CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS

David Zahrádka

Optimization-based control of the F1/10 autonomous racing car

Department of Control Engineering

Thesis supervisor: **Ing. Jaroslav Klapálek**

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne............................ ....

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zahrádka David**	Personal ID number:	**434763**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		
Branch of study:	**Cybernetics and Robotics**		

II. Master’s thesis details

Master’s thesis title in English:

Optimization-based control of the F1/10 autonomous racing car

Master’s thesis title in Czech:

Optimální řízení autonomního závodního auta F1/10

Guidelines:

1. Make yourself familiar with ROS and F1/10 autonomous racing cars.
2. Review literature related to kinematic and dynamic vehicle models, and trajectory planning based on these models.
3. Create and identify kinematic and dynamic single-track model of the F1/10 car.
4. Use the identified model for optimal control of the F1/10 car (e.g. using MPC) on a limited time horizon, using a map of the environment.
5. Test the created algorithm (ideally on the F1/10 car), identify limitations and computational demands of the implemented solution.
6. Document everything thoroughly.

Bibliography / sources:

ROS tutorials, http://wiki.ros.org/ROS/Tutorials
F1/10 rules version 2.0, http://f1tenth.org/race/rules-v2.pdf
Liniger, A., Domahidi, A., and Morari, M. (2015) Optimization□based autonomous racing of 1:43 scale RC cars. Optim. Control Appl. Meth., 36: 628– 647. doi: 10.1002/oca.2123

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:

Ing. Michal Sojka, Ph.D., Embedded Systems, CIIRC

Date of master’s thesis assignment: **05.02.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until: **30.09.2021**

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

Acknowledgements

I would like to thank my advisor, Ing. Jaroslav Klapálek, for his invaluable help and advice, my friends and family for their never-ending support, my teapot for its reliable service during difficult times, and Hedwig Norris for her extremely patient assistance during long debugging sessions and her reminding me that occasional breaks are essential to keep the brain healthy.

Abstract

The main interest of this thesis is the application of optimization-based control techniques on a 1/10 scale autonomous racing vehicle for the *F1/10 Autonomous Racing Competition.* The goal is to implement a control algorithm able to achieve fast lap times in time-trial races. Selected vehicle and tire models are identified and implemented using a genetic algorithm. A *Model Predictive Contouring Control* algorithm utilizing the identified models is integrated into the racing vehicle platform, and its performance is tested in simulation and real-life experiments. The control algorithm successfully optimized the vehicle's trajectory in the simulation and was able to drive the vehicle safely through the testing track in the real-life experiment.

Keywords: F1/10 competition, Autonomous racing, Model identification, Pacejka magic formula, Model Predictive Contouring Control, Trajectory optimization

Abstrakt

Tato práce se zabývá aplikací optimalizačních řídících technik na autonomním závodním vozidle v měřítku 1/10 pro soutěž *F1/10 Autonomous Racing Competition.* Cílem práce je implementace řídícího algoritmu schopného dosáhnout rychlých časů na kolo na závodní dráze. V práci jsou popsány vhodné matematické modely vozu a pneumatik, a vybraný model vozidla a pneumatik je identifikován a implementován. Do závodního vozidla je integrován řídící algoritmus *Model Predictive Contouring Control*, který využívá identifikované modely, a je otestován v simulaci a v reálných experimentech s autonomním závodním vozidlem. Implementovaný řídící algoritmus byl schopen optimalizovat trajektorii závodního vozida v simulaci, v reálných experimentech dokázal s vozidlem projet testovací dráhu.

Klíčová slova: F1/10 competition, Závody autonomních vozů, Identifikace parametrů modelu, Pacejka magic formula, Model Predictive Contouring Control, Optimalizace trajektorie

Contents

List of Figures

1 Introduction

The *F1/10 Autonomous Racing Competition* is a competition for standardized 1/10 scale autonomous racing cars, in which the vehicles compete against each other in time-trial and head-to-head races. In order to outperform other race competitors, the autonomous vehicle needs to perform its best at maximum possible velocities, even on challenging racing tracks. Furthermore, both the small 1/10 scale of autonomous cars and the competition rules significantly restrict the capabilities of the computational hardware, making efficiency a crucial requirement. Because of this, effective control algorithms are required to minimize the time in which an autonomous racing car is able to safely drive through the racing track. The problem of how to minimize the time in which the autonomous vehicle can finish a lap is generally referred to as the Minimum Lap Time (MLT) problem and has been widely studied throughout the history of automotive racing of all scales, such as the *Formula One* race [39], or the *Superbike World Championship* [42].

The field of small-scale autonomous racing is beneficial to the expanding automotive industry by serving as a low-cost test bench for the control algorithms, thus reducing the financial costs associated with autonomous car development. The small vehicle scale also limits the number and quality of onboard sensors, requiring robust control solutions due to noise and low information quantities.

In this thesis, at first, the mathematical models of the vehicle, the tires and the drivetrain are described and an appropriate combination is selected. Then, an optimization-based control algorithm that utilizes the track knowledge for the online optimization of the autonomous vehicle's trajectory is presented. The selected models are identified and verified, and together with the selected control algorithm are integrated into the vehicle platform, shown in Fig. 1. The performance of the integrated controller is then analyzed in a simulation and in real-life experiments.

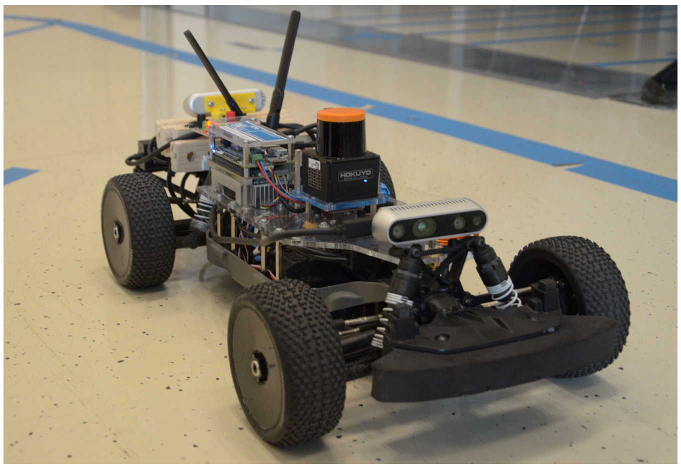


Figure 1: The F1/10 autonomous racing vehicle.

2 Related works

In Section 2.1, we explore the works related to the trajectory optimization approach for the MLT. Then, in Section 2.2, we describe the *Model Predictive Control* approach directly, and its specific *Contouring problem formulation* is discussed in Section 2.3.

2.1 Trajectory optimization

The interest in the minimum lap time problem dates as far back as 1958, where it was used in order to optimize the gear ratios and car body type for Formula 1 racing tracks, as described in [39]. Further analysis of the minimum lap-time problem appeared in 1970 and 1978, but the first to compute the optimal control sequence were *Metz et al.* in [32] by utilizing a quasi-steady state (QSS) optimization routine.

QSS optimization is used even in modern approaches, as in [10], where it was used to analyze the roll stiffness distribution of a race car, and [38], where it was used to optimize the gear-set. This method features low computational requirements, but provides less accurate results, as it doesn't concern the transient dynamics of the car between separate states.

In [12], *Cossalter et al.* used a two-point boundary value problem approach to solve the minimum lap-time problem for motorcycles on the Mugello raceway in order to evaluate vehicle handling and maneuverability by defining them using the cost function. They used a curvilinear coordinate system for the track model, which simplified the definition of the trajectory constraints representing the inner and outer track boundaries. This representation of the track is used commonly even in recent solutions.

Another used approach, while computationally difficult, is to use nonlinear optimization. In [37], a direct collocation scheme and a nonlinear programming algorithm were used to solve the MLT optimal control problem for a Formula One race car, while reducing the computational time by utilizing, apart from other means, a curvilinear coordinate system as introduced in [12]. Although utilized mainly for offline planning, in [51] and in [17] a direct numerical method was used for online trajectory optimization. However, the usage of nonlinear solvers resulted in limited reaction times.

While not optimization based, reactive algorithms are widely used for racing purposes because of their low sensor requirements and are mentioned here due to their prevalence. One of those is the Follow the Gap algorithm, presented in [40]. In each iteration, the algorithm extracts the visible gaps in front of the vehicle and steers the car towards the largest one. Since this algorithm doesn't consider the vehicle dynamics, it is not able to optimize the trajectory, thus providing sub-optimal results. It is however able to evade dynamic obstacles, such as debris and even other cars, and doesn't require prior knowledge of the racing track, and, therefore, it is computationally inexpensive.

2.2 Model Predictive Control

A popular solution for trajectory optimization is utilizing the Model Predictive Control (MPC) techniques, despite their usually short planning horizon. By combining trajectory optimization with reference tracking, the MPC can be used as a one-level control system for the optimization of both velocity and steering.

The MPC was successfully used for trajectory optimization on miniature racing cars in [49], where a nonlinear MPC tracking controller was modified to optimize the miniature car's trajectory by imposing an infeasible time reference. This manifested in the desired behavior of faster lap times achieved by deviating from the velocity reference and by cutting the track corners.

The MPC was also utilized for the longitudinal control of full-scale vehicles in [30], where it was applied in a scenario with a convoy of vehicles responding to a sudden cut-in of another vehicle. The MPC was also used for the lateral control of a full-scale vehicle in an obstacle evasion scenario using only steering in [21] and in [22], with the velocity reference tracking done by a separate controller. However, by combining both breaking and steering in an obstacle evasion scenario, the required distance for the evasive maneuver to be successful is reduced, as stated, for example, in [41] and [23].

An approach for online replanning with longer planning horizon was presented in [46], in which they approximated the optimal control problem as a convex quadratically constrained quadratic program rather than a nonlinear one. They utilized this for trajectory replanning in emergency obstacle avoidance scenarios and even demonstrated its performance in a racing scenario.

2.3 Model Predictive Contouring Control

The Model Predictive Contouring Control problem (MPCC) was introduced in [27]. It is an extension of the MPC framework suited for real time reference tracking and optimization in high speed multi-axis contouring systems, such as X-Y tables. Unlike other path following approaches, as seen in [5], the MPC-based solutions, such as the nonlinear MPC used in [18], are able to consider actuator constraints. However, due to the nonlinear nature, finding real-time solutions to the optimization problem is difficult. By utilizing a linear time-varying formulation in the MPCC, the computational complexity is reduced, which is beneficial for real-time applications.

The MPCC was utilized in [29] as a one-level online solution to the minimum lap-time problem. It was implemented on a 1:43 scale RC race car, formulated in a way that maximizes the progress along the tracked reference trajectory, represented by the track's center line, while respecting the vehicle model and track constraints. They also implemented an obstacle avoidance and overtaking algorithm using a high level path planner based on dynamic programming. The MPCC was later also implemented on a full-scale Formula Student car, as seen in [25].

3 Mathematical models

For a successful and reliable trajectory optimization and subsequent tracking, it is necessary to mathematically describe the behavior of the vehicle in such a way that it is possible to simulate and predict in the intended application scenarios. This mathematical description, together with its parameters and constraints, forms the vehicle model. The first necessary component of the model is the mathematical model of the behavior of the vehicle itself. The relevant models are introduced in Section 3.2.

Furthermore, since the racing car often operates at its dynamical limits, the tire forces and slip angles must be considered. This is mainly due to the lateral slip, which is commonly present while steering at higher velocities. In order to accurately model the slip dynamics, a suitable tire model is required. The available tire models include the classic *Dugoff model* [15] and its variations [26] and the widely used *Pacejka magic formula* [35] and its variations [7]. Description of those models can be found in Section 3.3.

3.1 Coordinate system

The coordinate system used in this thesis can be seen in Fig. 2. The vehicle and the racing track itself are described using planar Cartesian coordinates. The position of the vehicle in the track then consists of two numerical values $[X, Y]$, each denoting the distance to the coordinate system origin in the respective axes x, y.

The vehicle's state variables, e.g., the longitudinal forward velocity v_x or the lateral velocity v_y, are expressed with respect to the vehicle's own planar coordinate system, if not explicitly mentioned otherwise. The vehicle's coordinate system can be seen in Fig. 2, where the x_v, y_v are the axes of the coordinate system with the origin in $[X, Y]$. The coordinate system is rotated about the global coordinate system's origin by the vehicle's yaw (heading) angle φ.

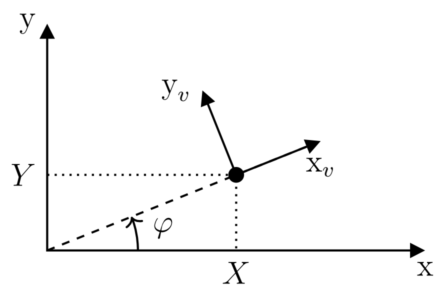


Figure 2: The utilized coordinate system.

3.2 Vehicle model

The simplest vehicle model is the point mass model, which is described in Section 3.2.1. In Section 3.2.2, the non-holonomic kinematic single-track model is described, and Section 3.2.3 describes the dynamic single-track model. The dynamic dual-track model is described in 3.2.4, and a brief description of the multi-body vehicle model is given in 3.2.5.

A list of parameters with their description is provided in Appendix C.

3.2.1 Point mass model

The point-mass model, as described in [6], represents the vehicle as a point mass with acceleration bounds. The acceleration can be applied in any direction, resulting in a simple description of the vehicle kinematics:

$$\begin{aligned} \dot{X} &= v_x, \\ \dot{Y} &= v_y, \\ \dot{v_x} &= a_x, \\ \dot{v_y} &= a_y, \end{aligned} \tag{1}$$

where $[X, Y]$ represent the global position, v_x, v_y represent the velocity in global coordinates and a_x, a_y are the input accelerations in the respective axes. The acceleration bounds can be then represented as:

$$\sqrt{a_x^2 + a_y^2} \leq a_{max} \tag{2}$$

where a_{max} is the maximal acceleration of the point mass. Due to its simplicity, the model can be useful for applications where it is not necessary to model the non-holonomic behavior of the vehicle.

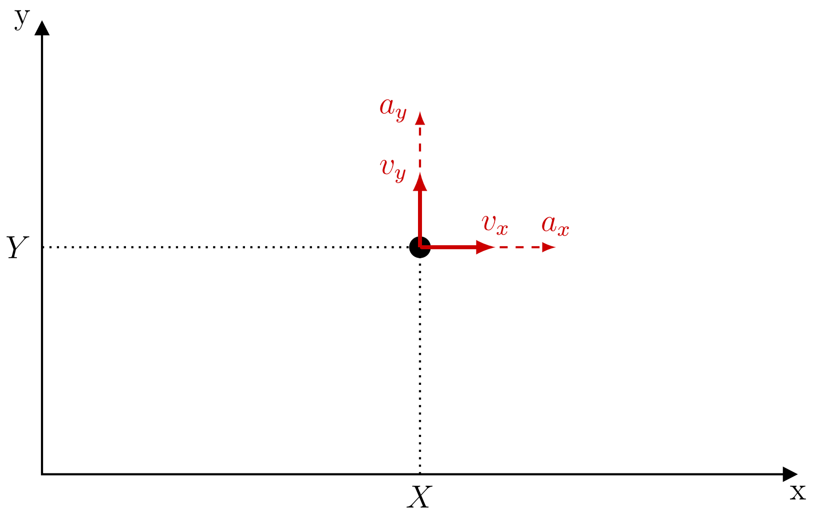


Figure 3: The point-mass model with its respective variables. X, Y [m] denote the vehicle's position, v_x, v_y [m s^{-1}] are the velocities in the global (track) coordinates, and a_x, a_y [m s^{-2}] are the input accelerations in the global (track) coordinates.

3.2.2 Kinematic single-track model

The kinematic single-track model, as presented in [6], describes the vehicle as two wheels connected by a central axle. In case of a road vehicle, this means that the front and rear wheels are grouped together. This simplification is possible for applications where the roll dynamics of the vehicle are not necessary to consider, e.g., modeling a vehicle in situations with a zero lateral velocity.

Unlike the point-mass model, the kinematic single-track model describes the non-holonomic behavior of the standard road vehicle's minimum turning radius. This is the reason that the kinematic single-track model is widely used for vehicle motion planning, since the holonomic behavior of the point-mass model is not accurate enough for some applications, e.g., motion planning for vehicle parking. However, as it is purely kinematic, it does not consider the longitudinal and lateral forces of the vehicle, and it requires the assumption of zero longitudinal and lateral slip. This makes it inaccurate in applications with non-zero lateral slip, such as steering in high velocities.

The corresponding differential equations describing the model are:

$$
\begin{aligned}
\dot{X} &= v\cos(\varphi),\\
\dot{Y} &= v\sin(\varphi),\\
\dot{\varphi} &= \frac{v}{l_{wb}}\tan\delta,\\
\dot{v} &= a_{long},
\end{aligned}
\tag{3}
$$

where the newly introduced variables a_{long}, δ stand for longitudinal acceleration and the steering angle, respectively, and are considered as inputs, v is the vehicle's longitudinal velocity, φ is the vehicle's yaw, and the l_{wb} is the length of the vehicle's wheelbase. A graphical representation of this model with a complete description of its variables can be seen in Figure 4.

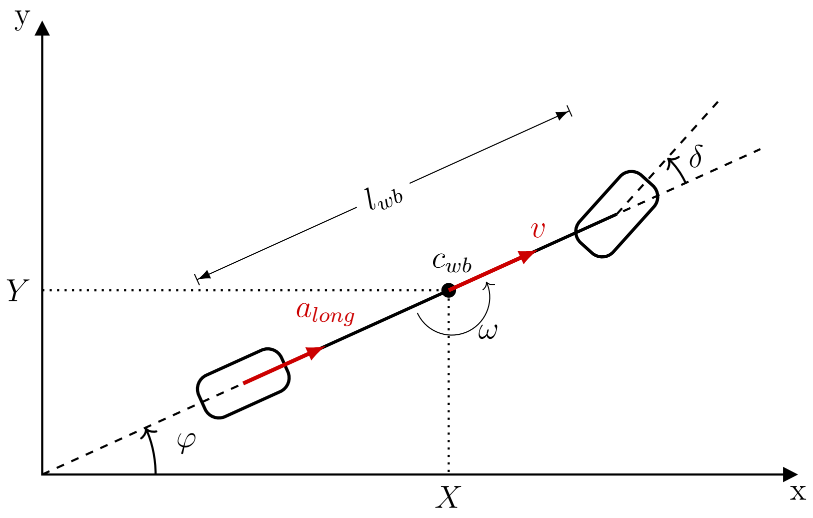


Figure 4: The kinematic single-track vehicle model with its respective variables. X, Y [m] denote the vehicle's position in global coordinates, a_{long} [m s^{-2}] is the input acceleration, φ [rad] is the vehicle's yaw (heading), ω [rad s^{-1}] is the yaw rate, δ [rad] is the input steering angle, v [m s^{-1}] is the longitudinal velocity, l_{wb} [m] is the length of the wheelbase and c_{wb} [m] is the center of the wheelbase.

3.2.3 Dynamic single-track model

The dynamic single-track model incorporates the longitudinal and lateral forces of the vehicle, as opposed to the kinematic model. By adding the dynamics of the vehicle, the model works better for higher velocities where the direction of the velocity vector does not necessarily correspond to the longitudinal vehicle axis due to a non-zero side-slip angle. Possible descriptions of such a model are introduced in [6] or in [29], where the dynamics of a rear wheel driven vehicle are described, using the longitudinal force applied on the rear axle $F_{r,x}$ and the steering angle δ as input. The differential equations describing this vehicle model are:

$$\begin{aligned}
\dot{X} &= v_x \cos(\varphi) - v_y \sin(\varphi), \\
\dot{Y} &= v_x \sin(\varphi) + v_y \cos(\varphi), \\
\dot{\varphi} &= \omega, \\
\dot{v_x} &= \frac{1}{m}(F_{r,x} - F_{f,y} \sin(\delta) + m v_y \omega), \\
\dot{v_y} &= \frac{1}{m}(F_{r,y} + F_{f,y} \cos(\delta) - m v_x \omega), \\
\dot{\omega} &= \frac{1}{I_z}(F_{f,y} l_f \cos(\delta) - F_{r,y} l_r),
\end{aligned} \tag{4}$$

where the newly introduced variables v_x, v_y are the longitudinal and lateral velocity, respectively, ω is the vehicle's yaw rate, m is the vehicle's mass, I_z is the moment of inertia corresponding to the vertical axis z and l_f, l_r are the distances of the front and rear axle, respectively, from the center of gravity. The lateral forces $F_{f,y}$, $F_{r,y}$ must be provided by, for example, a tire model. A graphical representation of this vehicle model with a complete description of its variables can be seen in Fig. 5.

3.2.4 Dynamic dual-track model

In contrast to the previous models, the dynamic dual-track model represents all four wheels instead of merging each axle into a single wheel. This represents the vehicle's geometry more accurately and allows it to consider that the normal force acting on each wheel of the front or rear axle is different. This helps us to model the dynamic forces acting on the car more accurately when dealing with laterally asymmetrical vehicles. It can also be extended to include roll and pitch dynamics, similarly to the single-track model.

The model is described by the differential equations:

$$\begin{aligned}
\dot{X} &= v_x \cos(\varphi) - v_y \sin(\varphi), \\
\dot{Y} &= v_x \sin(\varphi) + v_y \cos(\varphi), \\
\dot{v}_x &= a_x + v_y \omega, \\
\dot{v}_y &= a_y - v_x \omega, \\
\dot{\varphi} &= \omega, \\
\dot{\omega} &= \frac{M_z}{I_z},
\end{aligned} \tag{5}$$

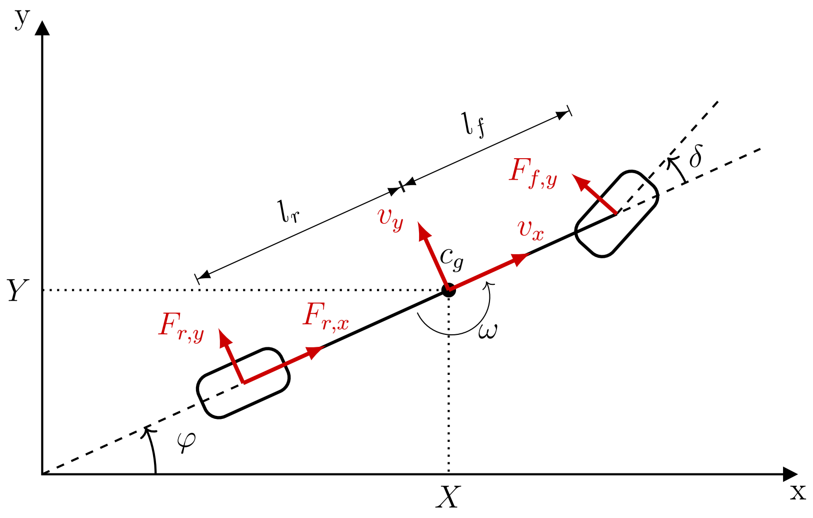


Figure 5: The dynamic single-track vehicle model with its respective variables. X, Y [m] denote the vehicle's position in the global coordinates, $F_{r,x}$ [N] is the rear longitudinal force acting on the vehicle, $F_{r,y}$, $F_{f,y}$ [N] are the lateral rear and front forces acting on the vehicle, φ [rad] is the vehicle's yaw (heading), ω [rad s^{-1}] is the yaw rate, δ [rad] is the input steering angle, v_x [m s^{-1}] is the longitudinal velocity of the vehicle, v_y [m s^{-1}] is the lateral velocity of the vehicle, c_g [m] is the center of gravity and l_f, l_r [m] are the distances between the center of gravity and the front and rear wheel, respectively.

where a_x, a_y, M_z are given as:

$$\begin{aligned}
a_x &= \frac{(F_{f,l,x} + F_{f,r,x})\cos(\delta) - (F_{f,l,y} + F_{f,r,y})\sin(\delta) + F_{r,l,x} + F_{r,r,x}}{m},\\
a_y &= \frac{(F_{f,l,x} + F_{f,r,x})\sin(\delta) + (F_{f,l,y} + F_{f,r,y})\cos(\delta) + F_{r,l,y} + F_{r,r,y}}{m},\\
M_z &= l_f((F_{f,l,x} + F_{f,r,x})\sin(\delta) + (F_{f,l,y} + F_{f,r,y})\cos(\delta)) - l_r(F_{r,l,y} + F_{r,r,y})\\
&\quad - \frac{t_f}{2}((F_{f,l,x} - F_{f,r,x})\cos(\delta) - (F_{f,l,y} - F_{f,r,y})\sin(\delta)) - \frac{t_r}{2}(F_{r,l,x} - F_{r,r,x}),
\end{aligned} \tag{6}$$

where the newly introduced variable $F_{a,b,c}$ are the forces acting on the vehicle, $a \in \{f, r\}$ signify the front and rear wheel, $b \in \{l, r\}$ signify the left and right wheel and $c \in \{x, y\}$ specify whether the force is longitudinal or lateral, respectively, and t_f, t_r are the lengths of the front and rear axle tracks. A graphical representation of the dynamic dual-track model with a complete description of its variables can be seen in Fig. 6.

The dynamic dual-track model is not widely used, since the simplified single-track model is often deemed sufficient. One example use case can be seen in [11], where the dynamic dual-track model was used to estimate the vehicle's states for use in active safety systems.

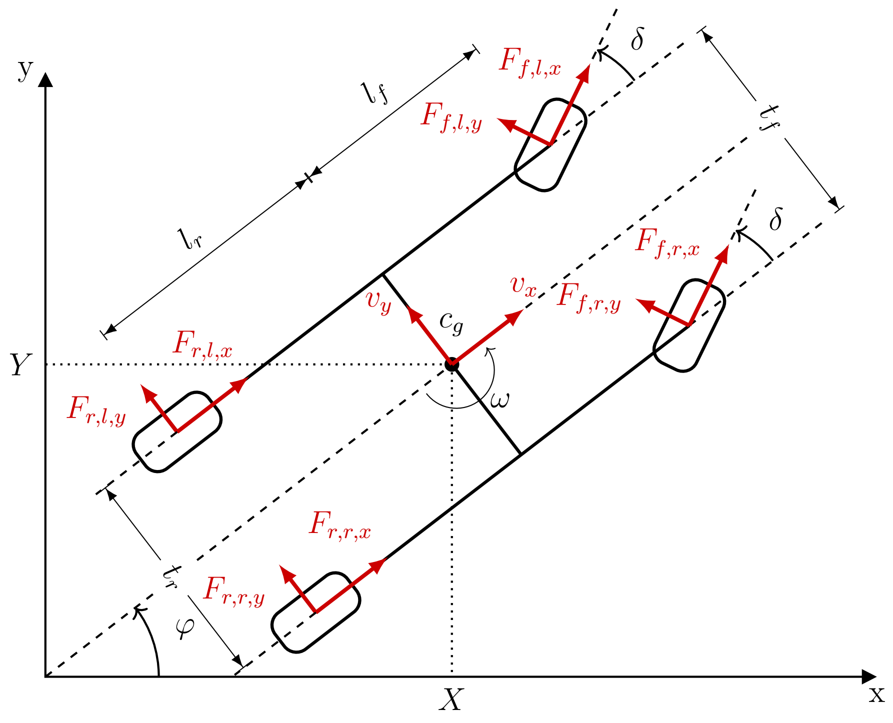


Figure 6: The dynamic dual-track vehicle model with its respective variables. X, Y [m] denote the vehicle's position in global coordinates, $F_{a,b,c}$ [N] are the forces acting on the vehicle, where $a \in \{f, r\}$ signify the front and rear wheel, $b \in \{l, r\}$ signify the left and right wheel and $c \in \{x, y\}$ specify whether the force is longitudinal or lateral, respectively. φ [rad] is the vehicle's yaw (heading), ω [rad s^{-1}] is the yaw rate, δ [rad] is the input steering angle, v_x [m s^{-1}] is the longitudinal velocity of the vehicle, v_y [m s^{-1}] is the lateral velocity of the vehicle, c_g [m] is the center of gravity, l_f, l_r [m] are the distances between the center of gravity and the front and rear wheel, respectively, and t_f, t_r [m] are the lengths of the front and rear axle tracks.

3.2.5 Multi-body dynamic model

Perhaps the most advanced commonly utilized vehicle model is the multi-body dynamic model, described in [6]. It considers the vehicle's pitch and roll dynamics and their effects on the vertical load of all 4 wheels, their individual spin, slip and nonlinear tire dynamics. In order to model the pitch and the roll dynamics, the multi-body model contains the description of the vehicle's suspension dynamics. The suspension dynamics, together with the tire dynamics, are then used to describe the forces between the sprung masses of the front and rear axle and the unsprung vehicle's mass. Since this vehicle model contains up to 29 state variables, it is not described here (see [6]).

3.3 Tire model

Since the wheeled land vehicle's behavior depends on the interaction between its wheels and the road, it is necessary to appropriately model the behavior of tires. The tire models describe the interaction between the vehicle's tire and the road. Therefore, their parameters depend both on the parameters of the tire and on the road the vehicle is driving on. Accurately describing this behavior necessitates complex mathematical formulations with many input parameters. This is the reason why many models only approximate the behavior while remaining fairly simple to identify, striving for the best accuracy possible.

3.3.1 Fiala tire model

The Fiala tire model was introduced in [19]. A variant of this model was utilized in [50] for the control of high side-slip maneuvers, and is described as:

$$F_y(\alpha) = \begin{cases} -C_\alpha \tan(\alpha) + \frac{C_\alpha^2 (2 - \frac{\mu_s}{\mu_p})}{3\mu_p F_z} |\tan(\alpha)| \tan(\alpha) - \frac{C_\alpha^3 (1 - 2\frac{\mu_s}{mu_p})}{9\mu_p^2 F_z^2} \tan^3(\alpha) & \text{if } |\alpha| < \alpha_{sl}, \\ -\mu_s F_z sgn(\alpha) & \text{if } |\alpha| \geq \alpha_{sl}, \end{cases} \tag{7}$$

where F_y is the lateral force the tire generates, C_α is the tire cornering stiffness, F_z is the normal load of the tire, μ_p is the peak friction coefficient between the tire and the road, μ_s is the sliding friction coefficient of the tire and α is the tire's slip angle. The α_{sl} is computed as:

$$\alpha_{sl} = \arctan(\frac{3\mu_p F_z}{C_\alpha}). \tag{8}$$

3.3.2 Dugoff physical tire model

The Dugoff physical tire model [15] is an analytical model based on the Fiala tire model, presented in [19] It characterizes the tire with the assumptions of constant vertical load distribution and using a linear dependence of the friction coefficient on the sliding velocity of the tire. This simplifies the task of characterizing the tire at the cost of reduced accuracy, as the tire forces are often non-linear [14]. The physical model of Dugoff was described in [26] as:

$$\begin{aligned} F_x &= C_s \frac{\kappa}{1-\kappa}\tau, \\ F_y &= C_\alpha \frac{\tan(\alpha)}{1-\kappa}\tau, \\ \tau &= \begin{cases} 2-\sigma & \text{if } \sigma < 1, \\ 1 & \text{if } \sigma \geq 1, \end{cases} \end{aligned} \tag{9}$$

where C_s is the longitudinal tire stiffness, C_α is the cornering tire stiffness, κ is the longitudinal slip of the tire, α is the lateral slip of the tire, and σ is defined by:

$$\sigma = \frac{(1-\kappa)\mu F_z}{2\sqrt{C_s^2\kappa^2 + C_\alpha^2 \tan^2(\alpha)}}, \tag{10}$$

where μ is the tire's friction coefficient.

3.3.3 Pacejka magic formula

There are many versions of the (so-called) Pacejka magic formula, as seen in [35] or [36]. It is a strictly empirical formula describing the combined slip and horizontal force generation of the tire. A brief description of the Pacejka magic formula, as presented in [36], is given here, and an example of a tire characteristic curve can be seen in Fig. 7.

The lateral or longitudinal force F produced by the tire is given by the formula:

$$F = D \cdot \sin(C \cdot \arctan(B \cdot x - E \cdot (B \cdot x - \arctan(B \cdot x)))), \tag{11}$$

with

$$\begin{aligned} Y(X) &= y(x) + S_v, \\ x &= X + S_h, \end{aligned} \tag{12}$$

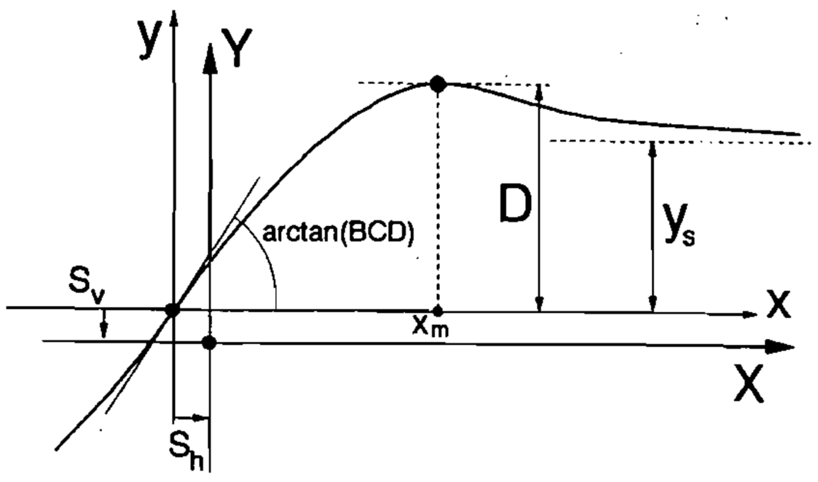


Figure 7: Typical Pacejka magic formula curve, representing the lateral or longitudinal force F generated by the tire, with some denoted parameters. The steady-state value y_s can be used to estimate the C and D Pacejka parameters. The parameters S_v and S_h correspond to the vertical and horizontal shift, respectively [35].

where F corresponds to either the lateral force F_y, or the longitudinal force F_x. The x then corresponds to the input variable, which is the lateral slip α for F_y, or the longitudinal slip κ for F_x. The parameters B, C, D, E are called Pacejka parameters. The parameter B is known as the stiffness factor, C as the shape factor, D as the peak value and E as the curvature factor. The S_h and S_v are two shift parameters, corresponding to the horizontal and vertical shift, respectively, introduced in [35].

The magic formula characterizes the force F generated by the tire by a curve that passes through the origin $x = y = 0$, reaches a maximum, and subsequently tends to a stable value. The curve (Fig. 7) shows an anti-symmetric shape with respect to the origin. Because of the addition of the shift parameters, the curve can have a horizontal or vertical offset.

3.3.4 Simplified Pacejka

The Simplified Pacejka magic formula was introduced in [7]. It is described by the equation:

$$Y = D \cdot \sin(C \cdot \arctan(B \cdot X)), \tag{13}$$

where the parameter D represents the peak function value, C describes the shape of the curve, B is the stiffness factor, Y is either the lateral force or brake force generated by the tire and X is either the lateral slip angle α, or the longitudinal slip angle κ. Compared to the original Pacejka magic formula, the simplified formula utilizes only three parameters, but additional formulations utilizing the E curvature factor are presented in [7].

In [29], the simplified Pacejka magic formula was utilized to model the tire dynamics of a 1/43 scale autonomous vehicle as follows:

$$
\begin{aligned}
F_{f,y} &= D_f \cdot \sin(C_f \cdot \arctan(B_f \cdot \alpha_f)), &\text{(14a)}\\
F_{r,y} &= D_r \cdot \sin(C_r \cdot \arctan(B_r \cdot \alpha_r)), &\text{(14b)}\\
\alpha_f &= -\arctan(\frac{\omega l_f + v_y}{v_x}) + \delta, &\text{(14c)}\\
\alpha_r &= \arctan(\frac{\omega l_r - v_y}{v_x}), &\text{(14d)}
\end{aligned}
$$

where B_f, C_f, D_f are the Pacejka parameters for the front tire of the single-track model, and B_r, C_r, D_r represent the rear tire. The simplified Pacejka magic formula was not used in [29] to calculate the longitudinal force F_x, as it was modeled as a part of the drivetrain model.

It is important to note that due to the dependence of the lateral tire forces $F_{f,y}$, $F_{r,y}$ on the vehicle slip angle α, this tire model can be applied only for non-zero longitudinal velocity.

3.4 Drivetrain model

In order to compute the tire dynamics, it is necessary to know the forward force F_x acting on the wheels that the drivetrain produces. In [29], a simple drivetrain model for a rear-wheel driven 1/43 scale racing car is utilized to calculate the force exhibited on the rear axle $F_{r,x}$ as:

$$F_{r,x} = (C_{m1} - C_{m2} v_x) d - C_r - C_d v_x^2, \tag{15}$$

where d is the PWM applied to the DC motor, the Cm_1, Cm_2 are empirical parameters representing the characteristics of the motor and the C_r, C_d are coefficients representing the rolling and drag resistances of the car, respectively.

Since the drag resistance coefficient C_d is difficult to measure without specialized equipment, such as a wind tunnel, the drivetrain model is generalized as:

$$F_x = (C_{m1} - C_{m2} v_x) d - C_{m3} - C_{m4} v_x^2, \tag{16}$$

where F_x is the longitudinal force exhibited on the rear or front axle and the $C_{m1\text{–}4}$ are the empirical parameters used to shape the model's response curve to fit the drivetrain characteristics. This generalization increases the flexibility of the model and allows us to introduce possible further parameters to shape the response.

As another approach, we can utilize the fact that the vehicle features a speed controller that accepts the engine revolutions per minute (ERPM) reference $\tilde{d} \in [0, 1]$ as an input, where $\tilde{d} = 1$ corresponds to maximal ERPM and $\tilde{d} = 0$ to minimal. Because of this, the drivetrain can be modeled as a first order system with the equation:

$$F_x = \frac{(C_m\tilde{d} - v_x)}{\tau_s}, \tag{17}$$

where $\tilde{d}$ is the ERPM reference input, τ_s is the time constant of the first order system response, and C_m is a general motor parameter. The first order drivetrain model can then be generalized as:

$$F_x = \frac{(C_{m1}\tilde{d})}{C_{m2}} - \frac{v_x}{C_{m3}}. \tag{18}$$

where $C_{m1\text{–}3}$ are empirical parameters used to shape the model's response curve.

Note that for $C_{m3} = C_{m2}$, the generalized first order drivetrain model, described by Eq. 18, becomes the original first order model, as described by Eq. 17. However, this generalization allows greater freedom in shaping the drivetrain model response curve.

The model described by Eq. 16 can also be used with the ERPM reference $\tilde{d}$ as an input, leading to the equation:

$$F_x = (C_{m1} - C_{m2}v_x)\tilde{d} - C_{m3} - C_{m4}v_x^2. \tag{19}$$

3.5 Tire friction ellipse

The tire friction ellipse forms a simple description of permissible control inputs, serving as a constraint for the optimization algorithms. The sum of the horizontal and longitudinal forces generated by a tire is bounded by the normal force F_z acting on the tire generated by the load. This can be described as a force budget that the controller can utilize when applying forces F_x and F_y. The tire friction ellipse is described in [9] and is defined as:

$$\frac{F_y^2}{\mu_y^2 F_z^2} + \frac{F_x^2}{\mu_x^2 F_z^2} \leq 1, \tag{20}$$

where F_y, F_x are the lateral and longitudinal forces generated by the tire, respectively, and μ_y, μ_x are the lateral and longitudinal sliding friction coefficients. The tire friction ellipse can be also described using the Pacejka parameter D, forming the friction ellipse for the rear and front tire forces F_r, F_f:

$$\begin{aligned} \frac{F_{r,y}^2}{D_r^2 F_{r,z}^2} + \frac{F_{r,x}^2}{D_r^2 F_{r,z}^2} \leq 1, \\ \frac{F_{f,y}^2}{D_f^2 F_{f,z}^2} + \frac{F_{f,x}^2}{D_f^2 F_{f,z}^2} \leq 1. \end{aligned} \tag{21}$$

In [25], a modified version of the tire friction ellipse was utilized, using additional parameters to modify the ellipse's shape:

$$\begin{aligned} \frac{F_{r,y}^2}{(p_{ellipse} D_r^2) F_{r,z}^2} + \frac{(p_{long} F_{r,x}^2)}{(p_{ellipse} D_r^2) F_{r,z}^2} \leq 1, \\ \frac{F_{f,y}^2}{(p_{ellipse} D_f^2) F_{f,z}^2} + \frac{(p_{long} F_{f,x}^2)}{(p_{ellipse} D_f^2) F_{f,z}^2} \leq 1, \end{aligned} \tag{22}$$

where the parameter $p_{long} \in [0, 1]$ influences the driving style so that with higher value, the vehicle is forced to corner less while accelerating, and higher $p_{ellipse} \in [0, 1]$ allows the tire forces to reach closer to their limits.

Eq. 22 can then be reformulated to obtain the formulation utilized in [29]:

$$\begin{aligned} F_{r,y}^2 + (p_{long} F_{r,x}^2) \leq F_{r,z}^2 (p_{ellipse} D_r^2), \\ F_{f,y}^2 + (p_{long} F_{f,x}^2) \leq F_{f,z}^2 (p_{ellipse} D_f^2). \end{aligned} \tag{23}$$

Note that in [29], the parameters D_r, D_f already contain the values of $F_{r,z}$, $F_{f,z}$, thus they are better representing the simplified Pacejka formula as described in Eq. 13. This results in the utilized and implemented tire friction ellipse formulation:

$$\begin{aligned} F_{r,y}^2 + (p_{long} F_{r,x}^2) \leq (p_{ellipse} D_r^2), \\ F_{f,y}^2 + (p_{long} F_{f,x}^2) \leq (p_{ellipse} D_f^2). \end{aligned} \tag{24}$$

4 Optimization Problem Formulation

Based on the results by *Liniger et al.*, we have selected the MPCC as utilized and described in [29] as the problem formulation for the MLT problem for its combined planning and tracking ability and showcased high performance. While the resulting controller is tracking the center line of the track, by applying low costs on the tracking error, the reference becomes mostly a measure of progress along the track that is to be maximized [29]. This eliminates the need to convert the track and vehicle model into curvilinear coordinates with the center lines arc length as the independent variable.

The MPCC is then solved in real-time using local convex quadratic programming (LCQP) approximations of the non-linear problem (NLP). The LCQP approximation is solved using the High-Performance Interior-Point Method Solver (HPIPM) presented in [20], which is a state of the art high-performance framework for quadratic programming (QP) designed to efficiently and reliably solve MPC problems.

The utilized MPCC problem formulation is taken directly from [29], and is presented here only as a means to a self-contained thesis. In order to formulate the MPCC for use in autonomous racing, it is first necessary to parametrize the reference trajectory, as described in Section 4.1. Afterward, to formulate the deviation of the autonomous car from its current reference point, error measures are introduced in Section 4.2.

Mathematical notation The set of real numbers is denoted as $\mathbb{R}$. Then, a set of n real column vectors is expressed as $\mathbb{R}^n$. The set of non-negative real numbers is expressed as $\mathbb{R}^0_+$, and the set of strictly positive real numbers as $\mathbb{R}_+$.

In case of a set of positive semi-definite matrices of size n, the used notation is $\mathbb{S}^n_+$ and a set of positive definite matrices as $\mathbb{S}^n_{++}$.

For a positive definite matrix $P \in \mathbb{S}^n_{++}$ and a vector $x \in \mathbb{R}^n$, the $||x||^2_P$ is defined as $||x||^2_P \triangleq x^t P x$.

The concatenation of two column vectors $a \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ is denoted as $(a, b) \triangleq [a^T, b^T]^T \in \mathbb{R}^{n+m}$.

4.1 Parametrization of Reference Trajectory

The parametrization of the reference trajectory is an offline procedure necessary for each unique track. The reference path is given as a set of center line points. The center line points are then interpolated with third order spline polynomials in a piece-wise manner.

The polynomials are then used to parametrize the reference path by its arc length:

$$\theta \in [0, L], \tag{25}$$

where L is the total length of the center line. Because of this parametrization, any center line reference point $[X^{ref}(\theta), Y^{ref}(\theta)]$ can be obtained by evaluating a third order polynomial for its argument θ. It also provides an accurate interpolation within the known points of the reference path.

By evaluating the equation:

$$\Phi(\theta) \triangleq \arctan\left\{\frac{\partial Y^{ref}(\theta)}{\partial X^{ref}(\theta)}\right\}, \tag{26}$$

it is also possible to find the angle of the tangent to the reference path $\Phi(\theta)$ at the specified reference point, which is used in error computation described in the following section.

4.2 Error measures

In order to calculate the optimal input sequence using the MPCC problem formulation, the error measures describing the deviation of the autonomous car's position from the reference path have to be defined.

Let θ_P, $\theta_P : \mathbb{R}^2 \rightarrow [0, L]$ be a projection operator on the reference trajectory defined by:

$$\theta_P(X, Y) \triangleq \arg\min_{\theta}(X - X^{ref}(\theta))^2 + (Y - Y^{ref}(\theta))^2, \tag{27}$$

where $[X, Y]$ marks the position of the vehicle and $[X^{ref}, Y^{ref}]$ the reference point. Then we can define the contouring error $e^c(X, Y, \theta_P)$ describing the orthogonal distance of the vehicle from the reference path as:

$$e^c(X, Y, \theta_P) \triangleq \sin(\Phi(\theta_P))(X - X^{ref}(\theta_p)) - \cos(\Phi(\theta_P))(Y - Y^{ref}(\theta_P)), \tag{28}$$

where Φ corresponds to the angle of a tangent to the reference path at a specified reference point, as defined in Eq. 26.

Due to the large computational costs of the projection operator θ_P, it is not well suited for online optimization algorithms. Therefore, we introduce an approximation θ_A of θ_P which is linked to θ_P by the equation:

$$e^l(X, Y, \theta_A) \triangleq |\theta_A - \theta_P|, \tag{29}$$

where $e^l(X, Y, \theta_A)$ forms the second error measure, lag error. The lag error serves as a measure of the quality of the approximation θ_A, which is an independent variable determined by the controller.

The contouring and lag errors $e^c(X, Y, \theta_P), e^l(X, Y, \theta_A)$ can be approximated as a function of the vehicle's position $[X, Y]$ and the approximate projection θ_A, which are variables controlled by the MPCC controller:

$$e^c \approx \hat{e}^c(X, Y, \theta_A) \triangleq \sin(\Phi(\theta_A))(X - X^{ref}(\theta_A)) - \cos(\Phi(\theta_A))(Y - Y^{ref}(\theta_A)), \tag{30a}$$

$$e^l \approx \hat{e}^l(X, Y, \theta_A) \triangleq -\cos(\Phi(\theta_A))(X - X^{ref}(\theta_A)) - \sin(\Phi(\theta_A))(Y - Y^{ref}(\theta_A)). \tag{30b}$$

This approximation (Eq. 30) has an effect that both the errors are independent on the projection operator θ_P. The approximate contouring error $\hat{e}^c$ and the approximate lag error $\hat{e}^l$ are defined as the orthogonal and tangential component of the error between $X^{ref}(\theta_A)$, $Y^{ref}(\theta_A)$ and the position X, Y, as shown in Fig. 8.

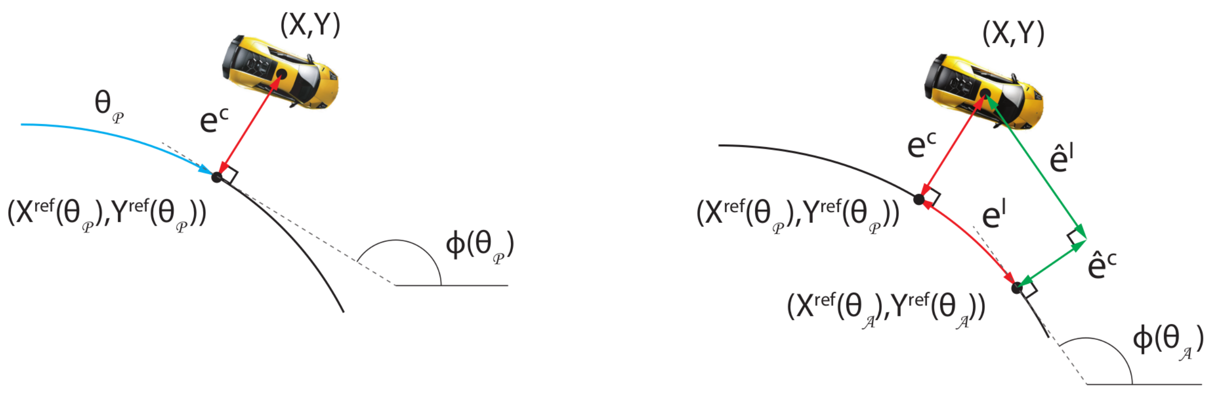


Figure 8: A visualization of the contouring error e^c (left) and the lag error e^l (right) with linear approximations $\hat{e}^c$ and $\hat{e}^l$ [29].

4.3 MPCC problem formulation

With the definition of the error measures, it is possible to formulate the MPCC problem, which is used for the MLT. The goal of the MPCC problem is to maximize the progress along the reference center line in a finite time horizon while simultaneously minimizing the contouring error describing the quality of reference tracking, constrained by the model dynamics, track constraints and input constraints. The time horizon of the MPCC is defined as N time samples. The optimization problem is formulated in the following way:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{N}\{||e^c_k(X_k, Y_k, \theta_P)||^2_{q_c}\} - \gamma\theta_{P,N}, && \text{(31a)}\\
\text{s.t.} \quad & x0 = x, && \text{(31b)}\\
& x_{k+1} = f(x_k, u_k), k = 0, \dots, N-1, && \text{(31c)}\\
& F_k x_k \leq f_k, k = 0, \dots, N-1, && \text{(31d)}\\
& \underline{\mathrm{x}} \leq x_k \leq \bar{x}, k = 0, \dots, N-1, && \text{(31e)}\\
& \underline{\mathrm{u}} \leq u_k \leq \bar{u}, k = 0, \dots, N-1, && \text{(31f)}
\end{aligned}$$

where $[X_k,\, Y_k]$ is the position of the autonomous vehicle in a time sample k. The position $[X_k,\, Y_k]$ is determined by the discrete-time model f, obtained by discretization of the utilized model equations (Eq. 35) with piece-wise constant control inputs. The $e^c_k(X_k, Y_k, \theta_{P,k})$ corresponds to the contouring error defined by Eq. 28, where $\theta_{P,k}$ is the associated path parameter such that $[X^{ref}(\theta_{P,k}), Y^{ref}(\theta_{P,k})]$ is the orthogonal projection of $[X_k, Y_k]$ onto the reference path. The associated weights, γ, $q_c \in \mathbb{R}_+$ for the progress along the reference center line and the contouring error, respectively, are then used to induce the trade-off between maximizing the progress and tight reference tracking. Constraints described by Eq. 31d are the parallel half space constraints for containing the position in the allowed corridor formed by the track boundaries, while Eqs. 31e,31f limit the states and inputs to physically admissible values.

Since the MPCC problem formulation relies on the projection operator $\theta_{P,k}$, it forms a bi-level NLP, because the approximation $\theta_{P,k}$ forms an NLP itself. The complexity of solving such a problem is unsuitable for online optimization, so the approximation. Therefore, $\theta_{A,k}$ of the projection operator $\theta_{P,k}$ is used instead, and the approximation quality is controlled by adding a cost on the lag error $\hat{e}^l_k$ to the objective.

In order to allow forming the lag error at each time step in the prediction horizon, it is necessary to introduce an integrator state with the dynamics:

$$\theta_{A,k+1} = \theta_{A,k} + \frac{v_k}{T_s}, \tag{32}$$

where v_k can be interpreted as the projected velocity and $\theta_{A,k}$ as the state of progress at time k. This approximation reduces the MPCC problem formulated in Eq. 31 to an optimal control problem in the form of an NLP that is amenable for a real-time implementation:

$$
\begin{aligned}
\min \quad & \sum_{k=1}^{N} ||\hat{e^c_k}(X_k, Y_k, \theta_{A,k})||^2_{q_c} + ||\hat{e^l_k}(X_k, Y_k, \theta_{A,k})||^2_{q_l} \\
& \qquad\qquad - \sigma v_k T_s + ||\Delta u_k||^2_{R_u} + ||\Delta v_k||^2_{R_v} - \sigma v_0 T_s, && \text{(33a)} \\
\text{s.t.} \quad & x_0 = x, \quad \theta_0 = \theta, && \text{(33b)} \\
& x_{k+1} = f(x_k, u_k), \quad \theta_{A,k+1} = \theta_{A,k} + \frac{v_k}{T_s}, \quad k = 0, \dots, N-1, && \text{(33c)} \\
& F_k x_k \leq f_k, \quad \underline{\mathrm{x}} \leq x_k \bar{x}, \quad 0 \leq \theta_k \leq L, \quad k = 1, \dots, N, && \text{(33d)} \\
& \underline{\mathrm{u}} \leq u_k \leq \bar{u}, \quad 0 \leq v_k \leq \bar{v}, \qquad k = 0, \dots, N-1, && \text{(33e)}
\end{aligned}
$$

where $\delta u_k \triangleq u_k - u_{k-1}$ and $\delta v_k \triangleq v_k - v_{k-1}$. Due to the utilization of the approximation $\theta_{A,k}$, the objective (Eq. 33a) utilizes the approximate contouring error $\hat{e}^c_k(X_k, Y_k, \theta_{A,k})$. The maximization of the final progress measure $\theta_{P,N}$ is furthermore replaced by $\sum_{k=0}^{N-1} v_k T_s$, which is equivalent if the approximation is accurate. Lower and upper bounds of the v_k and θ_k are imposed to avoid spurious solutions of the NLP, with $\bar{v}$ denoting the largest possible progress per sampling time. The cost on the lag error $e^l_k(X_k, Y_k, \theta_{A,k})$ links the state of progress to the dynamics of the car. To ensure an accurate progress approximation and thus a strong coupling between the cost function and the car model, the weight on the lag error $q_l \in \mathbb{R}_+$ is chosen high, as suggested in [27]. Furthermore, a cost term on the rate of change of the inputs is added to the objective in order to penalize fast changing controls, which helps to obtain smooth control inputs and preventing amplifying unmodeled dynamics.

4.4 Solving the MPCC problem

In order to solve the non-linear optimal control problem described by Eq. 33 in real-time, local convex approximations of the non-linear control problem in the form of the following QP formulations are built at each sampling time. This is done by a linearization of the non-linear terms:

$$
\begin{aligned}
\min_{x,u,\theta,v,s} \quad & \sum_{k=1}^{N} \begin{bmatrix} x_k \\ \theta_{A,k} \end{bmatrix}^T \Gamma_k \begin{bmatrix} x_k \\ \theta_{A,k} \end{bmatrix} + c_k^T \begin{bmatrix} x_k \\ \theta_{A,k} \end{bmatrix} - \gamma v_k T_s \\
& \qquad + \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix}^T R \begin{bmatrix} \Delta u_k \\ \Delta v_k \end{bmatrix} + q||s_k||_\infty - \gamma v_0 T_s, && \text{(34a)} \\
\text{s.t.} \quad & x_0 = x, && \text{(34b)} \\
& \theta_{A,0} = \theta, && \text{(34c)} \\
& x_{k+1} = A_k x_k + B_k u_k + g_k, \quad k = 0, \dots, N-1, && \text{(34d)} \\
& \theta_{A,k+1} = \theta_{A,k} + \frac{v_k}{T_s}, \quad k = 0, \dots, N-1, && \text{(34e)} \\
& F_k x_k \le f_k + s_k, \quad k = 1, \dots, N, && \text{(34f)} \\
& s_k \ \ge 0, \quad k = 1, \dots, N, && \text{(34g)} \\
& \underline{x} \le x_k \le \bar{x}, \quad k = 1, \dots, N, && \text{(34h)} \\
& 0 \le \theta_{A,k} \le L, \quad k = 1, \dots, N, && \text{(34i)} \\
& \underline{u} \le u_k \le \bar{u}, \quad k = 0, \dots, N, && \text{(34j)} \\
& 0 \le v_k \le \bar{v}, \quad k = 0, \dots, N, && \text{(34k)}
\end{aligned}
$$

where $\Gamma_k \in \mathbb{S}_+^7$ is formed by the quadratic part of the linearized contouring and lag error cost function and $c_k \in \mathbb{R}^7$ are from the linear part. In order to keep the linearization error small, the linear time variant approximation for the dynamics (Eq. 35) is used as well as for the contouring and lag errors. Each non-linear function is linearized around the output of the last QP iteration, shifted by one stage. The measurement $x = x_0$ is used as the first linearization point, and the last input of the previous iteration is kept constant to generate a new last input. The linearization point for the terminal state x_N is calculated by simulating the non-linear model for one time step. The track constraints (Eq. 34f) are formulated by two half space constraints tangential to the track per time step. These constraints are formulated as soft constraints, with slack variables $s_k \in \mathbb{R}^2$ and a corresponding infinity-norm penalty in the objective weighted by $q \in \mathbb{R}_+$, which is chosen quite high to recover the behavior of the hard constrained problem whenever possible.

The local QP is then solved in real-time using local convex QP approximations of the NLP, which is then solved using the HPIPM solver [20].

5 F1/10 Platform

The first iteration of the vehicle platform was developed in [48] for the *F1/10 Autonomous Racing Competition* [34] of standardized 1/10 scale vehicles. In the *F1/10 Autonomous Racing Competition*, university teams compete against each other in time-trial and head-to-head races on various racing tracks. The currently utilized version of the vehicle platform was developed in [16].

The organizers of the competition provide detailed build instructions, as seen in [47]. In order to maintain a balanced competition of algorithms, the vehicle's equipment is strictly defined and limited. The platform, together with its most relevant software, is briefly described in Section 5.1, and the Chapter 7 describes how the MPCC algorithm is integrated with the rest of the vehicle.

5.1 F1/10 compatible autonomous car description

An image of the utilized autonomous platform can be seen in Fig. 10. The autonomous vehicle platform used for the F1/10 competition is based on the Traxxas Slash 1/10 4WD chassis and is equipped with the NVIDIA Jetson TX2 embedded computer (Fig. 9).

The NVIDIA Jetson TX2 is a system on module embedded computing device built around the NVIDIA Pascal GPU architecture. It features two CPU units, the NVIDIA Denver2 dual-core and the ARM Cortex-A57 quad-core CPUs, both operating at 2GHz and 2GHz, respectively [33]. On top of that, it contains a 256-core NVIDIA Pascal-type GPU with 256 NVIDIA CUDA cores, together with 8GB LPDDR4 128-bit memory. The NVIDIA Jetson TX2 is connected to the Orbitty carrier board, providing USB and Gigabit Ethernet connectivity and a microSD slot.

The onboard sensors of interest are the Hokuyo UST-10LX Light Detection and Ranging (LIDAR) sensor and the SparkFun 9DoF Razor IMU, a 9 Degrees of Freedom Inertial Measurement Unit (IMU). The vehicle uses a Velineon 3500 DC motor, which is controlled by the Vedder Electronic Speed Controller (VESC), which allows to control the vehicle using the ERPM.

The vehicle is also equipped with a B3-STX Deluxe 2.4GHz transceiver, which allows us to take over the direct control of the vehicle with a remote transmitter. For safety reasons, it is used as an emergency brake, where the vehicle's autonomy is turned off whenever a command from the remote transmitter exceeds a dead-zone threshold. The authority over the vehicle controls is managed by a Teensy microcontroller, which serves as a bridge between the VESC and the transceiver or the onboard NVIDIA Jetson TX2. During the autonomous mode, the VESC receives the commands sent by the onboard NVIDIA Jetson TX2. Any command exceeding the dead-zone threshold from the transmitter then turns off

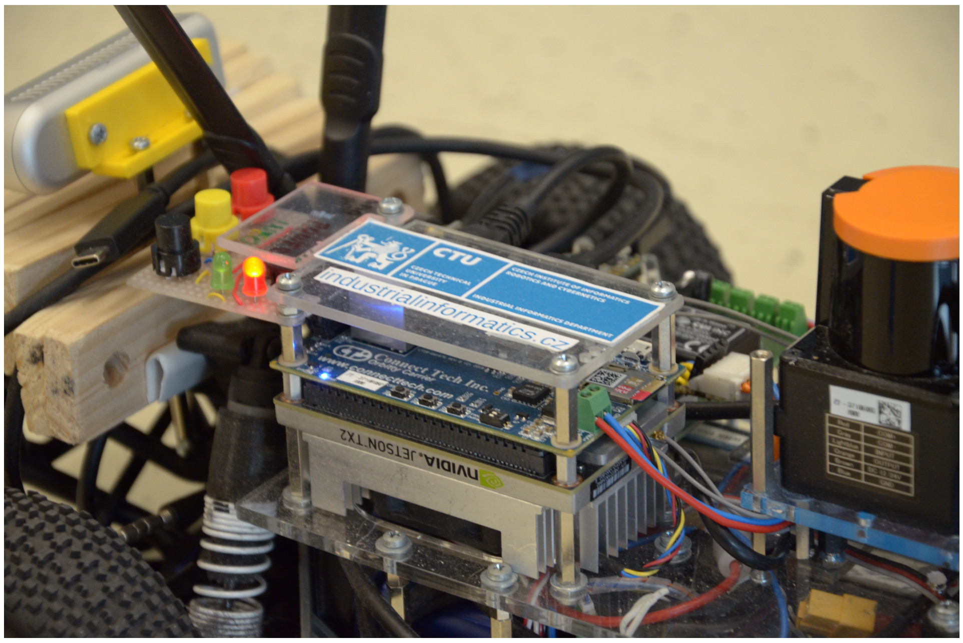


Figure 9: A closeup photo of the NVIDIA Jetson TX2 with the Orbitty carrier board.

the vehicle's autonomy, and the vehicle can be operated only using the remote transmitter until the autonomous mode is restored using a specific command.

The full list of the vehicle's hardware can be seen in Table 1.

Chassis	Traxxas Slash 1:10 4WD VXL
Onboard computer	NVIDIA Jetson TX2
Engine	Velineon 3500
Controller	VESC
Lidar	Hokuyo UST-10LX
IMU	SparkFun 9DoF Razor IMU
Transceiver	B3-STX Deluxe 2.4GHz
Power management	Powerboard, LiPo 3S
Camera	Intel RealSense D435
GPIO	Orbitty dev board v1

Table 1: List of hardware on the vehicle platform.

The utilized Linux operating system is Ubuntu Linux 16.04 equipped with the Robot Operating System (ROS) Kinetic Kame [43], a robotics middleware containing the libraries to interconnect the vehicle's control and data processing software. The vehicle's control and data processing software is formed by the so-called ROS nodes, which are computation performing processes [44]. The nodes are organized into ROS packages, which are a standardized format for modular libraries and nodes. Each package is a directory containing

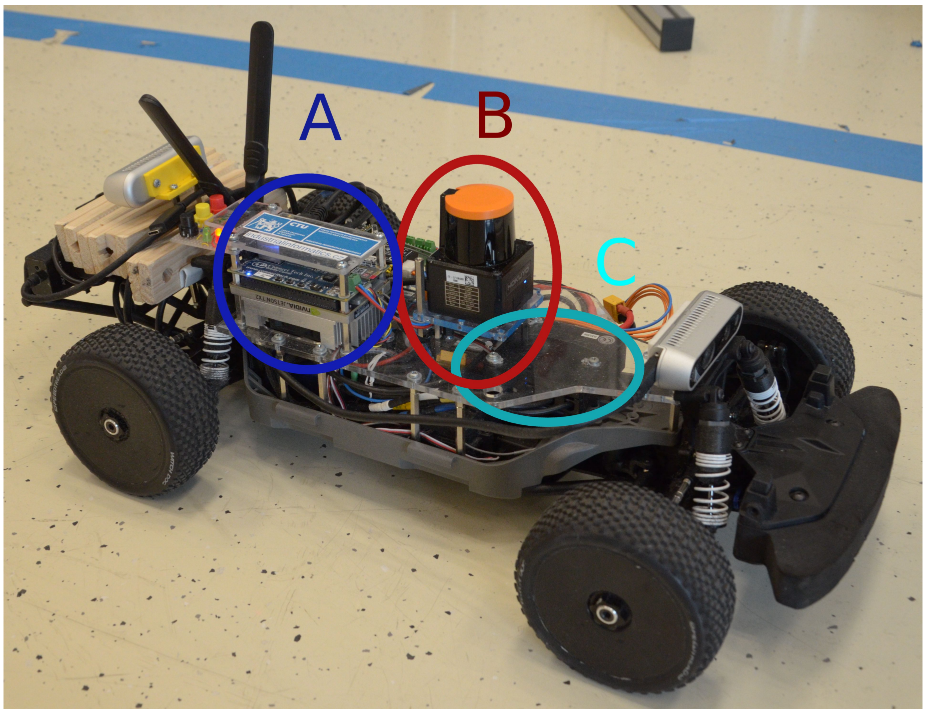


Figure 10: A photo of the utilized autonomous vehicle platform. **A** is the NVIDIA JETSON TX2 with the Orbitty carrier board, **B** is the LIDAR and **C** shows the VESC (under the acrylic glass). The IMU is hidden under the LIDAR.

the `package.xml` file, also called the package manifest. The package manifest defines the package's name, version, author and its dependencies on other packages.

The nodes communicate with each other using topics, which are named buses used for exchanging messages with anonymous publish-subscribe pattern. The messages are defined either in their own packages or as a part of a larger package with other messages or nodes. Each message is a data structure containing typed and named fields and has its own unique name.

The main packages of interest for optimization-based control are the `cartographer_slam`, containing the Cartographer SLAM algorithm described in Section 5.1.1, the `drive_api` package, providing a simple to use interface for low-level vehicle control and the respective drivers for vehicle's onboard sensors, which periodically publish collected sensor data in their respective topics.

5.1.1 Cartographer SLAM

It is necessary to know the track's layout and the car's position when considering planning-based control methods. Since the F1/10 competition prohibits the utilization of external sensors, the mapping and localization must be done using only the vehicle's onboard sensors. Simultaneous Localization And Mapping (SLAM) algorithms are well suited for such applications since they often allow to utilize only the onboard sensors, and they are able to both create and update environmental maps and localize the robot inside them. The F1/10 platform utilizes the Cartographer SLAM, presented in [24], which is a SLAM algorithm for 2D and 3D localization and mapping. It utilizes onboard planar, 3D, and multi-echo LIDARs together with the IMU and optionally GPS readings to provide planar occupancy-grid based maps and position readings, with a periodical loop-closing procedure to compensate for map drift due to sensor noise.

The data from the IMU is processed by the Ceres solver, presented in [4], which helps to mitigate measurement noise, and which is utilized to incorporate the knowledge about the robot's movement into map building and vehicle localization. This is necessary for localization and mapping in monotonous corridors, which are often present on racing tracks. In such track segments, the measurements from the onboard LIDAR may remain similar for various locations in the corridor, making it difficult to correctly estimate the robot's movement between separate measurements.

Due to the platform limitations by the competition rules, the vehicle is equipped with one onboard planar LIDAR and one IMU. Because the racing tracks used in the competition are planar, we can use only 2D maps, which helps to reduce the complexity of the SLAM task. The utilization of Cartographer SLAM in a racing scenario consists of 2 steps. First, the map is created during several laps, which are either manually driven, or driven autonomously using a reactive algorithm. The stored map is then loaded with the algorithm in localization mode, which has reduced computational requirements, and all map updates are short-term only.

The accuracy of the Cartographer SLAM was evaluated using the VICON external camera localization system [3]. The VICON localization system was evaluated against ground truth in [31], where the reported root mean squared error (RMSE) against the ground truth was $RMSE = 0.524\,\mathrm{mm}$ for velocities lower than $1\,\mathrm{m\,s^{-1}}$ and $RMSE = 0.329\,\mathrm{mm}$ for velocities higher than $3\,\mathrm{m\,s^{-1}}$. For the position update frequency of $100\,\mathrm{Hz}$, the mean absolute error (MAE) was $MAE = 0.367\,\mathrm{mm}$. This accuracy is sufficient to regard the VICON reported position with the update frequency $100\,\mathrm{Hz}$ as the ground truth for the evaluation of the Cartographer SLAM.

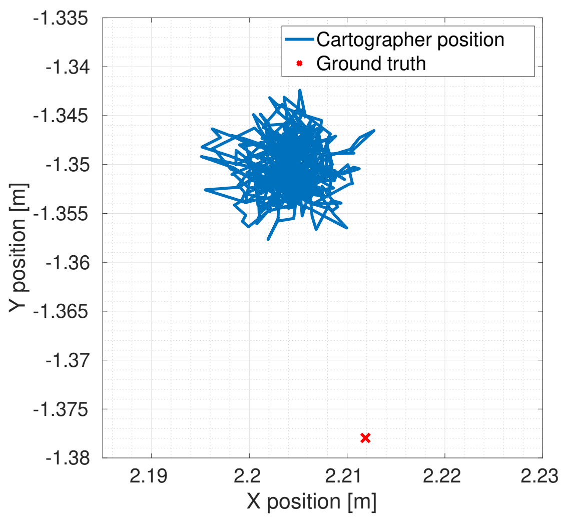
First, it is necessary to account for the different map coordinate system origins and rotations of VICON and Cartographer SLAM in a way that would not influence the evaluation process. To do this, the vehicle's position and yaw were recorded while standing still, and the map and yaw offsets were found as the difference between the averages of both source's respective measurements. This offset was then saved and was used to compensate the translational and rotational offset of the coordinate systems of both localization systems while processing data from further experiments. Measuring the offset in an independent experiment helps to ensure that the absolute localization error of Cartographer

SLAM at the beginning of each experiment is accounted for.

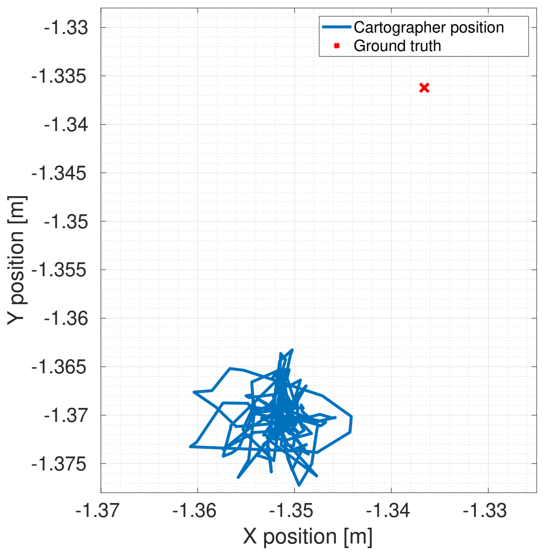
The accuracy was then evaluated using two experiments. In the first one, the reported position of the stationary vehicle from both localization systems was gathered. The offset-compensated measurements of the calibration experiment can be seen in Fig. 11a and the stationary experiment in Fig. 11. The second experiment consisted of localization data gathering from a manually driven trajectory. The offset-compensated measurements of the racing experiment can be seen in Fig. 12.

The non-zero $RMSE$ of the offset-compensated data from the calibration experiment can be attributed to the rotational compensation of the average yaw difference between the VICON and Cartographer SLAM coordinate systems. Since the $RMSE$ has a similar value for both the calibration and stationary experiment data, it can be assumed that the absolute error of the Cartographer SLAM is negligible and the non-zero $RMSE$ originates mainly from the transformation from the Cartographer SLAM map to the VICON map.

The $RMSE$ of the manually driven experiment, shown in Fig. 12, is low enough for the Cartographer SLAM to be usable for racing purposes, but must be taken into account during track processing, for example, by adjusting the track's safe corridor by the localization error.



(a) The calibration experiment.



(b) The stationary experiment.

Figure 11: The comparison between the ground truth and the Cartographer SLAM reported position during the calibration and stationary experiment measurements. The position error of the Cartographer SLAM is $RMSE = 2.050$[cm] for the calibration data and $RMSE = 2.680$[cm] for the stationary experiment.

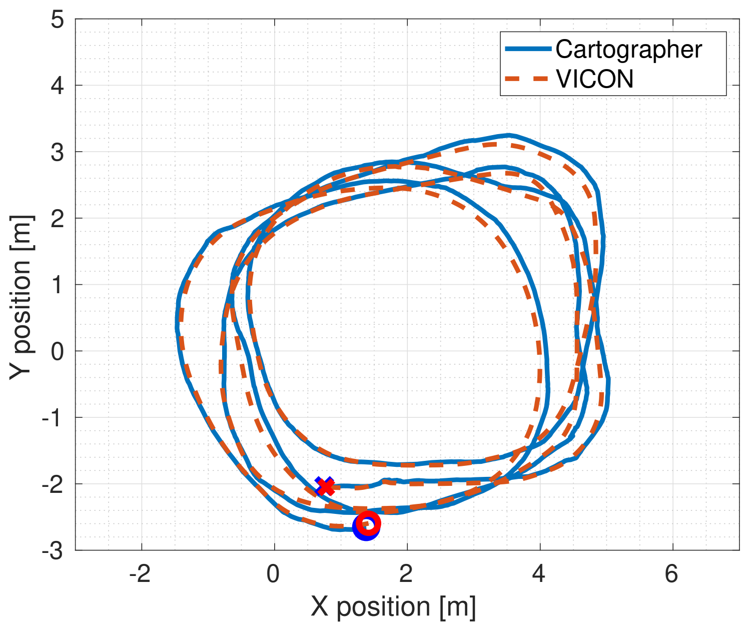


Figure 12: The comparison between the ground truth and the Cartographer SLAM reported position in an experiment with manual driving. The marker **x** denotes the starting position, the marker **o** the ending position. The position error of the Cartographer SLAM is $RMSE = 9.220$[cm].

6 F1/10 model identification

The process of vehicle model identification starts with selecting the appropriate mathematical models for the task from Chapter 3. The selected model, together with the explanation of the decision process, is described in Section 6.1.

The process of identification itself consists of parameter measurements and data processing from dynamic vehicle experiments, which are designed to clearly demonstrate the vehicle behavior that needs to be mathematically modeled. An example of the first category can be the vehicle's weight, or geometry, which can usually be directly measured. The second category contains, for example, the behavior of the vehicle during acceleration or during high side-slip maneuvers. The process of the identification, together with the experiments, is described in Section 6.2.

6.1 Utilized vehicle model

Due to the limited computational capacity of the vehicle's onboard computer, the complexity of the vehicle model is a necessary factor to be considered. However, the vehicle model must also sufficiently describe the dynamical behavior commonly present in the intended scenarios to be usable for motion planning.

Based on the results of model performance analysis during lane change and hairpin turn maneuvers published in [8], using the more advanced dual-track model leads to similar maneuver execution times, while requiring an order of magnitude higher solution time.

The comparison of the single-track model, the single-track model with pitch and roll dynamics, and the dual-track model with pitch and roll dynamics can be seen in Table 2. While the pure single track model did not consider pitch and roll dynamics, it was possible to utilize it to perform the desired maneuvers.

Furthermore, by comparing the single-track model with pitch and roll dynamics to the dual-track model, it can be assumed that the pitch and roll dynamics are not the reason for long execution times. Due to the successful application of all three models and the solution times, the single-track model without pitch and roll dynamics is selected for the purposes of this thesis.

To calculate the longitudinal and lateral forces acting on the tires, the simplified Pacejka tire model (as presented in Section 3.3.4) is used for its simplicity and its empirical nature, since measuring the tire coefficients, as required in other tire models, may not be done accurately with the available equipment. By combining the single-track model with the simplified Pacejka tire model and extending them for four-wheel driven drivetrain, the

Vehicle Model	Single-track	Single-track with pitch	Dual-track with pitch
Maneuver Time [s]	4.28	4.12	4.30
Solution Time [s]	8.00	16.90	137.80
No. of Iterations [-]	111	110	340

Table 2: Comparison of the dynamic single-track, dynamic single-track with pitch and roll dynamics and dynamic dual-track with pitch and roll dynamics models during lane changing and hairpin turn maneuvers, as presented in [8].

resulting utilized model is described by the equation:

$$\dot{X} = v_x cos(\varphi) - v_y \sin(\varphi), \tag{35a}$$
$$\dot{Y} = v_x sin(\varphi) + v_y \cos(\varphi), \tag{35b}$$
$$\dot{\varphi} = \omega, \tag{35c}$$
$$\dot{v}_x = \frac{1}{m}(F_{r,x} + F_{f,x}\cos(\delta) - F_{f,y}\sin(\delta) + m v_y \omega), \tag{35d}$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} + F_{f,y}\sin\delta + F_{f,x}\sin(\delta) - m v_x \omega), \tag{35e}$$
$$\dot{\omega}_y = \frac{1}{I_z}(F_{f,y} l_f \cos(\delta) + F_{f,x} l_f \sin(\delta) - F_{r,y} l_r), \tag{35f}$$

where X, Y represent the vehicle's position in global coordinates, v_x, v_y denote the longitudinal and lateral velocities, respectively, φ is the vehicle's yaw (heading) angle, ω is the vehicle's yaw rate, l_f, l_r are the distances of the front and rear axles, respectively, from the vehicle's center of gravity, m is the mass of the vehicle, I_z is the moment of inertia in the z axis and δ is the vehicle's steering angle, considered as an input. The $F_{a,b}$ signifies the forces, where $a \in \{f, r\}$ denotes the front and real forces and $b \in \{x, y\}$ denotes the longitudinal and lateral force, respectively. The lateral forces $F_{a,y}$ are obtained using the tire model:

$$F_{f,y} = D_f \cdot \sin(C_f \cdot \arctan(B_f \cdot \alpha_f)), \tag{36a}$$
$$F_{r,y} = D_r \cdot \sin(C_r \cdot \arctan(B_r \cdot \alpha_r)), \tag{36b}$$
$$\alpha_f = -\arctan\left(\frac{\omega l_f + v_y}{v_x}\right) + \delta, \tag{36c}$$
$$\alpha_r = \arctan\left(\frac{\omega l_r - v_y}{v_x}\right), \tag{36d}$$

where B_f, C_f, D_f are the Pacejka parameters and α_f, α_r are the front and rear slip angles, respectively.

The longitudinal forces $F_{a,x}$ are obtained using the drivetrain model, where two different approaches were utilized. The first one describes the longitudinal force $F_x = F_{f,x} = F_{r,x}$ the drivetrain applies as:

$$F_x = (C_{m1} - C_{m2}v_x)\tilde{d} - C_{m3} - C_{m4}v_x^2, \tag{37}$$

where $\tilde{d} \in [0, 1]$ is the ERPM reference input and $C_{m1\text{–}4}$ are the empirical drivetrain parameters.

The second describes the longitudinal force $F_x = F_{f,x} = F_{r,x}$ as:

$$F_x = \frac{(C_{m1}\tilde{d})}{C_{m2}} - \frac{v_x}{C_{m3}}. \tag{38}$$

where $\tilde{d} \in [0, 1]$ is the ERPM reference input and $C_{m1\text{–}3}$ are the empirical drivetrain parameters. A graphical representation of the utilized vehicle model, together with a complete description of its parameters is shown in Fig. 13.

Since the dynamic single-track vehicle model combined with the Pacejka tire model is applicable only for non-zero longitudinal velocities, it is assumed that the vehicle is always in motion. The minimal longitudinal velocity v_x is therefore set as $v_{x,min} = 0.1\,\{\mathrm{m\,s^{-1}}\}$. Because it is undesirable for the vehicle to stop in a race, this assumption is valid and should not pose any problems in real-life applications.

6.2 Identification

Due to the unavailability of specialized tools for motor torque and tire parameter measurement, it is necessary to design identification experiments that rely solely on the data the utilized vehicle platform provides, as described in Chapter 5. This includes the ERPM (see 3.4), the position of the car and the accelerometer readings.

At first, the vehicle's parameters must be obtained using the methods described in Section 6.2.1. Afterward, the drivetrain model needs to be identified, using the method described in Section 6.2.2. The drivetrain model is then used to simulate accurate vehicle velocities from known inputs for the purposes of the tire model identification described in Section 6.2.3.

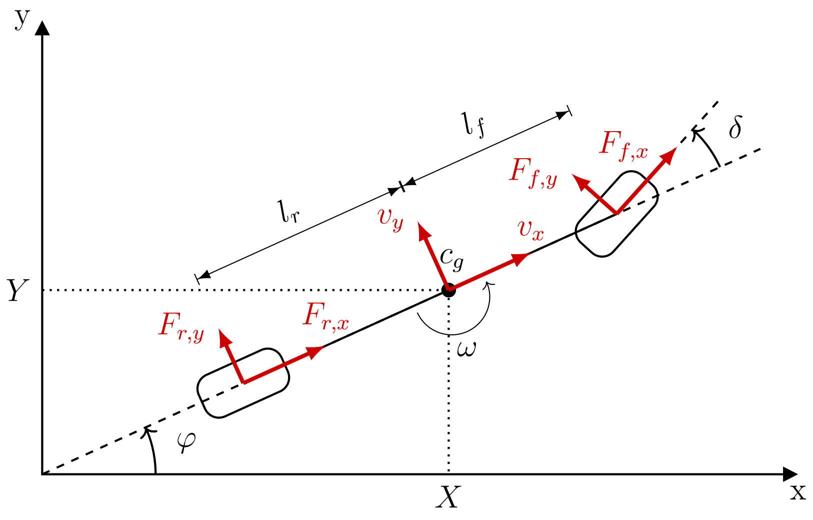


Figure 13: The utilized four-wheel driven single-track vehicle model with its respective variables. X, Y [m] denote the vehicle's position in global coordinates, $F_{r,x}$, $F_{f,x}$ [N] are the rear and front longitudinal forces acting on the vehicle, $F_{r,y}$, $F_{f,y}$ [N] are the lateral rear and front forces acting on the vehicle, φ [rad] is the vehicle's yaw (heading), ω [rad s^{-1}] is the yaw rate, δ [rad] is the input steering angle, v_x [m s^{-1}] is the longitudinal velocity of the vehicle, v_y [m s^{-1}] is the lateral velocity of the vehicle, c_g [m] is the center of gravity and l_f, l_r [m] are the distances between the center of gravity and the front and rear wheel, respectively.

6.2.1 Vehicle parameter identification

In the beginning, we have to measure the parameters of the vehicle itself. These parameters, also visualized in Fig. 14, are:

- The vehicle's mass m [kg],
- The moment of inertia of the z axis I_z [kg m^2],
- The distance l_f [m] from the center of gravity c_g to the front axle,
- The distance l_r [m] from the center of gravity c_g to the rear axle,
- The maximum steering angle δ_{max} [rad].

Length of the shaft The total length of the vehicle's driveshaft was measured as the distance between the front and rear wheel nuts, as the driveshaft itself is not accessible without disassembling the vehicle, and the manufacturer does not provide schematics with enough information. It was measured to be $l_{wb} = 0.33$ [m].

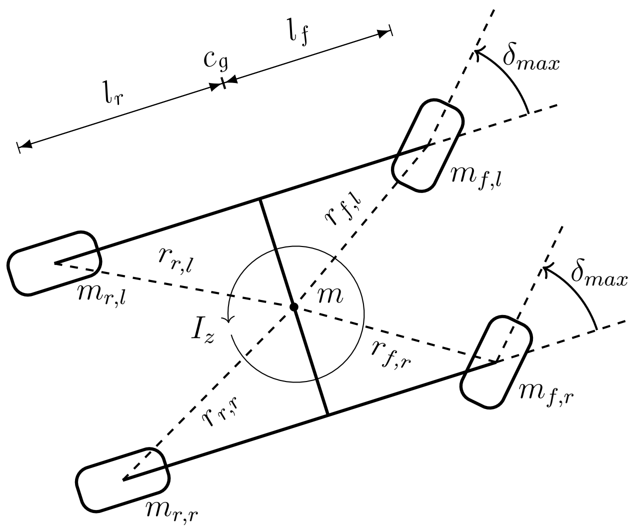


Figure 14: Visual description of the vehicle parameters required for the identification.

Mass distribution Afterward, the SkyRC Corner Weight System [2] was used to measure the mass distribution on the front and rear axle and each separate wheel, as seen in Fig. 15. This measurement provides the total mass of the vehicle $m = 3837 \pm 0.5$ [g] and the proportions of the mass distribution between the front and rear axle m_f, m_r, respectively, from which it is possible to calculate:

$$\begin{aligned} l_f &= l_{wb}\left(1 - \frac{m_f}{m}\right) = l_{wb} \cdot (1 - 0.42) = 0.191\,[\mathrm{m}], \\ l_r &= l_{wb}\left(1 - \frac{m_r}{m}\right) = l_{wb} \cdot (1 - 0.58) = 0.139\,[\mathrm{m}]. \end{aligned} \tag{39}$$

Moment of inertia The moment of inertia I_z is then calculated as the moment of inertia of a rigid assembly of point masses, formed by the vehicle's wheels. The measured distances from the vehicle's wheels can be seen in Table 4 together with their respective masses. The I_z is then calculated by the following equation:

$$I_z = \sum_{i=1}^{N} m_i r_i^2 = 0.152\,[\mathrm{kg\,m^2}], \tag{40}$$

where $N = 4$ is the number of point masses (wheels), m_i [kg] is the mass of the point mass i and r_i [m] is its distance from the center of gravity.

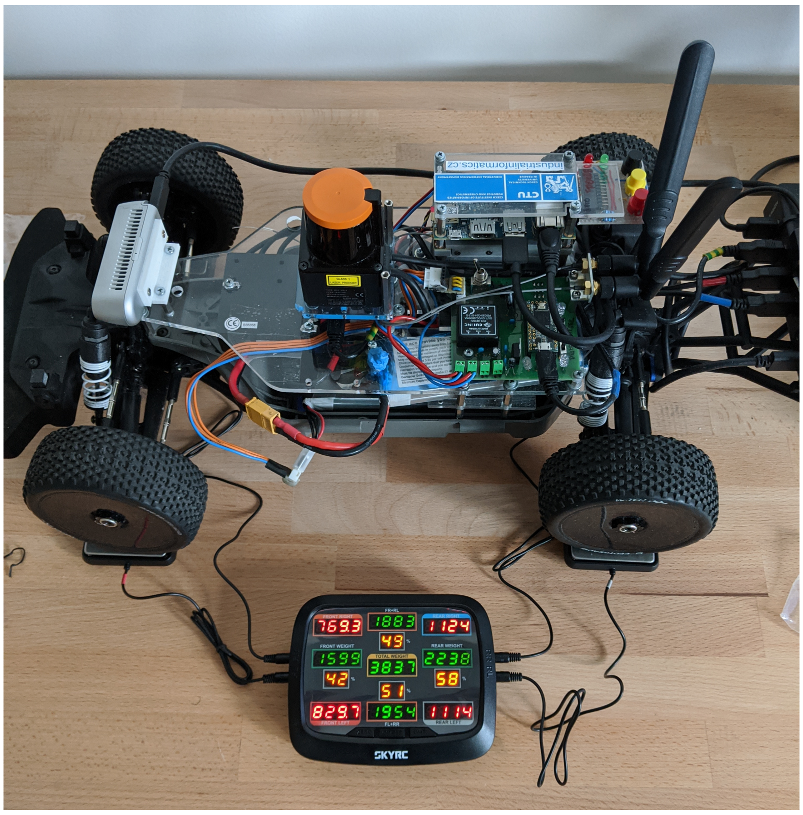


Figure 15: A photograph of the weight distribution measurement using the SkyRC Corner Weight System [2]. Displayed units are in [g].

Maximum steering angle The maximum steering angle δ_{max} [rad] is calculated from the measured turning radius of the vehicle during an experiment with constant maximum steering angle δ and low velocity so that there occurs no lateral slip. This experiment was conducted for both steering directions, and the measured data can be seen in Fig. 16. The minimal turning radius R_{min} [m] was then calculated as:

$$\begin{aligned} R_{min,l} &= \frac{Y_{max,l} - Y_{min,l}}{2} = 0.678\,[\mathrm{m}], \\ R_{min,r} &= \frac{Y_{max,r} - Y_{min,r}}{2} = 0.719\,[\mathrm{m}], \end{aligned} \tag{41}$$

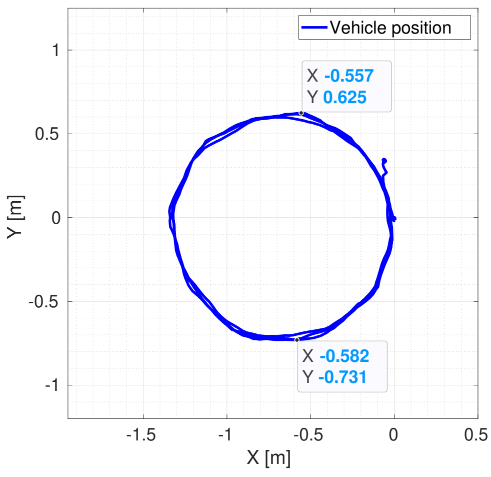
where Y_{max}, Y_{min} are the maximal and minimal values of the coordinate Y, respectively, and $Y_{max,l}$, $Y_{min,l}$ correspond to the leftward steering while $Y_{max,r}$, $Y_{min,r}$ to the rightward. The maximal steering angle $\delta_{max,l}$, $\delta_{max,r}$ can then be calculated as:

$$\begin{aligned} \delta_{max,l} &= \arcsin\left(\frac{l}{R_{max,l}}\right) = 0.508\,[\text{rad}], \\ \delta_{max,r} &= \arcsin\left(\frac{l}{R_{max,r}}\right) = 0.477\,[\text{rad}], \end{aligned} \tag{42}$$

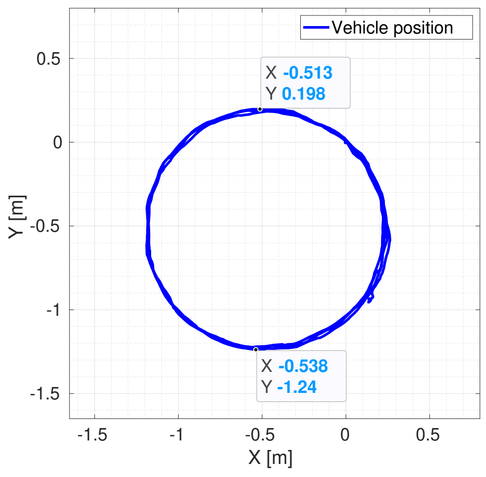
where l is the length of the wheelbase and $\delta_{max,l}$, $\delta_{max,r}$ correspond to the leftward and rightward maximum steering angle, respectively. The resulting maximal steering angles for the leftward and rightward steering together with the measured turning radii can be found in Table 3. Since the leftward maximal steering angle δ_{max} is larger than the rightward, the mean maximal steering angle is calculated and utilized. The calculated maximal steering angle roughly corresponds to the one utilized in [13].

Steering direction	Left	Right	Mean
R_{min} [m]	0.678	0.719	0.696
δ_{max} [rad]	0.508	0.477	0.492

Table 3: The measured turning radii R_{min} and calculated maximal steering angles δ_{max} for leftward and rightward steering, together with their average.



(a) Minimal leftward turning radius.



(b) Minimal rightward turning radius.

Figure 16: The measured vehicle's position during the turning radius measurement with depicted positions corresponding to the maximal and minimal Y position value.

Wheel	Front, left	Front, right	Rear, left	Rear, right
m [kg]	0.829	0.769	1.114	1.124
r [m]	0.223	0.223	0.180	0.180

Table 4: The masses m on each wheel and its distance r from the center of gravity c_g.

m [kg]	I_z [kg m^2]	l_f [m]	l_r [m]	δ_{max} [rad]
3.958	0.152	0.191	0.139	0.492

Table 5: Measured parameters of the utilized F1/10 autonomous vehicle platform.

6.2.2 Drivetrain model identification

The identification of the drivetrain model described in Section 3.4 consists of multiple acceleration and braking maneuvers with varying values of the ERPM reference input $\tilde{d}$ with the steering angle $\delta = 0$. Due to the zero value of δ, it can be assumed that there occurs no lateral slip. Furthermore, since we are unable to measure the longitudinal slip, it is also assumed to be nonexistent. The measured ERPM then directly corresponds to the vehicle's longitudinal velocity.

The data from the acceleration experiments consists of the ERPM as reported by VESC, and the vehicle input $\tilde{d}$. This measured data was then processed to form pairs of values based on the time in which they were obtained. Each of the measurements was ended using a slight braking input from the RC controller, which turns off the vehicle's autonomy, effectively simulating the reference command $\tilde{d} = 0$. Since turning off the vehicle's autonomy does not change the vehicle's input command, it was set as zero in post-processing when the operator's command arrived.

The drivetrain model is then fitted on the measured data from all acceleration experiments using the `ga()` function from the Mathworks Global Optimization Toolbox [1]. The `ga()` function is a genetic algorithm minimizing the cost function:

$$\texttt{cost} = \texttt{measured\_velocity} - \texttt{simulated\_velocity}, \tag{43}$$

to obtain a model that is accurate for the largest range of inputs possible. The parameters used for the optimization can be seen in Table 6. It is then verified on an independent second set of experiments performed under the same conditions.

Due to the limited area of the testing environment where the experiments were conducted, the vehicle could sustain the steady-state velocity only for a limited amount of time, especially in the case of higher ERPM reference inputs. This causes the genetic algorithm to produce parameters accurate for the steering and braking segment, but with higher steady-state velocities. Therefore, the measured data were processed by artificially lengthening the steady-state segment, effectively increasing the cost of the steady-state error. An example of the artificial steady-state lengthening can be seen in Fig. 17.

First, the generalized form of the model, as described by Eq. 19, is fitted onto the experimental data. The results can be seen in Fig. 18. While the acceleration and steady-state values represent the drivetrain accurately for $\tilde{d} = 0.5$, as seen in Fig. 18b, the steady-state value accuracy decreases for larger and smaller inputs, as shown for $\tilde{d} = 0.3$ in fig. 18a.

For this reason, the first order model, described in Eq. 18, is utilized and fitted onto the experimental data. The response of the model with parameters identified by the genetic algorithm, can be seen in Fig. 19. While the accuracy in the acceleration and braking segments is lower, its accuracy in the steady state segment does not decrease with varying ERPM reference inputs $\tilde{d}$. This can be seen when comparing the Figs. 19a and 19b that display the response to ERPM reference $\tilde{d} = 0.4$ and $\tilde{d} = 0.7$, respectively.

Since the steady state error is still significant, the identified parameters were afterward manually tuned at the cost of acceleration and braking accuracy, resulting in the modified drivetrain response, as seen in Fig. 20. The steady-state error was reduced at the expense of the braking segment error. The verification of this model on an independent set of data can be seen in Fig. 21.

Application	Population size	Max generations	Number of variables
Drivetrain model	250	2000	$\begin{cases} 4 & \textit{if} \text{ generalized model,} \\ 3 & \textit{if} \text{ first order model} \end{cases}$
Tire model	250	2000	5

Table 6: The parameters of the GA function used to obtain parameters of the drivetrain and tire models.

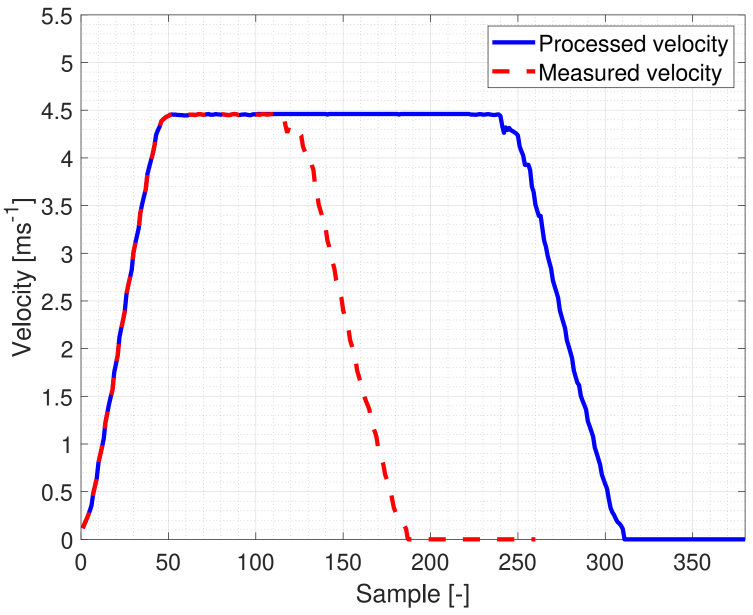
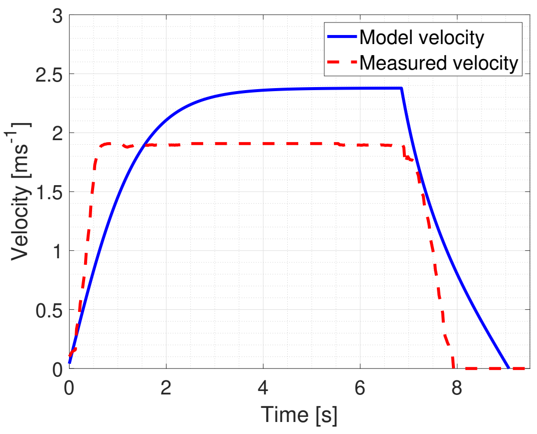
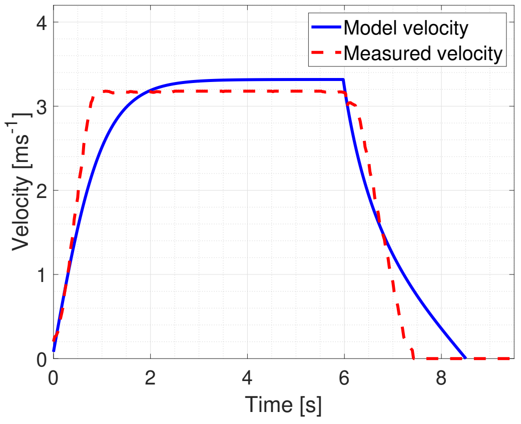


Figure 17: Velocity data with artificially lengthened steady state compared to original measured data.

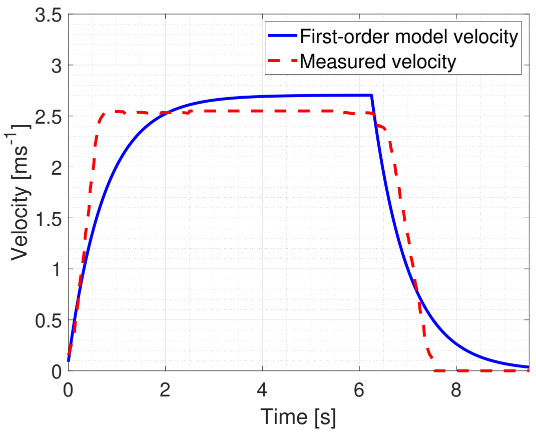


(a) ERPM reference response for $\tilde{d} = 0.3$.

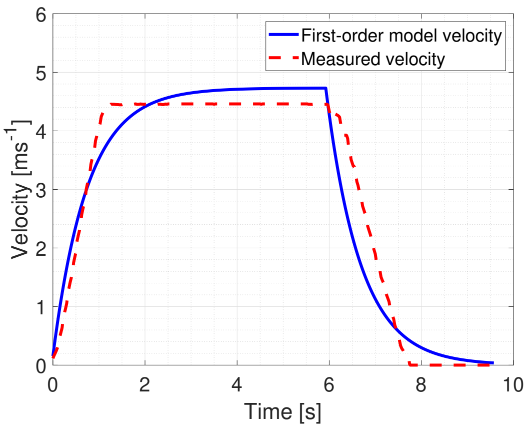


(b) ERPM reference response for $\tilde{d} = 0.5$.

Figure 18: The generalized drivetrain model's (Eq. 19) response compared with the vehicle's response to a common ERPM reference input. Parameters identified using a genetic algorithm are $C_{m1} = 14.959$; $C_{m2} = 0.001$; $C_{m3} = 1.327$; $C_{m4} = 0.559$.



(a) ERPM reference response for $\tilde{d} = 0.4$.



(b) ERPM reference response for $\tilde{d} = 0.7$.

Figure 19: The first order drivetrain model 18 response compared with the vehicle's response to a common ERPM reference input. Parameters identified using a genetic algorithm are $C_{m1} = 4.097$; $C_{m2} = 0.237$; $C_{m3} = 0.392$.

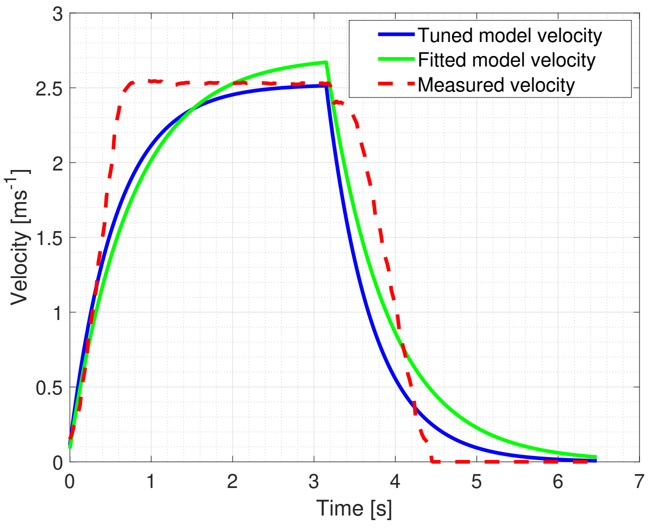
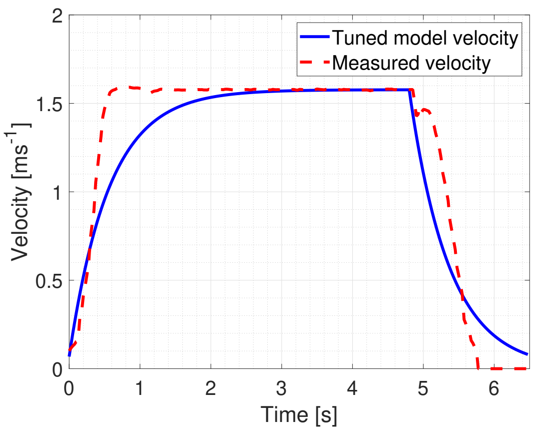
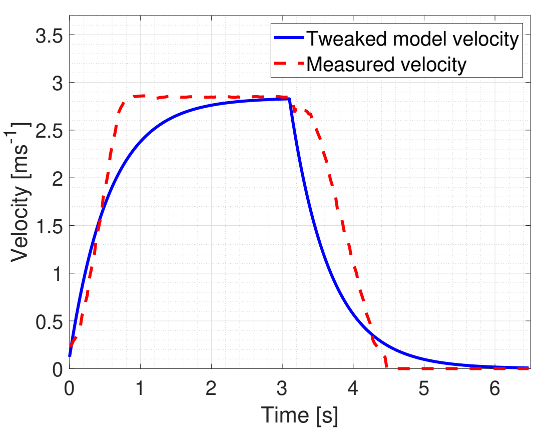


Figure 20: The manually tuned first order drivetrain model response compared to the fitted first order model with parameters from GA and the measured velocity for the ERPM reference input $\tilde{d} = 0.4$.



(a) ERPM reference response for $\tilde{d} = 0.25$.



(b) ERPM reference response for $\tilde{d} = 0.45$.

Figure 21: The manually tuned first order drivetrain model (Eq. 18) response compared to the vehicle's response to a common ERPM reference input from an independent, verification measurement set. Parameters identified using a genetic algorithm and manually tuned are $C_{m1} = 4.097$; $C_{m2} = 0.237$; $C_{m3} = 0.392$.

6.2.3 Tire model identification

The identification experiments for the tire model were performed as described in [50] by *Voser et al.* Since the utilized simplified Pacejka tire model is strictly empirical, the only performed experiment from [50] is a manually driven high side-slip maneuver.

Because the maneuver was manually performed, there is no possibility to use an ERPM reference as an input. Thus, the measured ERPM was converted to $\tilde{d} \in [0, 1]$ where 0 represents the minimum ERPM, and 1 represents the maximum ERPM.

The steering angle input δ was obtained using the recorded PWM commands from the transceiver unit. Since the experiment consisted of steering using the maximum steering angle δ_{max}, the change in the steering PWM command signaled the change from $\delta = 0$ to $\delta = 1$. After the steering PWM command returned to its original value, the steering angle δ was set to $\delta = 0$.

The recorded vehicle states of interest then include the longitudinal and lateral velocities v_x, v_y, the yaw rate ω and the side-slip angle α as reported by the `VehicleStatePublisher` (described in Section 7.2). The data is then modified in post-processing due to the high noise of the reported v_x, v_y of a stationary vehicle. The v_x, v_y are assumed to be zero when they are lower than $0.1\,\mathrm{m\,s^{-1}}$, and the parts of the recorded data where the vehicle was stationary were mostly cut out.

A genetic algorithm with the parameters described in Table 6 was then employed to fit the combined response of the vehicle model, identified in Section 6.2.1, the drivetrain model, identified in Section 6.2.2, and the Pacejka simplified tire model, as described in Section 3.3.4, using the Pacejka parameters B_a, C_a, D, where $a \in f, r$, f denotes the front and rear tires, respectively.

A single parameter D was used for both tires. This is because the drivetrain in a four-wheel driven vehicle applies the same forward force F_x to both the front and rear tires. Combined with the friction ellipse constraint, described in Section 3.5, the parameter D is used to limit the force the drivetrain can apply. By using two different parameters D_f, D_r, the drivetrain would be constrained by the smaller of those values, which is undesirable, as it would limit the drivetrain in a way that does not reflect the vehicle's behavior.

The combined model response with the identified parameters can be seen in Fig. 22. The model presents a significant understeering characteristic that is not present in the measured data, signified by a brief positive value of the side-slip angle at the beginning of the steering maneuver.

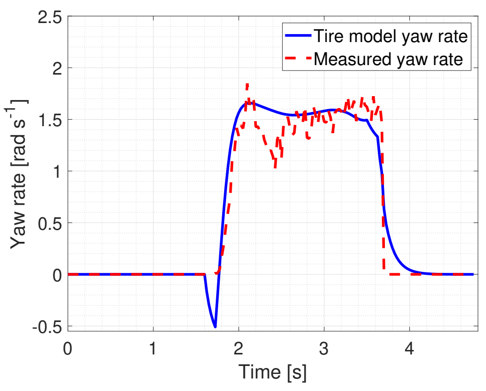
Therefore, another approach was utilized. Instead of fitting the measured side-slip angle α, as used in [50], the lateral velocity v_y was chosen as a reference for the genetic algorithm, together with the yaw rate ω. The response of the tire model identified using the lateral velocity can be seen in Fig. 23.

As the methods are very similar in nature, and differ only in a single utilized variable, to differentiate between them, the method seen in [50] is referred to as the side-slip approach, whereas the method using the lateral velocity is referred to as the lateral velocity approach.

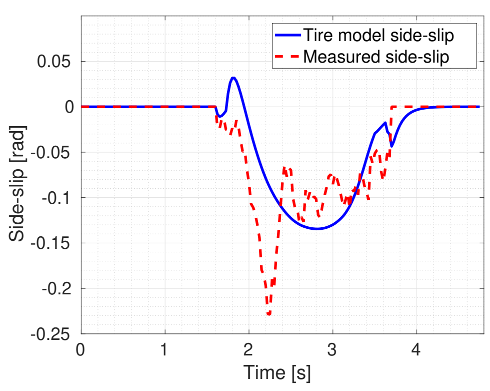
Because the parameter fitting using the side-slip angle α incorporates the longitudinal velocity v_x (since the α is obtained using $\alpha = \arctan2(v_y, v_x)$), the longitudinal velocity v_x simulated by the model with v_y fitted parameters is compared to the v_x from side-slip fitted and the measured data in Fig. 24. This is done to ensure that the lateral velocity identification method does not lower the accuracy of the identification compared to the one presented in [50].

It can be seen that the lateral velocity approach results in an identified model exhibiting less of the unwanted behavior, as seen in Fig. 24a, while the longitudinal velocity accuracy remains roughly the same, as seen in Fig. 24b.

The model obtained using the lateral velocity approach is then verified on an independent set of data to ensure that its behavior in comparison to the measured data is consistent. The verification results can be seen in Fig. 25.

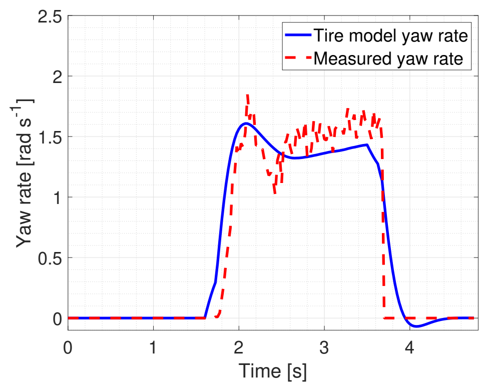


(a) The yaw rate ω of the tire model.

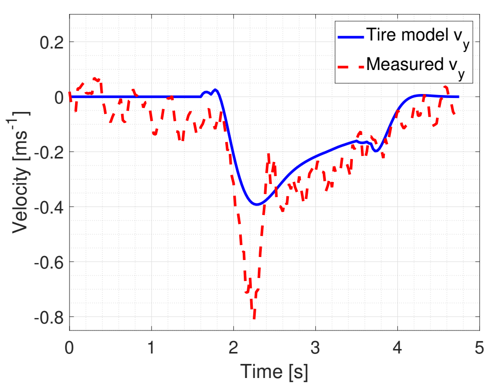


(b) The side-slip α of the tire model.

Figure 22: The response of the combined vehicle model identified using the side-slip angle α approach compared to the measured data in a high side-slip maneuver experiment. The identified parameters are $B_f = 10.603$; $C_f = 5.520$; $D_f = D_r = 0.475$; $B_r = 3.014$; $C_r = 3.413$.

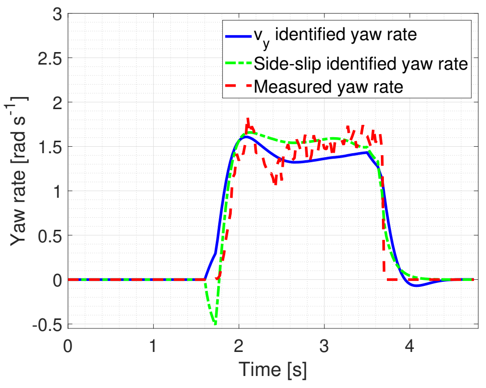


(a) The yaw rate ω of the tire model.

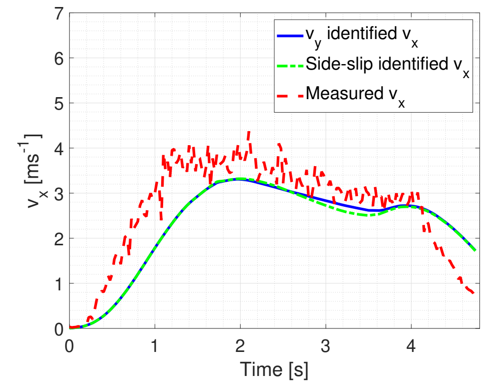


(b) The lateral velocity v_y of the tire model.

Figure 23: The response of the combined vehicle model identified using the lateral velocity v_y approach compared to the measured data in a high side-slip maneuver experiment. The identified parameters are $B_f = 0.711$; $C_f = 1.414$; $D_f = D_r = 0.892$; $B_r = 2.482$; $C_r = 1.343$.

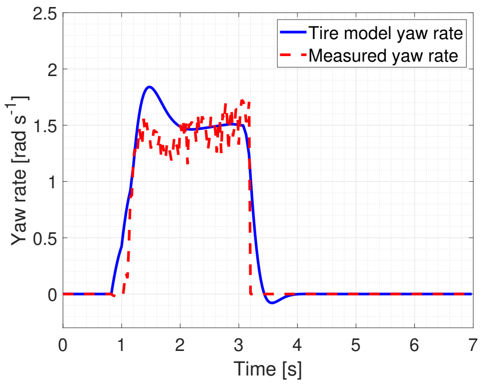


(a) The models' yaw rate ω compared to the measured.

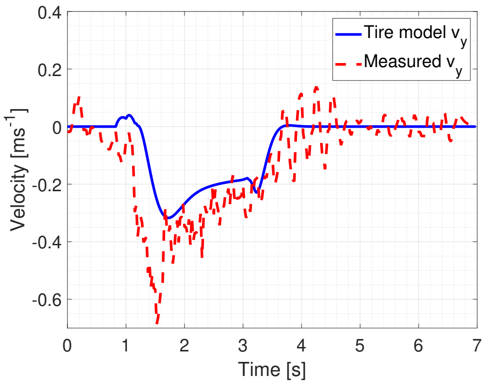


(b) The models' longitudinal velocity v_x compared to the measured.

Figure 24: The comparison of the tire models identified using the lateral velocity v_y and the side-slip angle α approach.



(a) The yaw rate ω of the tire model.



(b) The lateral velocity v_y of the tire model.

Figure 25: The response of the combined model identified using the lateral velocity approach compared to measured data in an independent high side-slip maneuver experiment.

7 MPCC integration

In order to use the MPCC as an online trajectory planner and tracker, it needs to be integrated into the existing F1/10 vehicle platform. This consists of the conversion of the occupancy grid-based track map into an MPCC-compatible format, which is described in Section 7.1, processing sensor data and providing them to the algorithm, the process of which is described in Section 7.2, and connecting the MPCC to the control interface of the F1/10 autonomous vehicle, as detailed in Section 7.3.

7.1 Map processing

The Cartographer SLAM provides a map in the form of a standardized ROS occupancy grid [45], which consists of an array of cells in row-major order with occupancy probability info and a structure containing the map's metadata, such as resolution, map width and height, and the origin point. The MPCC requires the map to be represented by a set of points forming the reference track's center line and corresponding closest inner and outer track border points, which correspond to the track's surrounding walls. It is necessary to first determine the center line of the track, as it is not provided, and then to extract the locations of the track's borders. This is done using the flooding algorithm as described in [13].

After extracting the necessary track information from the SLAM-provided occupancy grid, the center line points are sorted to correspond to the progression of the car on the racing track. Each center line point is then paired with the nearest inner and outer border points and their map coordinates X, Y are then stored each in a separate numerical array. These arrays are then stored in a *JSON* file under a corresponding key representing the set of points they are obtained from.

7.2 Sensor data processing

Since the MPCC algorithm requires data from multiple sources, a data aggregation and processing node was implemented, called `VehicleStatePublisher`. It collects the data from the Cartographer SLAM and VESC packages and periodically publishes them in a single ROS message, which allows for simple integration of control algorithms into the F1/10 platform. Furthermore, it is used to estimate information about the vehicle that is not measured by the vehicle's onboard sensors but is required for the MPCC algorithm.

A single iteration k of the `VehicleStatePublisher` node loads the transformation of the vehicle to the map provided by the Cartographer SLAM from which it extracts the vehicle's position $L_k = [X_k, Y_k]$. Afterward, it uses the information about the vehicle's position from the previous iteration L_{k-1} to calculate the average longitudinal and lateral velocities $v_k = [v_x, v_y]$ of the period between iterations. This is done by a double-rotation scheme in order to compensate for possible steering during the time period, where the position L_k is rotated about L_{k-1} by the difference in the yaw angle $\Delta\varphi = \varphi_k - \varphi_{k-1}$, generating the steering-compensated L_k^s. This is described by the following equation:

$$L_k^s = (L_k - L_{k-1}) \cdot R(\Delta\varphi) + L_{k-1}, \tag{44}$$

where $R(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$ is the rotation matrix. Both L_k^s, L_{k-1} are then rotated about the map's origin by the yaw angle φ_{k-1}:

$$\begin{bmatrix} L_k^c \\ L_{k-1}^c \end{bmatrix} = \begin{bmatrix} L_k^s \\ L_{k-1} \end{bmatrix} \cdot R(\varphi_{k-1}), \tag{45}$$

where L_k^c is the yaw-compensated position, and the velocities v_k are then simply calculated as:

$$v_k = \frac{(L_k^c - L_{k-1}^c)}{T_i}, \tag{46}$$

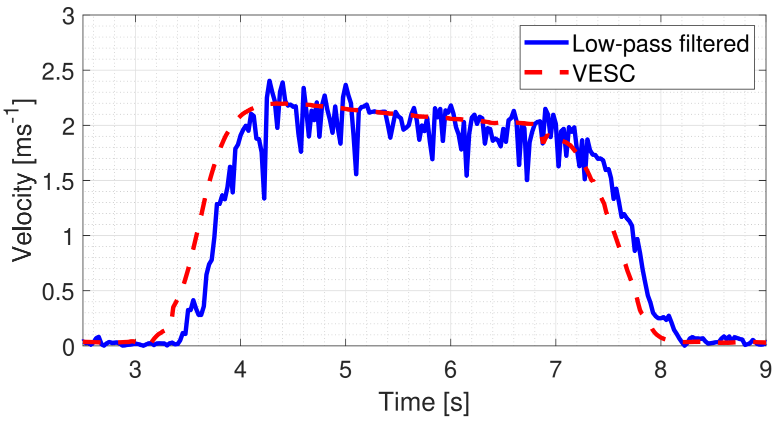
where $T_i = 0.025\text{s}$ is the period of one iteration of the `VehicleStatePublisher` loop.

Due to the measurement noise of the Cartographer SLAM localization (Section 5.1.1), the estimated velocities v_k are processed using a low-pass filter. We selected a first order infinite impulse response low-pass filter for its low computational requirements. The low-pass filter is described by the equation:

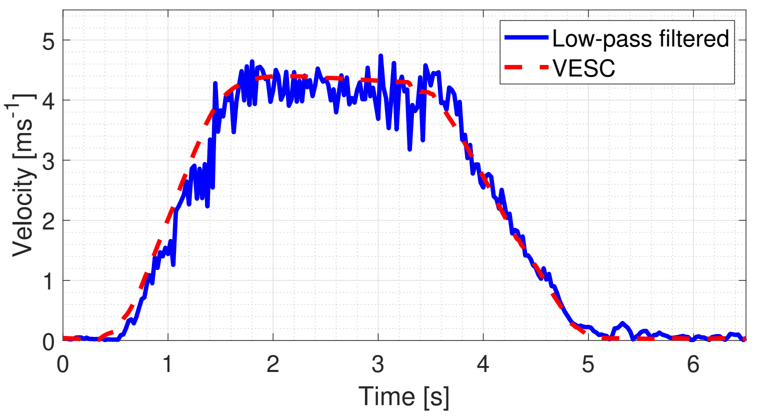
$$\hat{v}_k = \hat{v}_{k-1} + \alpha(v_k - \hat{v}_{k-1}), \tag{47}$$

where $\hat{v}_k$, v_k represent the filtered and unfiltered estimated velocities for the iteration k and α is the parameter of the low-pass filter. Its performance was evaluated using a comparison to the velocity calculated from ERPM as reported by VESC during an acceleration experiment under the assumption that no longitudinal slip occurred, and can be seen in Figs. 26a and 26b for different values of the reference ERPM input.

The last vehicle state variable that is estimated is the yaw rate ω, which is obtained as the rate of change of the measured yaw angle ϕ during the iteration period.



(a) Experiment with ERPM reference $d = 0.3$.



(b) Experiment with ERPM reference $d = 0.65$.

Figure 26: The performance of the low-pass filter described in Eq. 47 in an acceleration experiment with zero longitudinal slip.

7.3 MPCC node

For the purposes of the MPCC algorithm integration, the MPCC implementation released at [28] under the Apache 2.0 license was used and modified in order to fit this thesis's purposes. The MPCC was first modified to be buildable using the `catkin_make` tool by creating the package manifest `package.xml` listing the desired package name and package dependencies and modifying the `CMakeLists.txt` to link the required ROS packages, as is the standard for ROS.

The initialization of the algorithm consists of loading the vehicle, track and optimization parameters from the respective *JSON* configuration files. Afterward, the track is parametrized, as described in Section 4.1, and the HPIPM solver is initialized and set up using the specified solver parameters. The node subscribes to the `/vehicle_state` topic, and sets up a `/drive_api/command` publisher. Afterward, the x_0 vehicle initial state object is formed from the vehicle's initial position on the track and the initial nonzero velocity $v_{x0} = 0.1$, required by the utilized vehicle model. All other states are set to 0.

Algorithm 1 MPCC Main Loop

```
1: $\mathbf{x} = \mathbf{x_0}$
2: while running do
3:     Obtain $\mathbf{x_m} = [X, Y, v_x, v_y, \varphi, \omega]$ from VehicleState topic
4:     Update $\mathbf{x}$ with $\mathbf{x_m}$
5:     $\mathbf{u}$ = SolveMPCC(x)
6:     $\mathbf{x}$ = SimulateModel($\mathbf{u}$)
7:     Publish $[x.d, x.\delta]$ to drive_api/command topic
8: end while
```

The main loop of the algorithm is described in Algorithm 1. In each iteration i, the algorithm loads the latest measured state variables x_m, as provided by the `vehicle_state` subscriber, and replaces the respective vehicle state variables x where applicable. Since some of the vehicle states in x are virtual and provided by the controller (for example the virtual velocity v_s), they are kept from the previous iteration $i-1$. The MPCC problem is then solved for the vehicle state x, providing the sequence of N optimal inputs u, where N is the time horizon of the MPCC problem. The inputs u are used to run a simulation of the vehicle model, solved using the Runge-Kutta 4th order method, providing a sequence of N states x_{sim}. The state x is then declared as $x = x_{sim}[0]$, where the $x_{sim}[0]$ is the first of the sequence of N states.

The desired vehicle control input is then parsed as $D =$ `x.D`, $\delta =$ `x.delta` and processed in order to create a `command_msg` message defined by `drive_api_values.msg`. The `command_msg` message has four parameters: `velocity`, `forward`, `steering` and `right`. The `velocity` $\in < 0; 1 >$ denotes the ERPM (3.4) reference, where 1 is maximum and 0 is minimum, and is set to $x0.d$, The `forward` is a logical parameter signifying whether the vehicle should drive forward or backward, and is always set to `true` for the purposes of the MPCC control. The pair of parameters `steering`, `right` work similarly: `steering` $\in < 0; 1 >$ is the steering PWM reference, where 1 corresponds to the maximum value of PWM for the motor responsible for steering, and 0 to the minimum value. It's value is set as `steering` $= |\frac{\delta}{\delta_{max}}|$. Finally, the parameter `right`, containing a logical value specifying whether the vehicle should turn right or left, is set as:

$$\texttt{right} = \begin{cases} \texttt{true} & \text{if } \texttt{x0.delta} \leq 0, \\ \texttt{false} & \text{if } \texttt{x0.delta} > 0. \end{cases}$$

This message is then published using the `drive_api/command` publisher. While the MPCC controller provides the optimal input sequence u, it cannot be used to control the vehicle. This is because the controller-computed inputs are in the form of input derivatives, which the vehicle cannot process. Using the input derivatives in the controller is beneficial, as it enables to limit the rate of change of the inputs, thus preventing large oscillations in the d and δ states, that could result in an unwanted vehicle behavior that is not properly modelled.

The main loop of the algorithm is repeated with the iteration period T_i that is experimentally determined. The loop also includes two secondary publishers that are used to send the computed horizon of N states and the reference trajectory to their respective topics, `mpcc_horizon` and `mpcc_reference_trajectory`, which can be used for visualization and debugging purposes.

8 Experimental Results

First, the computational performance of the MPCC algorithm is analyzed to determine the sampling period provided to the algorithm. This is done in Section 8.1.

The implemented control algorithm together with the identified model is first tested in a simulation and its performance is reviewed. The results of the simulation are described in Section 8.2.

After a successful simulation, the control algorithm is deployed on the vehicle platform and its performance in a real-life experimental scenario is evaluated. This is described in Section 8.3.

8.1 Computational performance

The test of the computational performance consists of multiple runs of the algorithm and the measurement of it's mean computation time. Each run consists of an autonomous MPCC-controlled ride of the vehicle for the duration of 2000 algorithm iterations. This is used to set the sampling ratio for the optimization the algorithm utilizes.

The results are shown in Table 7. Even though the mean iteration time suggests setting the iteration period to $T_i = 0.050\,\mathrm{s}$, it is safer to set it as $T_i = 0.065\,\mathrm{s}$, so that in cases when the single iteration takes as long as the maximal measured iteration duration, there is still some overhead that would reduce the impact of delayed control computation. This would be insufficient for measurement 4, but the noticeably higher mean iteration time suggests that it has been an anomalous measurement.

Measurement	Max iteration time [s]	Mean iteration time [s]
1	0.068	0.047
2	0.066	0.044
3	0.069	0.042
4	0.086	0.062
5	0.069	0.045

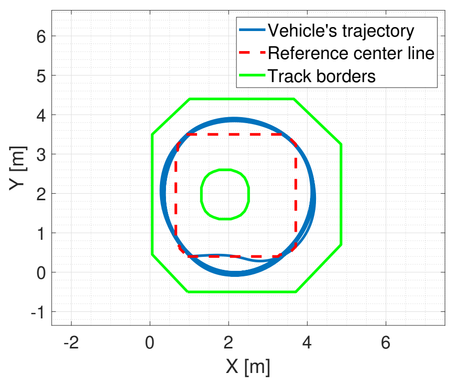
Table 7: Maximal and mean iteration times of the MPCC control algorithm from 2000 iterations.

8.2 Simulation results

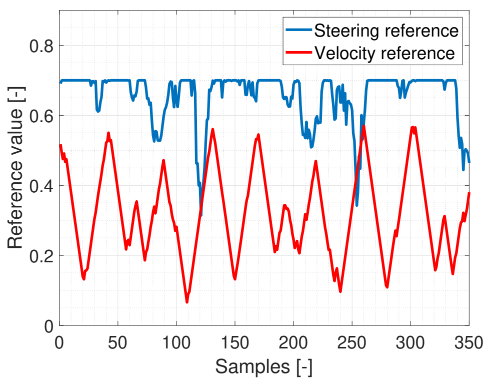
The implemented MPCC control algorithm was first tested in a simulation using the identified vehicle model on the racing track used for the experimental verification. The simulated trajectory and the computed control inputs can be seen in Fig. 27.

The optimization behavior of the algorithm is clearly seen in Fig. 27a, where the simulated vehicle deviates from the reference center line in the benefit of a stabler circular trajectory, allowing the simulated vehicle to retain it's velocity.

The computed control inputs, as seen in Fig. 27b, feature a somewhat steady leftward steering with a varying velocity reference.



(a) Simulated vehicle trajectory.



(b) Computed control signals.

Figure 27: The resulting vehicle trajectory and the computed control inputs from a simulation scenario.

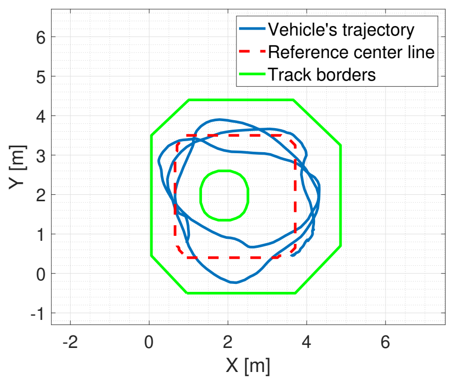
8.3 Real-life experimental results

The implemented MPCC control algorithm was then tested in an experimental scenario on the same racing track with the same parameters. The real-life trajectory and the computed control inputs can be seen in Fig. 28.

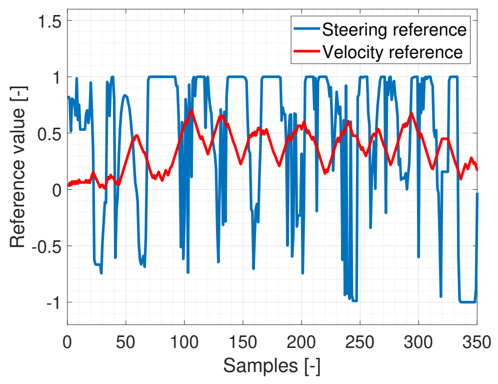
The vehicle's trajectory, as seen in Fig. 28a, shows that the MPCC algorithm tried to optimize the trajectory by deviating from the reference center line. However, the trajectory is not consistent compared to the simulated trajectory. Furthermore, the control inputs, seen in Fig. 28b, feature repeated segments of maximal leftward steering, followed by a corrective rightward steering input.

To analyze the reasons for this behavior, the longitudinal velocity v_x and the yaw rate ω from the simulated and real-life experiments are compared in Fig. 29. The simulated vehicle variables in Fig. 29a show that the peaks of the yaw rate ω were significantly lower than in the real-life experiment, as shown in Fig. 29b. Furthermore, the longitudinal velocity v_x was more stable in the simulation.

This result suggests that the execution frequency f = 15Hz of the MPCC control algorithm was not high enough to mitigate the identified model's inaccuracies. The computed control inputs then resulted in a response different from the response of the simulated model, which caused the steering correction and a general oscillatory behavior.

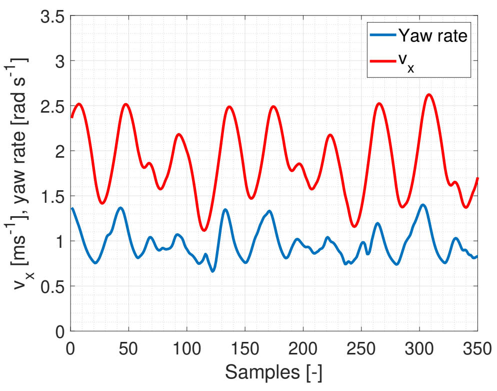


(a) Vehicle trajectory from the real-life experiment.

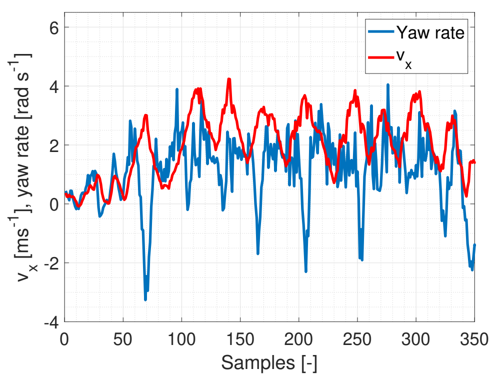


(b) Computed control signals.

Figure 28: The resulting vehicle trajectory and the computed control inputs from the real-life experimental scenario.



(a) Simulated vehicle variables.



(b) Measured vehicle variables from the real-life experiment.

Figure 29: The resulting vehicle trajectory and the computed control inputs from the real-life experimental scenario.

9 Conclusion

In this thesis, we have selected a dynamic single-track vehicle model for the 1/10 scale autonomous racing vehicle. The vehicle's tires were modeled using the Pacejka magic formula, and the drivetrain using a generalized first-order model. We identified the models using a set of real-life experiments and a genetic algorithm and verified their performance on an independent set of real-life experiments.

We selected the optimization-based control algorithm *Model Predictive Contouring Control* (MPCC) as the solution of the minimum lap-time problem for the *F1/10 Autonomous Racing Competition*. Together with the vehicle, drivetrain and tire models, we have integrated the MPCC in the autonomous racing vehicle platform. We have then evaluated the performance of the MPCC controller in a simulation and real-life experiments.

The MPCC managed to optimize the trajectory of a simulated racing vehicle through a virtual representation of a real-life testing track. In the real-life experiments with the F1/10 vehicle, the MPCC managed to drive the vehicle safely through the testing track. However, it displayed problematic behavior attributable to the combination of the low computational power of the vehicle's onboard computer and the slower dynamics of the utilized drivetrain model.

9.1 Future work

9.1.1 Improved drivetrain model

The drivetrain model utilized in this thesis has slower dynamics than the vehicle's drivetrain, which causes prediction errors in the control horizon. The future work is to develop a more accurate drivetrain model, which would improve the control performance of the MPCC algorithm.

9.1.2 Dynamic obstacle avoidance

The utilized control algorithm requires a full prior knowledge of the racing track. Since the *F1/10 Autonomous Racing Competition* occasionally features dynamic obstacles, it would be beneficial to extend the MPCC algorithm for dynamic obstacle avoidance. Such feature was already implemented in the MATLAB version of the MPCC algorithm in [28]. The future work is to implement a ROS-compatible online modification of the track constraints for obstacle avoidance.

9.1.3 Sub-map planning

The MPCC control algorithm optimizes the vehicle's trajectory only in a relatively short control horizon. The future work is to research and develop the option to supply the control algorithm with partial track segments, which would allow for optimization-based control without full prior track knowledge.

References

[1] MATLAB global optimization toolbox. `https://www.mathworks.com/help/gads/`. Accessed: 2020-07-21.

[2] SkyRC corner weight system. `https://www.skyrc.com/Corner_Weight_System`. Accessed: 2020-08-05.

[3] VICON localization system. `https://www.vicon.com/`. Accessed: 2020-08-05.

[4] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[5] A Pedro Aguiar, João P Hespanha, and Petar V Kokotović. Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, 44(3):598–610, 2008.

[6] Matthias Althoff. Commonroad: Vehicle models. *Technische niversität München, Garching*, pages 1–25, 2017.

[7] Egbert Bakker, Lars Nyborg, and Hans B Pacejka. Tyre modelling for use in vehicle dynamics studies. *SAE Transactions*, pages 190–204, 1987.

[8] Karl Berntorp, Björn Olofsson, Kristoffer Lundahl, and Lars Nielsen. Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres. *Vehicle System Dynamics*, 52(10):1304–1332, 2014.

[9] Raymond Brach and Matthew Brach. The tire-force ellipse (friction ellipse) and tire characteristics. Technical report, SAE Technical Paper, 2011.

[10] DL Brayshaw and MF Harrison. Use of numerical optimization to determine the effect of the roll stiffness distribution on race car performance. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 219(10):1141–1151, 2005.

[11] Jianfeng Chen, Congcong Guo, Shulin Hu, Jiantian Sun, Reza Langari, and Chuanye Tang. Robust estimation of vehicle motion states utilizing an extended set-membership filter. *Applied Sciences*, 10(4):1343, 2020.

[12] Vittore Cossalter, Mauro Da Lio, Roberto Lot, and Lucca Fabbri. A general method for the evaluation of vehicle manoeuvrability with special emphasis on motorcycles. *Vehicle system dynamics*, 31(2):113–135, 1999.

[13] Kopecký David. Lokalizace a pokročilé řízení autonomního modelu vozidla. 2019.

[14] Howard Dugoff. Tire performance characteristics affecting vehicle response to steering and braking control inputs. Final report. Technical report, 1969.

[15] Howard Dugoff, PS Fancher, and Leonard Segel. An analysis of tire traction properties and their influence on vehicle dynamic performance. *SAE transactions*, pages 1219–1243, 1970.

[16] Jan Dusil. Slip detection for F1/10 model car. 2019.

[17] Paolo Falcone, Francesco Borrelli, H Eric Tseng, Jahan Asgari, and Davor Hrovat. A hierarchical model predictive control framework for autonomous ground vehicles. In *2008 American Control Conference*, pages 3719–3724. IEEE, 2008.

[18] Timm Faulwasser, Benjamin Kern, and Rolf Findeisen. Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8642–8647. IEEE, 2009.

[19] E Fiala. Seitenkrafte am rollenden luftreifen, 1954.

[20] Gianluca Frison and Moritz Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. *arXiv preprint arXiv:2003.02547*, 2020.

[21] Joseph Funke, Matthew Brown, Stephen M Erlien, and J Christian Gerdes. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. *IEEE Transactions on Control Systems Technology*, 25(4):1204–1216, 2016.

[22] Benjamin Gutjahr, Lutz Gröll, and Moritz Werling. Lateral vehicle trajectory optimization using constrained linear time-varying mpc. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1586–1595, 2016.

[23] Ryuzo Hayashi, Juzo Isogai, Pongsathorn Raksincharoensak, and Masao Nagai. Autonomous collision avoidance system by combined control of steering and braking using geometrically optimised vehicular trajectory. *Vehicle system dynamics*, 50(sup1):151–168, 2012.

[24] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[25] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *arXiv preprint arXiv:1905.05150*, 2019.

[26] Moad Kissai, Bruno Monsuez, Adriana Tapus, and Didier Martinez. A new linear tire model with varying parameters. In *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 108–115. IEEE, 2017.

[27] Denise Lam, Chris Manzie, and Malcolm Good. Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6137–6142. IEEE, 2010.

[28] Alex Liniger. MPCC. `https://github.com/alexliniger/MPCC`, 2020.

[29] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[30] Peng Liu and Ümit Özgüner. Predictive control of a vehicle convoy considering lane change behavior of the preceding vehicle. In *2015 American Control Conference (ACC)*, pages 4374–4379. IEEE, 2015.

[31] Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur, and Xavier Savatier. A study of vicon system positioning performance. *Sensors*, 17(7):1591, 2017.

[32] D Metz and D Williams. Near time-optimal control of racing vehicles. *Automatica*, 25(6):841–857, 1989.

[33] NVIDIA. NVIDIA developer blog. `https://developer.nvidia.com/blog/jetson-tx2-delivers-twice-intelligence-edge/`. Accessed: 2020-08-07.

[34] University of Pennsylvania. F1/10 autonomous racing competition. `https://f1tenth.org/`. Accessed: 2020-08-08.

[35] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992.

[36] HB Pacejka and IJM Besselink. Magic formula tyre model with transient properties. *Vehicle system dynamics*, 27(S1):234–249, 1997.

[37] Giacomo Perantoni and David JN Limebeer. Optimal control for a formula one car with variable parameters. *Vehicle System Dynamics*, 52(5):653–678, 2014.

[38] Sergio M Savaresi, Cristiano Spelta, Davide Ciotti, Marco Sofia, Enrico Rosignoli, and Emiliano Bina. Virtual selection of the optimal gear-set in a race car. *International Journal of Vehicle Systems Modelling and Testing*, 3(1-2):47–67, 2008.

[39] H Scherenberg. Mercedes-benz racing cars—design and experience. *SAE Transactions*, pages 414–420, 1958.

[40] Volkan Sezer and Metin Gokasan. A novel obstacle avoidance algorithm: "follow the gap method". *Robotics and Autonomous Systems*, 60(9):1123–1134, 2012.

[41] Zvi Shiller and Satish Sundar. Emergency lane-change maneuvers of autonomous vehicles. 1998.

[42] Bobbo Simon, Cossalter Vittore, Massaro Matteo, and Peretto Martino. Application of the "optimal maneuver method" for enhancing racing motorcycle performance. *SAE International Journal of Passenger Cars-Mechanical Systems*, 1(2008-01-2965):1311–1318, 2008.

[43] Stanford Artificial Intelligence Laboratory. Robotic operating system. `https://www.ros.org`.

[44] Stanford Artificial Intelligence Laboratory. Robotic operating system nodes. `http://wiki.ros.org/Nodes`. Accessed: 2020-08-11.

[45] Stanford Artificial Intelligence Laboratory et al. Occupancy grid documentation. `http://docs.ros.org/melodic/api/nav_msgs/html/msg/OccupancyGrid.html`. Accessed: 2020-08-07.

[46] John K Subosits and J Christian Gerdes. From the racetrack to the road: Real-time trajectory replanning for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 4(2):309–320, 2019.

[47] University of Pennsylvania. F1Tenth build documentation. `https://f1tenth.org/build.html`. Accessed: 2020-08-07.

[48] Martin Vajnar. Model formule pro soutěž autonomních aut F1/10. 2017.

[49] Robin Verschueren, Stijn De Bruyne, Mario Zanon, Janick V Frasch, and Moritz Diehl. Towards time-optimal race car driving using nonlinear mpc in real-time. In *53rd IEEE conference on decision and control*, pages 2505–2510. IEEE, 2014.

[50] Christoph Voser, Rami Y Hindiyeh, and J Christian Gerdes. Analysis and control of high sideslip manoeuvres. *Vehicle System Dynamics*, 48(S1):317–336, 2010.

[51] Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for bertha—a local, continuous method. In *2014 IEEE intelligent vehicles symposium proceedings*, pages 450–457. IEEE, 2014.

Appendix A CD Content

In Table 8 are listed names of all root directories on CD.

Directory name	Description
thesis	the thesis in pdf format
thesis_sources	latex source codes
rosbags	rosbags from experimental measurements
src	ros packages and matlab scripts containing implemented code

Table 8: CD Content

Appendix B List of abbreviations

In Table 9 are listed abbreviations used in this thesis.

Abbreviation	**Meaning**
MLT	Minimum Lap-Time
QSS	Quasi-Steady State
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
ERPM	Engine Revolutions Per Minute
LCQP	Local Convex Quadratic Programming
NLP	Non-linear Programming
HPIPM	High-Performance Interior-Point Method
QP	Quadratic Programming
LIDAR	Light Detection and Ranging
IMU	Inertial Measurement Unit
VESC	Vedder Electronic Speed Controller
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
GA	Genetic Algorithm
PWM	Pulse Width Modulation
JSON	JavaScript Object Notation

Table 9: Lists of abbreviations

Appendix C List of parameters

In Table 10 are listed the various vehicle parameters used in this thesis.

Parameter	**Meaning**
X	Position in the X axis [m]
Y	Position in the Y axis [m]
v	Longitudinal velocity [$\mathrm{m\,s^{-1}}$]
v_x	Longitudinal velocity, or velocity in the X axis [$\mathrm{m\,s^{-1}}$]
v_y	Lateral velocity, or velocity in the Y axis [$\mathrm{m\,s^{-1}}$]
a_x	Longitudinal acceleration, or acceleration in the X axis [$\mathrm{m\,s^{-2}}$]
a_y	Lateral acceleration, or acceleration in the Y axis [$\mathrm{m\,s^{-2}}$]
δ	Vehicle's steering angle [rad]
φ	Vehicle's yaw (heading) [rad]
ω	Vehicle's yaw rate [$\mathrm{rad\,s^{-1}}$]
α	Vehicle's side-slip angle [rad]
l_{wb}	Length of the vehicle's wheelbase [m]
l_f	Distance from the front axle to the vehicle's center of gravity [m]
l_r	Distance from the rear axle to the vehicle's center of gravity [m]
m	Vehicle's mass [kg]
c_g	Vehicle's center of gravity [–]
t_r	Length of the vehicle's front axle [m]
t_f	Length of the vehicle's rear axle [m]
$F_{f,l,x}$	Longitudinal force acting on front left wheel [N]
$F_{r,r,y}$	Lateral force acting on rear right wheel [N]
$F_{f,x}$	Longitudinal force acting on front wheel [N]
$F_{r,y}$	Lateral force acting on rear wheel [N]
t_f	Length of the vehicle's rear axle [m]
t_f	Length of the vehicle's rear axle [m]
I_z	The moment of inertia of the z axis [$\mathrm{kg\,m^2}$]
$m_{f,l}$	Mass on the front left wheel [kg]
$F_{r,r}$	Mass on the rear right wheel [kg]

Table 10: List of parameters