Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

# Head-to-Head Racing with F1/10 Autonomous Car

Tomáš Nagy

Supervisor: Ing. Jaroslav Klapálek
Field of study: Cybernetics and Robotics
May 2024

## Acknowledgements

I would like to express my deep gratitude to my supervisor, Ing. Jaroslav Klapálek, for his guidance and valuable advice regarding this thesis. Moreover, I would like to extend my thanks to my family, friends, and colleagues for their support throughout my study.

## Declaration

I declare that the submitted thesis was developed individually and that I listed all used information sources in accordance with Methodical Guideline on Ethical Principles for College Final Work Preparation.

In Prague, 24. May 2024

## Abstract

This thesis focuses on the detection of overtaking zones on a racing track when competing against one opponent. A review of the current solutions and overtaking rules was conducted. Based on this review, we developed and implemented a planner that computes dynamically feasible overtaking manoeuvres. The maneuvers were planned with respect to the competition racing rules. By planning multiple trajectories on different parts of the track, we were able to successfully identify possible overtaking zones. The method was tested on multiple tracks with different opponent trajectories and vehicle parameters. To validate the feasibility of the planned manoeuvres, we executed a few of them in the simulation and real-life using the F1TENTH platform.

**Keywords:** overtaking, trajectory optimization, racing, F1TENTH

**Supervisor:** Ing. Jaroslav Klapálek

## Abstrakt

Táto diplomová práca sa zaoberá detekciou vhodných miest na predbiehanie jedného súpera na pretekárskej dráhe. Na základe prieskumu súčasných riešení danej problematiky, sme navrhli a implementovali plánovač predbiehacích manévrov. Plánovaním viacerých trajektórií na rozdieľnych miestach na trati, sme úspešne identifikovali vhodné miesta na predbiehanie. Aby sme ukázali realizovateľnosť naplánovaných manévrov, niektoré z nich sme otestovali v simulácií, ako aj v realite za použitia modelu vozdila F1TENTH.

**Kľúčové slová:** pretekanie, predbiehanie, optimalizácia trajektórie, F1TENTH

**Preklad názvu:** Pretekanie so súperom pre autonómne auto F1/10

# Contents

**1 Introduction** **1**

**2 Literature review** **3**
2.1 Reactive methods . . . . . . . . . . . . . . 3
2.2 Learning-based methods . . . . . . . . 4
2.3 Local planning . . . . . . . . . . . . . . . . 5
2.4 Overtaking-related rules in racing competitions . . . . . . . . . . . . . . . . . . . . 7
2.4.1 Defending a position . . . . . . . . 8
2.4.2 Contact with the opponent . . . 9
2.4.3 Overtaking . . . . . . . . . . . . . . . . . 9

**3 Metodology** **11**
3.1 Scenario definition . . . . . . . . . . . . 11
3.2 Marking convention . . . . . . . . . . . 12
3.3 Platform description . . . . . . . . . . 13
3.3.1 Hardware stack . . . . . . . . . . . . 13
3.3.2 Software stack . . . . . . . . . . . . . 14
3.4 Trajectory optimization . . . . . . . 16

**4 Method description** **19**
4.1 Algorithm improvements . . . . . . . 19
4.1.1 Trajectory evaluation . . . . . . . 19
4.1.2 Track progress calculation . . . 20
4.1.3 Detecting a collision with the opponent . . . . . . . . . . . . . . . . . . . . 20
4.1.4 Interaction between cars . . . . 21
4.2 Implementation details . . . . . . . . 22
4.2.1 Initial conditions and final state feasibility . . . . . . . . . . . . . . . . . . . . 22
4.2.2 Handling uncertainty . . . . . . . 23
4.2.3 Data gathering . . . . . . . . . . . . 23

**5 Experimental evaluation** **25**
5.1 The design of the experiments . . 25
5.1.1 Racing tracks description . . . 27
5.2 Results . . . . . . . . . . . . . . . . . . . . . . 28
5.2.1 Track 1 . . . . . . . . . . . . . . . . . . . 29
5.2.2 Track 2 . . . . . . . . . . . . . . . . . . . 30
5.2.3 Track 3 . . . . . . . . . . . . . . . . . . . 31
5.2.4 Track 4 . . . . . . . . . . . . . . . . . . . 33
5.2.5 Track 5 . . . . . . . . . . . . . . . . . . . 34
5.2.6 Track 6 . . . . . . . . . . . . . . . . . . . 36
5.3 Maneuver validation . . . . . . . . . . 38
5.3.1 Maneuver 1 . . . . . . . . . . . . . . . 38
5.3.2 Maneuver 2 . . . . . . . . . . . . . . . 39
5.3.3 Maneuver 3 . . . . . . . . . . . . . . . 40
5.3.4 Maneuver 4 . . . . . . . . . . . . . . . 41
5.3.5 Maneuver 5 . . . . . . . . . . . . . . . 42
5.3.6 Maneuver 6 . . . . . . . . . . . . . . . 43
5.3.7 Maneuver 7 . . . . . . . . . . . . . . . 43

**6 Future work** **47**

**7 Conclusion** **49**

**A Bibliography** **51**

**B Project Specification** **55**

# Figures

2.1 A comparison between FTG and Disparity Extender . . . 4
2.2 Local planning methods . . . 7
2.3 Defending a position. . . . 8
2.4 Overtaking in a corner . . . 10

3.1 F1TENTH platform top view. . . 14
3.2 F1TENTH platform photo . . . 15
3.3 Track segmentation example . . . 16
3.4 Matryoska mapping example . . . 17

4.1 Example of an overtake with and without interaction. . . . 21
4.2 Example of overtake positions . . 23

5.1 Images of racing tracks used for simulation experiments . . . 27
5.2 Track 6 used for real-life experiments . . . 28
5.3 Overtaking zones on Track 1 with different opponent trajectories . . . . 29
5.4 Overtaking zones on Track 1, while the ego car has increased friction . 29
5.5 Overtaking zones on Track 2. . . . 30
5.6 Overtaking zones on Track 3 with different opponent trajectories . . . . 31
5.7 Overtaking zones on Track 3 with optimal opponent trajectory . . . 32
5.8 Overtaking zones on Track 3 with ego's top speed change . . . 32
5.9 Overtaking zones on Track 3 with ego's friction change . . . 33
5.10 Overtaking zones on Track 4 with different opponent trajectories . . . . 34
5.11 Overtaking zones on Track 4 with ego's vehicle friction change . . . 34
5.12 Overtaking zones on Track 5 with different opponent trajectories . . . . 35
5.13 Overtaking zones on Track 5 with ego's vehicle friction change . . . 35
5.14 Overtaking zones on Track 4 while the opponent is driving its near-minimum time trajectory, and the ego vehicle has increased top speed. . . . 36
5.15 Overtaking zones on Track 6. . . 37
5.16 Maneuver 1 execution . . . 38
5.17 Maneuver 2 execution . . . 39
5.18 Maneuver 3 execution . . . 40
5.19 Maneuver 4 execution . . . 41
5.20 Maneuver 5 execution . . . 42
5.21 Execution of manoeuvre 6 in the simulation . . . 43
5.22 Execution of manoeuvre 7 in the simulation and in real life . . . 44
5.23 Comparison between simulated and real-life . . . 45

# Tables

3.1 Measurements of the F1TENTH platform . . . . . . . . . . . . . . . . . . . . . . . 14

5.1 Vehicle parameters. . . . . . . . . . . . 26
5.2 List of scenarios . . . . . . . . . . . . . . 26

# Chapter 1
# Introduction

Efficient and safe overtaking is a fundamental skill all drivers must have, particularly in the racing domain. While following the fastest racing line is crucial for a quick lap time, sticking to it alone does not secure a victory in a race. In comparison to a highway scenario where all drivers share the common goal of reaching their destinations safely, racing drivers compete for the first place. Moreover, in racing, the vehicles use higher speed, while competing against each other. Therefore, planning and executing efficient overtaking manoeuvres in racing is much more complex, making this an interesting problem to explore.

We study this problem in the context of the F1TENTH Autonomous Racing Competition [1]. This international event features 1:10 scale autonomous cars competing for first place. There are currently two types of races. In the *time trial*, the vehicle is supposed to complete a lap as fast as possible while racing alone on the track. To win this race, a lap time needs to be minimized, which is usually done by following a predetermined optimal race line. On the other hand, during a *head-to-head* race, two cars compete against each other on the track. This format adds more complexity since vehicles need to interact with each other to execute safe overtakes. However, many competitors utilize avoidance algorithms where the opponent is usually treated as a static obstacle. Therefore, most of the overtakes happen only as a result of avoiding the collision.

In this thesis, we aim to develop an algorithm that can identify possible overtaking zones on a track. This enables the development of future algorithms which can take advantage of this additional information. Therefore, they can plan manoeuvres only at parts of the track where the overtake is possible.

The structure of this thesis is as follows: In Chapter 2, we review common approaches for overtaking within the literature, followed by a discussion of overtaking rules used in various competitions. In Chapter 3, we specify the target scenario, describe the testing platform, and explain the algorithm we used as a base for our method. In Chapter 4, we introduce the changes made to the original algorithm and explain some important implementation details. In Chapter 5, we present the test results of our approach on various tracks

The implementation of the developed method, videos from experiments, and additional images can be found at: `https://github.com/CTU-F1T/DP-Overtaking-Zones`.

in simulation and real life. Finally, in Chapter 6, we propose some ideas for future improvements of the approach.

# Chapter 2

# Literature review

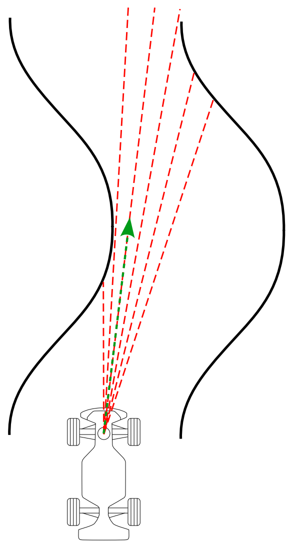
In this chapter, we discuss methods used in autonomous racing to overtake the opponent vehicle. We split these methods into three categories: (i) reactive methods, (ii) learning-based methods, and (iii) local planners. Afterwards, we review the rules for overtaking in various car racing competitions.

## 2.1 Reactive methods

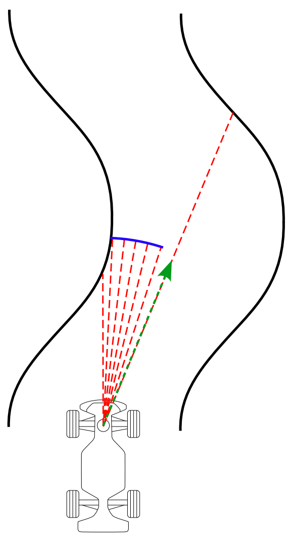
Reactive methods utilize only the information available from the latest sensor measurements; therefore, they do not use any historical data. The advantage of these methods is their low computation time, fast response, and robustness. On the other hand, they typically result in a sub-optimal behaviour.

**Follow the Gap.** Follow the Gap [2] (FTG) method works by finding the deepest gap (largest distance) from the current distance measurements. The algorithm then steers the vehicle towards that gap. This method was quite popular in the F1TENTH community due to its simplicity and robustness. Even though it was created for static obstacle avoidance only, it can be also used for dynamic obstacles by treating them as static obstacles in a single frame. In the F1TENTH races, it is usually used by itself or in combination with a reference trajectory tracking, where the algorithm switches to the FTG in case there is an obstacle blocking the progress on the reference trajectory. However, this approach only avoids the obstacles without planning the trajectory ahead. This can result in conservative overtakes.

**Disparity Extender.** Disparity Extender [3] is an extension of the FTG method. In contrast to the prior work, it augments the vector of lidar measurements. It starts by finding all disparities (disparity is a jump in value between two consecutive measurements in a single lidar scan) in the current lidar scan. Then, for every disparity, it finds a measurement with a lower value and overwrites a number of measurements in the direction of the bigger measurement so it covers at least half of the width of the vehicle. This method basically performs an inflation of obstacles in the lidar scan. After this, it continues as a standard FTG by finding the deepest gap. The difference between the FTG and the Disparity Extender can be seen in Fig. 2.1.



**(a) :** The FTG fails in this scenario because by following the deepest gap, it crashes into the side of the track.



**(b) :** The Disparity Extender shifts the disparity (blue line) and can avoid the crash into the track boundary.

**Figure 2.1:** A comparison between FTG and Disparity Extender. LiDAR scan values are red lines, and the travel direction is a green line.

## 2.2 Learning-based methods

Learning methods can be utilized for autonomous racing in various ways. They can be used in different stages of the pipeline (e.g., perception, opponent prediction, decision-making, trajectory tracking) or for the whole pipeline (end-to-end). A comparison between these learning-based architectures can be found in [4].

**Curriculum Reinforcement Learning.** In [5], an end-to-end three-stage Curriculum Reinforcement Learning was used. At first, the agent was trained for racing with no other car on the track. Then, the same agent was retrained using scenarios for overtaking and avoidance. The method was verified in a popular video game, Gran Turismo Sport, which contains realistic modelling of vehicles and tracks. The agent was able to learn racing strategy and behaviour with performance comparable to an expert driver while outperforming vanilla reinforcement learning and a built-in AI.

**Deep Latent Competition.** Another end-to-end reinforcement learning method called Deep Latent Competition (DLC) was introduced in [6]. DLC works by imagining multi-agent interaction sequences in the compact latent space of a learned world model. The method was evaluated in a simulator designed for learning competitive visual control policies.

**Opponent prediction with Deep recurrent neural network.** Authors in [7] used a deep recurrent neural network to predict positions, velocities and headings of multiple opponents on a racing track using a single camera. The approach was demonstrated using a virtual camera in a testing environment DeepRacing [8] built on the Formula One simulation racing game.

**Opponent prediction with Gaussian Processes.** In [9], a Gaussian Process (GP) is trained to predict the opponent's behaviour. In contrast to the previous method, this one considers only one opponent on the track. On the other hand, it uses GP, which is trained on many different laps of the same opponent. Therefore, it can predict the opponent's position as well as the variance.

## 2.3 Local planning

Local planning methods work by planning a local trajectory up to some horizon. These methods can be used to traverse an unmapped environment or to avoid a new obstacle in a known one. In racing, it can be used in combination with a global reference to plan an overtaking or avoidance manoeuvre while deviating from global reference as least as possible [10].

Many of the algorithms mentioned in this section generate a path; however, tracking algorithms usually require a trajectory. We can create a trajectory from a path by assigning a velocity to each position on the path, e.g., using the algorithm described in [11]. Moreover, if an algorithm works by evaluating paths to choose the best one, it can do so by, e.g., assigning velocities to create trajectories and choosing the fastest.

**Stochastic model predictive control.** As mentioned in Section 2.2, the GP is trained to predict the opponent's behaviour [9]. The predictions are then used in a stochastic model predictive control (MPC). The method was tested in a simple simulation scenario. The stochastic MPC generates optimistic, safe trajectories to overtake an opponent vehicle.

**Rapidly exploring random trees.** Rapidly exploring random trees (RRT) or its optimal version, RRT*, can be used to generate local paths. Using random sampling, a tree of paths is generated. Each tree node is evaluated, and the path is backtracked from a tree node with the best cost. This method is asymptotically optimal. An example of this method is on Fig. 2.2a.

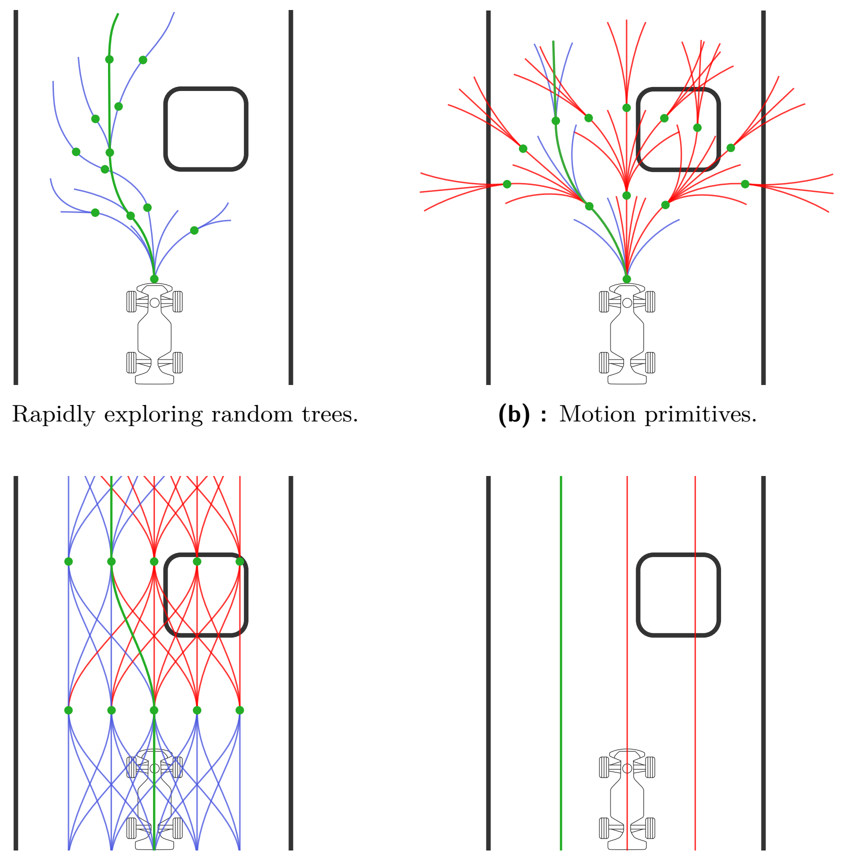
**Motion primitives.** Motion primitives [12, 13, 14] method is similar to local planning using the RRT method. However, instead of growing the tree each time a new plan is needed, motion primitives are generated once in a vehicle's local frame (Fig. 2.2b). When a new local plan is needed, motion primitives are evaluated based on the current vehicle state to pick the best one.

**Lattice planning.** Lattice planning from [15] works by creating a graph where each node represents a position on the track, and each edge represents a path between two points of the track (nodes). During an offline phase, points are sampled in a pattern to cover the whole racing track. Paths are then created to connect together points that are close to each other. This way, the whole track is covered with a graph of paths (Fig. 2.2c). During the execution, first the current position in the graph is found. In the end, a path that avoids obstacles and minimizes distance up to some horizon is found using a graph search algorithm.

**Lane switching.** The lane-switching algorithm was used in [16]. This algorithm is also popular in F1TENTH races due to its simplicity. First, a few global trajectories are generated, e.g., time optimal trajectory together with a centerline, a centerline shifted towards the left wall, and a centerline shifted towards the right wall (Fig. 2.2d). During the execution, the car first follows the optimal racing line. After an obstacle or an opponent is detected on the current trajectory, the vehicle switches to another free trajectory to avoid a collision. When possible, it switches back to the optimal trajectory.

**Spacial policies.** In [17], authors developed an algorithm to learn spacial policies and use them during racing. Their setup consists of a non-interactive obstacle vehicle (opponent) following a curvature-optimal racing line. Firstly, the track is divided into the following high-level segments: *Straight*, *Sweeper Curve*, *Hairpin Curve* and *Chicane*. Then, simulations are performed while varying certain simulation parameters, such as the starting positions of the opponent and ego vehicles and the opponent's speed. The ego vehicle utilizes Model Predictive Contouring Control (MPCC) with obstacle avoidance to plan dynamic trajectories. The algorithm counts a number of overtakes from each of the four high-level start zones: *left close behind*, *left far behind*, *right close behind*, and *right far behind*. During racing, this information is then used to encourage the ego vehicle to the position relative to the opponent that resulted in a high number of successful overtakes during simulations.

**Game theory.** A game theoretical approach was used in [18]. One of the assumptions of the used method is that the ego vehicle has $m$ and the opponent has $n$ trajectories to choose from. The trajectories are generated locally up to some horizon. These trajectories are then evaluated to choose the best trajectory pair. In this work, three different strategies were used: (i) a *sequential game*, where only the ego vehicle is responsible for the collision, (ii) a *cooperative game*, where both vehicles are responsible for the collision and the progress of the vehicle is used in the cost function, and (iii) a *blocking game*, where there is an additional cost for staying ahead at the end of the horizon. The results showed that the sequential game requires the least computational resources while resulting in the least amount of overtakes. The most overtakes happened during the cooperative games. During the race, the games are played in a moving horizon fashion.



**(a) :** Rapidly exploring random trees.

**(b) :** Motion primitives.

**(c) :** Lattice planner.

**(d) :** Lane switching.

**Figure 2.2:** Examples of a few local planning methods on a straight with a simple obstacle. Possible paths are blue, infeasible paths are red, and the chosen path is green.

## 2.4 Overtaking-related rules in racing competitions

In this section, we examine regulations from various F1TENTH and full-scale car racing competitions. Each competition has a specific set of rules drivers need to follow. Not following the racing regulations can be unsafe and can result in penalization, disqualification or even a crash. However, overtaking rules are not exactly defined. These rules are in a grey area; therefore, stewards have to decide on the cause and penalization based on their judgment if an accident occurs. If there is no apparent cause, no driver or even both drivers can be penalized. Even without an apparent cause, the racing incident is still a disadvantage. This is not a big issue in the F1TENTH competitions because the race is restarted from the last checkpoint. Moreover, these cars are pretty robust, so they usually end up without any damage. On the other hand, this may cause a huge issue in Formula 1 (F1) races. In the F1, any incident can easily cause damage to the vehicle, resulting in an unnecessary pit stop or even the inability to continue in the race. Therefore, to minimize the risk of an accident, we will also consider the unwritten rules and common racing etiquette.

Many of the car racing events are governed by the Federation Internationale de l'Automobile (FIA) [19] established in 1904. These events include the Formula One World Championship (F1), World Rally Championship, Intercontinental Drifting Cup, GT World CUP, various Karting competitions and many more. Their purpose is to represent motoring organizations, provide licensing, specify competition rules, etc. Many of the independent events, such as racing clubs during their private events, sim-racing sessions, and even F1TENTH events, have many of their rules based on the rules and guidelines developed by the FIA.

We split relevant rules and guidelines into three categories: (i) *defending a position*, where we discuss what kind of manoeuvres are usually allowed to prevent overtakes, (ii) *contact with the opponent*, where we discuss what kind of contact is allowed between vehicles within competition rules and what the penalties are for bumping into an opponent, and (iii) *overtaking*, where we discuss guidelines that both vehicles should follow during overtaking manoeuvre.

### 2.4.1 Defending a position

Defending a position can be done by taking a race-line that forces the opponent to take a worse trajectory through a corner. However, a sudden change of direction and blocking by going from side to side is generally not allowed. The latest FIA Code of Driving Conduct on Circuits states [20]:

> "More than one change of direction to defend a position is not permitted. Any driver moving back towards the racing line, having earlier defended his position off-line, should leave at least one car width between his own car and the edge of the track on the approach to the corner. However, manoeuvres liable to hinder other drivers, such as deliberate crowding of a car beyond the edge of the track or any other abnormal change of direction, are strictly prohibited."

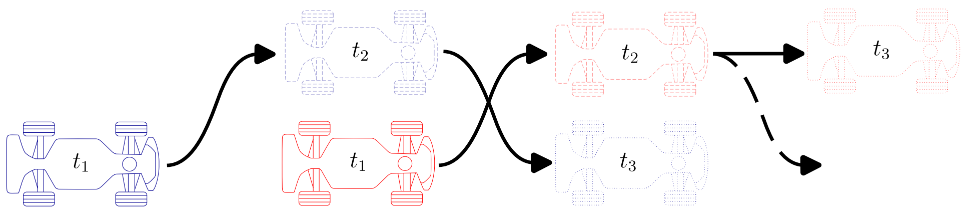


**Figure 2.3:** Example of a one defensive move rule. In time $t_2$, the red car can no longer block the blue car (dashed line) and must remain in its line.

In summary, it says that a car should not push the opponent off the track, should not go from side to side, and only use at maximum one defensive manoeuvre. This rule is also used in a similar form, e.g. by The Sim Racing Collective (TSRC) in [21], Sports Car Club of America (SCCA) [22], and in the F1TENTH rules [1]. However, some competitions may not allow any blocking move. An example of a defensive move is shown in Fig. 2.3.

### 2.4.2 Contact with the opponent

Deliberate crashing into an opponent is not allowed [21, 1, 20]. However, minor accidental bumps can be considered as a part of the racing during some events.

In F1, penalties for race incidents such as bumping into an opponent are five seconds, and a ten-second time penalty served during the next pit stop [23]. However, it is not exactly defined when to give which penalty. It depends on the cause, severity of the collision, and other factors. The F1 sporting regulations specify the following [23]:

> "It shall be at the discretion of the stewards to decide if any driver involved in an Incident should be penalised. Unless it is clear to the stewards that a driver was wholly or predominantly to blame for an Incident no penalty will be imposed."

In sim racing events by TSRC, it is not allowed to bump into an opponent and pass him. If the overtaking car would benefit from the bump, it should give back the position to avoid a potential penalty [21].

On the other hand, in the F1TENTH, only larger crashes are penalized. Minor bumps into an opponent are generally tolerated. The rule-book states [1]:

> "Touching the opponent is not penalized unless one of the cars significantly diverges from its expected trajectory."

### 2.4.3 Overtaking

As discussed in previous sections, overtaking incidents are a grey area of rules. There are no exact definitions of these manoeuvres, and incidents are decided by stewards. However, the following two documents are great sources for overtaking guidelines: (i) Racing Room & Passing Guidlines [24] by Randy Pobst (SCCA Hall of Fame member and professional driver champion) and Terry Earwood (professional driver champion and professional driver's coach), and (ii) Overtaking Guidelines [25] for F1 by FIA. However, it is important to note that these are not rules. For example in FIA overtaking guidelines for F1 is written [25]:

> "For the avoidance of doubt, these are merely guidelines to assist the stewards in their decision-making and are non-binding."

The main purpose of overtaking is to pass the opponent safely and without a collision. It is important to communicate the intentions by the vehicle movement clearly and without sudden changes of direction. Moreover, the most important part is the following [24]:

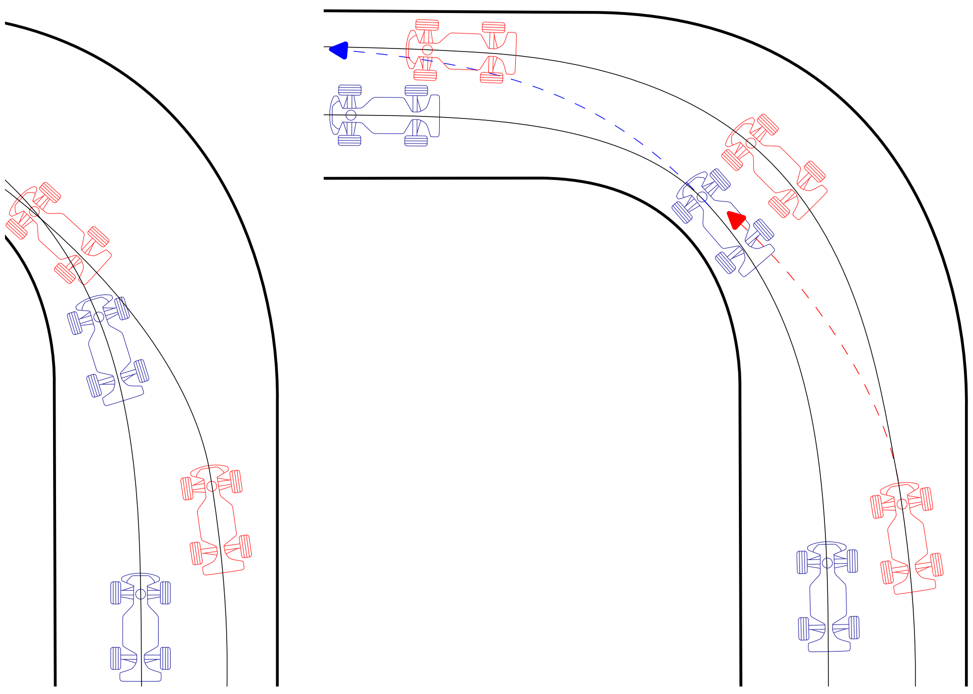


**Figure 2.4:** In the left image, the blue car does not have a significant portion of the car alongside the red vehicle. It is the responsibility of the blue car to avoid the accident. On the other hand, in the right image, the blue car has a significant portion of the car alongside the red vehicle. It is now the responsibility of both vehicles to avoid the crash. The red car cannot take the same path as in the left image (red dashed line) and has to leave space for the blue car. Moreover, on the corner exit, the blue car cannot use the whole track (blue dashed line) and has to leave enough space for the red car.

> "Safe, successful passing depends on what a driver can see. Do not hit what you can see!"

During an overtake, the overtaking car bears the largest percentage of responsibility [24, 21]. This is because the vehicle in the front cannot always see cars behind. Moreover, in F1TENTH, vehicles cannot see behind at all since these cars consist only of one front-facing LiDAR. However, guidelines state that the car being overtaken must leave enough space (more than one width of the car) if the overtaking car has a significant portion of the car alongside [25, 24]. If this is the case, then it is the responsibility of both vehicles to pass through a corner without an accident. This can be seen in Fig. 4.1.

# Chapter 3

# Metodology

In this chapter, we start by introducing the problem we aim to solve. Then, we continue by defining the target scenario and the rules of overtaking (Section 3.1). Next, we introduce our algorithm's target platform (Section 3.3), which we also use for the evaluation. In the end, we describe a method on which we base our algorithm (Section 3.4). We also list the specific changes that need to be done to this method in order to use it for our purpose.

Competitors in the F1TENTH competition are usually using simple overtaking methods. The most common ones are motion primitives, Follow the Gap and lane switching due to their simplicity and low computational cost. However, these algorithms are usually utilized only as an avoidance algorithm. They are used when an obstacle is detected a few meters ahead of the vehicle on the reference trajectory. Therefore, overtakes only happen as a result of avoiding collisions.

We aim to develop a method that can identify possible overtaking zones on the racing track when we know the opponent's trajectory. To achieve this, we create an algorithm that can plan overtaking trajectories. We run this planner multiple times with different initial conditions, vehicle parameters, and opponent trajectories. In the end, we analyze the successful manoeuvres to create a map of possible overtaking zones on the track. The opponent's trajectory can be known in advance because of, e.g., previous laps or because it is using common racing strategies.

Other algorithms that can take advantage of the additional information about the possible overtaking zones can be later developed to plan better manoeuvres and make better decisions. Therefore, the future methods do not need to make spontaneous decisions, as in the case of avoidance methods.

## 3.1 Scenario definition

Since overtaking is a difficult problem, especially in racing, we narrow it down by specifying the target scenario. Moreover, to have a fair and safe race, all vehicles need to follow certain racing rules. Since we aim to use the method in the F1TENTH setting, we have to consider the F1TENTH competition rules [1]. However, the F1TENTH competition does not have well-defined overtaking rules. Therefore, we specify the considered racing rules ourselves below.

The scenario we consider is defined as follows:

- The goal of the ego vehicle is to overtake the opponent.
- The racing track is a flat surface with walls on each side as a boundary.
- The opponent's trajectory and current position are known.
- Only one opponent is considered.
- Both cars have to follow the racing rules.
- Only simple closed tracks are considered (no intersections or splits of the track).

**Racing rules.** As was previously mentioned, the F1TENTH competition does not have well-defined overtaking rules. Therefore, the considered rules are additionally based on the common racing rules from other competitions as well as racing guidelines and etiquette, which we discussed in Section 2.4.

We consider the following racing rules:

- Vehicles cannot crash into the opponent on purpose.
- The vehicle about to be overtaken cannot make a blocking move since F1TENTH cars usually do not have a rear-facing sensor.
- The vehicle cannot push the opponent to the track boundary.
- Vehicles cannot change directions suddenly.
- If a vehicle is trying to overtake on the inside of a corner, it has to have a significant portion of the vehicle alongside the opponent's vehicle during the corner entry.

## 3.2 Marking convention

**Ego vehicle.** A vehicle that is trying to overtake the opponent is called the ego vehicle. We use the blue color to depict this vehicle.

**Opponent vehicle.** A vehicle that is being overtaken is called the opponent vehicle. We use the red color to depict this vehicle.

**Position $(x, y)$.** The Cartesian coordinates of a vehicle. As we consider the car driving on a flat surface, the $z$-axis is omitted.

**Orietation $\theta$.** Orientation, also called heading, is the rotation of the vehicle around the $z$-axis.

**Pose.** A tuple of the position and the orientation of the vehicle.

**Path.** A sequence of Poses that compose a line the car is driving on.

**Trajectory.** Path where each Pose has an assigned time.

**Closed trajectory.** A trajectory where the first pose and the last pose are the same.

**Occupancy grid.** A 2D array of cells. Each cell can be unknown, empty, occupied, or something in between. If a cell is occupied, it has a value of 100, and free if it has a value of 0.

**Racing track.** A road made for the racing of vehicles. We represent the racing track as an occupancy grid.

**Corner apex.** The apex is a point on the inside of a corner that is the closest to the vehicle's trajectory.

**Corner entry.** The part of the corner before the corner apex.

**Corner exit.** The part of the corner after the corner apex.

## 3.3 Platform description

In this section, we describe the platform, which is the primary aim of our algorithm. We briefly explain the platform's hardware (Section 3.3.1) and the relevant parts of the software stack (Section 3.3.2) used to perform the real-life experiments in Section 5.3.7.

### 3.3.1 Hardware stack

**Mechanics.** We use the F1TENTH [26] race car as a platform for real-life experiments as well as its model for simulation experiments. The platform chassis is the Traxxas Slash 4x4 which is a front-wheel steering and all-wheel-drive 1:10 scale model of a car. It contains a single Velineon 3500 BLDC motor, which is driven using a Vedder Electronic Speed Controller (VESC). The motor provides torque to the centre shaft that drives the front and rear differentials. For the steering it utilizes a Traxxas 2075R servo motor. Dimensions of the vehicle can be seen in Fig. 3.2 and Table 3.1.

**Sensors.** The most important sensor on the platform is the Hokuyo UST-10LX LiDAR sensor. It has a 270° angular detection range with an angular resolution of 0.25°, 10 m detection range, and 40 Hz scanning frequency. The next sensor is a SparkFun 9DoF Razor IMU. It contains a three-axis accelerometer, gyroscope, and magnetometer. And finally, the VESC includes an observer of the motor speed, which uses a back EMF from the BLDC motor.

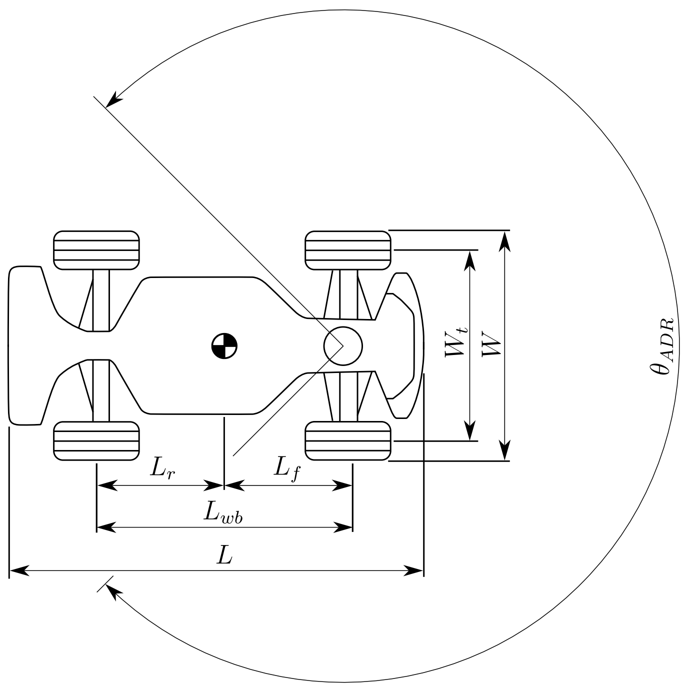


**Figure 3.1:** F1TENTH platform top view.

| Description | Symbol | Value | Unit |
|---|---|---|---|
| Vehicle width | $W$ | 0.3 | m |
| Vehicle length | $L$ | 0.55 | m |
| Track-width | $W_t$ | 0.25 | m |
| Wheelbase | $L_{wb}$ | 0.32 | m |
| CoG to front wheel axis | $L_f$ | 0.16 | m |
| CoG to rear wheel axis | $L_r$ | 0.16 | m |
| LiDAR angular detection range | $\theta_{ADR}$ | 270 | deg |

**Table 3.1:** Measurements of the F1TENTH platform

**Onboard computer.** The NVidia Jetson TX2 is used as an onboard computing unit. It contains a Dual-Core NVIDIA Denver 2 64-bit CPU, Quad-Core ARM® Cortex®-A57 MPCore, and NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores. Most of the necessary algorithms run on the CPU. The GPU is utilized only for a part of the localization.

### 3.3.2 Software stack

The software stack uses ROS2 Humble and can be divided into: (i) *an offline part*, which includes algorithms used to prepare for the race beforehand and (ii) *an online part*, that includes algorithms used during the deployment in the race. The offline part includes mapping and global racing line optimization algorithms, while the online part includes perception, local planning, and control algorithms.

### Offline part

**Mapping.** We use the `SlamToolbox` [27] as a mapping algorithm. It is a SLAM algorithm that uses the odometry of the vehicle and merges it with the lidar scan measurements to create a map representation of the environment. The resulting map is represented as an occupancy grid with values containing the probability of a cell being occupied.

**Global trajectory optimization.** An `ng_trajectory` [28] optimization toolbox was used for generating global time-optimal, curvature-optimal, and length-optimal trajectories. This toolbox is also a base for the local planning algorithm used to plan overtaking manoeuvers in this work. Therefore, we describe this algorithm in more detail in Chapter 4.

### Online part

**Localization.** For the localization on the track during the experiment, we use a particle filter from [29]. At first, it samples vehicle poses over the track, so-called particles, based on the current belief. Next, it uses odometry to get the future state. In the end, it compares the predicted laser scan with the measured one to update the belief of the vehicle's state.

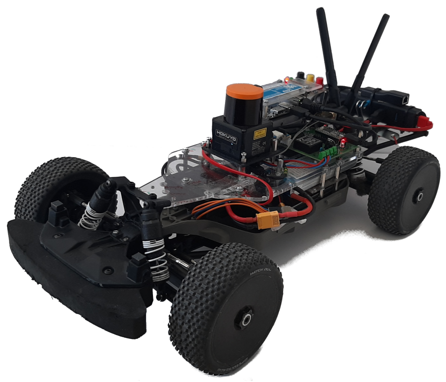


**Figure 3.2:** A photo of the F1TENTH platform.

**Trajectory tracking.** We use a Pure Pursuit algorithm for tracking the trajectory. It works by finding a look-ahead point on the trajectory in front of the vehicle, which is in the distance defined by a look-ahead value. The look-ahead value changes based on the current speed. The algorithm then uses inverse kinematics to calculate a steering angle that guides the vehicle towards the look-ahead point.

**Odometry.** The odometry, sometimes also called the dead reckoning, is a method of estimating the movement of a vehicle from sensors such as wheel encoders, speed sensors, etc. It is used by mapping and localization algorithms described above. The odometry is calculated using data from the IMU and the motor speed sensor. The measurements are propagated through a kinematic single-track vehicle model to estimate the movement of the car.

## 3.4 Trajectory optimization

In this section, we describe the method on which we base our algorithm and list all of the changes that need to be done to use it as a planner for the overtaking manoeuvres. The method is a tool for global trajectory optimization called `ng_trajectory` described in [28]. This tool uses the `DoubleFastGADiscreteOnePlusOne` method from the `Nevergrad` [30] library to perform the optimization.

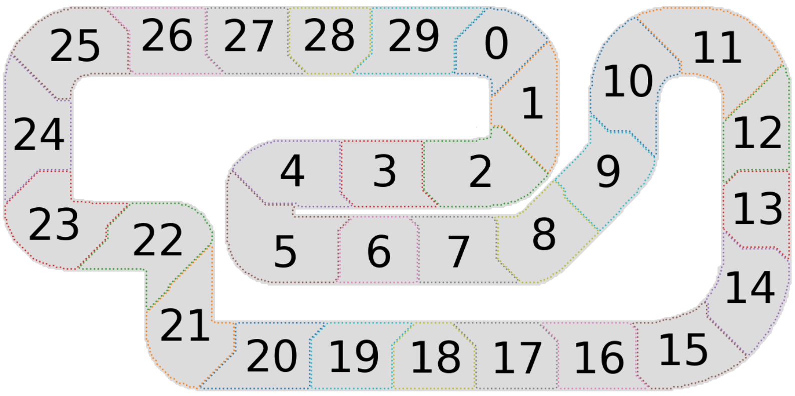


**Figure 3.3:** Example of a track segmented into 30 zones.

It works by first splitting the track into a predefined number of zones, such as in Fig. 3.3. Then, in every iteration of the optimization algorithm, it generates one point per zone using the Matryoska mapping from [28]. The Matryoska mapping uses a homeomorphism from a square to a track zone, which can be seen in Fig. 3.4. The mapped points are interpolated using a cubic spline, resampled, and checked if they are feasible. If this is not the case, the candidate is rejected by setting its fitness value to a large number. Otherwise, it continues by computing a speed profile for this path using the forward-backward method based on [11]. In the end, the fitness value is computed for this candidate and passed back to the optimization algorithm. The algorithm then samples the next generation of candidates based on the success of the candidates from the previous generation. The optimization algorithm repeats these steps to minimize the fitness value.

To use this global trajectory planner as an overtake manoeuvre planner, we need to extend it with the following:

- *Trajectory evaluation* that calculates the fitness value based on the success of the overtaking manoeuvre (Section 4.1.1),
- *Track progress calculation* to accurately determine which vehicle is ahead (Section 4.1.2),
- *Collision detection* with the opponent (Section 4.1.3), and
- *Interaction between cars* (Section 4.1.4).

These changes to the algorithm are described in detail in the following chapter.

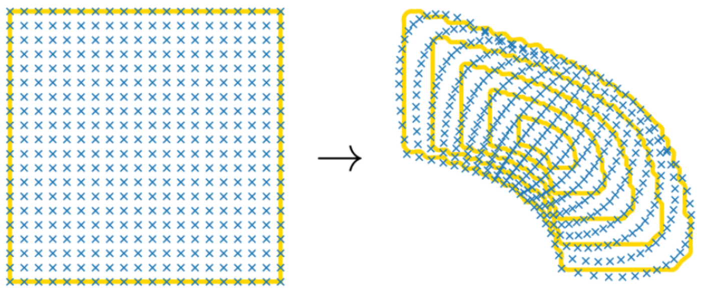


**Figure 3.4:** Example of the Matryoska mapping from a square to a track zone.[28]

# Chapter 4

# Method description

In this chapter, we explain in detail the algorithm changes introduced in Section 3.4. In the end, we show a few implementation details needed for the correct functionality of the planner.

## 4.1 Algorithm improvements

In this section, we explain the changes in the `ng_trajectory` algorithm.

### 4.1.1 Trajectory evaluation

After a trajectory is generated, it is checked whenever it is valid. At first, we check for curvature limits, track boundary violations, and acceleration limits. These checks can be performed fast, so they can quickly reject a lot of invalid candidate solutions. Next, we need to check for a collision with the opponent. Collision checks are described in detail in Section 4.1.3. If a collision with the opponent is found, it needs to be decided if it is the ego's or the opponent's fault. Vehicle interaction and determining the fault of the crash is described in Section 4.1.4. If the crash was ego's fault, we return the penalty. Equations for all the penalties are

$$\text{penalty} = \begin{cases} 100 \cdot N_{ib} \cdot P & \text{if out of track bounds,} \\ 100 \cdot N_{ic} \cdot P & \text{if curvature limit violation,} \\ P \cdot |a| & \text{if acceleration limit violation,} \\ N_{inv} \cdot P & \text{if collision (EGO's fault),} \end{cases} \tag{4.1}$$

where $N_{ib}$ is number of points that violate track boundaries, $N_{ic}$ is number of points that violate curvature limits, $a$ is the acceleration, and $P$ is a fixed penalty parameter. A valid trajectory was found if no penalty was applied. If the trajectory is valid, the cost is calculated in the following way:

$$\text{cost} = t_N + C_N + (1 - y_c) \cdot (1 - y_o) \cdot (f_n + 2C_{fo}) + y_c \cdot (f_n + C_{fo}), \tag{4.2}$$

where $t_N$ is the final time of the manoeuvre, $C_N$ is the final state cost, $C_{fo}$ is parameter to help favor overtaking, $f_n$ is an average progress difference, and $y_c$ and $y_o$ are variables defined as follows

$$y_c = \begin{cases} 1 & \text{if collision occured (not EGO's fault),} \\ 0 & \text{if no collision,} \end{cases} \tag{4.3}$$

$$y_o = \begin{cases} 1 & \text{if opponent was overtaken,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

The average progress difference is defined as

$$f_c = \frac{\sum_{i=0}^{N-1}(\hat{p}_i - p_i)}{N}, \tag{4.5}$$

where $N$ is the number of trajectory points, $\hat{p}_i$ and $p_i$ are the opponent's and ego's progress on the track, respectively. The progress calculation is described below.

### 4.1.2 Track progress calculation

We need to calculate the trajectory progress for both vehicles to determine which of them is ahead. This can be done using a centerline and searching for the closest point to the vehicle. Then, the progress is calculated as the length of the centerline from the beginning of the track to that point. However, if the centerline is not smooth, the closest point can jump a lot, even during small changes in the vehicle's position. We found that a more reliable way is to use the opponent's trajectory for the progress calculation.

### 4.1.3 Detecting a collision with the opponent

To determine if a collision occurred, we need to check for an intersection between vehicles. However, the vehicles are nonconvex; therefore, the intersection check would be hard to compute. To simplify the problem, we represent the vehicle's geometry by an oriented rectangle. We use the hyperplane separation theorem [31] to decide if the two rectangles are colliding. The theorem works by finding a separation axis onto which the two rectangles are projected. There is no collision if sets created by this orthogonal projection are disjoint.

However, this does not have to be checked all the time. If vehicles are far away or too close, we can easily determine the collision by calculating the distance between vehicles. There is always a collision if

$$d \leq \frac{W_1 + W_2}{2}. \tag{4.6}$$

where $d$ is the distance between centres of cars, and $W_1$, $W_2$ are the widths of the first and the second vehicles, respectively. On the other side, there is no collision if

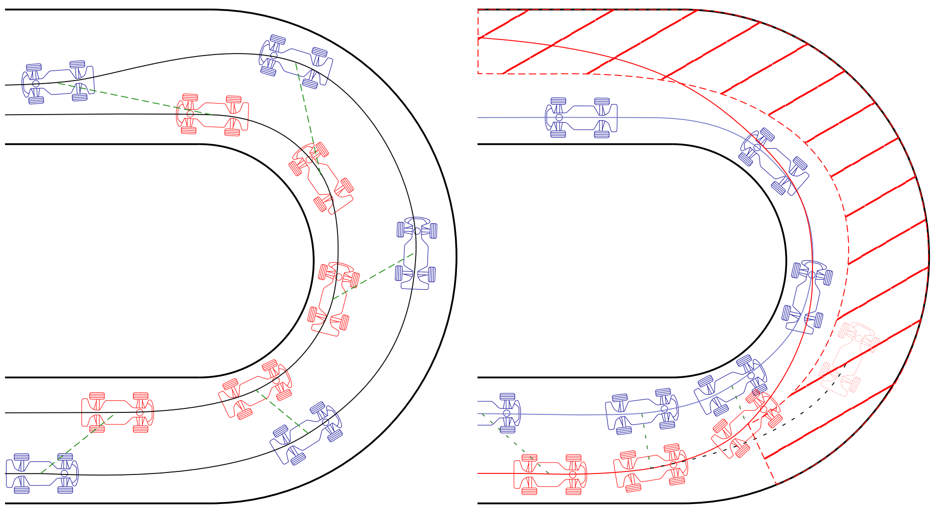
$$d > \sqrt{\left(\frac{W_1 + W_2}{2}\right)^2 + \left(\frac{L_1 + L_2}{2}\right)^2}, \tag{4.7}$$

where $L_1$, $L_2$ are the lengths of the first and the second vehicle, respectively. Moreover, in our case, we consider both cars to be the same, so $W_1 = W_2 = W$ and $L_1 = L_2 = L$. Therefore, the theorem needs to be used only if

$$\sqrt{W^2 + L^2} < d \leq W. \tag{4.8}$$

### 4.1.4 Interaction between cars

To plan effective overtakes, we need to model the interaction between vehicles. It would be almost impossible to perform an overtake with no interaction between cars at all. In the Fig. 4.1a, the opponent is following a minimum length trajectory. This is a very slow trajectory through a corner; moreover, it leaves a lot of space for the ego car. In this case, it is possible to make an overtake without disrupting the opponent's trajectory. On the other hand, in Fig. 4.1b, the opponent follows a trajectory that goes throughout the whole width of the track. Therefore, the ego vehicle would need to be much faster than the opponent to make an overtake without forcing him to choose a different trajectory.



**(a) :** No interaction between cars. **(b) :** Interaction between cars.

**Figure 4.1:** Example of an overtake with and without interaction. In the left figure **(a)**, the ego car (blue) overtakes the opponent car (red) without a simulated collision. Therefore, we assume no interaction between cars. However, in the right figure **(b)**, the simulation shows that the collision would occur if the opponent continued on its trajectory. However, at the time of the collision, more than half of the length of the ego car is along the opponent. Therefore, we assume that the opponent would choose a different trajectory to avoid causing a collision.

As written in Section 3.1, the opponent vehicle is not allowed to make a blocking move since the platform does not contain a rear-facing sensor Section 3.3.1. Therefore, the interaction between cars starts happening only after the ego car gets into the opponent's field of view (the side on the opponent's car).

As was mentioned in the rules (Section 2.4), if the overtake happens on the inside, the ego car needs to have a significant portion of the vehicle alongside the opponent during the corner entry. The opponent then has to leave space for the ego car so it does not push the ego car into the wall. Therefore, the simulated collision in Fig. 4.1b would be the opponent's fault, and we assume that the opponent is going to deviate from his original trajectory to give the ego vehicle enough space. We can now conclude that the overtake was successful. Moreover, we know the side that the opponent is on during the modelled crash. Therefore, the ego has to leave enough space on that side during the following few corners (red zone in Fig. 4.1b) since the opponent might drive alongside.

If the manoeuvre does not disrupt the opponent's trajectory, we assume no interaction between cars.

## 4.2 Implementation details

In this section, we explain the important implementation details of our method.

### 4.2.1 Initial conditions and final state feasibility

We assume that before the start of an overtake, the ego vehicle is already following the opponent for some time. As mentioned in Chapter 3, this is one of the ways to obtain the opponent's trajectory. Therefore, the initial position, orientation, and speed of the ego vehicle are determined based on the opponent's trajectory at a certain distance behind the opponent's current position.

To fix the initial position and orientation, we fix the first two consecutive points of the manoeuvre. Therefore, we do not optimize the trajectory points in the first two track segments that were described in Section 3.4.

To make sure that the final state of the manoeuvre is feasible, we compute the trajectory for 10 more meters. The chosen distance is enough for the car to stop from its maximum velocity. During these final 10 meters, the ego car cannot overtake. Therefore, if this final part of the trajectory is feasible, we can conclude that the final state of the manoeuvre is feasible as well. Moreover, we force the last position and the orientation of the manoeuvre to be close to the reference trajectory using additional cost. This way, the vehicle can continue on the reference trajectory after the manoeuvre is finished.

Another method to ensure the final state feasibility is to optimize the manoeuvre as a closed trajectory throughout the whole track. This automatically ensures the final state feasibility without a need for additional cost or increased horizon. On the other hand, this method does not scale well with the increasing size of the track. Therefore, we did not use this method.

### 4.2.2 Handling uncertainty

The uncertainties in the opponent's trajectory can be caused by an imperfect measurement of the opponent's position or by the opponent's imprecise trajectory tracking. We need to consider this uncertainty to be able to perform real-world experiments. We handle this uncertainty by inflating the rectangles that represent the vehicles by a constant value. The improvement to this method is described in Chapter 6.

### 4.2.3 Data gathering

As we described in Chapter 3, our main goal is to find possible overtaking zones on a track. After a manoeuvre is successfully found, we have the data about one possible overtake. However, during the optimization, the algorithm tries many different candidate solutions to find the optimal one (Section 3.4). Moreover, some of the candidate solutions can contain successful overtaking manoeuvres, even if they are not the best ones. Therefore, we save all of the candidate solutions that contain a successful overtake to collect much more data about the possible overtaking zones from a single optimization.

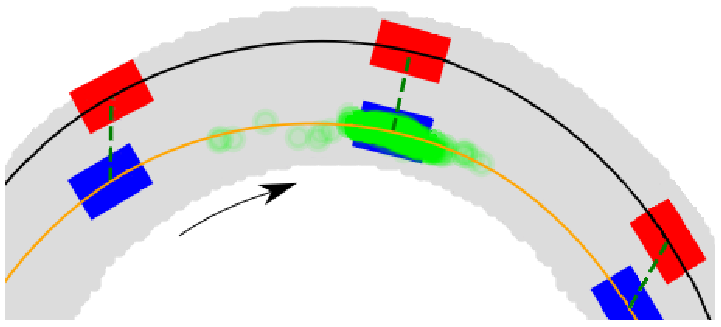


**Figure 4.2:** Example of different overtake positions of different manoeuvres gathered from a single optimization. Overtake positions are green (darker means more overtakes at that point), the ego vehicle is blue, which follows the manoeuvre with the best cost (yellow line), and the opponent is red. The positions of vehicles are drawn in one-second intervals.

# Chapter 5

# Experimental evaluation

In this chapter, we evaluate the developed method (described in Chapter 4) on multiple tracks. We use multiple opponent trajectories and ego vehicle parameters on each track to showcase how different vehicle parameters and trajectories affect overtaking with F1TENTH vehicles. To show the feasibility of the planned manoeuvres, we validate the results both in simulation and real-life experiments. The design of experiments is described in Section 5.1, racing tracks are introduced in Section 5.1.1, experimental results are shown in Section 5.2, and finally, the maneuver validation is in Section 5.3

## 5.1 The design of the experiments

In this section, we describe the design of experiments used to test the method. Furthermore, we introduce all of the tracks used for the experiments.

Each experiment consists of running the overtake planner (described in Chapter 4) 60 times while changing the initial position of the vehicles. We use the successful overtakes to create a map of possible overtaking zones. The opponent's initial positions are sampled uniformly on its racing trajectory. The initial position of the ego vehicle is always behind the opponent in a constant-time distance on the opponent's trajectory. The opponent's vehicle parameters are the same across all of the simulation experiments, and they are shown in Table 5.1. The default parameters of the ego vehicle are the same as the opponent's parameters; however, in each experiment, we change up to one parameter to show its effect on overtaking ability. We chose these parameters since we expect them to have a major effect on the overtaking zones. Moreover, these parameters can be easily changed on the real vehicle in the following way:

- The top speed ($v_{max}$) of the vehicle can be changed by changing the setting of the top speed limiter as well as switching to a battery with a different voltage.

- The weight ($m$) of the vehicle can be increased by adding additional mass. However, reducing the weight can be quite challenging due to the necessity of the used components.

- The friction coefficient ($\mu$) can be changed by changing a driving surface or using a different tire type.

However, during the testing of the algorithm on Track 6, we specified a different set of parameters to make these manoeuvres safe for real-life experiments. Additionally, in each experiment, we choose one of the following opponent's reference trajectories: (i) near-time-optimal, (ii) near-curvature-optimal, (iii) near-length-optimal, and (iv) Follow the Gap. The full list of performed experiments can be found in Table 5.2.

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Coefficient of friction | $\mu$ | 0.3 | - |
| Top speed | $v_{max}$ | 7.0 | m/s |
| Weight | $m$ | 3.68 | kg |

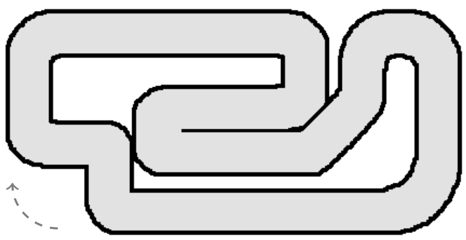
**Table 5.1:** Parameters of the opponent's vehicle and the default parameters of the ego's vehicle.

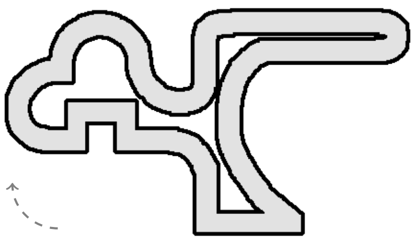
| ID | Ego's changed parameter | Value | Unit | Change [%] | Opponent's trajectory |
|---|---|---|---|---|---|
| 0 | - | - | - | 0 | near-time-optimal |
| 1 | weight | 2.95 | kg | -20 | near-time-optimal |
| 2 | weight | 4.42 | kg | +20 | near-time-optimal |
| 3 | friction | 0.315 | - | +5 | near-time-optimal |
| 4 | friction | 0.33 | - | +10 | near-time-optimal |
| 5 | friction | 0.36 | - | +20 | near-time-optimal |
| 6 | top speed | 7.35 | m/s | +5 | near-time-optimal |
| 7 | top speed | 7.7 | m/s | +10 | near-time-optimal |
| 8 | top speed | 8.4 | m/s | +20 | near-time-optimal |
| 9 | - | - | - | 0 | near-curvature-optimal |
| 10 | - | - | - | 0 | near-length-optimal |
| 11 | - | - | - | 0 | Follow the Gap |

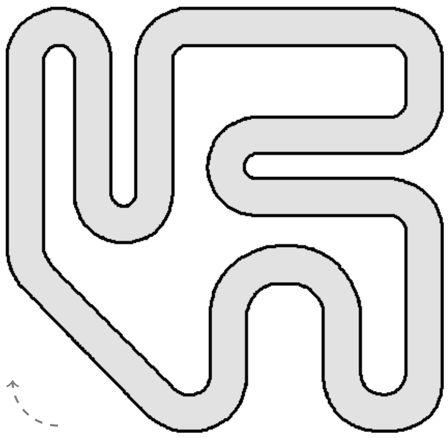
**Table 5.2:** List of scenarios.

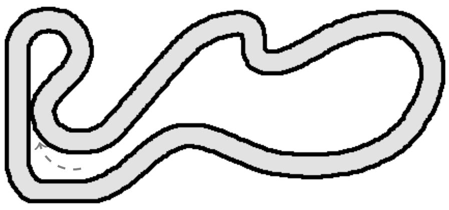
We use the F1TENTH gym [32] simulator to validate some of the manoeuvres planned by the overtaking planner. To communicate with the simulator, we use a ROS wrapper from [33]. Both vehicles run the exact same pipeline and have access to their as well as to the opponent's full state $(x, y, \theta, v)$.

Furthermore, we choose one manoeuvre that we execute on the real cars, which were described in Section 3.3. During the real-life experiment, the whole pipeline runs on an onboard computer. Both vehicles localize only using onboard sensors, and the opponent shares its position with the ego vehicle over the WiFi.


**(a) :** Track 1.


**(b) :** Track 2.


**(c) :** Track 3.


**(d) :** Track 4.

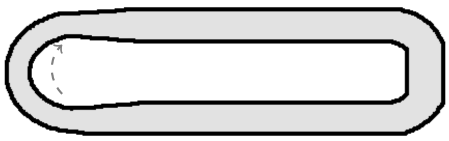

**(e) :** Track 5.

**Figure 5.1:** Images of racing tracks used for simulation experiments.

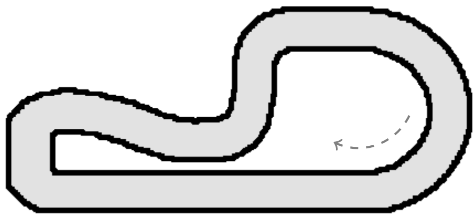
### 5.1.1 Racing tracks description

In this section we describe what kind of racing tracks we used to perform the experiments. We chose six tracks in total, from which we used five of them only in simulation and one for real-life experiments.

**Track 1.** This track (Fig. 5.1a) is a scaled-down, simplified version of the *Motorsport Arena Oschersleben*. This track was chosen since it was already used, in slight variations, multiple times at the F1TENTH competitions. It contains many 90° and 180° turns and a few straight sections. The width of this track is 2.7 m, and the length is 98 m.

**Track 2.** This track (Fig. 5.1b) contains many consecutive left and right corners while also being narrow; therefore, it is expected to be quite hard to perform an overtake. It also contains two straight sections with lengths of around 10 m. The width of this track is 2 m, with a length of 98 m.

**Track 3.** This track (Fig. 5.1c) has a width of 4.2 m and a length of 256 m, making this the widest and the longest of the tested tracks. It contains many high-speed corners between a lot of long straight sections. This track was selected so the cars would have a lot of overtaking opportunities since we expect it to be easier to overtake on a wider track.


(a) : The layout of Track 6.

(b) : A photo of Track 6.

**Figure 5.2:** Track 6 used for real-life experiments.

**Track 4.** This track (Fig. 5.1d) has a lot of curves and two 10-meter-long straight sections. It is the narrowest track, with a width of 1.8 m and a length of 126 m. Because this track contains many consecutive left and right corners while being narrow (similar to Track 2), we expect it to be quite hard to overtake the opponent on this track.

**Track 5.** This track (Fig. 5.1e) is a simple round track with a long straight. It was chosen to show how different parameters affect overtaking on a straight and during a simple 180° corner. It contains one corner with a width of 3.3 m, one corner with a width of 2 m, and two straights, each 28 m long. The whole length of this track is 83 m long.

**Track 6.** This track (Fig. 5.2) is used for real-life experiments. It contains a 15-meter-long straight, one long, fast corner, one hairpin, and two 90-degree turns. It was designed to contain many different features while still being able to fit inside a conference room. The track is 50 meters long, and the width is 1.95 meters.

## 5.2 Results

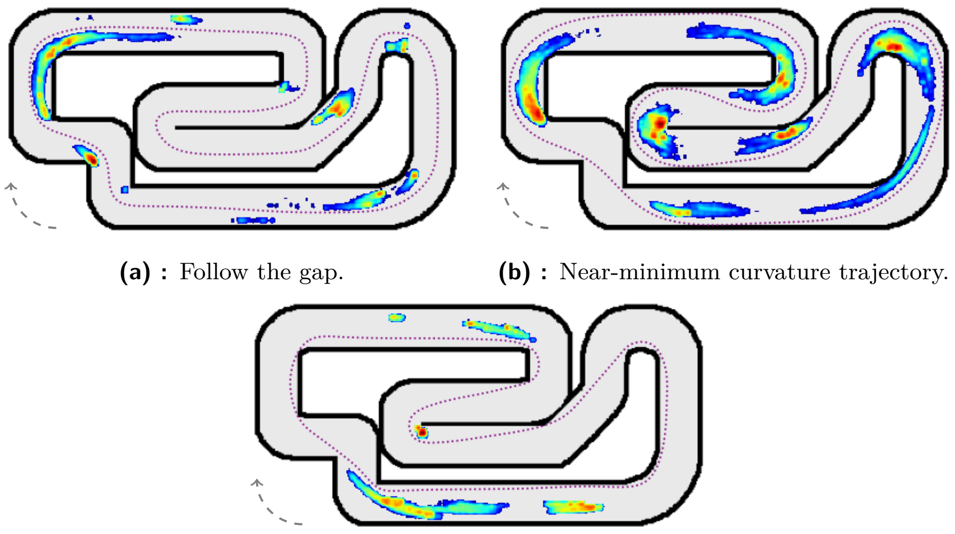
In this section, we show the results of the experiments described in Section 5.1. We present the results in the form of a so-called heat map, where parts of the racing track are coloured based on the number of overtakes. The used heat map has four main colours:

- *red* – most of the overtake happened at this place,
- *green* – some overtakes happened at this place,
- *blue* – only a few overtakes happened at this place, and
- *grey* – no overtakes happened at this place.

Every image contains the opponent's racing line drawn using a purple dotted line.

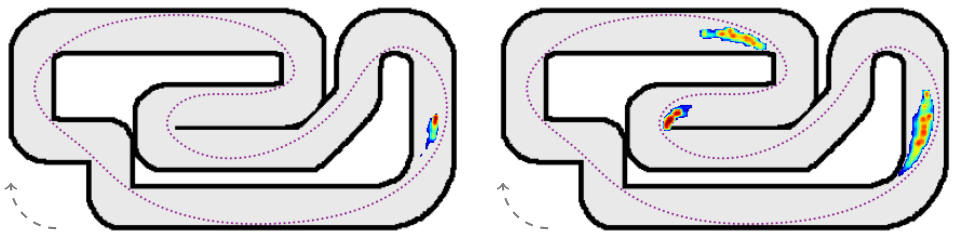
### 5.2.1 Track 1

The effect of different opponent trajectories on overtaking on Track 1 while the vehicles are equal is shown in Fig. 5.3. The opponent can be easily overtaken on the inside when following the near-minimum curvature trajectory (Fig. 5.3b). When the opponent uses the FTG algorithm, it takes up a lot of space on the track because it is driving in the middle of it. This behaviour blocks the ego vehicle, and therefore, the algorithm finds fewer overtakes. However, the FTG algorithm cannot drive that fast, and the ego vehicle can still overtake easily. Overtaking zones when the opponent follows a near-minimal length trajectory can be seen in Fig. 5.3c.



**(a) :** Follow the gap. **(b) :** Near-minimum curvature trajectory. **(c) :** Near-minimum length trajectory.

**Figure 5.3:** Overtaking zones on Track 1, while both vehicles are the same and the opponent drives different trajectory types. We can see that the most overtakes happen when the opponent follows a near-minimum curvature racing line.



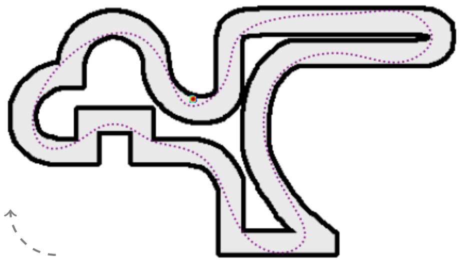
**(a) :** Increase by 10% ($\mu = 0.33$). **(b) :** Increase by 20% ($\mu = 0.36$).

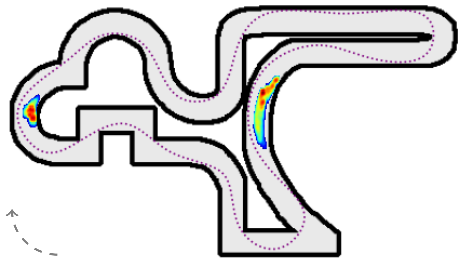
**Figure 5.4:** Overtaking zones on Track 1, while the ego car has increased friction, and the opponent drives its near-time-optimal trajectory.

When the opponent follows its near-time optimal trajectory, we did not find any overtakes when the vehicles are the same or even when the friction coefficient of the ego vehicle increased by 5%. We found some overtakes in one part of the track when we increased the friction coefficient by 10% (Fig. 5.4a). Naturally, most overtakes were found with the highest friction coefficient (20% increase); however, still only in three corners (Fig. 5.4a). Moreover, we also found no overtakes when we changed the weight of the ego vehicle. Moreover, even the increase in the top speed of the ego vehicle by 20% did not result in any overtakes.

### 5.2.2 Track 2

As we described in Section 5.1.1, we do not expect a lot of overtakes to happen on this track since it is narrow and contains many consecutive left-right turns. This hypothesis turned out to be true. The most overtakes on this track happen when the opponent follows a near-minimum curvature trajectory and the vehicles are equal; however, still only in two corners.


**(a) :** The opponent is driving on the near-minimum time trajectory while both vehicles have the same parameters.


**(b) :** The opponent is driving on the near-minimum curvature trajectory while both vehicles are equal.

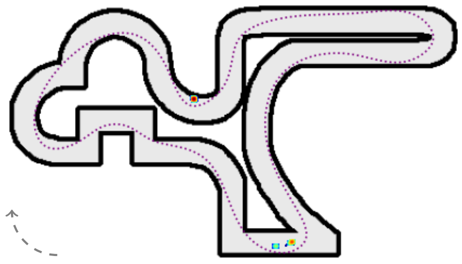
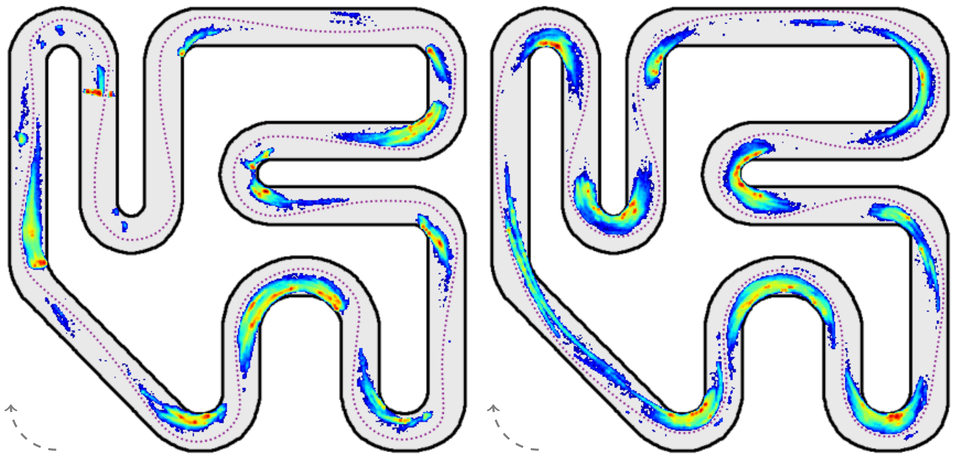

**(c) :** The opponent is driving its near-minimum time trajectory. The ego vehicle has improved friction coefficient by 20% ($\mu = 0.36$).

**Figure 5.5:** Overtaking zones on Track 2.

Interestingly, we found a few overtakes at one point on the track, even when both cars are equal, and the opponent follows a near-minimum time trajectory (Fig. 5.5a). But it is hard to show that the opponent's trajectory is, in fact, the minimum time trajectory. Therefore, we cannot conclude if this overtake would be possible even when the opponent follows the true time optimal trajectory. On the other hand, this demonstrates that even a small mistake can result in an overtake.

When the opponent follows his near-minimum time trajectory, even increasing the friction of the ego vehicle by 20% did not result in many overtakes (Fig. 5.5c). Moreover, changing the weight of the ego vehicle did not result in an increase or decrease in overtakes when compared to the number of overtakes with equal vehicles. A change in the top speed also did not change overtaking zones as the cars did not reach it for a long enough time.


**(a) :** Follow the gap. **(b) :** Near-minimum curvature trajectory.

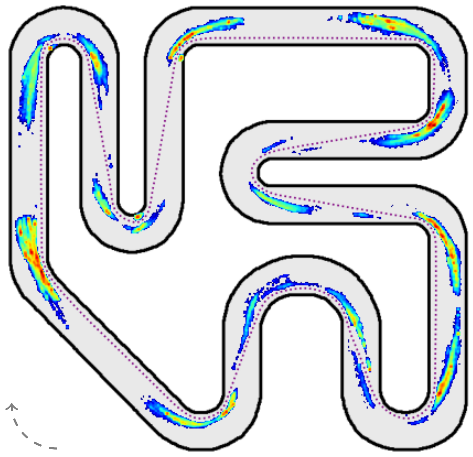

**(c) :** Near-minimum length trajectory.

**Figure 5.6:** Overtaking zones on Track 3, while both vehicles are the same and the opponent drives different trajectory types.

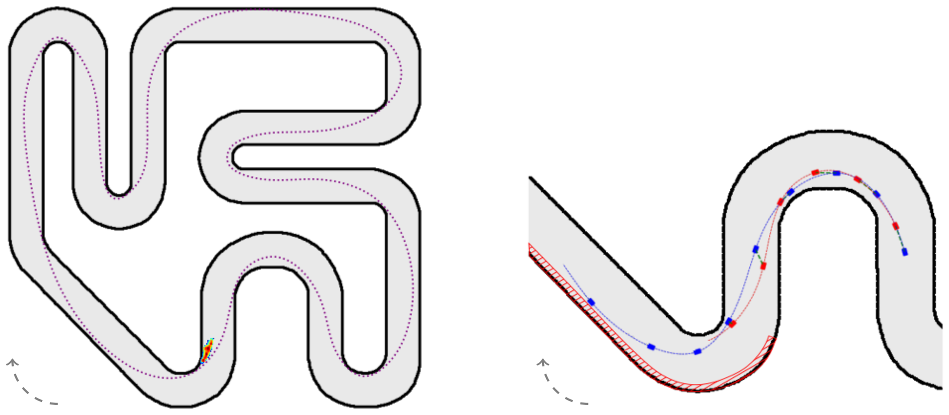
### 5.2.3 Track 3

As we discussed in Section 5.1.1, we chose this wide track as an example of a track where it should be easy to overtake. In fact, many overtakes were found when both vehicles were the same and the opponent used the FTG algorithm, followed a near-minimum curvature trajectory, or followed a near-minimum length trajectory. Since this track is very wide, the near-minimum length trajectory and the FTG algorithm, even though it drives in the middle of the track, cannot effectively block the ego vehicle.

When the opponent drives near-time optimal trajectory, some overtakes were found in the case of increasing ego's vehicle friction coefficient by 20%, 10%, 5% (Fig. 5.9), or even in the case of equal cars (Fig. 5.7).

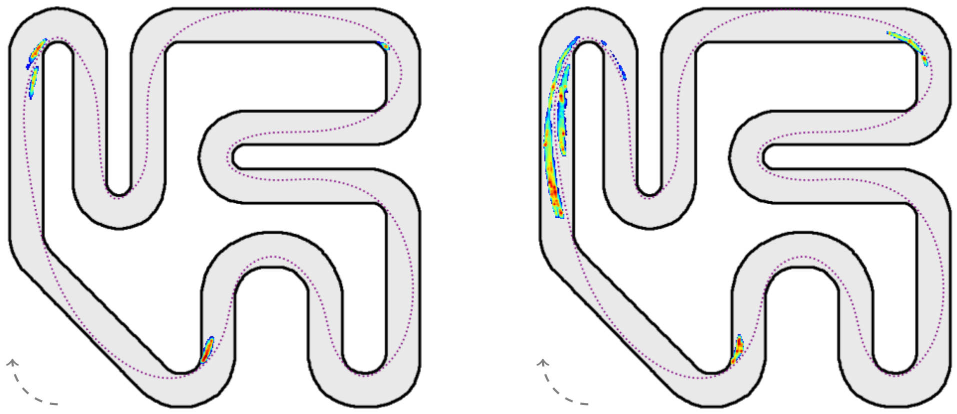
Increasing the top speed of the ego vehicle by 5% while the opponent follows its near-time optimal trajectory did not result in an increased number of overtakes. Some overtakes started occurring at the end of the long straight when the top speed of the ego vehicle was increased by 10% (Fig. 5.8a). An increase of the ego's top speed by 20% resulted in many overtakes on the long straight (Fig. 5.8b).

Changing the weight of the ego vehicle while the opponent follows its near-time optimal trajectory did not affect the number of overtakes on this track.



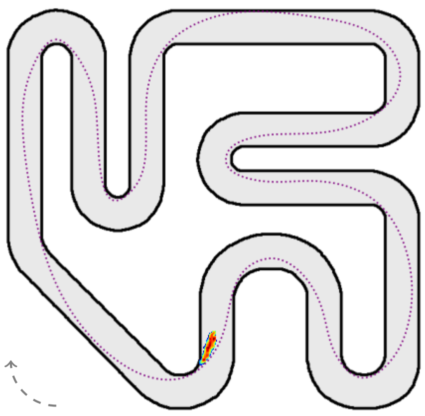
**(a) :** Overtaking zones.    **(b) :** Example of a planned maneuver.

**Figure 5.7:** Overtaking zones on Track 3 while both vehicles are the same and the opponent drives near-minimum time trajectory.

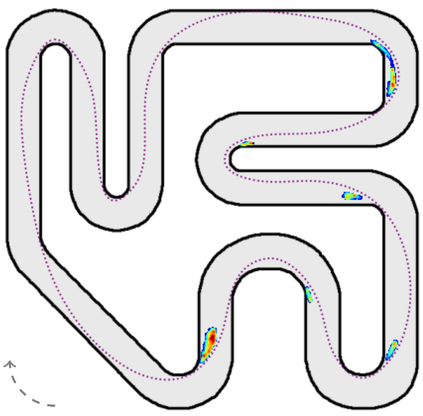


**(a) :** Increase by 10% ($v_{max} = 7.7\,\mathrm{m \cdot s^{-1}}$).    **(b) :** Increase by 20% ($v_{max} = 8.4\,\mathrm{m \cdot s^{-1}}$).

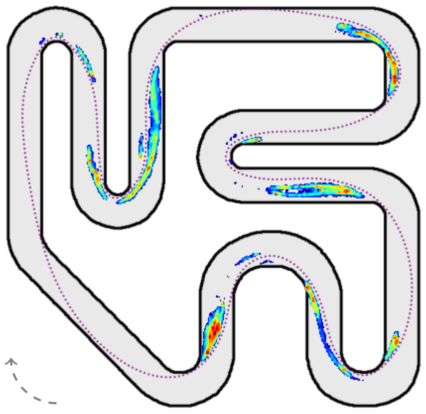
**Figure 5.8:** Overtaking zones on Track 3 while the ego vehicle has increased top speed and the opponent is driving its near-minimal time trajectory.



**(a) :** Increase by 5% ($\mu = 0.315$).



**(b) :** Increase by 10% ($\mu = 0.33$).



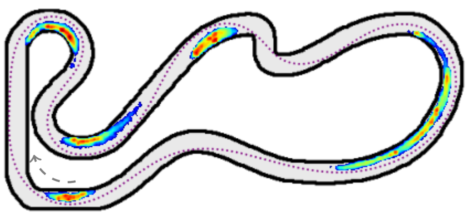
**(c) :** Increase by 20% ($\mu = 0.36$).

**Figure 5.9:** Overtaking zones on Track 3 while the opponent is driving its near-minimum time trajectory, and the ego vehicle has increased friction.

## 5.2.4 Track 4

As discussed in Section 5.1.1, we do not expect it to be hard to perform an overtake on this track (similar to Track 2). Moreover, the FTG algorithm we used was not able to finish this track because of the sudden left-right turn in the middle of the track.

We show the effect of different opponent trajectories on the overtaking zone while the vehicles are equal in Fig. 5.10. When the opponent follows the near-minimum curvature trajectory, the overtakes always happen on the inside of the corners in various parts of the track (Fig. 5.10a). On the other hand, when the opponent follows a near-minimum length trajectory, overtakes happen only in one part of the track (Fig. 5.10b). This is because the opponent always takes the inside line, and this track contains many consecutive left-right turns. Therefore, it does not leave enough space for the ego vehicle to perform an overtake in other parts of the track.


**(a) :** Near-minimal curvature.

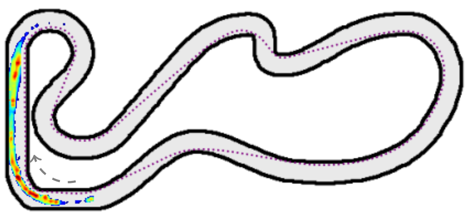

**(b) :** Near-minimal length.

**Figure 5.10:** Overtaking zones on Track 4, while both vehicles are the same and the opponent drives different trajectory types.

Moreover, it is also difficult to find overtaking manoeuvres when changing vehicle parameters. When the opponent follows its near-minimum time trajectory, we had to increase the ego's friction coefficient by 20% to find some overtakes (Fig. 5.11). Changing the weight of the ego vehicle again did not result in any overtakes. Moreover, this track does not contain any straights that are long enough to result in an overtake even when the top speed was increased by 20%.

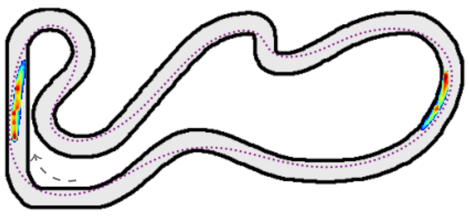
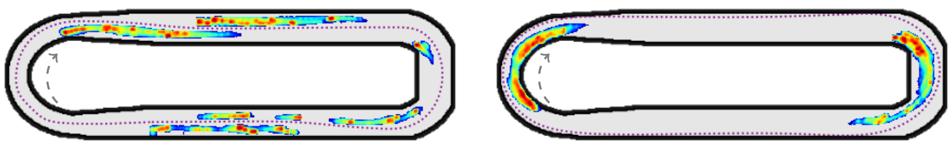


**Figure 5.11:** Overtaking zones on Track 4 while the opponent is driving its near-minimum time trajectory, and the ego vehicle has increased friction by 20% ($\mu = 0.36$).

### 5.2.5 Track 5

As discussed in the Section 5.1.1 this track was specifically designed to show the difference between overtaking on straights and in the corners.

In the Fig. 5.12, we can see the difference in overtaking zones when the opponent drives different trajectories, and both vehicles are equal. When the opponent uses the FTG, most overtakes happen on the straight (Fig. 5.12a). This is because the FTG algorithm is slow during straight sections since it is a reactive algorithm and cannot achieve the top speed. However, because the FTG drives in the middle of the track and the left corner is narrow, it effectively blocks the ego vehicle, which is unable to perform an overtake there. On the other hand, when the opponent follows a near-minimum curvature trajectory, no overtakes happen on the straight, and all of the overtakes happen in the corners (Fig. 5.12b). This happens because the minimum curvature trajectory is slower in corner sections. However, during the straight, the opponent's speed is the same as the ego's, so the ego car cannot catch up to it. The fewest overtakes happen when the opponent follows a near-minimum


**(a) :** Follow the gap. **(b) :** Near-minimum curvature.

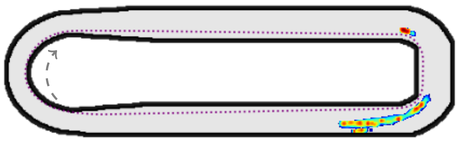
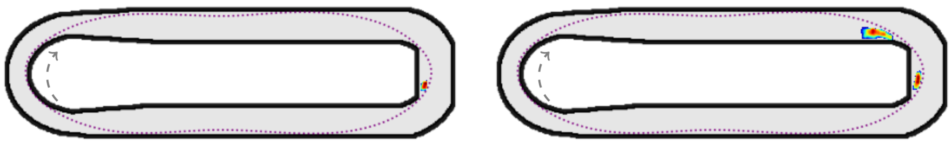

**(c) :** Near-minimum length.

**Figure 5.12:** Overtaking zones on Track 5, while both vehicles are the same and the opponent drives different trajectory types.

length trajectory (Fig. 5.12c).


**(a) :** Increase by 5% ($\mu = 0.315$). **(b) :** Increase by 10% ($\mu = 0.33$).

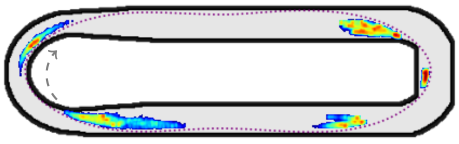
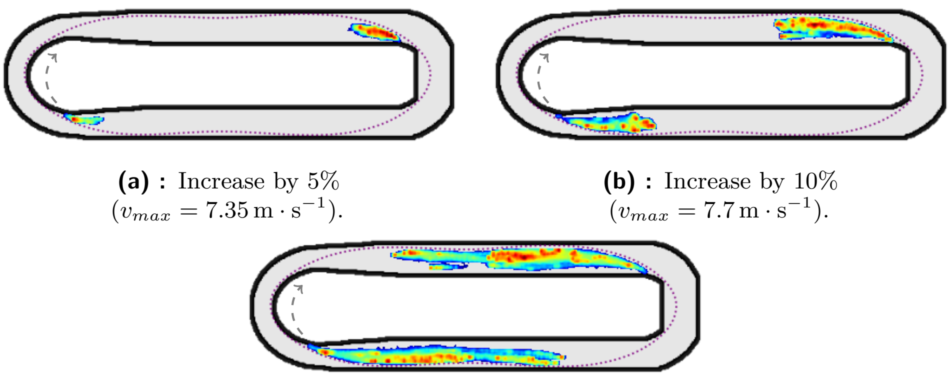

**(c) :** Increase by 20% ($\mu = 0.36$).

**Figure 5.13:** Overtaking zones on Track 5 while the opponent is driving its near-minimum time trajectory, and the ego vehicle has increased friction.

When we increased the vehicle's friction coefficient by only 5%, while the opponent follows its near-minimum time trajectory, the algorithm already found some overtakes, as shown in the Fig. 5.13a. When the friction coefficient was increased by 20% (Fig. 5.13c), overtakes happened during the corner, on the corner entry, and on the corner exit. However, no overtakes happened in the middle of the straight since the vehicles have the same maximum speed, and the friction coefficient does not have a significant effect anymore.

On the other hand, the opposite situation happened when we increased the maximum speed of the ego vehicle while the opponent followed its near-minimum time trajectory (Fig. 5.14). Since the maximum speed has an effect on the straights, no overtakes happened during corners. Even after the ego car reaches a higher maximum speed, it still takes some time to catch up to the opponent. Therefore, when we increased the top speed by 5%, we only observed overtakes at the end of the straights, where the ego vehicle had enough time to catch up to the opponent (Fig. 5.14a). The more we increased



**(a) :** Increase by 5% ($v_{max} = 7.35\,\mathrm{m \cdot s^{-1}}$).

**(b) :** Increase by 10% ($v_{max} = 7.7\,\mathrm{m \cdot s^{-1}}$).

**(c) :** Increase by 20% ($v_{max} = 8.4\,\mathrm{m \cdot s^{-1}}$).

**Figure 5.14:** Overtaking zones on Track 4 while the opponent is driving its near-minimum time trajectory, and the ego vehicle has increased top speed.

the top speed, the sooner the ego vehicle caught up to the opponent and performed the overtake.

No overtakes were observed on this track when the cars were equal, and the opponent was tracking its near-time optimal racing line. Moreover, changing the weight of the vehicle also did not result in any overtaking.

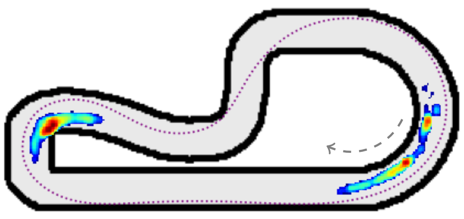
### 5.2.6 Track 6

As previously mentioned in Section 5.1.1, we aim to validate an overtaking manoeuvre performed on this track in real life. Therefore, we have to slightly adjust the vehicle parameters to make it safe for real-life experiments. We performed three different experiments on this track.

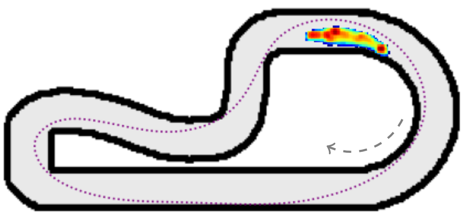
The first experiment consists of the opponent following near-minimum curvature trajectory while the vehicles are equal (Fig. 5.15a). The method found two possible overtaking zones in two different corners. As mentioned in the previous experiments, the minimum curvature trajectory is locally slow during corner sections, and the ego can perform the overtake on the inside. We successfully executed an overtaking manoeuvre from this experiment in the simulation as well as in real life (Section 5.3.7).

The second experiment consists of the opponent following the near-minimum time trajectory while its friction coefficient is reduced by 30%. One possible overtaking zone was found as shown in Fig. 5.15b. We successfully executed an overtaking manoeuvre from this experiment in the simulation (Section 5.3.5).

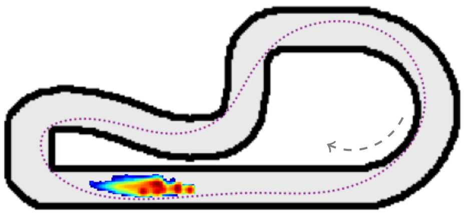
The final experiment on this track consists of the opponent following the near-minimum time trajectory while his top speed is decreased by 30%. A possible overtaking zone was found at the end of the straight section, as the ego can achieve a higher top speed. We successfully executed an overtaking manoeuvre from this experiment in the simulation (Section 5.3.6).



**(a) :** Overtaking zones while the vehicles are exactly the same and the opponent follows near-minimum curvature trajectory.



**(b) :** Overtaking zones while the opponent has reduced the friction coefficient and follows its near-minimum time trajectory.



**(c) :** Overtaking zones while the opponent has reduced top speed and follows its near-minimum time trajectory.

**Figure 5.15:** Overtaking zones on Track 6.

## 5.3 Maneuver validation

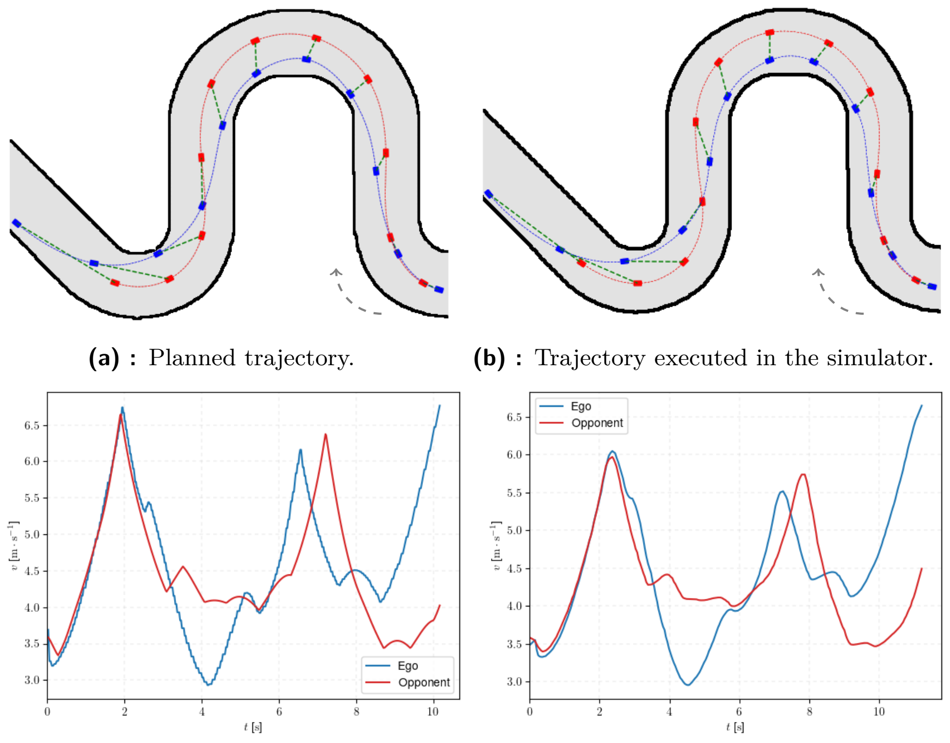
In this section, we validate some of the manoeuvres found during the experiments in the previous section. We execute them in the simulation and in real life to show that it is possible to execute them and, therefore, to show their feasibility.

### 5.3.1 Maneuver 1

In this manoeuvre, both cars are equal, and the opponent is following a near-minimum curvature trajectory.

Even though both vehicles are equal, the opponent's trajectory throughout the corner is so wide that we can easily find a faster trajectory.

There is no interaction between vehicles in this manoeuvre, and the ego vehicle overtakes the opponent successfully. This is an easy manoeuvre to execute because of the large distance between vehicles during the manoeuvre.



**(a) :** Planned trajectory. **(b) :** Trajectory executed in the simulator.

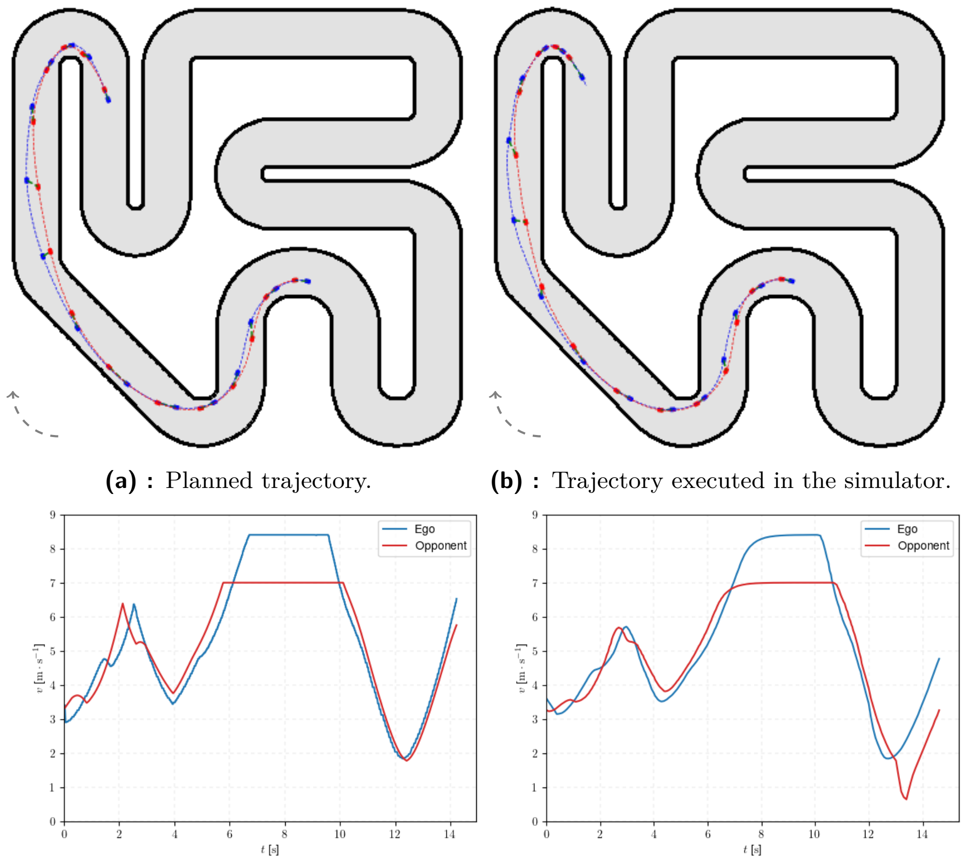
**(c) :** Planned vehicle speed. **(d) :** Vehicle speed in the simulator.

**Figure 5.16:** A comparison between the planned manoeuvre and the manoeuvre executed in the simulation. The manoeuvre is planned on Track 3 while both vehicles are equal and the opponent is following a near-minimum curvature trajectory. Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

### 5.3.2 Maneuver 2

In this manoeuvre, the opponent follows its near-minimum time trajectory while the ego vehicle has its top speed increased by 20%.

From the planned speed in Fig. 5.17b, it can be seen that both vehicles have similar speeds up to the point of reaching the top speed of the opponent's vehicle in around 7 s. At this point, the ego vehicle starts catching up to the opponent and the overtake starts. No collision was detected in the planned manoeuvre. However, in Fig. 5.17d in a time around 13.5 s the opponent vehicle slows down to avoid a collision. This can be caused by the imperfect execution of the planned manoeuvre in the simulation. However, the overtake was successful in the simulation.



**(a) :** Planned trajectory. **(b) :** Trajectory executed in the simulator.

**(c) :** Planned vehicle speed. **(d) :** Vehicle speed in the simulator.

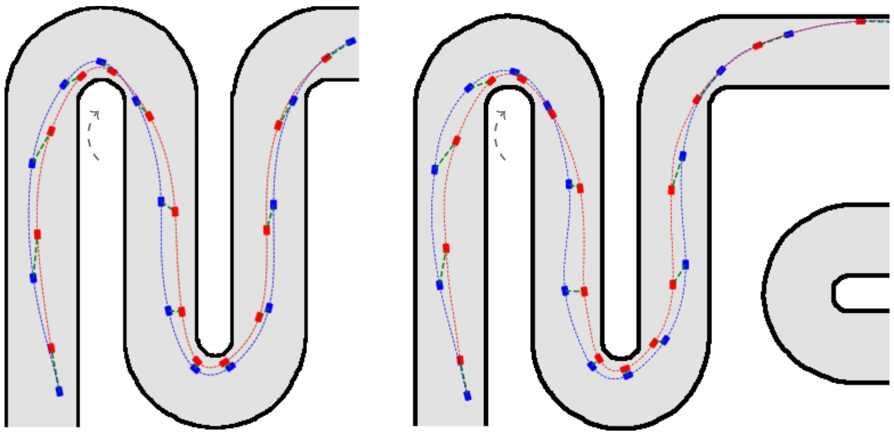
**Figure 5.17:** A comparison between the planned manoeuvre and the manoeuvre executed in the simulation. The manoeuvre is planned on Track 3 while the opponent is following its near-minimum time trajectory, and the ego vehicle has increased top speed by 20%. Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

### 5.3.3 Maneuver 3

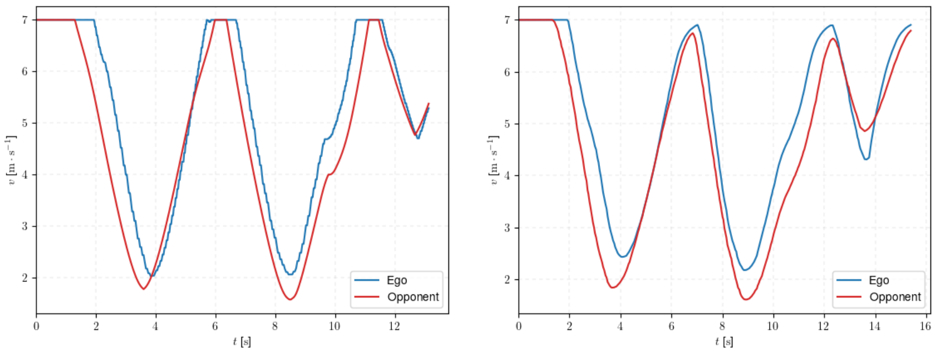
In this manoeuvre, the opponent is following its near-minimum time trajectory, while the ego vehicle has its friction coefficient increased by 20%. The ego vehicle takes advantage of it by taking the outside line throughout the corner and overtaking the opponent on the outside without a collision.

This manoeuvre showcases the ability of the algorithm to always prioritize the overtaking manoeuvres without collision.

The positions of the vehicles, as well as velocities in the simulation, are close to the planned ones (Fig. 5.18).



**(a) :** Planned trajectory. **(b) :** Trajectory executed in the simulator.



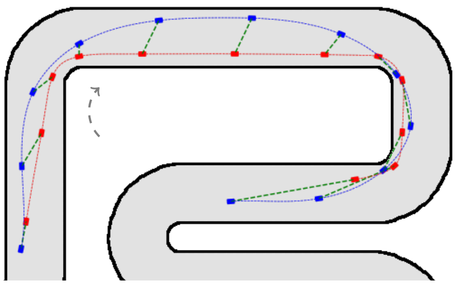
**(c) :** Planned vehicle speed. **(d) :** Vehicle speed in the simulator.

**Figure 5.18:** A comparison between the planned manoeuvre and the manoeuvre executed in the simulation. The manoeuvre is planned on Track 3 while the opponent is following its near-minimum time trajectory, and the ego vehicle has increased friction coefficient by 20%. Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

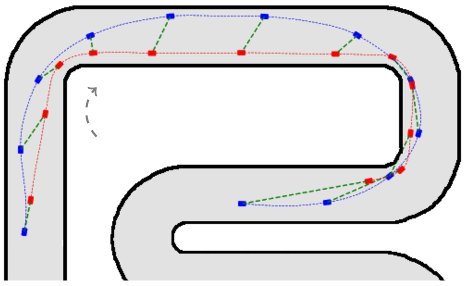
### 5.3.4 Maneuver 4

In this manoeuvre, both vehicles are the same; however, the opponent is following a near-minimum length racing line. The minimum length trajectory is fast on the straights but suboptimal in the corners. The ego vehicle follows a trajectory that goes from the outside of the corner to the inside and then again to the outside. This trajectory has smaller curvature which allows the ego vehicle to go faster, and in this case, it is enough to overtake the opponent even though the vehicles are exactly the same.

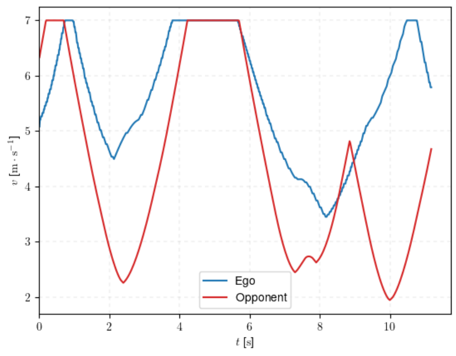
The vehicle positions in the simulation, as well as velocities, are close to the planned ones.



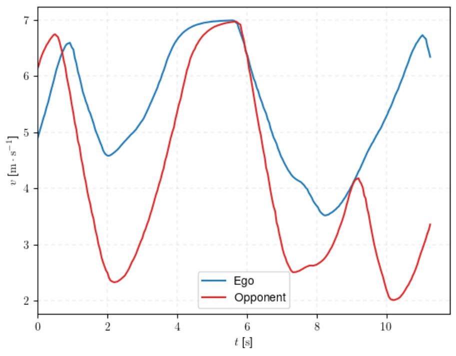
**(a) :** Planned trajectory.



**(b) :** Trajectory executed in the simulator.



**(c) :** Planned vehicle speed.



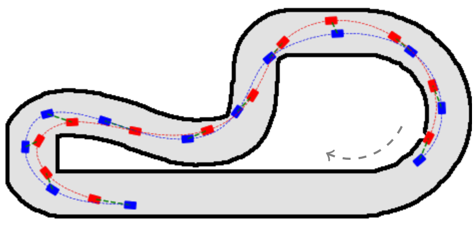
**(d) :** Vehicle speed in the simulator.

**Figure 5.19:** A comparison between the planned manoeuvre (left) and the manoeuvre executed in the simulation (right). The manoeuvre is planned on Track 3 while both vehicles are equal and the opponent is following a near-minimum length trajectory. Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

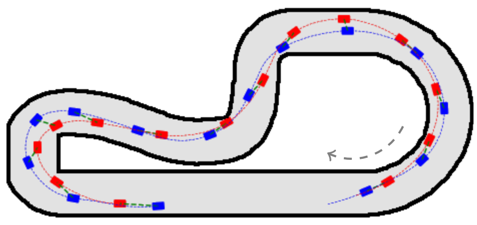
### 5.3.5 Maneuver 5

In this manoeuvre, the opponent is following its near-minimum time trajectory while its friction coefficient is reduced by 30%. The ego vehicle is able to take advantage of an increased friction coefficient and overtake the opponent on the inside on the small straight on the top of the track (Fig. 5.20).

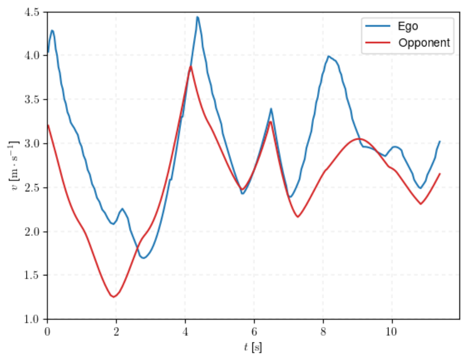
The ego vehicle successfully executed the manoeuvre in the simulation and overtook the opponent. The vehicle velocities in the simulation are close to the planned ones, as can be seen in Fig. 5.20.



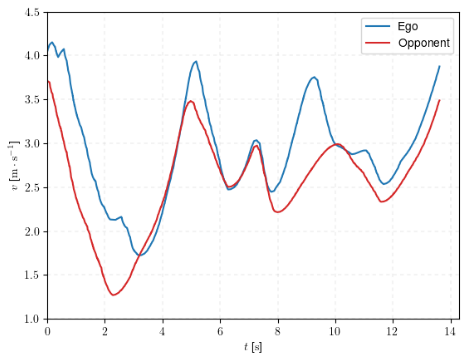
**(a) :** Planned trajectory.



**(b) :** Trajectory executed in the simulator.



**(c) :** Planned vehicle speed.



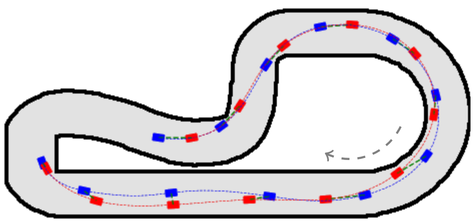
**(d) :** Vehicle speed in the simulator.

**Figure 5.20:** A comparison between the planned manoeuvre (left) and the manoeuvre executed in the simulation (right). The manoeuvre is planned on Track 6 while the opponent is following near-minimum time trajectory while its friction coefficient is decreased by 30% ($\mu = 0.19$). Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

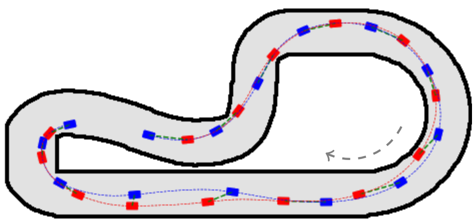
### 5.3.6 Maneuver 6

In this manoeuvre, the opponent follows its near-minimum time trajectory while its maximum speed is reduced by $1\,\mathrm{m}\cdot\mathrm{s}^{-1}$ when compared to the ego car. The overtake is happening on the straight.

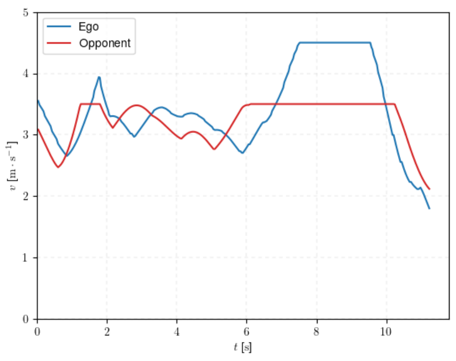
The planning algorithm found a collision of vehicles during these manoeuvres, which can be seen in Fig. 5.21a on the left. However, the ego vehicle correctly overtook the opponent, and the opponent would crash into the ego vehicle from behind, causing the crash. Therefore, the algorithm correctly calculated that it would be the opponent's fault and also classified the manoeuvre as successful. During the execution of this manoeuvre in the simulation, the opponent has to slow down to avoid the collision. The speed reduction can be seen in Fig. 5.21d in time around 11.5 s and 13 s.



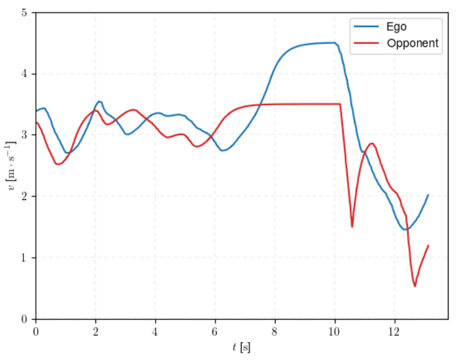
**(a) :** Planned trajectory.



**(b) :** Trajectory executed in the simulator.



**(c) :** Planned vehicle speed.



**(d) :** Vehicle speed in the simulator.

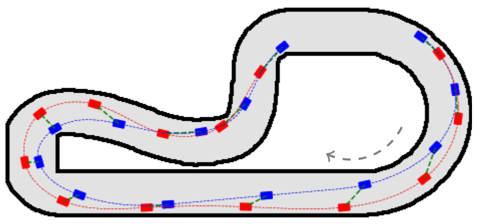
**Figure 5.21:** A comparison between the planned manoeuvre (left) and the manoeuvre executed in the simulation (right). The manoeuvre is planned on Track 6 while the opponent is following near-minimum time trajectory while its top speed is decreased by 30% ($v_{max} = 3.5$). Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

### 5.3.7 Maneuver 7

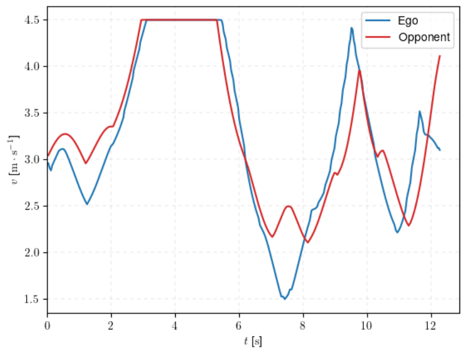
We chose to verify this manoeuvre in a real-life experiment. Both vehicles have the same parameters, however the opponent is following the near-curvature optimal racing line.

The position of the vehicles in the simulation (Fig. 5.22c) is close to the planned position (Fig. 5.22a). However, the position of the vehicles during the execution of the manoeuvre in real life is a little bit different. The most noticeable difference is in the Fig. 5.22e on the straight, where it looks like the vehicles collided. However, as can be seen in Fig. 5.23, this is not the case. The reason is that the position of the vehicles was measured using the localization onboard the vehicles which is not perfect.

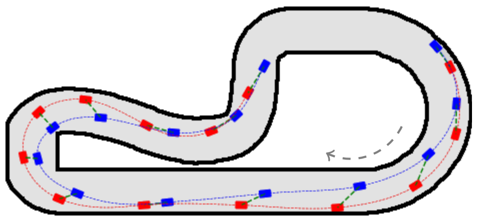
The velocity of the vehicles is close to the planned velocity in the simulation as well as in the real-life experiment.



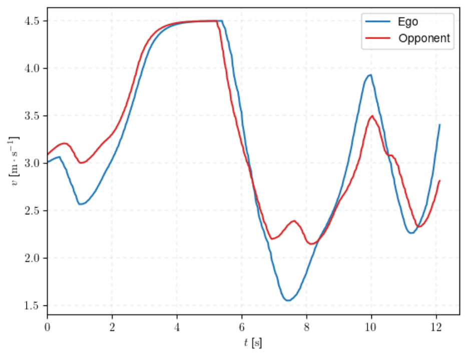
**(a) :** Planned trajectory.



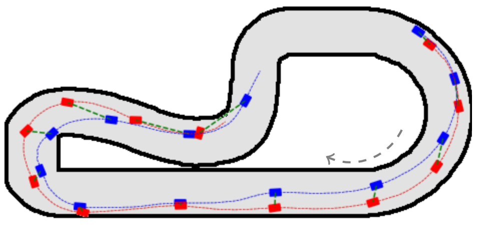
**(b) :** Planned vehicle speed.



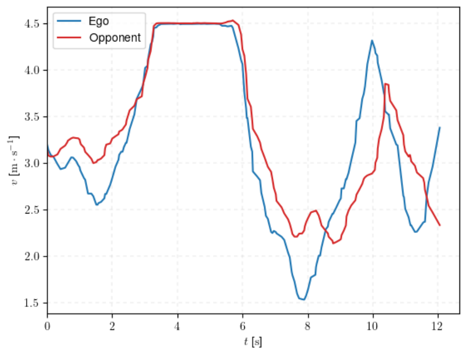
**(c) :** Trajectory executed in the simulator.



**(d) :** Vehicle speed in the simulator.



**(e) :** Trajectory executed in the real life.



**(f) :** Vehicle speed in the real life.

**Figure 5.22:** A comparison between the planned manoeuvre and the manoeuvre executed in the simulation. The manoeuvre is planned on Track 6 while both vehicles are the same and the opponent is following a near-minimum curvature trajectory. Vehicle positions are shown every second. The green line connects vehicles at the same time instance.

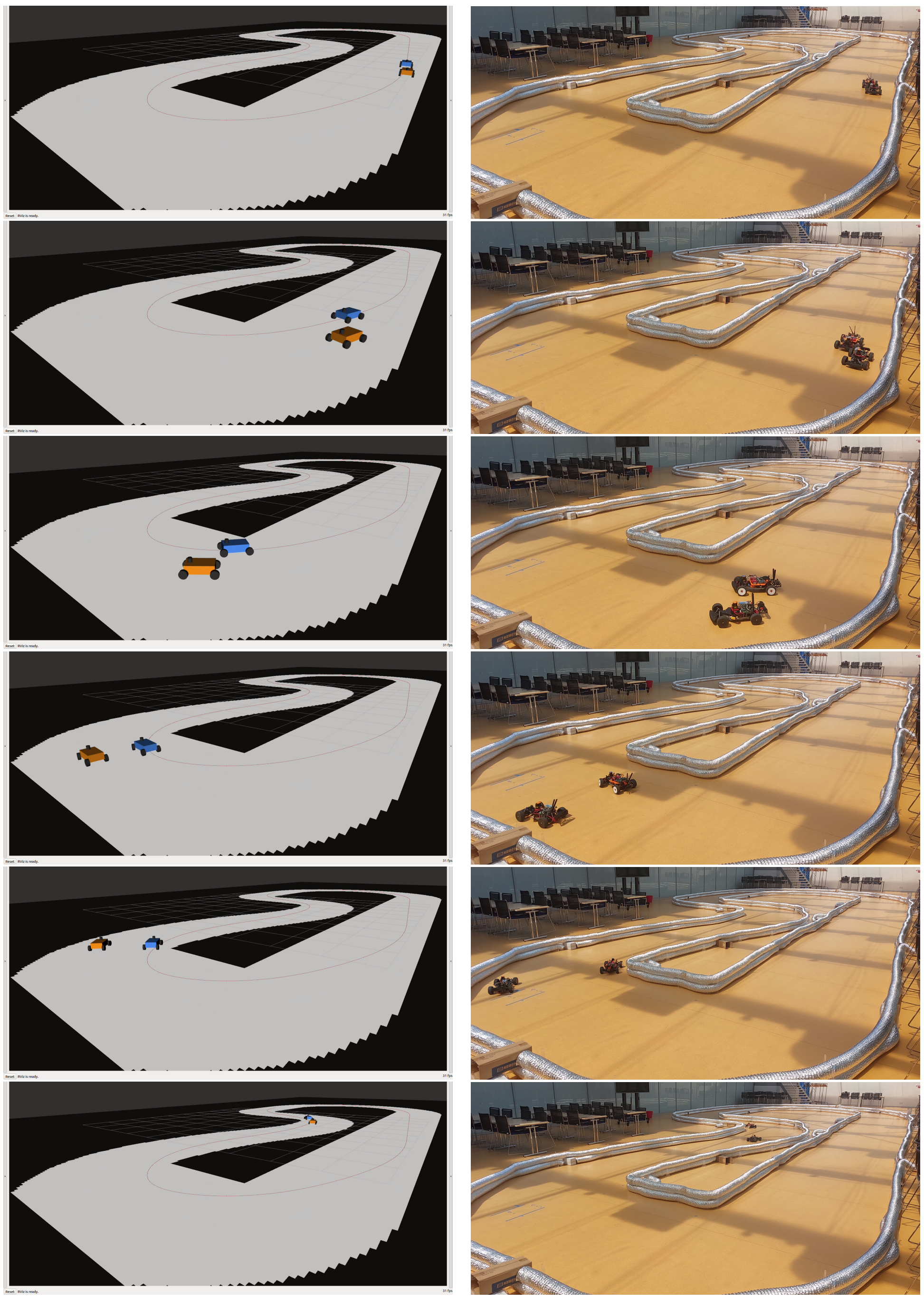


**Figure 5.23:** Comparison between simulated (left) and real-life (right) overtake on the Track 6. The opponent is following a near-minimal-curvature trajectory. Both vehicles are the same.

# Chapter 6

# Future work

For future work, we aim to improve on the drawbacks of the method presented in this work. The main drawback is the localization of the ego and the opponent vehicles (Fig. 5.22e). This greatly affects the ability of the ego vehicle to execute the overtake manoeuvre in real life and can cause a crash with the opponent.

**Runtime.** For the runtime, we have a few options. Firstly, we can improve it by optimizing the code (better use of Numpy vectorization), trying to implement this method in another programming language (e.g. Julia), or trying a different optimization method. However, a better solution would be to develop a different method that can run in real-time and is able to utilize the information about the possible overtaking zones computed by our algorithm. This way, the possible zones could be computed in advance for different types of opponents. Then, during the race, the algorithm would identify the type of opponent and try to pass him at parts of the track with a high chance of overtaking.

**Uncertainty handling.** The uncertainty within this method comes from the localization of the ego vehicle as well as the opponent and imperfect trajectory tracking. For example, due to imperfect trajectory tracking, the opponent does not have to take the exact same route every lap. The current handling of this uncertainty is inflating the size of the rectangles representing the vehicles by a constant value. However, this is not optimal since the opponent's tracking can have different performances in different parts of the track. Therefore, a possible improvement could be to record the opponent's trajectory over a few different laps and use inflation of vehicles based on the disparity of trajectories.

**Deployment in the race.** To use the preplanned trajectories in the race, we need to develop some safety methods. If an opponent goes off the expected trajectory, e.g., due to an error in the localization, the algorithm needs to detect the change and safely avoid the collision.

We can improve the odometry of the vehicle from Section 3.3.2 by using a more complex vehicle model or data from additional sensors such as wheel encoders. This can greatly improve the estimation accuracy of the vehicle's movement, which can further improve the accuracy of the localization.

Additionally, in the real race, the opponent won't share its position with the ego vehicle. Therefore, some methods to detect and estimate the opponent's position need to be used.

# Chapter 7

# Conclusion

In this work, we successfully implemented an algorithm that identifies overtaking zones when the opponent's trajectory is known in advance. To identify these zones, we ran the planning of the overtaking manoeuvres multiple times on each track and used the successful manoeuvres to create a heat map of the overtaking zones. Our implementation of the overtake planner is based on the `ng_trajectory` toolbox that uses non-gradient optimization to find an optimal racing line on the track (Section 3.4). We improved this toolbox so it can perform planning of overtaking manoeuvres with one opponent on the track (Chapter 4).

To test the method, we ran the algorithm on seven different tracks with different opponent trajectories and vehicle parameters (Chapter 5). The algorithm successfully identified possible overtaking zones. The results show that changing the weight of the F1TENTH vehicles does not change the overtaking zones on the tracks. This can be because we do not use a good enough model for manoeuvre planning, or the weight does not have that big of an effect on overtaking in such a small car. Increasing the top speed of the vehicles improved overtaking on straights when the top speed was achieved for enough time (Section 5.2.5). However, the tracks in the F1TENTH races usually contain a lot more corners than straights. Therefore, the most important of the tested parameters is the friction coefficient. Increasing it allows vehicles to go faster through the corners, take tighter turns, accelerate faster, and decelerate faster than the opponent (Section 5.2.2, Section 5.2.3). Therefore, increasing the friction coefficient is the most important, from the tested parameters, to increase the chance of the overtake. So good and clean tyres should be the priority when racing.

When analyzing different trajectories of the opponent, we found that the easiest one to overtake is usually the minimum curvature trajectory (Section 5.2.1). This trajectory always takes the outside line in corners while leaving enough space for the overtaking vehicle, resulting in a locally slow trajectory and many overtaking opportunities. The Follow the Gap algorithm is easy to overtake since it is a reactive algorithm and cannot utilize the full power of the vehicle. However, sometimes, it can still be tricky to overtake the FTG algorithm. Depending on the track, the FTG can block a significant portion of it since the FTG drives in the middle of the road. The minimum length trajectory is slow in the corners but fast on the straight. If the track is wide enough, it is easy to overtake the opponent that follows this trajectory (Section 5.2.3). However, if the track is narrow with many consecutive left-right turns, it can be extremely hard to overtake the opponent (Section 5.2.2).

To show the dynamic feasibility of the manoeuvres, we tested a few of them in an F1Tenth gym simulator and in real life (Section 5.3). The vehicle could perform the planned manoeuvres in the simulator without major problems. On the other hand, in real-life experiments, the onboard localization and imperfect trajectory tracking caused major problems during the execution of the manoeuvres (Section 5.3.7).

The biggest issue we had in this work was the localization of the opponent and ego vehicles during real-life experiments. The vehicle position during the execution of manoeuvres sometimes jumped by tenths of a meter, which sometimes caused a collision with the opponent. This localization error needs to be addressed to be able to execute more complex overtaking manoeuvres. This could be solved by an algorithm that could change the manoeuvre online to avoid unanticipated changes in the positions of both vehicles. Other drawbacks of this method and some solution proposals are described in detail in Chapter 6.

# Appendix A

# Bibliography

[1] F1TENTH, "F1TENTH Rules CPS-IoT 2024." `https://cpsweek2024-race.f1tenth.org/rules.html`, 2024. Accessed: 2024-05-08.

[2] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: "follow the gap method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[3] N. Otterness, "The "disparity extender" algorithm, and f1/tenth." https://www.nathanotterness.com/2019/04/the-disparity-extender-algorithm-and.html, April 2019. Accessed: 2010-09-30.

[4] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, "Comparing deep reinforcement learning architectures for autonomous racing," *Machine Learning with Applications*, vol. 14, p. 100496, 2023.

[5] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9403–9409, IEEE, 2021.

[6] W. Schwarting, T. Seyde, I. Gilitschenski, L. Liebenwein, R. Sander, S. Karaman, and D. Rus, "Deep latent competition: Learning to race using visual control policies in latent space," 11 2020.

[7] T. Weiss, J. Chrosniak, and M. Behl, "Towards multi-agent autonomous racing with the deepracing framework.," *International Conference on Robotics and Automation (ICRA) - Workshop on Opportunities and Challenges with Autonomous Racing.*

[8] T. Weiss and M. Behl, "Deepracing: A framework for autonomous racing," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1163–1168, 2020.

[9] T. Brüdigam, A. Capone, S. Hirche, D. Wollherr, and M. Leibold, "Gaussian processbased stochastic model predictive control for overtaking in autonomous racing," in *2021 IEEE International Conference on Robotics*

*and Automation (ICRA) Workshop on Opportunities and Challenges with Autonomous Racing*, 2021.

[10] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[11] N. R. Kapania, J. Subosits, and J. Christian Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091005, 2016.

[12] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[13] A. Liniger and J. Lygeros, "A viability approach for fast recursive feasible finite horizon path planning of autonomous rc cars," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, (New York, NY, USA), p. 1–10, Association for Computing Machinery, 2015.

[14] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, E. H. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," *2012 American Control Conference (ACC)*, pp. 4239–4244, 2012.

[15] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3149–3154, 2019.

[16] Z. Zang, R. Tumu, J. Betz, H. Zheng, and R. Mangharam, "Winning the 3rd japan automotive ai challenge - autonomous racing with the autoware.auto open source software stack," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1757–1764, 2022.

[17] J. Bhargav, J. Betz, H. Zehng, and R. Mangharam, "Deriving spatial policies for overtaking maneuvers with autonomous vehicles," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pp. 859–864, 2022.

[18] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2020.

[19] Federation Internationale de l'Automobile, "Organisation." `https://www.fia.com/organisation`, 2024. Accessed: 2024-05-09.

[20] Federation Internationale de l'Automobile, "Appendix L to the international sporting code." https://www.fia.com/sites/default/files/wmsc_appendix_l_2024_publie_le_28_fevrier_2024.pdf, 2024. Accessed: 2024-05-08.

[21] The Sim Racing Collective, "Rules & Regulations." https://www.tsrc.club/rules. Accessed: 2024-05-08.

[22] Sports Car Club of America, "General Competition Rules: SCCA Road Racing," 2024.

[23] Federation Internationale de l'Automobile, "Formula One Sporting Regulations Issue 6." https://www.fia.com/sites/default/files/fia_2024_formula_1_sporting_regulations_-_issue_6_-_2024-04-30_v2.pdf, April 2024. Accessed: 2024-05-08.

[24] R. Pobst, T. Earwood, "Appendix P. Racing Room & Passing Guidlines." https://cdn.connectsites.net/user_files/scca/downloads/000/050/288/Appendix_P_Updated_Jan22.pdf?1650377995, January 2022. Accessed: 2024-05-08.

[25] Federation Internationale de l'Automobile, "F1 Driving Standard Guidelines." https://www.fia.com/sites/default/files/doc_2_-_2022_imola_event_-_fia_f1_driving_standard_guidelines.pdf, April 2022. Accessed: 2024-05-08.

[26] F1TENTH, "F1tenth - build documentation." https://f1tenth.org/build.html, 2023. Accessed: 2024-30-04.

[27] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[28] J. Klapálek, A. Novák, M. Sojka, and Z. Hanzálek, "Car racing line optimization with genetic algorithm using approximate homeomorphism," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 601–607, 2021.

[29] C. H. Walsh and S. Karaman, "Cddt: Fast approximate 2d ray casting for accelerated localization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3677–3684, IEEE, 2018.

[30] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform." https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[31] Stephen Boyd and Lieven Vandenberghe, "Convex Optimization," *Cambridge University Press*, 2004.

[32] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *NeurIPS 2019 Competition and Demonstration Track*, pp. 77–89, PMLR, 2020.

[33] F1TENTH, "F1TENTH gym ROS2." `https://github.com/f1tenth/f1tenth_gym_ros`, 2024. Accessed: 2024-20-05.

# MASTER‘S THESIS ASSIGNMENT

## I. Personal and study details

| | | | |
|---|---|---|---|
| Student's name: | **Nagy Tomáš** | Personal ID number: | **483574** |
| Faculty / Institute: | **Faculty of Electrical Engineering** | | |
| Department / Institute: | **Department of Control Engineering** | | |
| Study program: | **Cybernetics and Robotics** | | |

## II. Master’s thesis details

Master’s thesis title in English:

**Head-to-Head Racing with F1/10 Autonomous Car**

Master’s thesis title in Czech:

**Závodění s oponentem pro autonomní auto F1/10**

Guidelines:

1. Get familiar with ROS2 and F1/10 project.
2. Conduct a review of algorithms used for planning of overtaking maneuvers in the topic of autonomous racing.
3. Conduct a review of overtaking-related rules in racing competitions. Use them to define the rules for overtaking that are applicable in an F1/10 competition.
4. Design an algorithm for planning overtaking maneuvers. Consider only one opponent on the racing track. In early stages assume that the position and the trajectory of the opponent is known in advance.
5. Define a set of testing scenarios considering opponents with various control algorithms (e.g., tracking optimal trajectory, reactive control, etc.) and parameters (e.g., same maximum velocity).
6. Evaluate your algorithm on the testing scenarios and demonstrate at least one of the scenarios on the F1/10 car.
7. Evaluate your results and document everything thoroughly.

Bibliography / sources:

[1] S. Macenski et al., "Robot Operating System 2: Design, architecture, and uses in the wild," Science Robotics vol. 7, May 2022, doi: 10.1126/scirobotics.abm6074
[2] J. Betz et al., "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing," in IEEE Open Journal of Intelligent Transportation Systems, vol. 3, pp. 458-488, 2022, doi: 10.1109/OJITS.2022.3181510
[3] J. Bhargav et al., "Deriving Spatial Policies for Overtaking Maneuvers with Autonomous Vehicles," 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 2022, pp. 859-864, doi: 10.1109/COMSNETS53615.2022.9668548

Name and workplace of master’s thesis supervisor:

**Ing. Jaroslav Klapálek Department of Control Engineering FEE**

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

Date of master’s thesis assignment: **16.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

| | | |
|---|---|---|
| Ing. Jaroslav Klapálek | prof. Ing. Michael Šebek, DrSc. | prof. Mgr. Petr Páta, Ph.D. |
| Supervisor’s signature | Head of department’s signature | Dean’s signature |

## III. Assignment receipt

The student acknowledges that the master'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 master's thesis, the author must state the names of consultants and include a list of references.

| | |
|---|---|
| _______ . ______________ | ______________________________ |
| Date of assignment receipt | Student's signature |