/1905.05150, 2019.

[14] Solange D. R. Santos, José Raul Azinheira, Miguel Ayala Botto, and Duarte Valério. Path planning and guidance laws of a formula student driverless car. *World Electric Vehicle Journal*, 13(6), 2022.

[15] Jacob Olausson and Jacob Larsson. Optimal control and race line planning for an autonomous race car. Master's thesis, Linköping University, Vehicular Systems, 2021.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[17] Roman Šíp. Visual detection of traffic cones for autonomous student formula. Bachelor's thesis, Czech Technical University in Prague, 2022.

[18] Daniel Štorc. Detection of traffic cones from lidar point clouds. Bachelor's thesis, Czech Technical University in Prague, 2022.

[19] Ondřej Kuban. Vývoj algoritmu vedení po trati pro autonomní studentskou formuli. Bachelor's thesis, Czech Technical University in Prague, 2024. To be defended.

[20] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23:661–692, 1 2006.

[21] Tomáš Roun. Navigation system for autonomous student formula. Master's thesis, Czech Technical University in Prague, 2021.

[22] Bo Persson, U. Tartaglino, O. Albohr, and Erio Tosatti. Rubber friction on wet and dry road surfaces: The sealing effect. *Physical Review B*, 71, 03 2005.

[23] Jinhyun Park, In Gyu Jang, and Sung-Ho Hwang. Torque distribution algorithm for an independently driven electric vehicle using a fuzzy control method: Driving stability and efficiency. *Energies*, 11(12), 2018.

[24] eForce FEE Prague Formula. Engineering design report: Aerodynamic devices. 2018.

[25] Jan Filip. Trajectory tracking for autonomous vehicles. Master's thesis, Czech Technical University in Prague, 2018.