KRISTÓF ZOLTÁN FLOCH
BSc THESIS

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS

FACULTY OF MECHANICAL ENGINEERING

DEPARTMENT OF MECHATRONICS, OPTICS AND MECHANICAL ENGINEERING INFORMATICS

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS
FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF MECHATRONICS, OPTICS AND MECHANICAL ENGINEERING INFORMATICS

INSTITUTE FOR COMPUTER SCIENCE AND CONTROL
SYSTEMS AND CONTROL LABORATORY

Kristóf Zoltán FLOCH
BSc Thesis

Model-based motion control of the F1TENTH autonomous electrical vehicle

Consultant:
Dr. Tamás PÉNI
Senior research fellow
SZTAKI Systems and Control Lab

Supervisor:
Dr. Csaba BUDAI
Assistant professor
BME – GPK MOGI

Advisor:
Dr. Roland TÓTH
Senior research fellow / Associate professor
SZTAKI Systems and Control Lab
Eindhoven University of Technology

Budapest, 2022

Copyright © Kristóf Zoltán FLOCH, 2022.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Gépészmérnöki Kar
Mechatronika, Optika és Gépészeti Informatika Tanszék
https://mogi.bme.hu

SZAKDOLGOZAT-FELADAT

NYILVÁNOS

AZONOSÍTÁS			
Név: **Floch Kristóf Zoltán**		Azonosító: **73917317089**	
Képzéskód: 2N-AM0	Specializáció kódja: 2N-AM0-OT-2017	Feladatkiírás azonosítója: GEMI:2023-1:2N-AM0:I8SNRG	
Szak: Mechatronikai mérnöki alapszak (BSc)			
Szakdolgozatot kiadó tanszék: Mechatronika, Optika és Gépészeti Informatika Tanszék	Záróvizsgát szervező tanszék: Mechatronika, Optika és Gépészeti Informatika Tanszék		
Témavezető: Dr. Budai Csaba (73554263569), adjunktus			

FELADAT	
Cím	**F1TENTH autonóm elektromos autó modell alapú mozgásszabályozása** Model-based motion control of the F1TENTH autonomous electrical vehicle
Részletes feladatok	Készítsen irodalmi összefoglalót az autonóm járművek dinamikus modellezéséről, azok modell alapú szabályozásáról! Ismertesse a dolgozatban használt F1TENTH járműplatformot és a hozzá kapcsolódó kísérleti környezetet! Alkossa meg az F1TENTH autó dinamikus modelljét és kísérletek segítségével becsülje meg a modellparamétereket! Készítsen alacsony szintű mozgásszabályozó algoritmusokat a pályakövetés megvalósítására, majd elemezze ezeket szimulációs környezetben! Implementálja az elkészült algoritmusokat az F1TENTH járműplatformon és elemezze az elvégzett kísérletek eredményét! Foglalja össze eredményeit magyar és angol nyelven!
Hely	A szakdolgozat készítés helye: Számítástechnikai és Automatizálási Kutatóintézet 1111 Budapest, Kende utca 13-17. Konzulens: Dr. Péni Tamás, tudományos főmunkatárs

ZÁRÓVIZSGA		
1. záróvizsga tantárgy(csoport)	2. záróvizsga tantárgy(csoport)	3. záróvizsga tantárgy(csoport)
ZVEGEMIBMIE Irányításelmélet	**ZVEGEMIBMMT** Mechanikai tervezés	**ZVEVIAUA040** Elektronikai rendszerek tervezése

HITELESÍTÉS		
Feladat kiadása: 2022. szeptember 5.	Beadási határidő: 2022. december 9.	
Összeállította: Dr. Budai Csaba (73554263569) témavezető	Ellenőrizte: *Dr. Kiss Rita Mária* s.k. tanszékvezető	Jóváhagyta: *Dr. Györke Gábor* s.k. dékánhelyettes
Alulírott, a feladatkiírás átvételével egyúttal kijelentem, hogy a Szakdolgozat-készítés c. tantárgy előkövetelményeit maradéktalanul teljesítettem. Tudomásul veszem, hogy jogosulatlan tantárgyfelvétel esetén a jelen feladatkiírás hatálytalan. [illegible] .... *Floch Kristóf Zoltán*		

DECLARATION OF INDIVIDUAL WORK

I, *Kristóf Zoltán Floch* (I8SNRG), the undersigned, student of the Budapest University of Technology and Economics hereby declare that the present thesis has been prepared by myself without any unauthorized help or assistance such that only the specified sources (references, tools, etc.) were used. All parts taken from other sources word by word or after rephrasing but with identical meaning were unambiguously identified with explicit reference to the sources utilized.

Budapest, 2022

Kristóf Zoltán Floch

Contents

Abstract

F1TENTH is a 1/10 scale, electric model of a real car. The aim of its developers was to create an open-source vehicle platform that can support autonomous systems research and education. The main objective of this work is the development of the nonlinear, dynamic model of the F1TENTH vehicle. Then, with the help of the obtained model, accurate path-following control algorithms are designed that are capable of agile maneuvering.

First, the dynamic model of the F1TENTH platform is derived. This model can be split into three main part. The nonlinear vehicle dynamics describes the general behaviour of the race car. The drivetrain model expresses the connection between the motor control input and the force acting on the wheels. Finally, the tire model incorporates the lateral behaviour of the wheels. After the introduction of the model structure, the parameters are obtained. While some of these can be directly measured, there are also empirical parameters that can only be estimated from measurements. Here the design process and execution steps of the identification experiments and the data acquisition is also outlined. Then, the collected data is processed and the model parameters are calculated, using optimization algorithms. The resulting dynamic model is validated by comparing it with the observed behavior the real car.

The next part of the work describes the design of model-based trajectory-tracking control algorithms. First, the nonlinear dynamics are decoupled into lateral and longitudinal subsystems. After further simplifications, individual lateral and longitudinal controllers are developed for both of the subsystems, via full state feedback. The robustness and performance of the designed controllers are then examined in both numerical simulations as well as in real world, implemented on the F1TENTH platform.

In addition to the modelling and control, the work presented in this paper also required a software framework for the management of the F1TENTH platform. This framework is responsible for the operation of onboard and external sensors, the data acquisition and the control of the vehicle actuators. The main advantage of this software is its Python interface, that provides an easy solution for the high-level control and the management of the onboard – ROS based – software stack of the vehicle. Furthermore, the framework is also capable of managing multiple vehicles simultaneously, so a group of cars can also be controlled. As a result to this development, a vehicle test environment based on the F1TENTH platform is designed, which can provide the fundamentals for further research projects related to autonomous mobile robots.

Acknowledgement

I would like to thank *Tamás Péni* and *Roland Tóth* at SZTAKI for their continuous support during my work. I would also like to thank *Csaba Budai* at BME Department of Mechatronics, Optics and Mechanical Engineering Informatics for his help and supervision.

The research was supported by the European Union within the framework of the National Laboratory for Autonomous Systems. (RRF-2.3.1-21-2022-00002)

Budapest, 2022

Kristóf Zoltán Floch

Notations

The table contains the names of the notations that occur several times, and in the case of physical quantities, its unit of measurement. The designation of each quantity is, where possible, the same as that accepted in the domestic and international literature. An explanation of the rarely used notations can be found at their first location.

Latin letters

Notation	Name, comment, value	Dimension
a	acceleration	$\mathrm{m\ s^{-2}}$
c_c	path curvature	$\mathrm{m^{-1}}$
d	motor reference input	1
e_η	lateral orthogonal error	m
l	wheelbase length	m
l_f	front axle distance from center of mass	m
l_r	rear axle distance from center of mass	m
m	mass	kg
n_u	number of system inputs	
n_x	number of system states	
s	curvilinear abscissa of the path	m
s_1	longitudinal position in the moving coordinate frame	m
t	time	s
u	system input vector	
v	absolute velocity	$\mathrm{m\ s^{-1}}$
v_η	lateral velocity	$\mathrm{m\ s^{-1}}$
v_ξ	longitudinal velocity	$\mathrm{m\ s^{-1}}$
x	system state vector	
z_1	lateral position in the moving coordinate frame	m
A	system matrix	
B	input matrix	
C_f	front wheel cornering stiffness	$\mathrm{N\ rad^{-1}}$
C_r	rear wheel cornering stiffness	$\mathrm{N\ rad^{-1}}$
$C_{\mathrm{m}\{1,2,3\}}$	drivetrain parameters	N, $\mathrm{Ns\ m^{-1}}$, N
E	disturbance matrix	
I_z	inertia around the Z axis	$\mathrm{kg\ m^2}$
$F_{\mathrm{f},\eta}$	lateral tire force of the front wheel	N
$F_{\mathrm{f},\xi}$	longitudinal tire force of the front wheel	N
$F_{\mathrm{r},\eta}$	lateral tire force of the rear wheel	N
$F_{\mathrm{r},\xi}$	longitudinal tire force of the rear wheel	N

Notation	Name, comment, value	Dimension
K	feedback matrix	
Q	LQR state weighting matrix	
R	LQR input weighting matrix	
X	position in the X axis	m
Y	position in the Y axis	m

Greek letters

Notation	Name, comment, value	Dimension
α_f	front tire side-slip angle	rad
α_r	rear tire side-slip angle	rad
β	center of mass side-slip angle	rad
δ	steering angle	rad
ϵ	disturbance vector	
θ_e	heading error	rad
θ_p	rotation angle of the moving coordinate frame	rad
φ	heading angle	rad
ρ	general scheduling variable	
ω	yaw rate	rad s^{-1}

Indices, exponents

Notation	Name, comment, value
i	general running index (integer)
n	general quantity (integer)
k	discrete system time step index (integer)
p	path property
max	maximum
min	minimum
ref	reference

1 Introduction

Nowadays, autonomous systems are becoming increasingly important. As a result of the continuous development, intelligent mobile robots such as drones or ground vehicles are getting deployed in many different areas of industry. From transportation and logistics to manufacturing and production support, there are a growing number of applications.

In order to unlock the full potential of these autonomous systems, efficient algorithms are required that can solve problems related to mobile robot navigation and control. To aid the research and development, there has been a growing interest in small scale vehicle platforms on which such algorithms can be properly evaluated. At SZATKI, an autonomous vehicle test area, called *AIMotionLab*, has been developed for the research of algorithms related to cooperative navigation, path planning and precise trajectory tracking. While small scale quadcopters have already been integrated into the environment, autonomous ground vehicle research has just started. This motivates the main contributions of this work: the development of well applicable modelling and identification techniques for these types of small scale vehicles, and the design of motion control solutions that are capable of agile maneuvering.

In the relevant literature, many different vehicle platforms can be found. In [19], 1/43 scale RC cars were used for the evaluation of path planning and path following algorithms. Another vehicle, namely a 1/24 scale mobile robot was used in [2], to test localization algorithms. MIT has also developed its own platform, called racecar [22], which is a 1/10 scale autonomous racing car for educational purposes. Taking inspiration from the MIT racecar, [29] presents MuSHR, an alternative vehicle platform, with cheaper hardware components and an open-source software environment. Designed mainly for autonomous racing purposes, [1] also introduces an 1/10 scale platform called F1TENTH. While this vehicle shares most of its hardware components with the MIT racecar, it has much better software support thanks to its large developer community. Consequently, this platform has been chosen at *AIMotionLab*.

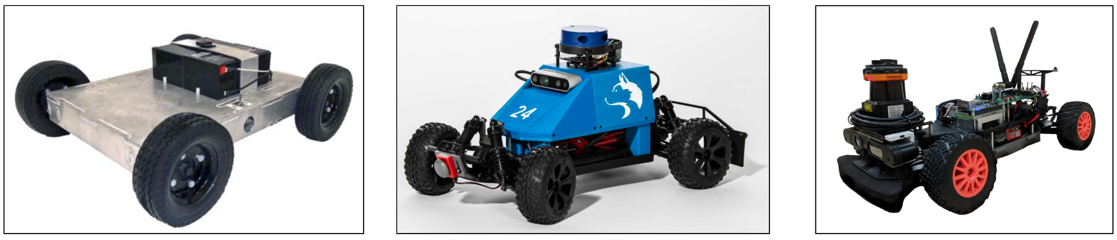


Figure 1.1: Different small-scale vehicle platform used in research and the industry: SuperDroid[1](left), MuSHR[2](center), F1TENTH[3](right).

[1]https://www.superdroidrobots.com/
[2]https://mushr.io/
[3]https://f1tenth.org/

The first step of the work is to construct the dynamical model of the F1TENTH vehicle. An accurate model is not only essential for the design of efficient control algorithms, but it can also be used to perform realistic simulations, e.g., for validation of the control algorithms or collect accurate data later for machine learning purposes, as in [23]. While the modelling of cars has a rich scientific literature [35] [13] [3], the specialties of small scale vehicles are often not considered by them. Although [19] and [37] present modelling techniques for these small scale platforms, their solutions mainly focus on autonomous racing instead of precise navigation. Therefore, a different approach is presented in this work, with the used models and parameter estimation methods adjusted to our goals. As a result, an accurate digital representation of the vehicle can be obtained with well applicable identifications steps, to allow fast and straight forward modelling that can be used not only for the F1TENTH, but also for other small scale platforms.

The introduced model consists of three different parts. First, the dynamic single track model [3] describes the motion of the vehicle. This model is chosen according to [37], where it was deemed sufficient for the F1TENTH platform. It is then complemented with a drivetrain model, which expresses the connection between the motor control input and the forces acting on the tires. This model incorporates the motor characteristics and the transmission between the drive shaft and the wheels. In most relevant publications, the motors are described as a first order system that can be augmented augmented with friction effects and the gear ratio of the transmission [37] [19] to obtain this drivetrain model. The last important part is the application of a tire model, which is used to describe the interaction between the tires of the vehicle and the ground surface. There is a wide range tire models that engineers can use, from analytical [11] [8] to empirical ones [4], depending on the complexity of the current task. For this work, a linear tire model is used, like in [25]. After the derivation of the required components, an important task is to determine the unknown parameters of these models. While some of these parameters can be measured directly, there are also empirical ones, that can only be estimated from measurements. In the literature, there are multiple estimation procedures and techniques outlined, such as [37] [25]. This work uses a combination of these methods to achieve precise model identification, specifically in environments where the size of the test area is limited.

In order to accomplish precise navigation with the vehicle, accurate trajectory tracking is required. The problem of autonomous trajectory tracking can be achieved by multiple control approaches [36]. The most simple ones originate from geometric relations such as the Pure Pursuit [5] or the Stanley controller [14]. One step in complexity is the introduction of *proportional–integral–derivative* (PID) based control, but the tuning of the controller gains can be a challenging task for rapidly changing environments [15]. To tackle the problematic gain adjustment, model-based feedback control, tuned by *linear quadratic regulator* LQR [27] can be applied. Nowadays, *model predictive control* (MPC) [10] is also getting more attention in the field of autonomous vehicles, but the application is currently constrained by the limited computational capacity of the onboard systems.

The introduced model-based control methods often require simplified, linear systems for the control design. However, to achieve better performance, linear parameter-varying (LPV) [26] models are developed in this work. The utilization of LPV models can be beneficial, as they can describe complex, nonlinear behavior, while preserving the advantageous properties of the linear model structure. A simple solution for the control of parameter varying systems is the application of a gain-scheduling [18] control. With the varying parameters considered as scheduling variables, a set of linear controllers can be designed for different operating points of the nonlinear system. Then, the optimal gains of the controller can be chosen from this set, based on the current values of the scheduling variables. Beyond gain scheduling, advanced LPV control design [7] has also matured to a powerful framework of methods with mathematically guaranteed stability and performance properties for all possible scheduling trajectories.

The main contributions of this work can be summarized as:

- Configuration of the *AIMotionLab* area to provide a test environment for the F1TENTH platform;
- Derivation of a dynamic model for F1TENTH cars, completed with the estimation of the model parameters, to obtain an accurate representation of the vehicle;
- Design of model-based, gain-scheduled and LPV control algorithms for accurate trajectory tracking with the vehicle;
- Development of a complete software framework, not only with the implementation of designed control algorithms, but also for simulations and convenient high-level management of the cars in the lab.

This work is organized into seven sections. First, the *AIMotionLab* environment is presented, where all the development takes place. In Section 3, the mathematical model of the F1TENTH vehicle is derived. Next, Section 4 details the experiments required for the estimation of the unknown model parameters. After the model parameters are obtained, open-loop validation measurements confirm the accuracy of the identified system. Section 5 details the control design procedure. Here, the nonlinear vehicle model is decoupled into LPV subsystems for control purposes. Then, for the decoupled models, gain-scheduled and LPV feedback control algorithms are designed and evaluated. In Section 6, real world experiments are presented, with the control algorithms implemented on the F1TENH platform. Finally, the conclusions and the future work are summarized in Section 7.

Complementing the thesis, the theoretical background of the work is collected in the appendix. Appendix A contains the derivation of the vehicle models used in this thesis. In Appendix B, the fundamentals of optimal control theory is presented for both time invariant and parameter-varying systems. Lastly, Appendix C contains a brief description of the control framework developed for the F1TENTH platform.

Thanks to the continuous development of autonomous systems, intelligent mobile robots such as drones or ground vehicles are getting deployed in many different areas of the industry. In order to unlock the full potential of these autonomous systems, efficient algorithms are required that can solve problems related to mobile robot navigation and control. To aid the research and development, there has been a growing interest in small scale vehicle platforms on which such algorithms can be properly evaluated. This presentation introduces modelling and control techniques specifically developed for these vehicle platforms. The motion of the vehicles are described by a nonlinear dynamic model, which is complemented with drivetrain and tire models. To easily obtain the parameters of the model, experiment based parameter estimation methods are introduced. Then, based on the resulting model, a low-level path-following algorithm is introduced, which is capable of precise maneuvering in constrained space. The introduced modelling and control techniques can provide the proper foundation for high level navigation and control algorithm research and development.

2 The F1TENTH test environment

This sections presents the F1TENTH test environment developed at SZTAKI as part of *AIMotionLab*. The main objective of *AIMotionLab* is to provide a test area where complex trajectory-tacking and path-planning algorithms can be evaluated not only in simulation, but also on real vehicles. While miniature quadcopters are already successfully integrated into the vehicle management system, autonomous ground vehicle development has just started recently. Therefore, at first, an F1TENTH autonomous vehicle test environment was built, with the utilization of the existing lab infrastructure. The structure of the environment is depicted in Figure 2.1. The main components of the system are introduced in the following subsections.

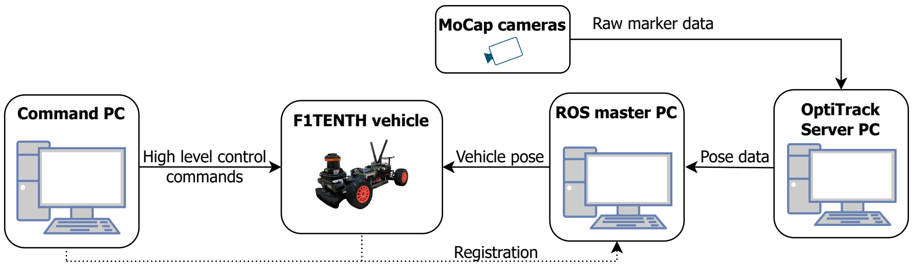


Figure 2.1: The hardware configuration of the F1TENTH test environment.

2.1 F1TENTH vehicle

F1TENTH is a 1/10 scale autonomous vehicle [1], developed and maintained by the F1TENTH community [9]. It is an open-source platform to aid autonomous systems research and education.

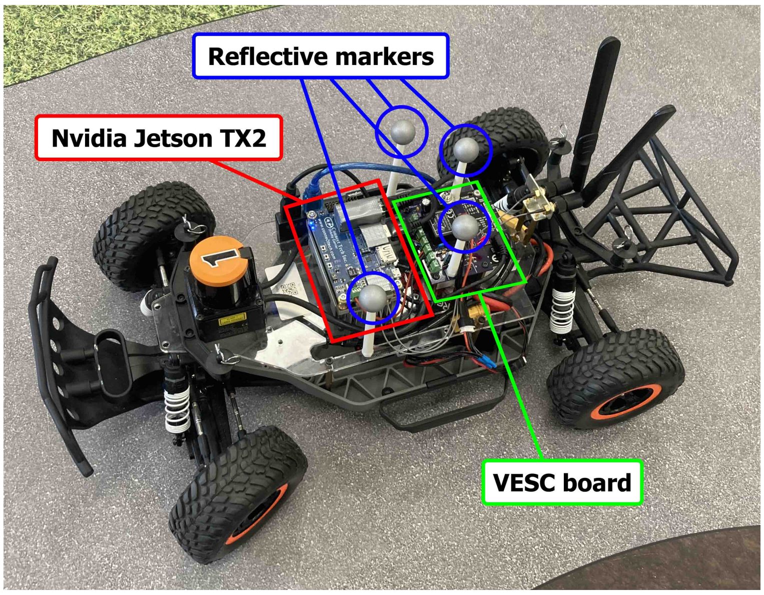


Figure 2.2: The F1TENTH vehicle and its main components.

The vehicle is built on the *Traxxas Slash 4x4 Ultimate* [32] chassis. It is a four wheel driven car, powered by a brushless DC motor, while the steering is controlled by a high-torque servo. These actuators are driven by a *Vedder Electronic Speed Controller* (VESC) [33]. This device handles the electrical commutation and allows the BLDC motor to be controlled with a simple PWM input. This is a useful feature, because the motor characteristics can be modelled as a first order system, like a simple DC motor. Moreover, the device has voltage and current sensors, but there is also an IMU and a speed sensor built in the VESC board.

The main computation unit is an *Nvidia Jetson TX2* embedded computer. Thanks to its multiple implemented communication ports, sensors and other hardware interfaces can be easily integrated into the system. In the current configuration that is used throughout this work, there are two external devices connected to the board. First, the VESC uses serial communication through an USB port to receive control inputs and send sensor information from the motor. Second, the data from external motion capture system, described in Section 2.2, is accessed via the Wifi module of the Jetson.

2.2 OptiTrack motion capture system

Sensing and environment mapping have a key part in the development of efficient mobile robots. For model-based motion control algorithms, it is essential to have reliable information about the current state of the actuated robot and the environment. In normal sized autonomous vehicles, there are usually a large collection of sensors, such as IMUs and speed sensors, lidars, cameras and GPS. Using their individual measurements, sensor fusion techniques can provide sufficiently accurate position and velocity measurements. However, this work introduces a different localization approach.

As the GPS cannot be used indoors, another external positioning solution is proposed that relies on the OptiTrack motion capture system. The main advantage of motion capture is that it can provide submilimeter and milirad position and orientation data. Thanks to the exceptional accuracy, other vehicle state properties such as velocity and acceleration information can be estimated by numerical differentiation without the application of other complementary sensors.

The motion capture system of the *AIMotionLab* environment consists of 14 high precision *OptiTrack Prime X 13*[1] infrared cameras and one central server PC. With reflective markers placed on the F1TENTH vehicles, each camera tracks the position of the markers independently and transmits the information to the OptiTrack Server PC at 120 Hz. On the server, the OptiTrack motion capture software, called Motive[2], is responsible for processing the incoming raw data. In Motive, rigid bodies can be defined from unique marker configurations, corresponding to the objects that should be tracked. Then, the position

[1]`https://optitrack.com/cameras/primex-13/`
[2]`https://optitrack.com/software/motive/`

and orientation data of these objects can be broadcasted through the local network, as shown in Figure 2.1.

2.3 Robotic Operating System

The test environment introduced in this work is built with the utilization of the *Robotic Operating System* (ROS) [24]. ROS is an open-source robotics framework, that provides useful tools for the development of robotic applications. There are multiple reasons to use ROS. First, the existing, out-of-the-box software stack of the vehicle created by the F1TENTH community [9] is built around it. Second, thanks to the large developer community, there is a great collection of drivers and hardware interfaces already implemented in the framework.

ROS is a modular, distributed communication and computation environment. The independent tasks running in the environment are organized into separate processes, so-called nodes, that can run individually, even across multiple machines. For the inter-process communication, a TCP and UPD based data transfer solution is provided, that enables convenient and fast information transmission between nodes, even across multiple machines on the same network. In the current environment, the ROS master PC is responsible for the overall management. It keeps track of the available devices and the running processes and provides a central access point for external connections.

Based on the ROS architecture, a complete management framework is developed for the vehicles as part of this work. More details on this software can be found in Appendix C, and the associated GitHub repository[1].

2.4 Command PC

As the lab architecture in Figure 2.1 shows, the Command PC is responsible for the high level management the cars. During normal operation, its main job is to start up the selected vehicles and to provide reference trajectories for them to follow.

It is important to note, that the tasks of the Command PC and the ROS master PC can be handled by one single computer. They are now separated only for multitasking and more convenient development purposes in the lab.

[1]`https://github.com/AIMotionLab-SZTAKI/aimotion-fleet1tenth`

3 Dynamic model of the F1TENTH car

The goal of this work is to achieve precise trajectory tracking with the F1TENTH vehicles. To design control algorithms for this task, an accurate model of the car is required. Therefore, in this section, the dynamic model of the F1TENTH vehicle is presented. The modelling is mainly based on [37], where the author constructs the model for the F1TENTH vehicle and proposes different experiments for estimating the parameters of it. However, parts of the models and the methods, such as the utilized tire model and the identification procedures differs from the ones used in that thesis. The main reason for that is to provide a more faster and more convenient modelling workflow and to adapt the to the environment setup of *AIMotionLab*.

3.1 Vehicle dynamics

This section gives an overview of the single track models which are used to describe the motion dynamics of the F1TENTH vehicle. These models are often referred to as bicycle models, because they lump together the front and rear wheel pairs into one single wheel each. This simplification results in a much more compact formulation, while they can accurately describe the motion of the vehicle. In this work, multiple models are developed with different complexity levels. First, the most simple kinematic single track model is introduced. Next, building on the basic geometric relations of it, a more complex dynamic model is derived.

The kinematic single track model can be expressed by the following equations [3]:

$$\dot{x} = v\cos(\varphi), \tag{3.1a}$$

$$\dot{y} = v\sin(\varphi), \tag{3.1b}$$

$$\dot{\varphi} = \frac{v}{l}\tan(\delta), \tag{3.1c}$$

$$\dot{v} = a, \tag{3.1d}$$

where $(x,\ y)$ is the position of the center of mass, φ denotes the heading angle measured from the X axis, v is the length of the velocity vector and l is the wheelbase. The two inputs of the system are the steering angle δ and the acceleration a. The model is also depicted in Figure 3.1 and the derivation steps are detailed in Appendix A. While this low complexity model can be easily utilized for control design, the control performance is limited by the simplifications and assumptions that were used in the model construction (see Appendix A). Another key limitation is the fact that the kinematic models can not be used to forward simulate the dynamics of the system, hence model-based control solutions require dynamic models.

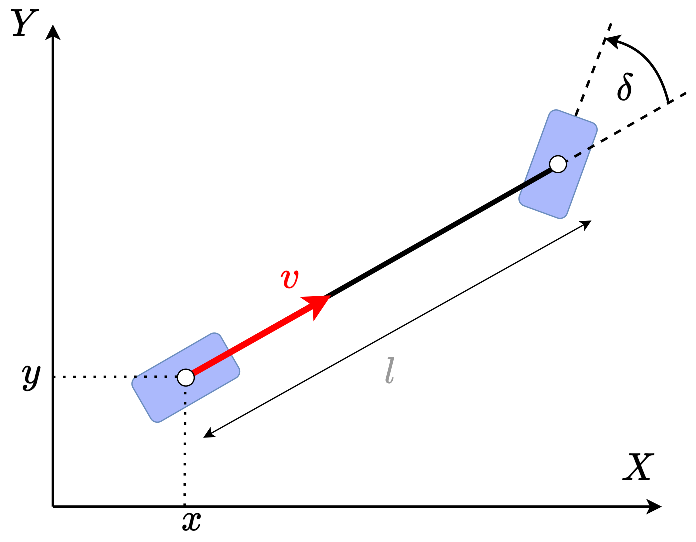


Figure 3.1: Kinematic single track model.

Therefore, to archive a more accurate digital representation of the vehicle, the introduction of a dynamic model is necessary. These models build on top of the fundamentals introduced by kinematic ones, but they representation capabilities are extended, as they also consider the dynamic effects.

Considering the F1TENTH platform, [37] used a dynamic single track model, as their validation tests showed that it can sufficiently describe the motion of the vehicle. It is also stated, that more advanced dual track modelling solutions [3] do not lead to notable accuracy improvements, but their computational cost increases significantly. Moreover, [19] also uses the dynamic single track representation to model a 1/43 sized RC car in their autonomous racing research, which also shares some similarities with this project. Therefore, for the F1TENTH vehicles, the dynamic single track model is chosen in this work.

The model is defined by the following differential equations [3]:

$$\dot{x} = v_\xi \cos(\varphi) - v_\eta \sin(\varphi), \tag{3.2a}$$

$$\dot{y} = v_\xi \sin(\varphi) + v_\eta \cos(\varphi), \tag{3.2b}$$

$$\dot{\varphi} = \omega, \tag{3.2c}$$

$$\dot{v}_\xi = \frac{1}{m}\left(F_{\mathrm{r},\xi} + F_{\mathrm{f},\xi}\cos(\delta) - F_{\mathrm{f},\eta}\sin(\delta) + m v_\eta \omega\right), \tag{3.2d}$$

$$\dot{v}_\eta = \frac{1}{m}\left(F_{\mathrm{r},\eta} + F_{\mathrm{f},\xi}\sin(\delta) + F_{\mathrm{f},\eta}\cos(\delta) - m v_\xi \omega\right), \tag{3.2e}$$

$$\dot{\omega} = \frac{1}{I_z}\left(F_{\mathrm{f},\eta} l_\mathrm{f}\cos(\delta) + F_{\mathrm{f},\xi} l_\mathrm{f}\sin(\delta) - F_{\mathrm{r},\eta} l_\mathrm{r}\right), \tag{3.2f}$$

where (x, y) is the position of the center of mass, φ denotes the heading angle measured from the X axis and v_ξ, v_η, ω represent the longitudinal, lateral and angular velocity respectively, as it is depicted in Figure 3.2. Constant parameters of the model are the distance of the front and rear axis from the centre of mass, denoted as l_f and l_r, respectively, and the mass of the vehicle m. The inputs are the steering angle δ and the longitudinal tire forces $F_{\mathrm{f},\xi}$, $F_{\mathrm{r},\xi}$ acting on the front and rear wheels. As the F1TENTH is a four wheel driven vehicle, it is assumed that the longitudinal force acting on the two tires are equal, therefore

$$F_{\xi} = F_{\mathrm{f},\xi} = F_{\mathrm{r},\xi}, \tag{3.3}$$

where F_{ξ} will be obtained from the drivetrain model, described in Section 3.2. The lateral tire forces $F_{\mathrm{f},\eta}$ and $F_{\mathrm{r},\eta}$ are computed with the help of tire models in Section 3.3.

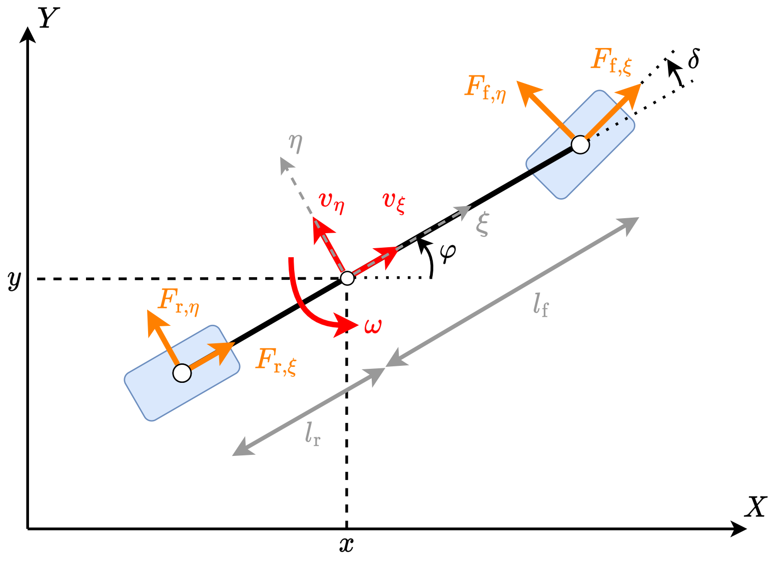


Figure 3.2: Single track vehicle model. The longitudinal (v_{ξ}) and lateral velocity (v_{η}) of the center of mass is defined in the moving (ξ, η) coordinate frame, which is fixed to the vehicle. The heading angle φ denotes the angle between the ξ and the x axis and the yaw rate is defined as its time derivative: $\omega = \dot{\varphi}$.

Throughout this work, this dynamic representation of the vehicle will be used for the control design and simulation tasks. The only exception, where the kinematic one is utilized instead, is the design of the steering controller for backward driving, as this simpler model can eliminate undesired side-effects of non-collocated control.

3.2 Drivetrain model

As it was already mentioned in Section 3.1, the longitudinal tire force (F_{ξ}) is generated by the drivetrain. In this section a model for this drivetrain is presented, to provide a connection between the motor control input and the acting forces. The model incorporates the characteristics of the BLDC motor, the transmission between the drive shaft and the wheels of the vehicle and also the effect of dry and viscous friction.

The VESC board provides two different ways to actuate the motor. It can be used as a voltage controller where the control input is the PWM duty cycle, which corresponds to the speed (rpm) of the motor. Another approach is actuating the applied direct current, which has a direct effect on the torque output. This section gives an overview of the modelling method for both of these approaches, but in the latter sections the VESC will be used only in voltage control mode.

As the BLDC motor dynamics are significantly faster than the vehicle dynamics, it is unnecessary to use a complex motor model. Therefore, this work models the motor characteristics as a first order system, extended with the effect of static friction. A variation of this approach was presented in [19] with 1/43 scale RC cars and in [37], the author used the same first order model for the F1TENTH vehicles.

The first order characteristics of the motor model are derived from the steady state equations of the DC motor. If the control input, is a unitless value, denoted as d, that varies in the range of $[-1,\ 1]$, the voltage applied on the motor can be calculated as

$$U = U_{\mathrm{max}} d, \tag{3.4}$$

where U_{max} is the maximal voltage and U is the actual voltage value, respectively. Similarly, if the input current is actuated

$$I = I_{\mathrm{max}} d, \tag{3.5}$$

where I_{max} is the maximal current value and I is the actual current value, respectively.

The steady-state torque-speed characteristics of a DC motor can be represented by the following equation, as detailed in [34]:

$$T_{\mathrm{m}} = \frac{\psi}{R} U - \frac{\psi^2}{R} \Omega, \tag{3.6}$$

where U is the input voltage, ψ is the amplitude of the motor flux vector, R is the armature resistance and Ω is the shaft speed. The value of ψ is assumed to be constant. The connection between the applied current and the motor torque can also be expressed as a linear equation, using a lumped model from [34]:

$$T_{\mathrm{m}} = K_T I, \tag{3.7}$$

where I is the input current and K_T is the motor torque constant. T_{m} denotes the motor torque.

The relation between the torque applied by the motor and the longitudinal force acting on the wheels can be expressed using the wheel radius (r) and the gear-ratio (G) of the drivetrain:

$$F_{\xi,\mathrm{m}} = \frac{G}{2r} T_{\mathrm{m}}. \tag{3.8}$$

Similarly the relation between the shaft speed and the longitudinal velocity of the model is known:

$$v_\xi = \frac{r}{G} \Omega. \tag{3.9}$$

To consider the mechanical losses, the following linear model can be introduced:

$$F_{\mathrm{d}} = C_{\mathrm{d},1} v_\xi + \mathrm{sign}(v_\xi) C_{\mathrm{d},0}, \tag{3.10}$$

where $C_{d,1}$ is the viscous friction constant and $C_{d,0}$ represents the dry friction. Note that this model is only valid for moving vehicles ($v_\xi \neq 0$), since the dry friction acting on a stationary object requires a more elaborate jump flow model than a single constant.

By subtracting the friction force from the motor force, F_ξ can be obtained:

$$F_\xi = F_{\xi,\mathrm{m}} - F_{\mathrm{d}}. \tag{3.11}$$

Next, by substituting (3.4), (3.6), (3.9), (3.8) and (3.10) into (3.11), the following drivetrain model can be obtained for the calculation of the longitudinal force, under voltage actuation:

$$F_\xi = \frac{\psi U_{\max}}{2rR} d - \left(\frac{G^2 \psi^2}{2r^2 R} + C_{\mathrm{d},1} \right) v_\xi - \mathrm{sign}(v_\xi) C_{\mathrm{d},0}. \tag{3.12}$$

Similarly, from (3.5), (3.7), (3.8), (3.10) and (3.11), the current actuated drivetrain model can be expressed in terms of the actuation command d as

$$F_\xi = \frac{G K_T I_{\max}}{2r} d - C_{\mathrm{d},1} v_\xi - \mathrm{sign}(v_\xi) C_{\mathrm{d},0}. \tag{3.13}$$

Both (3.12) and (3.13) can be rewritten to the following form:

$$F_\xi = C_{\mathrm{m}1} d - C_{\mathrm{m}2} v_\xi - \mathrm{sign}(v_\xi) C_{\mathrm{m}3}. \tag{3.14}$$

where $C_{\mathrm{m}1}$, $C_{\mathrm{m}2}$ and $C_{\mathrm{m}3}$ are the drivetrain parameters that can be identified as detailed in the subsequent sections.

3.3 Tire model

Tire models are used to describe the interaction between the tires of the vehicle and the ground surface, namely the acting tire forces. There are many different tire models from complex FEM based models to simplified and empirical formulas. The more complex methods are applicable for both longitudinal and lateral tire force calculations, but are also harder to identify. Since the drivetrain model introduced in Section 3.2 already provides the longitudinal tire force, the tire model in our current application is only required for the lateral tire force calculation.

In [19], the tire model of a 1/43 sized autonomous car was described by the Simplified Pacejka magic formula [4]. In [37], the same model was used for the modeling and identification of an F1TENTH vehicle. The formula is described by the following equations:

$$
\begin{aligned}
F_{\mathrm{f},\eta} &= D_{\mathrm{P,f}} \cdot \sin(C_{\mathrm{P,f}} \cdot \arctan(B_{\mathrm{P,f}} \cdot \alpha_{\mathrm{f}})), && \text{(3.15a)}\\
F_{\mathrm{r},\eta} &= D_{\mathrm{P,r}} \cdot \sin(C_{\mathrm{P,r}} \cdot \arctan(B_{\mathrm{P,r}} \cdot \alpha_{\mathrm{r}})), && \text{(3.15b)}\\
\alpha_{\mathrm{f}} &= -\arctan\left(\frac{\omega l_{\mathrm{f}} + v_\eta}{v_\xi}\right) + \delta, && \text{(3.15c)}\\
\alpha_{\mathrm{r}} &= \arctan\left(\frac{\omega l_{\mathrm{r}} - v_\eta}{v_\xi}\right), && \text{(3.15d)}
\end{aligned}
$$

where $B_{\mathrm{P,f}}$, $C_{\mathrm{P,f}}$, $D_{\mathrm{P,f}}$ and $B_{\mathrm{P,r}}$, $C_{\mathrm{P,r}}$, $D_{\mathrm{P,r}}$ are the Pacejka parameters of the front and rear wheel of the model respectively. α_{f} and α_{r} represent the lateral slip angles.

Considering that our work focuses on indoor navigation and motion planning for an obstacle rich environment, the vehicle rarely accelerates to high velocities. This means that a simpler, linearized model may also be sufficient. As v_ξ is magnitudes larger than v_η, ωl_{r} and ωl_{f}, the tire side slip angles can be approximated as:

$$
\begin{aligned}
\alpha_{\mathrm{f}} &\approx -\frac{\omega l_{\mathrm{f}} + v_\eta}{v_\xi} + \delta, && \text{(3.16a)}\\
\alpha_{\mathrm{r}} &\approx \frac{\omega l_{\mathrm{r}} - v_\eta}{v_\xi}. && \text{(3.16b)}
\end{aligned}
$$

For small α angles the lateral force can also be approximated as:

$$
F_\eta = D_{\mathrm{P}} \cdot \sin(C_{\mathrm{P}} \cdot \arctan(B_{\mathrm{P}} \cdot \alpha)) \approx D_{\mathrm{P}} C_{\mathrm{P}} B_{\mathrm{P}} \alpha. \tag{3.17}
$$

By introducing $C_{\mathrm{f}} = D_{\mathrm{P,f}} C_{\mathrm{P,f}} B_{\mathrm{P,f}}$ and $C_{\mathrm{r}} = D_{\mathrm{P,r}} C_{\mathrm{P,r}} B_{\mathrm{P,r}}$, the so-called cornering stiffness parameters, a linear tire model can be obtained, which is described by the following equations:

$$
\begin{aligned}
F_{\mathrm{r},\eta} &= C_{\mathrm{r}} \alpha_{\mathrm{r}}, && \text{(3.18a)}\\
F_{\mathrm{f},\eta} &= C_{\mathrm{f}} \alpha_{\mathrm{f}}, && \text{(3.18b)}\\
\alpha_{\mathrm{r}} &= \frac{-v_\eta + l_{\mathrm{r}} \omega}{v_\xi}, && \text{(3.18c)}\\
\alpha_{\mathrm{f}} &= \delta - \frac{v_\eta + l_{\mathrm{f}} \omega}{v_\xi}. && \text{(3.18d)}
\end{aligned}
$$

This model only has two unknown parameters C_{r} and C_{f} that are needed to be estimated compared to the 6 parameter Pacejka magic formula. However, it is important to note, that this simplified, linear model introduces a singularity at $v_\xi = 0$, because of the tire slip estimation in (3.16).

4 Parameter estimation

In this section, the identification procedure of the above described dynamical model is outlined. Some of the model parameters can be measured directly such as m, l. Another set of parameters can be estimated from the measured ones like l_r, l_f and I_z. However, the parameters of the drivetrain (C_m1, C_m2, C_m3) and the tire model (C_f, C_r) can only be obtained by performing experiments and fitting measurement data.

First, the onboard velocity sensor of the motor controller is configured and calibrated, as it will be used later in the drivetrain identification. Then, it is followed by the direct measurement and calculation of the physical parameters. Next, the steering servo parameters are obtained, which is followed by the drivetrain parameter identification. Lastly, the tire model parameters are estimated.

The estimation procedures heavily rely on the methods presented in [37]. However, as there are new and modified model components in this work, the identification methods also differ. New experiments are introduced for the linear tire model estimation, and a different curve fitting approach is used for the calculation of the drivetrain model parameters.

4.1 Onboard velocity sensor

Despite high sampling frequency and precision, the motion capture system has limitations. With the current configuration of the lab, the 14 OptiTrack cameras are capable of covering a 4.5x5 m surface. While this is sufficient for most applications, some identification procedures cannot be carried out, as accurate position data is limited to this area. To obtain the drivetrain parameters, the required acceleration and braking maneuvers need larger operating space than the available covered area. To resolve this issue, the onboard sensor built into the VESC motor controller is utilized, which is capable of measuring the rotational speed of the motor. As this is a scalar value, most of the properties of the vehicle such as lateral velocity or side-slip cannot be estimated from this sensor. On the other hand, for straight line experiments it can produce accurate results, assuming that no longitudinal slip occurs between the ground surface and the tire.

Another issue is that the sensor measures the *electric revolutions per minute* (ERPM) of the motor. So the relationship between the longitudinal velocity v_ξ [m/s] and ERPM [rpm] must be determined. To achieve this, a set of straight line drive experiments have been performed. First, the vehicle is accelerated to reach a steady-state velocity. Then, the measured ERPM can be compared with the v_ξ longitudinal velocity, which is computed by the differentiation of the OptiTrack position data, logged at 25 Hz. From the measurements, it is clear that the connection is linear, therefore

$$v_\xi = k_{v,1} \cdot \mathrm{ERPM} + k_{v,0}, \tag{4.1}$$

where v_ξ is the longitudinal velocity of the vehicle and $k_{v,1}$ and $k_{v,0}$ are the unknown parameters. The $k_{v,0}$ offset parameter is necessary, as the sensor sends nonzero velocity measurements for stationary vehicles. By the least squares method, a linear model can be fitted on the collected data to identify the v_ξ-ERPM relationship. The measurements and the line fitted on the collected data points are shown in Figure 4.1, and the identified parameters are summarized in Table 4.1, located in Section 4.6.

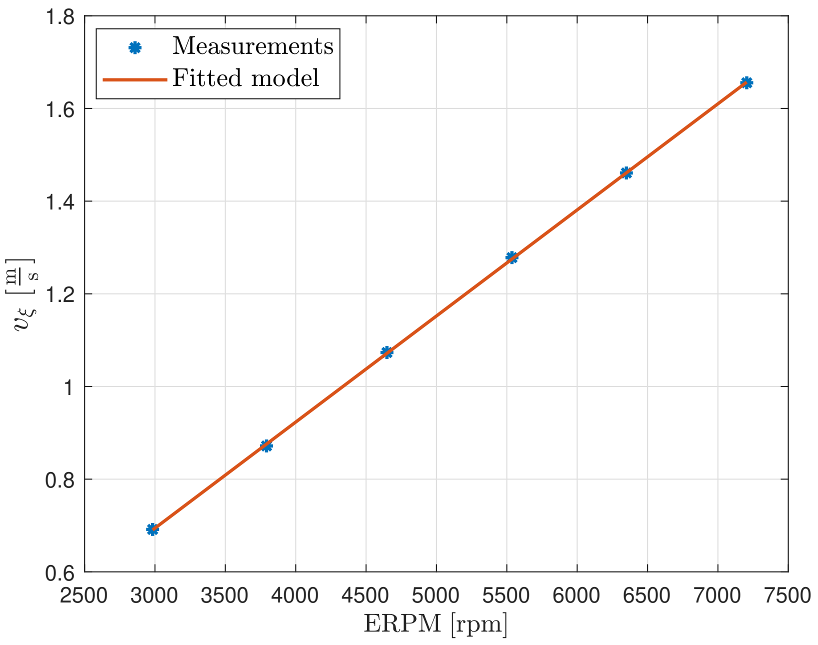


Figure 4.1: Onboard velocity sensor calibration. The collected velocity data from the different sensors and the line fitted on the data points.

4.2 Physical parameters

The distance of the front and the rear axis (l) and the mass of the vehicle (m) can be measured directly. By measuring the weight of the front and rear axis separately, the mass distribution can also be approximated. With the following equations, the l_f and l_r values can be obtained:

$$l_\mathrm{r} = l\left(1 - \frac{m_\mathrm{r}}{m}\right), \tag{4.2a}$$

$$l_\mathrm{f} = l\left(1 - \frac{m_\mathrm{f}}{m}\right), \tag{4.2b}$$

where m_r and m_f denote the mass measured at the rear and at the front axis, respectively.

The inertia around the z axis can also be approximated from the previously obtained parameters as

$$I_z = m_\mathrm{r} l_\mathrm{r}^2 + m_\mathrm{f} l_\mathrm{f}^2. \tag{4.3}$$

4.3 Steering angle

One of the control inputs of the dynamic vehicle model from Section 3 is the steering angle, denoted as δ. This is realized with a steering servo that can directly actuate the heading of the front wheels of the vehicle. The input signal to the servo is defined by the actuator hardware as $u_\mathrm{s} \in [0, 1]$, where 0 represents a full left and 1 represents a full right turn. However, due to the mechanical construction of the vehicle, there is an offset in the steering angle as well as a backlash between a servo and the actual position of the wheels. This effect is easily observable: for $u_\mathrm{s} = 0.5$ the trajectory of the vehicle will be an arc, rather than the expected straight line.

The goal in this section is to obtain the steering offset ($k_{\delta,0}$) and calculate the gain ($k_{\delta,1}$) between the steering angle δ and input to the servo u_s.

$$u_\mathrm{s} = k_{\delta,1}\delta + k_{\delta,0} \tag{4.4}$$

To determine these coefficients multiple measurements were performed with different u_s values, ranging from $u_\mathrm{s} = 0.1$ to $u_\mathrm{s} = 0.9$ with 0.1 step size. To drive the vehicle, a low $d = 0.05$ reference control input was used to minimize the effect of side slip. The trajectory of the vehicle was recorded with the OptiTrack camera system. In the post processing, a circle is fitted on the collected position data, from which the actual steering angle can be determined by the following equation, introduced in [21]:

$$\delta = \arcsin\left(\frac{l}{R}\right), \tag{4.5}$$

where l is the wheelbase of the vehicle and R is the radius of the circle fitted on the measurement data. From multiple measurements, $k_{\delta,1}$ and $k_{\delta,0}$ can be identified by fitting a linear model on the (u_s, δ) data pairs with the least squares method.

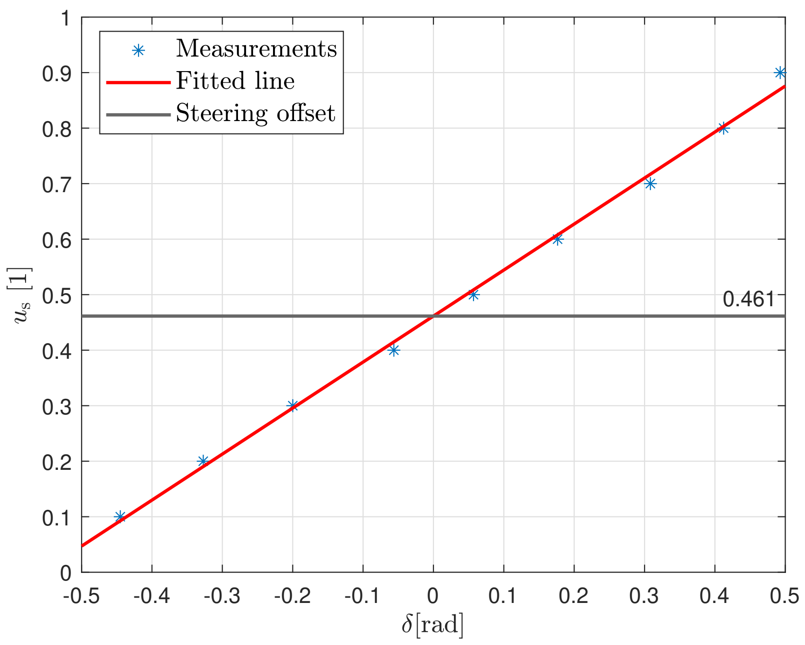


Figure 4.2: Linear connection between the steering angle and the steering servo actuator input. The model is obtained by linear regression using the logged control input and the steering angle calculated from the measurement. The calculated parameters are collected in Table 4.4 of Section 4.6.

4.4 Drivetrain model

In the following section, the parameters of the drivetrain model described in Section 3.2 are estimated. During the identification experiments, the VESC is used in voltage control mode. After the parameters are obtained, the effect of static friction is observed by mapping out a hysteresis curve of the acceleration and deceleration experiments.

4.4.1 Model parameters

The model parameter estimation is carried out by performing multiple acceleration and braking maneuvers with different d reference input values and constant $\delta = 0$ steering angle, as it is described in [37].

These maneuvers can be described with the following steps: take a stationary vehicle with initial conditions $q(0) = [x(0) \;\; y(0) \;\; \varphi(0) \;\; v_\xi(0) \;\; v_\eta(0) \;\; \omega(0)]^T = 0$. At $t = 0$, the motor reference input is set to $d = d_{\mathrm{acc},i}$ until $t = t_{\mathrm{acc}}$, where the reference is reduced to $d = 0$. The time evolution of the test input is also displayed in Figure 4.3.

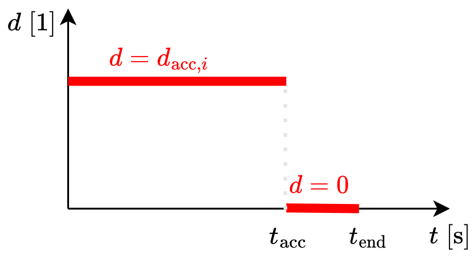


Figure 4.3: Time evolution of the test input used in the drivetrain identification.

The experiments are performed with $d_{\mathrm{acc},i}$ ranging from 0.05 to 0.225 with 0.025 steps. The d reference input is nonzero for $t_{\mathrm{acc}} = 6$ s and the tests finished at $t_{\mathrm{end}} = 8$ s. For higher d values, the vehicle could not keep the steady-state velocity for the whole $t \in [0, t_{\mathrm{acc}}]$ period as the testing area of the lab is limited. To overcome this issue, in the processing of the measurement data, the steady-state segment was artificially lengthened as in [37].

By constraining the trajectory of the vehicle to a straight line, it can be assumed that no side-slip will occur ($\alpha_{\mathrm{f}} = \alpha_{\mathrm{r}} = 0$). Based on the tire model in (3.18) and the vehicle model in (3.2), this implies zero lateral tire forces ($F_{\mathrm{r},\eta} = F_{\mathrm{f},\eta} = 0$) and zero lateral velocity ($v_\eta = 0$).

With this assumption, the drivetrain model in (3.14) can be substituted into the longitudinal dynamics of the vehicle in (3.2d) to get the following formula:

$$\dot{v}_\xi = \frac{2}{m}\left(C_{\mathrm{m1}}d - C_{\mathrm{m2}}v_\xi - \mathrm{sign}(v_\xi)C_{\mathrm{m3}}\right). \tag{4.6}$$

The obtained equation describes a one dimensional linear time-invariant (LTI) system that can be rewritten into the standard state-space representation as $\dot{x} = Ax + Bu$:

$$\underbrace{\dot{v}_\xi}_{\dot{x}} = \underbrace{-\frac{2C_{\mathrm{m2}}}{m}}_{A}\underbrace{v_\xi}_{x} + \underbrace{\begin{bmatrix}\frac{2C_{\mathrm{m1}}}{m} & -\frac{2C_{\mathrm{m3}}}{m}\end{bmatrix}}_{B}\underbrace{\begin{bmatrix}d\\ \mathrm{sign}(v_\xi)\end{bmatrix}}_{u}. \tag{4.7}$$

The inputs of the system are the d reference input and $\mathrm{sign}(v_\xi)$.The first input (d) is piecewise constant with values $d = d_{\mathrm{acc},i}$ for $t \in [0, t_{\mathrm{acc}}]$ and $d = 0$ for $t \in (t_{\mathrm{acc}}, t_{\mathrm{end}}]$. The second input, $\mathrm{sign}(v_\xi)$, is 1 during the measurements, as the identification is performed in forward motion.

Next, the obtained continous system is discretized by zero order hold method with 0.04 s sampling time. The discrete time state-space model can be expressed as

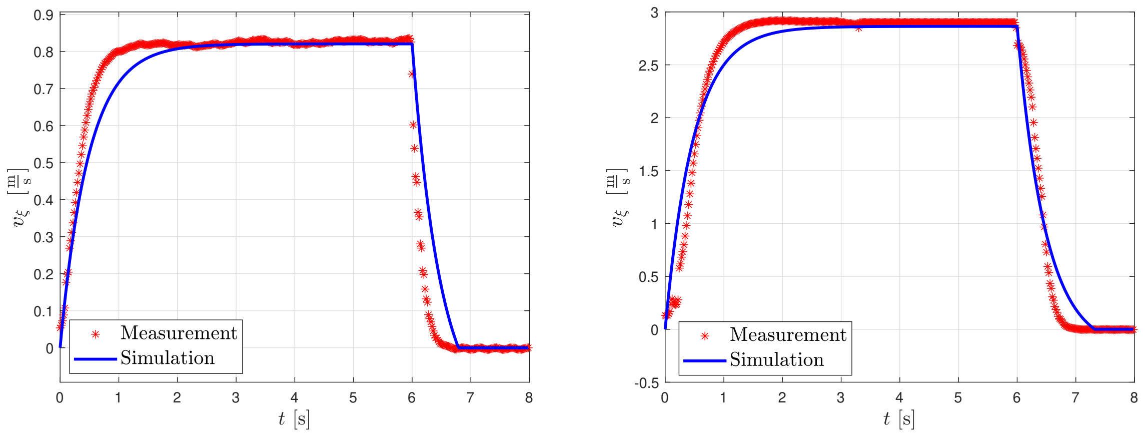
$$v_\xi[k+1] = A_{\mathrm{d}}v_\xi[k] + B_{\mathrm{d}}u[k], \tag{4.8}$$

where A_{d} and B_{d} are the discrete time state and input matrices. Using the discrete time response, the longitudinal velocity of the vehicle model can be easily simulated by a simple iteration, since the initial conditions and the control inputs are all known at every time step.

The simulated velocity then can be fitted on the measurement data, collected from the conducted experiments. This results in an unconstrained nonlinear optimization problem:

$$\min_{C_{\mathrm{m}i}} \quad \sum_{k=1}^{N} (v_\xi[k] - v_{\xi,\mathrm{mes}}[k])^2, \tag{4.9}$$

where $v_\xi[k]$ is the simulated velocity and $v_{\xi,\mathrm{mes}}[k]$ is the velocity data collected form the measurements at k. The curve fitting was carried out using nonlinear least squares fitting, with the MATLAB `lqscurvefit` function from the Optimization toolbox [30].



(a) $d = 5$ % reference input　　**(b)** $d = 15$ % reference input

Figure 4.4: Comparison of the drivetrain model with real world measurements. The estimated parameters can be found in Table 4.5 of Section 4.6.

4.4.2 Effect of static friction

Since the vehicle model is only valid for a non-stationary vehicle, due to the linear tire model and the dry friction constant in the drivetrain, the friction effects that are dominant at low speeds have not been considered.

To investigate the vehicle behavior at low reference speeds a hysteresis curve is presented that maps the longitudinal velocity of the vehicle to the corresponding control input in both acceleration and deceleration case.

The experiments required for the data acquisition are straight line ($\delta = 0$) acceleration and deceleration tests with motor control inputs ranging from 0 to 0.08. The longitudinal velocity of the vehicle is calculated by numerical differentiation, using the OptiTrack position data, logged at 25 Hz.

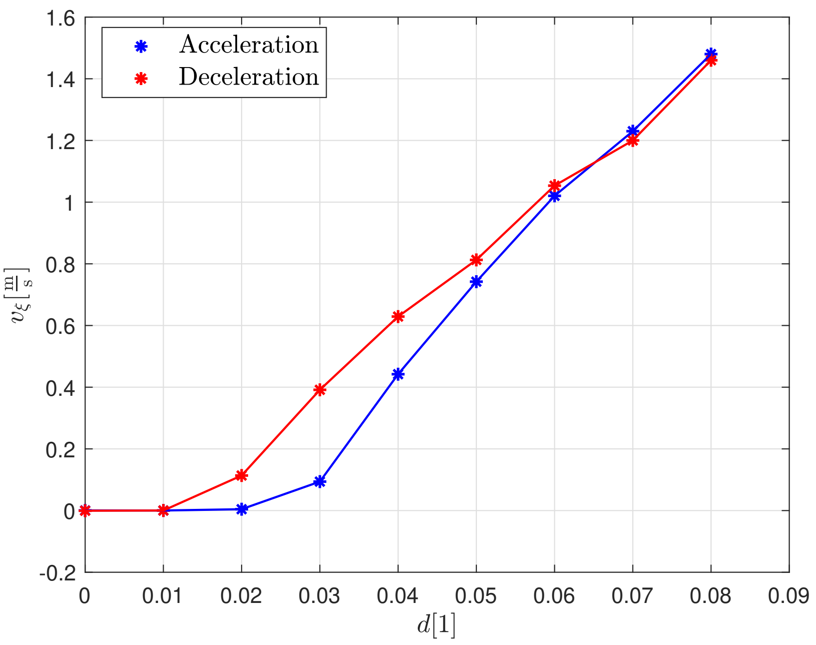


Figure 4.5: The hysteresis curve between the control input d and v_ξ longitudinal velocity for both the acceleration and the deceleration.

From the Figure 4.5, it is visible that for $d < 0.05$ there is significant difference between the acceleration and deceleration case. Moreover, these experiments highlighted another undesired effect. As the motor controller provides stable torque, and the desired rotational speeds are set to low values by the d reference input, the vehicle starts to shake. As this effect cannot be eliminated in voltage control mode without the modification of the VESC controller, such low speeds should be avoided.

Therefore, to eliminate these undesired effects, a minimal velocity $|v_{\xi,\mathrm{min}}| = 0.75$ m/s is introduced. Above $v_{\xi,\mathrm{min}}$ the motor characteristics are linear, the hysteresis effect does not occur.

4.5 Tire model

In this section the cornering stiffness values of the linear tire model, introduced in Section 3.3, are estimated.

The tire model identification can be carried out in different ways. In [37], the author used manually executed high-speed slip maneuvers to identify the parameters of the Simplified Pacejka magic formula. However, as these maneuvers require high velocities, it was unsafe to perform them in the limited space of the lab.

The authors in [35] present another technique, that uses a quasi-steady state ramp-up maneuver for the identification experiments. Here the steering angle δ is slowly increased while the throttle input is kept constant so the vehicle can be assumed to be in a steady-state condition. However, due to the capture volume limitations of the OptiTrack system, we could not perform this maneuver. In [25], they overcome the same limitation by running a sequence of circular motion tests with different δ angles as a substitute. During these tests, both the steering angle and the throttle input are kept constant to achieve steady state condition after the initial acceleration transients.

If the motion of the vehicle is assumed to be in steady-state condition ($\dot{v}_\xi = 0$, $\dot{\omega} = 0$), the lateral acceleration can be approximated with the following equation:

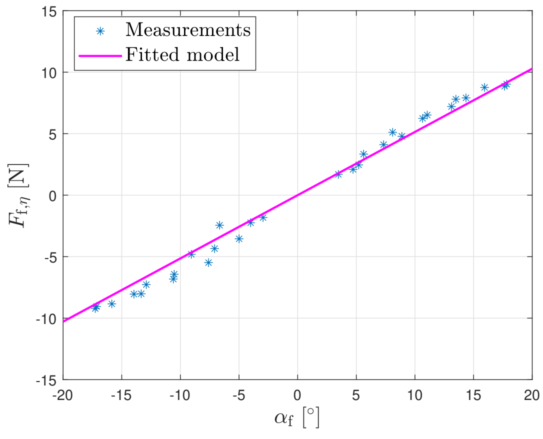
$$a_\eta^{\mathrm{ss}} \approx \omega v_\xi, \tag{4.10}$$

and the lateral tire forces can be expressed from the vehicle model described in (3.2):

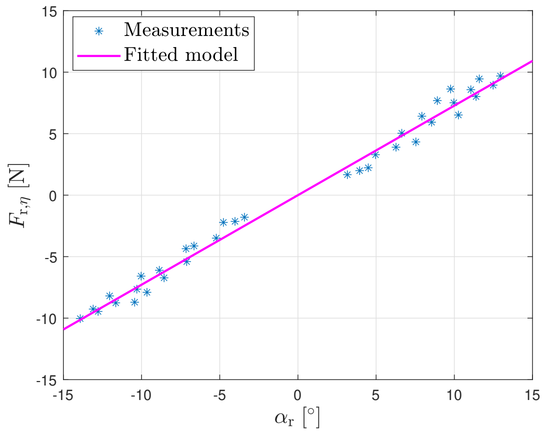
$$F_{\mathrm{f},\eta} = \frac{m l_\mathrm{r}}{(l_\mathrm{f} + l_\mathrm{r})\cos(\delta)} v_\xi \omega = \frac{m l_\mathrm{r}}{(l_\mathrm{f} + l_\mathrm{r})\cos(\delta)} a_\eta^{\mathrm{ss}}, \tag{4.11a}$$

$$F_{\mathrm{r},\eta} = \frac{m l_\mathrm{f}}{(l_\mathrm{f} + l_\mathrm{r})} v_\xi \omega = \frac{m l_\mathrm{f}}{(l_\mathrm{f} + l_\mathrm{r})} a_\eta^{\mathrm{ss}}. \tag{4.11b}$$

Since the tire forces from (4.11) and the side slip angles from (3.18) both can be calculated by substituting the measured vehicle states into the equations, the cornering stiffness values can be obtained by fitting a linear model with least squares onto the data.



(a) Front tire characteristics



(b) Rear tire characteristics

Figure 4.6: Linear tire model fitted on the measurement data, obtained by performing a sequence of circular motion tests with different δ steering angle and d reference input. The calculated cornering stiffness values are collected in Table 4.2.

4.6 Summary of model equations and parameters

In this section all the equations are collected to get a clear view of the model.

$$\dot{x} = v_\xi \cos(\varphi) - v_\eta \sin(\varphi) \tag{4.12a}$$

$$\dot{y} = v_\xi \sin(\varphi) + v_\eta \cos(\varphi) \tag{4.12b}$$

$$\dot{\varphi} = \omega \tag{4.12c}$$

$$\dot{v}_\xi = \frac{1}{m}\left(F_\xi + F_\xi \cos(\delta) - F_{\mathrm{f},\eta} \sin(\delta) + m v_\eta \omega\right) \tag{4.12d}$$

$$\dot{v}_\eta = \frac{1}{m}\left(F_{\mathrm{r},\eta} + F_\xi \sin(\delta) + F_{\mathrm{f},\eta} \cos(\delta) - m v_\xi \omega\right) \tag{4.12e}$$

$$\dot{\omega} = \frac{1}{I_z}\left(F_{\mathrm{f},\eta} l_\mathrm{f} \cos(\delta) + F_{\mathrm{f},\xi} l_\mathrm{f} \sin(\delta) - F_{\mathrm{r},\eta} l_\mathrm{r}\right) \tag{4.12f}$$

$$F_\xi = C_{\mathrm{m1}} d - C_{\mathrm{m2}} v_\xi - \mathrm{sign}(v_\xi) C_{\mathrm{m3}} \tag{4.12g}$$

$$F_{\mathrm{r},\eta} = C_\mathrm{r} \alpha_\mathrm{r} \tag{4.12h}$$

$$F_{\mathrm{f},\eta} = C_\mathrm{f} \alpha_\mathrm{f} \tag{4.12i}$$

$$\alpha_\mathrm{r} = \frac{-v_\eta + l_\mathrm{r}\omega}{v_\xi} \tag{4.12j}$$

$$\alpha_\mathrm{f} = \delta - \frac{v_\eta + l_\mathrm{f}\omega}{v_\xi} \tag{4.12k}$$

The two control inputs of the system are d motor reference and δ steering angle.

Parameter	Value	Dimension
$k_{v,1}$	$2.2894 \cdot 10^{-4}$	$30\mathrm{m} \cdot \pi^{-1}\mathrm{rad}^{-1}$
$k_{v,0}$	$7.4655 \cdot 10^{-3}$	s^{-1}

Table 4.1: VESC sensor parameters.

Parameter	Value	Dimension
C_r	41.7372	$\mathrm{N} \cdot \mathrm{rad}^{-1}$
C_f	29.4662	$\mathrm{N} \cdot \mathrm{rad}^{-1}$

Table 4.2: Tire model parameters.

Parameter	Value	Dimension
m_r	1.439	kg
m_f	1.484	kg
m	2.923	kg
l	0.33	m
l_r	0.168	m
l_f	0.163	m
I_z	0.0796	$\mathrm{kg} \cdot \mathrm{m}^2$

Table 4.3: Vehicle model parameters.

Parameter	Value	Dimension
$k_{\delta,1}$	0.8288	rad^{-1}
$k_{\delta,0}$	0.4615	1
$\delta_{\mathrm{r,max}}$	0.4967	rad
$\delta_{\mathrm{l,max}}$	0.5162	rad

Table 4.4: Steering parameters.

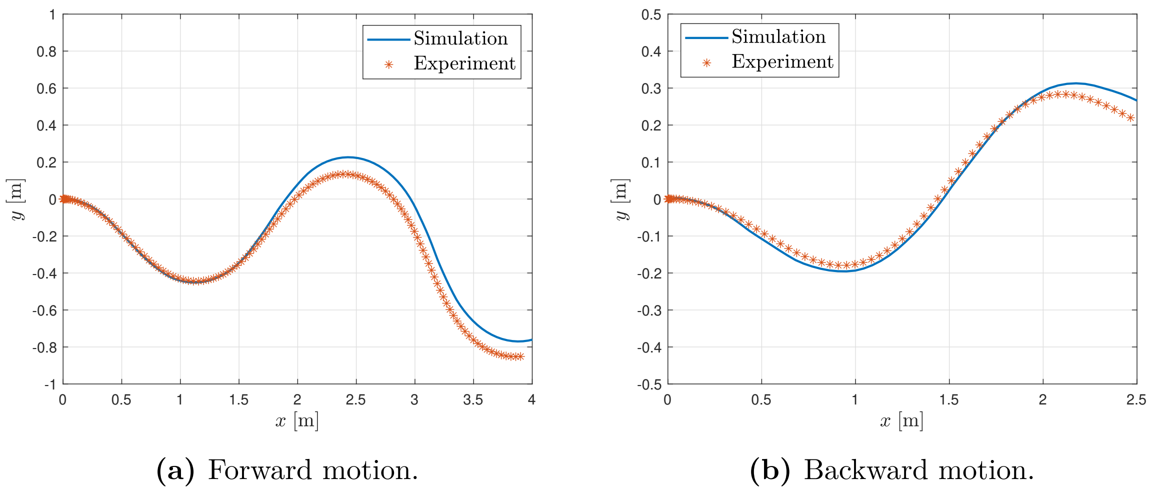
Parameter	Value	Dimension
C_{m1}	41.7960	N
C_{m2}	2.0152	$\mathrm{Ns} \cdot \mathrm{m}^{-1}$
C_{m3}	0.4328	N

Table 4.5: Drivetrain model parameters.

4.7 Validation

After all parameters of the vehicle are obtained, the resulting dynamic model is validated by comparing it with the observed behavior the real car. During the validation, two types of experiments are performed. In both cases the full state and the control inputs of the vehicle have been recorded at 25 Hz. After the experiments, using the log files, the same control input sequence is applied on the simulated vehicle, that uses the identified dynamic model in (4.12).

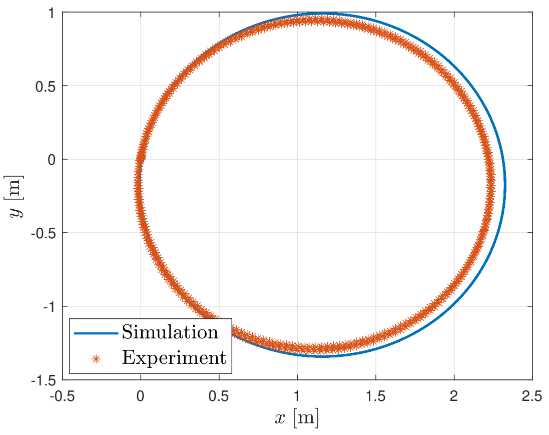
First, the validation procedure is carried out with manually performed maneuvers. Here, the remote controller application of the Command PC is used by a human driver, to send d and δ control inputs directly to the vehicle. As these control inputs are logged during the experiments, a simulation is done using the same input sequence, to compare the accuracy of the model to the real vehicle. Two experiments have been performed: one for forward and one for backward motion. The results of these experiments are shown in Figure 4.7 where both the logged and the simulated position are displayed. Since these are open-loop simulations, some divergence from the logged trajectory is expected, as time proceeds, due to numerical errors and integrated effects of small inaccuracies.



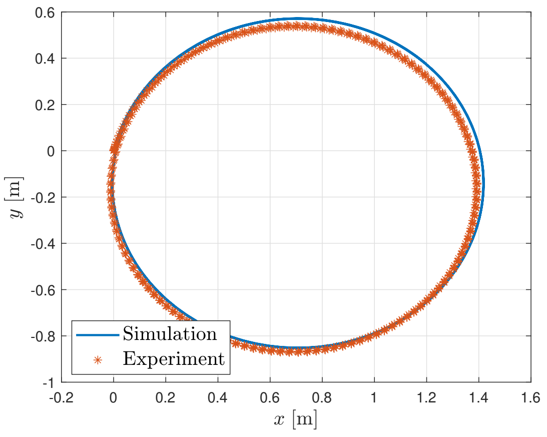
(a) Forward motion. **(b)** Backward motion.

Figure 4.7: The results of the manually driven validation experiments. The maximal deviation of the simulated path compared to the logged trajectory was 0.168 [m] in the forward moving scenario (fig a) and 0.203 [m] in the backward motion scenario (fig b).

The second type of validation experiment is a circular motion test where the vehicle is driven around a circle with a constant steering angle. It was performed with $\delta = \pm 0.3$ rad and $\delta = \pm 0.5$ rad while the motor reference input was $d = 0.06$. After the measurements, the same control inputs are applied to the simulated vehicle model. Using the simulation results and the measured states, the model and the real car is once again compared with each other. The results of these validations are depicted in Figure 4.8.



(a) $\delta = 0.3$ rad steering angle.



(b) $\delta = 0.5$ rad steering angle.

Figure 4.8: Results of the circular motion validation experiments.

As the experiments show, deviation that develops over time between the simulated response and the measured one is relatively small. Such model inaccuracy can be easily handled by integral action added to model based controller design, therefore the identified model is deemed satisfactory for further utilization.

5 Control design for trajectory tracking

The aim of the following sections is to develop accurate path-following algorithms for the F1TENTH vehicle, utilizing the previously derived and identified model.

5.1 Control problem formulation

The trajectories to follow are given as two dimensional spline curves $\psi(s^{\mathrm{ref}})$, defined by coordinate functions $(x(s^{\mathrm{ref}}), y(s^{\mathrm{ref}}))$. Both $x(s^{\mathrm{ref}})$, and $y(s^{\mathrm{ref}})$ are monotonic in s^{ref} and we assume $0 \leq s^{\mathrm{ref}} \leq s^{\mathrm{ref}}_{\mathrm{max}}$, where $(x(0), y(0))$ and $(x(s^{\mathrm{ref}}_{\mathrm{max}}), y(s^{\mathrm{ref}}_{\mathrm{max}}))$ assign the endpoints of the curve. The speed profile $v^{\mathrm{ref}}(s^{\mathrm{ref}})$ along the trajectory is also given. During this work, the arc length of the prescribed trajectory will be used as the s^{ref} parameter. These types of reference motion trajectories can be obtained by regular path planning algorithms.

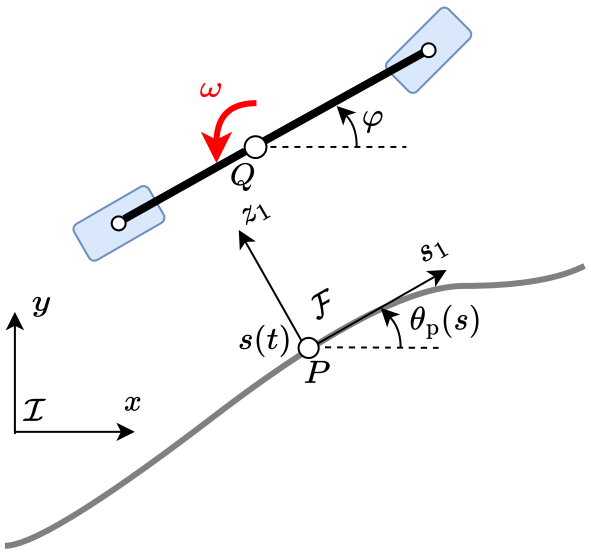


Figure 5.1: Vehicle coordinates in the global and in the moving coordinate frames.

To formulate the control objective, it is useful to express the vehicle model in path coordinates, as described in [28]. The trajectory tracking scenario is depicted in Figure 5.1. $\mathcal{I}$ represents the fixed global coordinate, P is the reference point that is moving along the desired trajectory $\psi(s)$. $\mathcal{F}$ is the moving coordinate frame that is associated with P so that one of its axes is tangent to the trajectory curve. The signed curvilinear abscissa of P along the path is denoted by s. The center of mass of the vehicle Q can be expressed in both $\mathcal{I}$ as (x, y) and in $\mathcal{F}$ as (s_1, z_1). Additionally, the rotation matrix from $\mathcal{I}$ to $\mathcal{F}$ is

$$R(\theta_{\mathrm{p}}(s)) = \begin{bmatrix} \cos(\theta_{\mathrm{p}}(s)) & \sin(\theta_{\mathrm{p}}(s)) \\ -\sin(\theta_{\mathrm{p}}(s)) & \cos(\theta_{\mathrm{p}}(s)) \end{bmatrix}, \tag{5.1}$$

where $\theta_{\mathrm{p}}(s)$ is the angle of the path tangent measured from the x axis. Let the curvature of the path at s be denoted as $c_{\mathrm{c}}(s)$. Moreover, define $\omega_{\mathrm{p}}(s) = \dot{\theta}_{\mathrm{p}}(s)$. Then $\omega_{\mathrm{p}}(s)$ can be expressed as

$$\omega_{\mathrm{p}}(s) = c_{\mathrm{c}}(s)\dot{s}. \tag{5.2}$$

From [28], the dynamics of Q can be expressed in the moving coordinate frame as

$$\dot{s}_1 = \begin{bmatrix}\cos(\theta_{\mathrm{p}}(s)) & \sin(\theta_{\mathrm{p}}(s))\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} - \dot{s}\left(1 - c_{\mathrm{c}}(s)z_1\right), \tag{5.3a}$$

$$\dot{z}_1 = \begin{bmatrix}-\sin(\theta_{\mathrm{p}}(s)) & \cos(\theta_{\mathrm{p}}(s))\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} - c_{\mathrm{c}}(s)\dot{s}s_1. \tag{5.3b}$$

The control objective is to achieve $s_1(t) = 0$ and $z_1(t) = 0$ over the entire reference trajectory, as the vehicle travels through it. In the following parts of this section, the previously introduced kinematic and dynamic models are expressed in terms of path coordinates using (5.3). After these so-called path-following models are obtained, model based control solutions can be developed to achieve the control objective.

First, the kinematic bicycle model is expressed in path coordinates. Substituting (3.1a) into (5.3) yields:

$$\dot{s}_1 = v_\xi \cos(\varphi - \theta_{\mathrm{p}}) - \dot{s}(1 - c_c(s))z_1, \tag{5.4a}$$

$$\dot{z}_1 = v_\xi \sin(\varphi - \theta_{\mathrm{p}}) - c_c(s)\dot{s}z_1, \tag{5.4b}$$

$$\dot{\varphi} - \dot{\theta}_{\mathrm{p}}(s) = \frac{v_\xi}{l}\tan(\delta) - c_{\mathrm{c}}(s)\dot{s}, \tag{5.4c}$$

$$\dot{v}_\xi = a. \tag{5.4d}$$

Let the heading error and its derivative be defined as $\theta_{\mathrm{e}} = \varphi - \theta_{\mathrm{p}}(s)$ and $\dot{\theta}_{\mathrm{e}} = \dot{\varphi} - \dot{\theta}_{\mathrm{p}}(s)$. Furthermore, if P is constructed by projecting Q onto the trajectory $\dot{s}_1 = s_1 = 0$ will hold, which means that the control objective $s_1 = 0$ is achieved, and (5.4) can be simplified as

$$\dot{s} = \frac{v_\xi \cos(\theta_{\mathrm{e}})}{1 - c_{\mathrm{c}}(s)z_1}, \tag{5.5a}$$

$$\dot{z}_1 = v_\xi \sin(\theta_{\mathrm{e}}), \tag{5.5b}$$

$$\dot{\theta}_{\mathrm{e}} = \frac{v_\xi}{l}\tan(\delta) - c_{\mathrm{c}}(s)\dot{s}, \tag{5.5c}$$

$$\dot{v}_\xi = a. \tag{5.5d}$$

Next, the dynamic path-following model is calculated. By substituting the vehicle dynamics from (4.12) into (5.3), the motion of the dynamic vehicle model can also be expressed in $\mathcal{F}$ with the following equations:

$$\dot{s}_1 = v_\xi \cos(\varphi - \theta_{\mathrm{p}}(s)) - v_\eta \sin(\varphi - \theta_{\mathrm{p}}(s)) - \dot{s}\left(1 - c_{\mathrm{c}}(s)z_1\right), \tag{5.6a}$$

$$\dot{z}_1 = v_\xi \sin(\varphi - \theta_{\mathrm{p}}(s)) + v_\eta \cos(\varphi - \theta_{\mathrm{p}}(s)) - c_{\mathrm{c}}(s)\dot{s}s_1. \tag{5.6b}$$

With executing the same projection step ($\dot{s}_1 = s_1 = 0$) as in the kinematic model and the substitution of the introduced heading error term (θ_e), the model can be expressed as:

$$\begin{aligned}
\dot{s} &= \frac{v_\xi \cos(\theta_\mathrm{e}) - v_\eta \sin(\theta_\mathrm{e})}{1 - c_\mathrm{c}(s) z_1}, && (5.7\mathrm{a})\\
\dot{z}_1 &= v_\xi \sin(\theta_\mathrm{e}) + v_\eta \cos(\theta_\mathrm{e}) && (5.7\mathrm{b})\\
\dot{\theta}_\mathrm{e} &= \omega - c_\mathrm{c}(s)\dot{s}, && (5.7\mathrm{c})\\
\dot{v}_\xi &= \frac{1}{m}\left(F_\xi + F_\xi \cos(\delta) - F_{\mathrm{f},\eta}\sin(\delta) + m v_\eta \omega\right), && (5.7\mathrm{d})\\
\dot{v}_\eta &= \frac{1}{m}\left(F_{\mathrm{r},\eta} + F_\xi \sin(\delta) + F_{\mathrm{f},\eta}\cos(\delta) - m v_\xi \omega\right), && (5.7\mathrm{e})\\
\dot{\omega} &= \frac{1}{I_z}\left(F_{\mathrm{f},\eta} l_\mathrm{f} \cos(\delta) + F_{\mathrm{f},\xi} l_\mathrm{f} \sin(\delta) - F_{\mathrm{r},\eta} l_\mathrm{r}\right), && (5.7\mathrm{f})\\
F_\xi &= C_\mathrm{m1} d - C_\mathrm{m2} v_\xi - \mathrm{sign}(v_\xi) C_\mathrm{m3}, && (5.7\mathrm{g})\\
F_{\mathrm{r},\eta} &= C_\mathrm{r}\alpha_\mathrm{r}, && (5.7\mathrm{h})\\
F_{\mathrm{f},\eta} &= C_\mathrm{f}\alpha_\mathrm{f}, && (5.7\mathrm{i})\\
\alpha_\mathrm{r} &= \frac{-v_\eta + l_\mathrm{r}\omega}{v_\xi}, && (5.7\mathrm{j})\\
\alpha_\mathrm{f} &= \delta - \frac{v_\eta + l_\mathrm{f}\omega}{v_\xi}. && (5.7\mathrm{k})
\end{aligned}$$

It is important to note that these path-following models introduce a singularity at $z_1 = (c_\mathrm{c}(s))^{-1}$. As the maximal steering angle of the vehicle is known from Section 4.3, the upper limit of $c_\mathrm{c}(s)$ can be obtained using (4.5):

$$\max_s\{c_\mathrm{c}(s)\} = \frac{1}{R_\mathrm{max}} = \frac{\sin(\delta_\mathrm{max})}{l} = 1.44\,\frac{1}{\mathrm{m}}, \tag{5.8}$$

which results in the following limit for z_1:

$$\max_s\{z_1\} = \frac{1}{\max_s\{c_\mathrm{c}(s)\}} = 0.69\ \mathrm{m}. \tag{5.9}$$

However, this limitation should not raise any concerns, because a precise path tracking controller should not allow such high deviation from the reference trajectory.

5.2 Decoupling of the dynamics

In the following sections, to simplify the control design procedure, the dynamic nonlinear model in (5.7) is decoupled into two SISO subsystems, describing the lateral and longitudinal motion of the vehicle, respectively. This way two independent controllers can be designed for stabilizing the whole dynamic system. In case of the lateral motion, a decoupled kinematic model is also introduced for the control design of backward moving vehicles.

The first, longitudinal controller is designed to track the the speed profile $v^\mathrm{ref}(s)$ along the path accurately by actuating the motor reference input d. The second, lateral controller is responsible for path tracking by controlling the steering angle δ of the system. The overall control structure is depicted in Figure 5.2. From the vehicle position data, a state estimator module is implemented which can calculate the full state of the vehicle by numerical differentiation. Using this data, full state feedback control algorithms are designed.

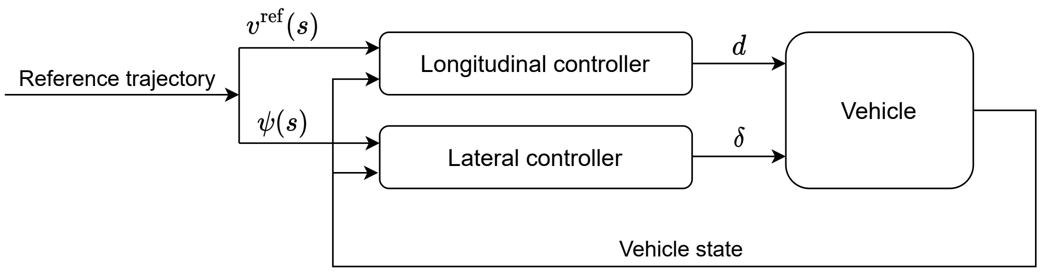


Figure 5.2: The proposed path-following control structure.

5.2.1 Longitudinal dynamics

In this section, the longitudinal dynamics of the vehicle is described with respect to the desired path. Considering a forward moving vehicle, $\dot{s}$ from (5.7) can be expressed as

$$\dot{s} = \frac{v\cos(\gamma)}{1 - c_{\text{c}}(s)z_1}, \tag{5.10}$$

where $v = \sqrt{v_\xi^2 + v_\eta^2}$ is the absolute velocity of the center of mass and γ is the angle between the path tangent and v. The geometric relations are also shown in Figure 5.3.

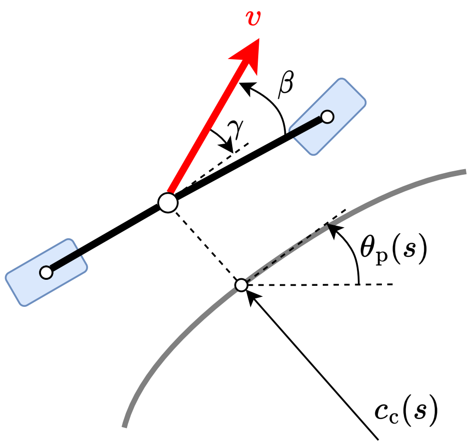


Figure 5.3: The geometric relations of the decoupled longitudinal model.

In this case, the absolute velocity v can also be expressed as

$$v = \frac{v_\xi}{\cos(\beta)}, \tag{5.11}$$

where $\beta = \text{arctan2}(v_\eta, v_\xi)$ is the side slip angle of the center of mass. Using β, γ can also be expressed:

$$\gamma = \varphi + \beta - \theta_\text{p}(s). \tag{5.12}$$

Considering (4.12d), it can be observed that the lateral dynamics only have substantial effects on the longitudinal characteristics when the vehicle drives above 3 m/s. However, due to the limited space available in the lab, it would not be safe to operate the cars with such high velocities. Therefore, under the assumption of operation speed lower than 3 m/s, the lateral parts can be neglected from the longitudinal dynamics which results in the following motion equation:

$$\dot{v}_\xi = \frac{2}{m}\left(C_\text{m1}d - C_\text{m2}v_\xi - \text{sign}(v_\xi)C_\text{m3}\right). \tag{5.13}$$

From (5.10) and (5.13), the longitudinal dynamics can be described as

$$\dot{s} = \frac{v_\xi \cos(\gamma)}{\cos(\beta)(1 - c_\text{c}(s)z_1)}, \tag{5.14a}$$

$$\dot{v}_\xi = \frac{2}{m}\left(C_\text{m1}d - C_\text{m2}v_\xi - \text{sign}(v_\xi)C_\text{m3}\right). \tag{5.14b}$$

By introducing scheduling variable $p = \frac{\cos(\gamma)}{\cos(\beta)(1-c_\text{c}(s)z_1)}$, a linear parameter-varying (LPV) model can be obtained:

$$\begin{bmatrix}\dot{s}\\ \dot{v}_\xi\end{bmatrix} = \begin{bmatrix}0 & p\\ 0 & -\frac{2C_\text{m2}}{m}\end{bmatrix}\begin{bmatrix}s\\ v_\xi\end{bmatrix} + \begin{bmatrix}0\\ \frac{2C_\text{m1}}{m}\end{bmatrix}d + \begin{bmatrix}0\\ -\frac{2C_\text{m3}}{m}\end{bmatrix}\text{sign}(v_\xi). \tag{5.15}$$

To design a gain scheduling or parameter varying controller for this model, the range of p must be determined. The calculation of the absolute upper limit of p is straight forward:

$$\max\{p\} = \frac{1}{\cos\left(\max\{\beta\}\right)\left(1 - \max\{c_\text{c}(s)\}\max\{z_1\}\right)}. \tag{5.16}$$

From the data collected during the high speed circular motion tests in Section 4.5, $\max\{\beta\} = 0.16$. The extremum of z_1 depends on the efficiency of the lateral path tracking controller. For normal operation, $\max\{z_1\} = 0.2$ m is a reasonable limit. By substituting the maximal values into (5.16), the range of p can be determined:

$$p \in (-\max\{p\}, \max\{p\}) = (-3.52,\ 3.52). \tag{5.17}$$

5.2.2 Lateral dynamics

After the longitudinal dynamics, the lateral behavior of the system is also decoupled to obtain the models for the controller design. First, the lateral system based on the kinematic model is derived, which will be used for the control design of a reversing vehicle. This system can be described by the following equations:

$$\dot{z}_1 = v_\xi \sin(\theta_\text{e}), \tag{5.18a}$$

$$\dot{\theta}_\text{e} = \frac{v_\xi}{l}\tan(\delta) - c_\text{c}(s)\dot{s}. \tag{5.18b}$$

Assuming small steering angles and heading errors, the trigonometric functions can be linearized as $\sin(\alpha) \approx \alpha$ and $\tan(\alpha) \approx \alpha$. Furthermore, v_ξ is considered as a scheduling-parameter to obtain a linear parameter-varying model:

$$\begin{bmatrix} \dot{z}_1 \\ \dot{\theta}_\mathrm{e} \end{bmatrix} = \begin{bmatrix} 0 & v_\xi \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ \theta_\mathrm{e} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{v_\xi}{l} \end{bmatrix} \delta \begin{bmatrix} 0 \\ 1 \end{bmatrix} \omega_\mathrm{p}(s). \tag{5.19}$$

As (5.19) solely relies on the kinematics of the vehicle, better control performance can be achieved if the control algorithm is based on the dynamic vehicle model. Therefore, the more complex, dynamic single track model is also decoupled, to obtain a more accurate representation of the lateral behavior.

The lateral dynamics of the full nonlinear dynamic model in (4.12) can be expressed as

$$\dot{v}_\eta = \frac{C_\mathrm{r}(-v_\eta + l_\mathrm{r}\omega)}{m v_\xi} + \frac{F_\xi}{m}\sin(\delta) + \frac{C_\mathrm{f}}{m}\left(\delta - \frac{v_\eta + l_\mathrm{f}\omega}{v_\xi}\right)\cos(\delta) - v_\xi \omega, \tag{5.20a}$$

$$\dot{\omega} = \frac{C_f}{I_z}\left(\delta - \frac{v_\eta + l_\mathrm{f}\omega}{v_\xi}\right) l_\mathrm{f}\cos(\delta) + \frac{F_\xi}{I_z} l_\mathrm{f}\sin(\delta) - \frac{C_\mathrm{r}}{I_z}\left(\frac{-v_\eta + l_\mathrm{r}\omega)}{v_\xi}\right) l_\mathrm{r}. \tag{5.20b}$$

(5.20c)

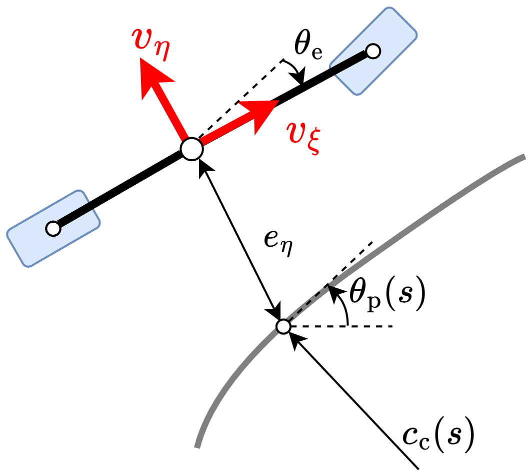


Figure 5.4: Lateral dynamics in path coordinates.

If the longitudinal acceleration is assumed to be minimal during normal operation, the longitudinal tire force can be neglected ($F_\xi = 0$ N) to simplify the model. Here, the longitudinal velocity (v_ξ) is also considered as a scheduling parameter for an LPV model. Furthermore, for small steering angles (less than 0.5 rad) $\sin(\delta) \approx \delta$, $\cos(\delta) \approx 1$ approximations can be used to linearize the model:

$$\dot{v}_\eta = \frac{-(C_\mathrm{r} + C_\mathrm{f})}{m v_\xi} v_\eta + \left(\frac{C_\mathrm{r} l_\mathrm{r} - C_\mathrm{f} l_\mathrm{f}}{m v_\xi} - v_\xi\right)\omega + \frac{C_\mathrm{f}}{m}\delta, \tag{5.21a}$$

$$\dot{\omega} = \frac{l_\mathrm{r} C_\mathrm{r} - l_\mathrm{f} C_\mathrm{f}}{I_z v_\xi} v_\eta + \frac{-(l_\mathrm{f}^2 C_\mathrm{f} + l_\mathrm{r}^2 C_\mathrm{r})}{I_z v_\xi}\omega + \frac{l_\mathrm{f} C_\mathrm{f}}{I_z}\delta. \tag{5.21b}$$

The obtained equations can also be reorganized into the standard state-space representation:

$$\begin{bmatrix} \dot{v}_\eta \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{-(C_\mathrm{r}+C_\mathrm{f})}{mv_\xi} & \frac{C_\mathrm{r}l_\mathrm{r}-C_\mathrm{f}l_\mathrm{f}}{mv_\xi} - v_\xi \\ \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{I_z v_\xi} & \frac{-(l_\mathrm{f}^2C_\mathrm{f}+l_\mathrm{r}^2C_\mathrm{r})}{I_z v_\xi} \end{bmatrix} \begin{bmatrix} v_\eta \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{C_\mathrm{f}}{m} \\ \frac{C_\mathrm{f}l_\mathrm{f}}{I_z} \end{bmatrix} \delta. \tag{5.22}$$

The next step is to include the path dynamics into the lateral model. Using the notation described in Section 5.1, the reference yaw rate derived from the path can be expressed as

$$\omega_\mathrm{p}(s) = c_\mathrm{c}(s)|v_\xi|, \tag{5.23}$$

and the path defined lateral acceleration is

$$\dot{v}_\eta(s) = c_\mathrm{c}(s)v_\xi^2. \tag{5.24}$$

The introduction of the absolute sign in (5.23) is necessary for the model to be valid in reversing path tracking scenarios. Let the lateral error e_η be defined as the orthogonal distance of the center of mass from the path. Furthermore, the heading error and its derivative are denoted and calculated as $\theta_e = \varphi - \theta_p(s)$ and $\dot{\theta}_\mathrm{e} = \omega - \omega_\mathrm{p}(s)$. The introduced variables are also depicted in Figure 5.4. The dynamics of e_η is described by

$$\ddot{e}_\eta = (\dot{v}_\eta + v_\xi\omega) - \dot{v}_\eta(s) = \dot{v}_\eta + v_\xi(\omega - \omega_\mathrm{p}(s)) = \dot{v}_\eta + v_\xi\dot{\theta}_\mathrm{e}, \tag{5.25a}$$

$$\dot{e}_\eta = v_\eta + v_\xi \sin(\theta_\mathrm{e}). \tag{5.25b}$$

Substituting (5.25) into (5.21) yields the following equations, where the lateral dynamics is expressed with the lateral and heading errors as state variables:

$$\begin{bmatrix} \dot{e}_\eta \\ \ddot{e}_\eta \\ \dot{\theta}_\mathrm{e} \\ \ddot{\theta}_\mathrm{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(C_\mathrm{r}+C_\mathrm{f})}{mv_\xi} & \frac{C_\mathrm{r}+C_\mathrm{f}}{m} & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{mv_\xi} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{I_z v_\xi} & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{I_z} & \frac{-(l_\mathrm{r}^2C_\mathrm{r}+l_\mathrm{f}^2C_\mathrm{f})}{I_z v_\xi} \end{bmatrix} \begin{bmatrix} e_\eta \\ \dot{e}_\eta \\ \theta_\mathrm{e} \\ \dot{\theta}_\mathrm{e} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{C_\mathrm{f}}{m} \\ 0 \\ \frac{C_\mathrm{f}l_\mathrm{f}}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{mv_\xi} - v_\xi \\ 0 \\ \frac{-(l_\mathrm{r}^2C_\mathrm{r}+l_\mathrm{f}^2C_\mathrm{f})}{I_z v_\xi} \end{bmatrix} \omega_\mathrm{p}(s). \tag{5.26}$$

5.3 Longitudinal control

As a result of the previous modelling presented in Section 5.2.1, an individual SISO LPV model has been obtained that describes the longitudinal behavior of the vehicle. Based on this model, a longitudinal controller is designed for the vehicle, which is responsible for tracking the given speed profile, that is defined along the path as $v^\mathrm{ref}(s)$. However, it is also important to track the time evolution of the position along the reference trajectory. To achieve this objective, model-based feedforward and feedback controllers are developed.

First, Section 5.3.1 introduces a gain-scheduled control design, then the LPV method is discussed in Section 5.3.2.

5.3.1 Gain-scheduled control

The desired position of the vehicle along the defined path can be calculated by the integration of the speed profile as $s^{\text{ref}} = \int_0^t v^{\text{ref}}(s)\mathrm{d}t$, therefore the desired state reference can be defined as

$$x^{\text{ref}} = \begin{bmatrix} s^{\text{ref}} \\ v^{\text{ref}}(s) \end{bmatrix}, \tag{5.27}$$

and the LPV model to control in state-state representation is

$$\underbrace{\begin{bmatrix} \dot{s} \\ \dot{v}_\xi \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & p \\ 0 & -\frac{2C_{\text{m2}}}{m} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} s \\ v_\xi \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{2C_{\text{m1}}}{m} \end{bmatrix}}_{B} \underbrace{d}_{u} + \underbrace{\begin{bmatrix} 0 \\ -\frac{2C_{\text{m3}}}{m} \end{bmatrix}}_{E} \underbrace{\text{sign}(v_\xi)}_{\epsilon}, \tag{5.28}$$

where the value of ϵ is either -1 or 1, depending on the direction. Therefore, it is considered as another scheduling variable next to p. As (5.28) is an LPV model with scheduling variables ϵ and p, gain scheduled controllers are designed.

To achieve the desired state reference, the following feedforward-feedback control law is applied:

$$u = u_{\text{ff}} + u_{\text{fb}}, \tag{5.29}$$

where u_{ff} is the feedforward and u_{fb} is the feedback term of the control input. The feedforward term is calculated from the steady-state condition of the system and the feedback term is obtained with a gain scheduled LQR controller.

First the **feedforward term** of the control law is calculated. In steady-state condition, $\dot{x}^{\text{ss}} = 0$ holds and the steady-state vector needs to be equal to the control reference ($x^{\text{ss}} = x^{\text{ref}}$) which results in

$$0 = Ax^{\text{ref}} + Bu_{\text{ff}} + E\epsilon, \tag{5.30}$$

where u_{ff} is the feedforward control input, required to achieve $x^{\text{ss}} = x^{\text{ref}}$. From (5.30), u_{ff} can be expressed as

$$u_{\text{ff}} = -(B^TB)^{-1}B^T(Ax^{\text{ref}}+E\epsilon) = \frac{m\left(C_{\text{m2}}v^{\text{ref}}(s) + C_{\text{m3}}\text{sign}(v_\xi)\right)}{C_{\text{m1}}} = f_{long,1}v^{\text{ref}}(s) + f_{\text{long},2}(\epsilon). \tag{5.31}$$

Next, the feedforward term is complemented with an LQR based **feedback term** to improve the disturbance rejection and the system transients.

First, introduce error variables as

$$\Delta x = x - x^{\text{ref}}, \tag{5.32a}$$

$$\Delta u = u - u_{\text{ff}}. \tag{5.32b}$$

Substituting (5.32) into the system dynamics in (5.28) yields

$$\dot{x} = \dot{x}^{\text{ss}} + \Delta\dot{x} = A(x^{\text{ref}} + \Delta x) + B(u_{\text{ff}} + \Delta u). \tag{5.33}$$

As $\dot{x}^{\text{ss}} = Ax^{\text{ref}} + Bu_{\text{ff}} = 0$ thanks to the feedforward term, (5.33) can be simplified as

$$\Delta\dot{x} = A\Delta x + B\Delta u, \tag{5.34a}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} s - s^{\text{ref}} \\ v_\xi - v^{\text{ref}}(s)\end{bmatrix} = \begin{bmatrix} 0 & p \\ 0 & -\frac{2C_{\text{m2}}}{m}\end{bmatrix}\begin{bmatrix} s - s^{\text{ref}} \\ v_\xi - v^{\text{ref}}(s)\end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2C_{\text{m1}}}{m}\end{bmatrix} d. \tag{5.34b}$$

As the controllability matrix of this system $M_{\text{c}} = \begin{bmatrix} B & AB \end{bmatrix} = 2$ has full rank for the whole range of the scheduling variables, the system can be stabilized with full state feedback. Next, the design process follows the steps detailed in Section B.2.2. After the system is discretized with the zero order hold method using 0.025 s sampling time, a discrete time, infinite horizon LQR based feedback control law is obtained, that has the following form:

$$u_{\text{fb}}[k] = -K^{\text{GS}}_{\text{long}}(p)\Delta x[k] = -\begin{bmatrix} k^{\text{GS}}_{\text{long},1}(p) & k^{\text{GS}}_{\text{long},2}(p)\end{bmatrix}\begin{bmatrix} s[k] - s^{\text{ref}}[k] \\ v_\xi[k] - v^{\text{ref}}(s)[k]\end{bmatrix}, \tag{5.35}$$

$k^{\text{LPV}}_{\text{long},2}(p)$ are third order polynomials.

The controller can be tuned by altering Q and R weighting matrices. The weights are chosen to emphasize accurate position tracking compared to the speed profile tracking and the penalty of the control input is increased to avoid hard transients:

$$Q = \begin{bmatrix} 20 & 0 \\ 0 & 2\end{bmatrix}, \qquad R = 100. \tag{5.36a}$$

In the case of the longitudinal control of a reversing vehicle, the above proposed algorithm can perform effectively, only minor modifications are required. While the curvilinear abscissa of the path (s) has been defined as a non-negative number in Section 5.1, s^{ref} needs to redefined for reversing motion as $s^{\text{ref}} = -\int_0^t v^{\text{ref}}(s)\mathrm{d}t$.

5.3.2 LPV control

After the gain-scheduling control has been successfully tested in simulations, a more advanced LPV-LQR controller is also synthesised. This controller is also split into two main components.

$$u = u_{\text{ff}} + u_{\text{fb}}. \tag{5.37}$$

The **feedforward** term is the same as that used in the gain-scheduled controller:

$$u_{\text{ff}} = f_{long,1}v^{\text{ref}}(s) + f_{\text{long},2}(\epsilon). \tag{5.38}$$

The **feedback** term is also calculated from the error model introduced in (5.34a). Using the LPV-LQR controller introduced in Section B.2.3, the following controller can be obtained:

$$u_{\text{fb}}[k] = -K^{\text{LPV}}_{\text{long}}(p)\Delta x[k] = -\begin{bmatrix} k^{\text{LPV}}_{\text{long},1}(p) & k^{\text{LPV}}_{\text{long},2}(p)\end{bmatrix}\begin{bmatrix} s[k] - s^{\text{ref}}[k] \\ v_\xi[k] - v^{\text{ref}}(s)[k]\end{bmatrix} \tag{5.39}$$

where $k^{\mathrm{LPV}}_{\mathrm{long},1}(p)$ and $k^{\mathrm{LPV}}_{\mathrm{long},2}(p)$ are third order polynomials.

Tuning the controller can be done by altering the Q and R matrices. Based on simulation results, the following weights are chosen, to penalize the position error more aggressively compared to the velocity error:

$$Q^{\mathrm{LPV}} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R^{\mathrm{LPV}} = 1000. \tag{5.40a}$$

There is a clear difference between the weighting matrices of the gain-scheduled and the LPV controller. This is a result of the fact that the LPV controller considers the whole operating range and not only the local design points. Moreover, the LMI-based method is a suboptimal solution to the LQR problem, which can also result in deviation from the Ricatti equation based solution.

5.4 Lateral control

After the longitudinal controller of the vehicle has been successfully designed, the lateral control of the vehicle is the next step. The objective of the lateral controller is to enable accurate path tracking, by eliminating the lateral and the heading errors. All the control algorithms developed are model-based feedback control methods, using the models derived in Section 5.2.2.

This section heavily relies on the work presented in [27]. However, unlike in [27], v_ξ longitudinal velocity is considered as a varying parameter. Therefore, gain-scheduled controllers are developed first, with the design steps outlined in B.2.2. Then, after finding a sufficient control architecture, more advanced LPV-LQR controllers are designed, which is based on the theory introduced in Section B.2.3.

5.4.1 Gain-scheduled LQR feedback control

The first solution for the lateral control is a simple gain-scheduled stabilizing feedback control that is based mainly on [28]. From (5.26), the system and can be expressed as

$$\underbrace{\begin{bmatrix} \dot{e}_\eta \\ \ddot{e}_\eta \\ \dot{\theta}_{\mathrm{e}} \\ \ddot{\theta}_{\mathrm{e}} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(C_{\mathrm{r}}+C_{\mathrm{f}})}{mv_\xi} & \frac{C_{\mathrm{r}}+C_{\mathrm{f}}}{m} & \frac{l_{\mathrm{r}}C_{\mathrm{r}}-l_{\mathrm{f}}C_{\mathrm{f}}}{mv_\xi} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_{\mathrm{r}}C_{\mathrm{r}}-l_{\mathrm{f}}C_{\mathrm{f}}}{I_z v_\xi} & \frac{l_{\mathrm{r}}C_{\mathrm{r}}-l_{\mathrm{f}}C_{\mathrm{f}}}{I_z} & \frac{-(l_{\mathrm{r}}^2C_{\mathrm{r}}+l_{\mathrm{f}}^2C_{\mathrm{f}})}{I_z v_\xi} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} e_\eta \\ \dot{e}_\eta \\ \theta_{\mathrm{e}} \\ \dot{\theta}_{\mathrm{e}} \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ \frac{C_{\mathrm{f}}}{m} \\ 0 \\ \frac{C_{\mathrm{f}}l_{\mathrm{f}}}{I_z} \end{bmatrix}}_{B} \delta + \underbrace{\begin{bmatrix} 0 \\ \frac{l_{\mathrm{r}}C_{\mathrm{r}}-l_{\mathrm{f}}C_{\mathrm{f}}}{mv_\xi} - v_\xi \\ 0 \\ \frac{-(l_{\mathrm{r}}^2C_{\mathrm{r}}+l_{\mathrm{f}}^2C_{\mathrm{f}})}{I_z v_\xi} \end{bmatrix}}_{E} \underbrace{\omega_{\mathrm{p}}(s)}_{\epsilon}, \tag{5.41}$$

where δ is the control input and ϵ is considered as external disturbance. As the disturbance $\epsilon = \omega_{\mathrm{p}}(s) = c_{\mathrm{c}}(s)|v_\xi|$ is continuous and only changes slowly, it is neglected from the control design.

After the model is specified, the control design procedure follows the steps presented in Section B.2.2. First, the controllability of the system is checked by calculating the rank of the controllability matrix $M_\mathrm{c} = \begin{bmatrix} B & AB & A^2B & A^3B \end{bmatrix}$. As $\operatorname{rank}(M_\mathrm{c}) = 4$ for the whole range of v_ξ scheduling parameter, the system is controllable, therefore it can be stabilized with full state feedback. As the implemented algorithms will run on an embedded computer, a discrete time feedback control is designed. Accordingly, the continuous system discretized with zero order hold method, using 0.025 s sampling time. Then, for the discrete time system, a full state feedback control is designed which has the following linear law:

$$\delta[k] = -K^{\mathrm{GS}}_{\mathrm{lat,LQR}}(v_\xi)x[k]. \tag{5.42}$$

The controller is designed by LQR technique, to enable convenient tuning by the alteration of Q and R weighting matrices. To increase the cost of the lateral error compared to the heading error, the following gains are chosen:

$$Q^{\mathrm{GS}} = \begin{bmatrix} 139 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad R^{\mathrm{GS}} = 100. \tag{5.43a}$$

After calculating the optimal gains for different v_ξ values, ranging from 0.5 m/s to 3.5 m/s with 0.05 m/s steps, a third order polynomial is fitted on the gain-velocity data pairs to obtain the parameter varying feedback gains.

To properly evaluate the performance of the designed controller, the simulator package detailed in Section C is used, which utilizes the nonlinear dynamic vehicle model identified in Section 4.6. The reference trajectory is a 2 dimensional spline curve that is completely realizable with the vehicle and the reference velocity is constant 1.5 m/s for the whole track. The comparison between the reference and the realized trajectory is shown in Figure 5.5.

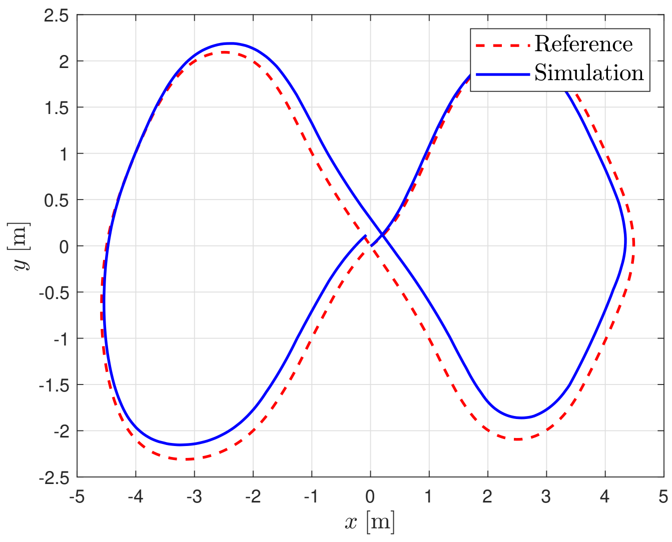


Figure 5.5: Reference path and the actual path of the vehicle using state feedback

It is clear from the figures that although the system has been successfully stabilized, it builds up significant lateral error, as a result of the external disturbance. Therefore this steering method fails to deliver accurate path tracking and improvements are necessary, which are presented in the following sections.

5.4.2 Gain-scheduled LQ-servo control

This section presents an improved control design approach, the LQ-servo, to compensate the effects of external disturbance ϵ. In the LQ-servo, the integral of the lateral error $q = \int e_\eta \, \mathrm{d}t$ is introduced as a state variable. The extended system in state-space form can be expressed as

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} q \\ e_\eta \\ \dot{e}_\eta \\ \theta_\mathrm{e} \\ \dot{\theta}_\mathrm{e} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-(C_\mathrm{r}+C_\mathrm{f})}{m v_\xi} & \frac{C_\mathrm{r}+C_\mathrm{f}}{m} & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{m v_\xi} \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{I_z v_\xi} & \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{I_z} & \frac{-(l_\mathrm{r}^2C_\mathrm{r}+l_\mathrm{f}^2C_\mathrm{f})}{I_z v_\xi} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q \\ e_\eta \\ \dot{e}_\eta \\ \theta_\mathrm{e} \\ \dot{\theta}_\mathrm{e} \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{C_\mathrm{f}}{m} \\ 0 \\ \frac{C_\mathrm{f} l_\mathrm{f}}{I_z} \end{bmatrix}}_{B} + \delta \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{l_\mathrm{r}C_\mathrm{r}-l_\mathrm{f}C_\mathrm{f}}{m v_\xi} - v_\xi \\ 0 \\ \frac{-(l_\mathrm{r}^2C_\mathrm{r}+l_\mathrm{f}^2C_\mathrm{f})}{I_z v_\xi} \end{bmatrix}}_{E} \underbrace{\omega_\mathrm{p}(s)}_{\epsilon}. \tag{5.44}$$

If this system is stabilized by feedback control, the steady-state condition $\dot{x} = 0$ corresponds to zero lateral error, as $\dot{q} = e_\eta$. This means that the effects of the disturbance can be compensated if the controller is sufficiently fast, compared to the rate of $\omega_\mathrm{p}(s)$.

After the introduction of the extended system, the control design is carried out with the steps detailed in Section B. As

$$\mathrm{rank}(M_\mathrm{c}) = \mathrm{rank}\left(\begin{bmatrix} A & AB & A^2B & A^3B & A^4B \end{bmatrix}\right) = 5 \tag{5.45}$$

for the whole range of v_ξ, the system is controllable, therefore full state feedback is applied. After discretization with 0.025 s sampling time, an LQR state feedback control is designed. The control law has the following form:

$$\delta[k] = -K^{\mathrm{GS}}_{\mathrm{lat,LQ-servo}}(v_\xi)x[k]. \tag{5.46}$$

The weighting matrices of the LQR are tuned to enhance the cost of lateral error compared to the heading error for more accurate path tracking:

$$Q^{\mathrm{GS}} = \begin{bmatrix} 65 & 0 & 0 & 0 & 0 \\ 0 & 328 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad R =^{\mathrm{GS}} 392. \tag{5.47a}$$

The performance of the controller was evaluated using the simulator detailed in Section C. During the simulation the same reference trajectory is used as in Section 5.4.1. A comparison between the the reference and the actual vehicle trajectory is displayed in Figure 5.6.

It is clear, that the application of the LQ-servo can successfully compensate the steady state error of the system. However, another important issue is raised. As a result of the system dynamics, the feedback gains of e_η and θ_e need to have high values to stabilize the system. Furthermore, these values cannot be reduced by altering the weighting matrices of the LQR, as lower gains would result in an unstable closed loop system. Consequently, because of the large gains, there are rapid changes in the control input of the system, as shown in Figure 5.7.

As the dynamics of the steering servo motor is not included in the model, the controller performs well in simulations. However, the real hardware cannot realize such fast changes in the input. Therefore, this control method cannot be implemented on the real system with the same performance as shown in the simulations.

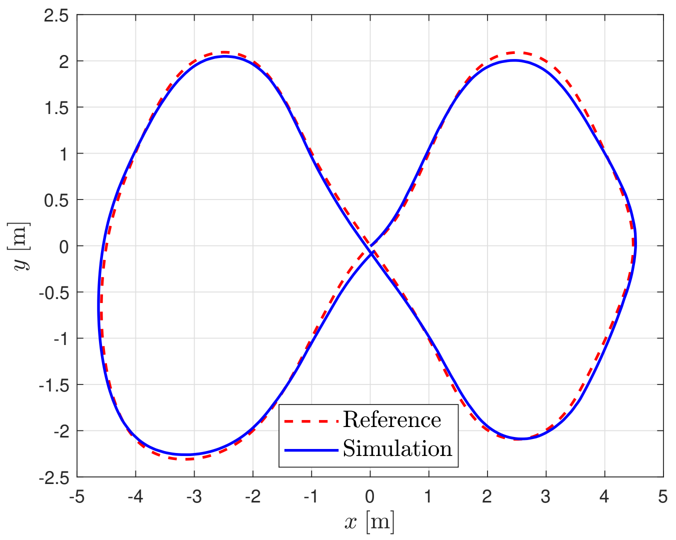


Figure 5.6: Reference track compared to the actual path of the vehicle with the gain-scheduled augmented state feedback control.

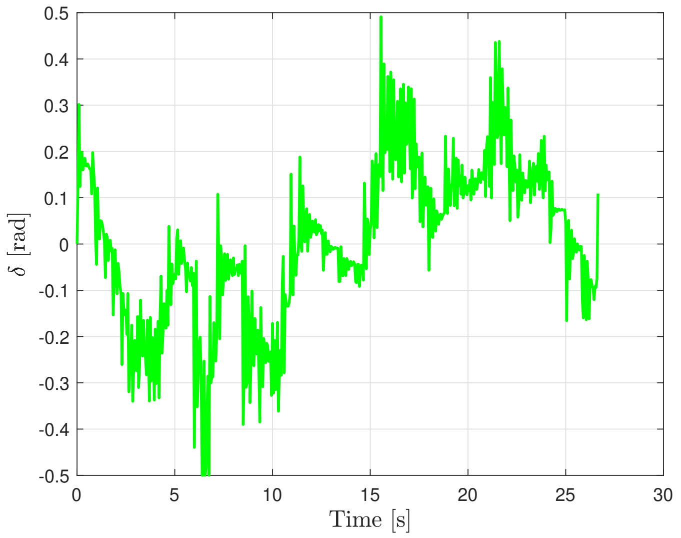


Figure 5.7: The steering inputs of the simulation with the feedback law based on the augmented lateral dynamics.

5.4.3 Enhanced Stanley method with simplified gain-scheduled control

The issues of the controllers designed in the previous sections are all originate from the same problem. The dynamics of the model in (5.26) require high feedback gain of the heading error to be able to stabilize the system. As the lateral and the heading error work against each other, one cannot be eliminated without the temporary increase of the other. While the augmented system produces more accurate results, the gains are increased even further to guarantee the closed loop stability and the control input alternates too quickly to realize it on a real system, or leads to saturation.

To eliminate this undesired effect of the control gains "working against each other", a third control solution is presented. This includes the introduction of a third, simplified lateral model. Here, the dynamics of the heading error is first neglected from the extended system model and considered only as disturbance. However, the integral term from the LQ-servo is preserved, as it has been successfully applied in disturbance compensation. The model can be expressed as:

$$\underbrace{\begin{bmatrix} \dot{q} \\ \dot{e}_\eta \\ \ddot{e}_\eta \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \frac{-(C_\mathrm{r}+C_\mathrm{f})}{mv_\xi} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} q \\ e_\eta \\ \dot{e}_\eta \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{C_\mathrm{f}}{m} \end{bmatrix}}_{B} \underbrace{\delta}_{u} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{C_\mathrm{r}+C_\mathrm{f}}{m} & \frac{-(l_\mathrm{r}^2 C_\mathrm{r}+l_\mathrm{f}^2 C_\mathrm{f})}{I_z v_\xi} & \frac{l_\mathrm{r} C_\mathrm{r}-l_\mathrm{f} C_\mathrm{f}}{mv_\xi} - v_\xi \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} \theta_\mathrm{e} \\ \dot{\theta}_\mathrm{e} \\ \omega_\mathrm{p}(s) \end{bmatrix}}_{\epsilon}. \tag{5.48}$$

As $M_\mathrm{c} = \begin{bmatrix} B & AB & A^2B \end{bmatrix}$ controllability matrix has full rank, the system can be stabilized by full state feedback. After discretization, the control law can be expressed as

$$\delta[k] = -K^{\mathrm{GS}}_{\mathrm{lat,Stanley}}(v_\xi)x[k] \tag{5.49}$$

The optimal feedback gains are calculated using LQR with weighting matrices

$$Q^{\mathrm{GS}} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 15200 & 0 \\ 0 & 0 & 2580 \end{bmatrix}, \qquad R^{\mathrm{GS}} = 2340. \tag{5.50a}$$

The performance of the controller is evaluated in the `fleet1tenth` simulator module. Using the same test trajectory as in the previous sections, a comparison between the actual and the reference path is shown in Figure 5.8.

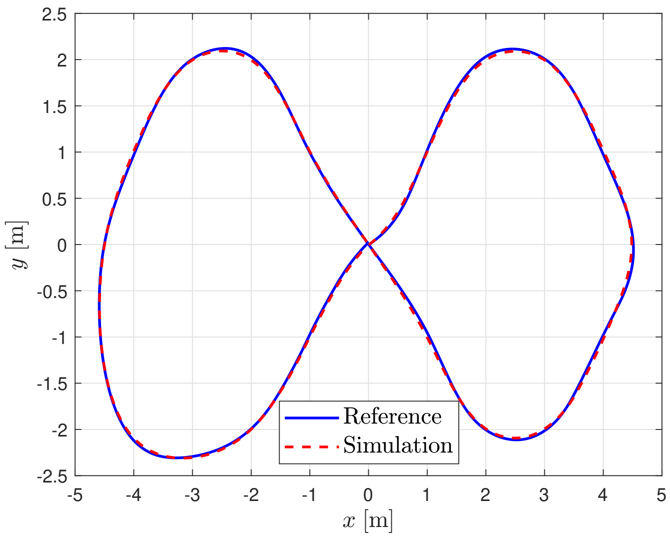


Figure 5.8: Reference track compared to the actual path of the vehicle with the gain-scheduled simplified state feedback control.

Despite the fact that this controller completely neglects the dynamics of the heading error, the tracking already outperforms the previously designed controllers. To further increase the accuracy, θ_{e} is introduced as a feedforward term in the control law, like it is used in the so-called Stanley controller [14]. The final proposed control law can be expressed as

$$\delta[k] = -\theta_{\mathrm{e}}[k] - K^{\mathrm{GS}}_{\mathrm{lat,Stanley}}(v_\xi)x[k]. \tag{5.51}$$

The control algorithm is validated with the same numerical simulation as the previous ones. As it is shown in Figure 5.9, the combined controller with the Stanley-like feed-forward term and the model-based feedback term provide further accuracy improvement, therefore this control algorithm is implemented on the F1TENTH vehicle for real world experiments.

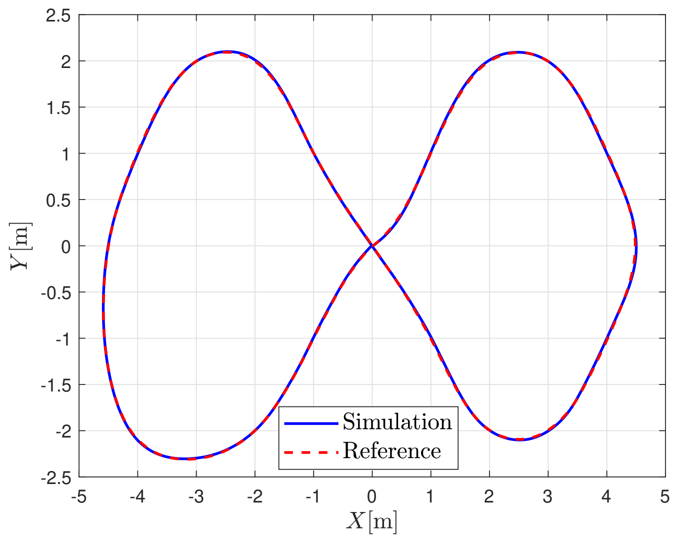


Figure 5.9: Reference track compared to the actual path of the vehicle with the gain-scheduled enhanced Stanley control algorithm.

5.4.4 Enhanced Stanley method with simplified LPV control

As the simulations show, the enhanced Stanley method proved to be efficient for the path following task, with the gain-scheduled LQR feedback term.

To further improve the algorithm, this section designs the feedback term with an LPV-LQR technique, introduced in Section B.2.3. The utilization of this LPV controller is beneficial, as it can provide mathematically guaranteed stability and performance for the whole range of the scheduling variables of the system, whereas gain-scheduling only takes into account the design points.

The control law is identical to the one introduced in the previous section:

$$\delta[k] = -\theta_\mathrm{e}[k] - K^{\mathrm{LPV}}_{\mathrm{lat,Stanley}}(v_\xi)x[k]. \tag{5.52}$$

where the feedback gains are tuned by altering the weighting matrices of the LPV-LQR. Based on simulation results, the following weights are chosen:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 15.24 & 0 \\ 0 & 0 & 2.58 \end{bmatrix}, \qquad R = 567. \tag{5.53a}$$

These differ from the gain-scheduled results, as the LPV controller considers the whole operating range of the scheduling variable, unlike the gain-scheduled technique which only takes into account the local design point.

The performance of the resulting controller is examined with numerical simulations, using the same parameters as in the previous sections. The results of path tracking is displayed in Figure 5.10. As the figure shows, the performance of the LPV-LQR is similar to the gain-scheduled one.

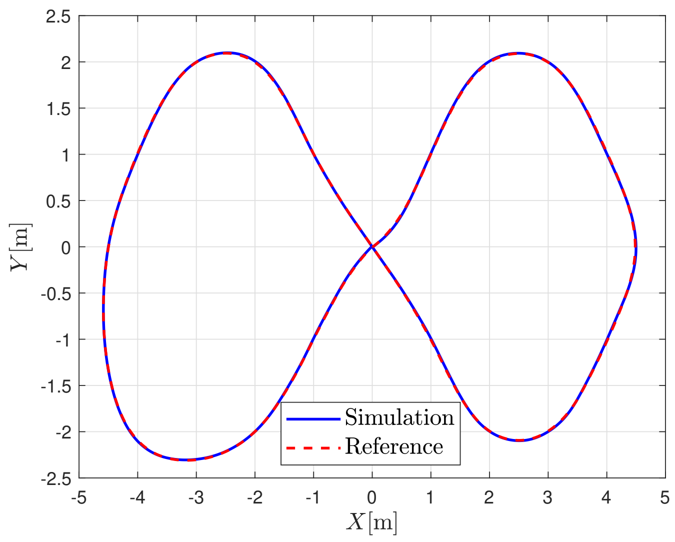


Figure 5.10: Reference track compared to the actual path of the vehicle with the LPV enhanced Stanley control algorithm.

5.4.5 Comparison of lateral control algorithms

The previous pages have introduced multiple lateral control strategies. To get a better view of the accuracy and performance of these controllers, this section presents a brief comparison between them.

While the first, simple model based controller was able to stabilize the lateral system, the tracking accuracy was insufficient. To increase the accuracy, an integral term was introduced, to improve the disturbance rejection of the controller. This method showed promising results in terms of accuracy, but the jumps in the control inputs made it impossible for the controller to be implemented on the real hardware. From simulations, it became clear that the controllers are unable to compensate the lateral and the heading error simultaneously. Therefore, the two effects were countered separately, with the introduction of the enhanced Stanley-controller. This solution used a simplified lateral model, that neglects the dynamics of the heading error. The lateral error was countered with a state-feedback controller based on this model and the heading error was compensated with an introduced feedforward term.

The last, Stanley-based control method showed the most promising results in simulations. Therefore, this controller was not only designed with the gain-scheduled LQR method, but the more advanced LPV-LQR, as the latter can provide mathematical guarantees for the stability and performance in whole operation range.

A comparison between the all the introduced controllers are also displayed in Figure 5.11. It is clear, that the enhanced Stanley-based algorithms were the most effective, with minor differences between the gain-scheduled and the LPV controllers. Therefore, both of these methods are also implemented on the real hardware, to conduct further experiments.

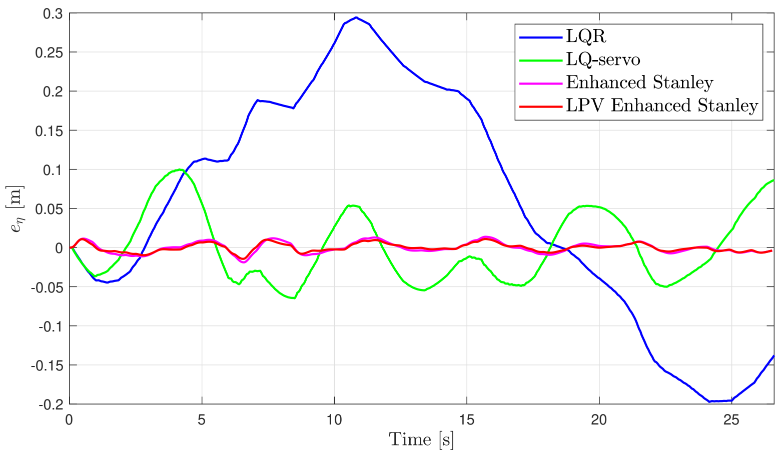
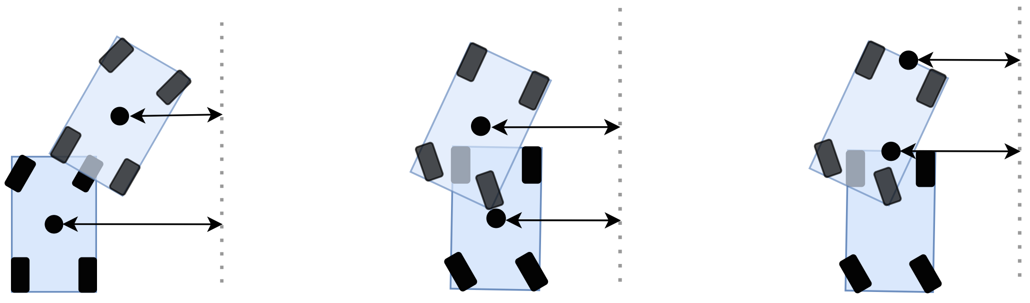


Figure 5.11: Comparison of the simulated time evolution of the lateral errors using the different control algorithms.

5.4.6 Backward driving

Special consideration is required in the case of a backward driven vehicle. For better visualization the path-tracking scenario, for a forward and backward driven car is depicted in Figures 5.12a and 5.12b, respectively.



(a) Forward motion **(b)** Backward motion **(c)** Backward motion with replaced reference

Figure 5.12: The geometry of path tracking of the forward and backward driven vehicles with different reference points, considering Ackermann steering [21].

In case of the forward driven vehicle, the application of the appropriate steering angle will immediately result in the decrease of the lateral error. However, in the case of backward driving, there is a momentary increase on the lateral error at the start of the actuation, called the initial undershoot. As a consequence, the lateral controller provides an even larger actuation signal, which leads to oscillations. This is the result of the non-collocated control of the lateral behavior also described in [16].

This effect can be countered by replacing the reference point of the vehicle to the rear axle as shown in Figure 5.12c. While dynamic models are hard to derive for points other than the center of mass, the kinematic model already has its reference placed at the desired location. Therefore, for the reversing control design, the model in (5.19) is used.

$$\underbrace{\begin{bmatrix}\dot{z}_1\\ \dot{\theta}_\mathrm{e}\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & v_\xi\\ 0 & 0\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}z_1\\ \theta_\mathrm{e}\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\ \frac{v_\xi}{l}\end{bmatrix}}_{B}\underbrace{\delta}_{u}\underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{E}\underbrace{\omega_\mathrm{p}(s)}_{\epsilon} \tag{5.54}$$

First, using the obtained LPV model with scheduling variable v_ξ, a **gain-scheduled** state feedback controller is designed to counter the effect of the external disturbance ϵ. As $M_\mathrm{c} = \begin{bmatrix}A & AB\end{bmatrix}$ has full rank for the range of v_ξ, the system can be stabilized by full state feedback control. After discretization with 0.025 s sampling time, the control law can be expressed as

$$\delta[k] = -K^\mathrm{GS}_\mathrm{lat,kin}(v_\xi)x[k] \tag{5.55}$$

To calculate the optimal feedback gains a discrete time, infinite horizon LQR is used, with weighting matrices

$$Q^\mathrm{GS} = \begin{bmatrix}50 & 0\\ 0 & 3.3\end{bmatrix}, \qquad R^\mathrm{GS} = 34. \tag{5.56a}$$

In simulations, this gain-scheduled controller showed promising results. However, the interpolation of the feedback gains cannot provide mathematical guarantees for stability and performance. Therefore, a more advanced **LPV-LQR** based feedback controller is also designed, based on the theory introduced in Section B.2.3. The feedback law can be expressed as

$$\delta[k] = -K^\mathrm{LPV}_{\mathrm{lat_k in}}(v_\xi)x[k]. \tag{5.57}$$

The optimal weighing matrices based on simulations are

$$Q^\mathrm{LPV} = \begin{bmatrix}50 & 0\\ 0 & 6.67\end{bmatrix}, \qquad R^\mathrm{LPV} = 650. \tag{5.58a}$$

The designed controllers are validated with the same numerical simulations as introduced in the previous section. As it is shown in Figure 5.13, this simpler kinematic model based method provides accurate path tracking for the reversing vehicle, with both the gain-scheduled and the LPV controllers. Therefore, these controllers are also implemented on the F1TENTH vehicles for real world experiments. The only concern raised is the vehicle behavior at higher speeds, as the kinematic model does not consider dynamic effects such as the tire behavior. However, this does not impose problems in our applications since backward driving is only applied for small length trajectories with limited prescribed velocities ($v_\xi > -1$ m/s).

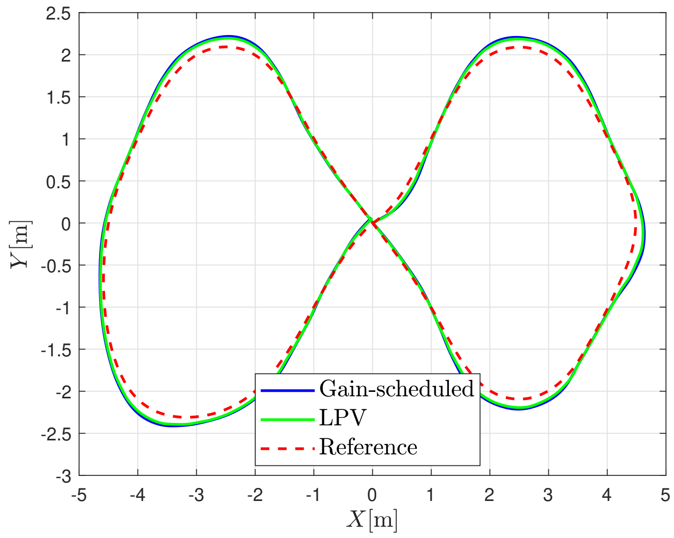


Figure 5.13: Path tracking simulation of a backward driven vehicle compared with the reference. During the simulations the center of mass of the vehicle was logged, not the front axle, therefore the experienced 10 cm deviation is normal.

6 Real world experiments

This section presents the performance of the developed motion control algorithms, implemented on the F1TENTH vehicles. To show a proper evaluation of the control algorithms, complex sequence of maneuvers are performed in an obstacle rich environment to illustrate the navigation capabilities of the cars. A video recording of the experiments is also available online: `https://youtu.be/X_GDFcN_bQE`.

The reference trajectory of the experiment can be split into six sections, that are constructed individually. Each section is defined as a two dimensional, third order spline curve representing the desired path, and a one dimensional spline that describes the speed profile along the path, as introduced in the control problem formulation in Section 5.1. The paths are designed with the limitations of the vehicle taken into consideration, therefore, the resulting path is fully realizable by the car. The speed profile in the current experiment is piecewise constant for the sections: 1.2 m/s for forward and -0.75 m/s for backward motion. The reference trajectory, with each section marked is displayed in Figure 6.1. After the construction of the path and speed splines, only the vector knots, the coefficients and the degree of the splines are stored, as this information is sufficient for the vehicle to reconstruct the path.

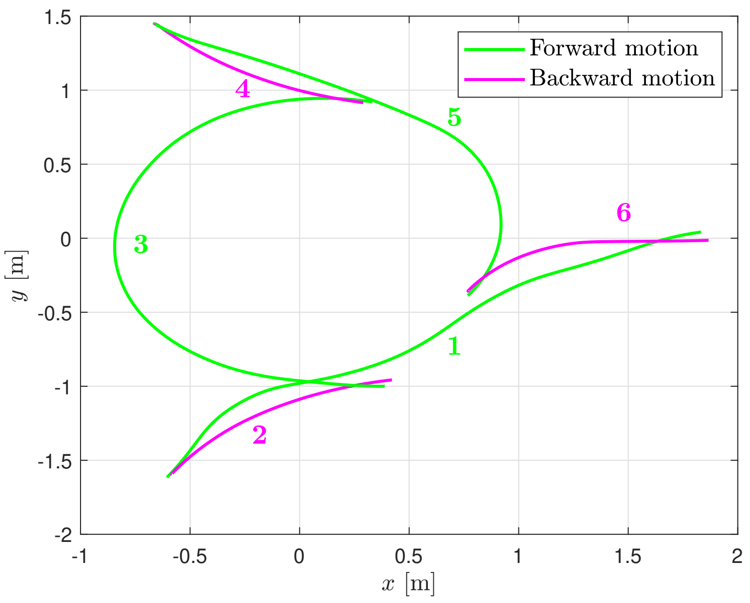


Figure 6.1: The six sections of the reference trajectory used in the experiment.

As part of this work, a complete vehicle control framework was developed for the test environment. This framework is introduced in Appendix C, therefore, this section only gives an overview of the used algorithms, without going into the implementation details. After the necessary components of the test environment, such as the OptiTrack server and the ROS master have been successfully started, the Command PC executes the high-level experiment management script shown in Algorithm 1.

Algorithm 1 Experiment management

```
1: Load the six parts (see Fig. 6.1) of the reference trajectory (vector of knots, spline
   coefficients, degree) into a list $\mathcal{T}^{\text{ref}}$.
2: Set control parameters and start up the F1TENTH onboard stack
3: for $i = 1, ..., \text{len}(\mathcal{T}^{\text{ref}})$ do
4:     Send the vector of knots, coefficients and the degree from $\mathcal{T}^{\text{ref}}(i)$ to vehicle
5:     Wait for the vehicle to execute the trajectory
6:     if trajectory execution was unsuccessful then
7:         Notify the user about the error
8:         break
9:     end if
10: end for
11: Shut down the vehicle
```

On the vehicle side, after software startup, the system waits for reference trajectories. After a trajectory is received, it is first validated by checking if the vehicle is at the staring position of the path. If the validation is successful, the low-level motion control algorithm starts and guides the vehicle through the provided reference. The implementation is explained in Algorithm 2.

Algorithm 2 Low level motion control (F1TENTH)

```
1: while the trajectory is not completed do
2:     Get the state from the state estimator module
3:     Obtain control model in path coordinates (5.7)
4:     Calculate the control inputs (5.31), (5.35) and (5.51)
5:     Apply control inputs on the vehicle
6: end while
```

A comparison between the resulting and the reference path is shown in Figure 6.2, with both the gain-scheduled (GS) and the LPV-LQR (LPV) controller. Is is clear, that the vehicle was capable of tracking the provided reference accurately with both controllers.. In case of forward motion, there is no significant difference between the gain-scheduled and the LPV controller. For backward motion, the LPV performs slightly better. However, most of the deviation of the gain-scheduled controller is the result of the reference point replacement. While in case of forward motion, the vehicle center of mass is used as reference, in the backward driving scenario it is placed on the rear axle, but during the experiments only the position of the center of mass is recorded.

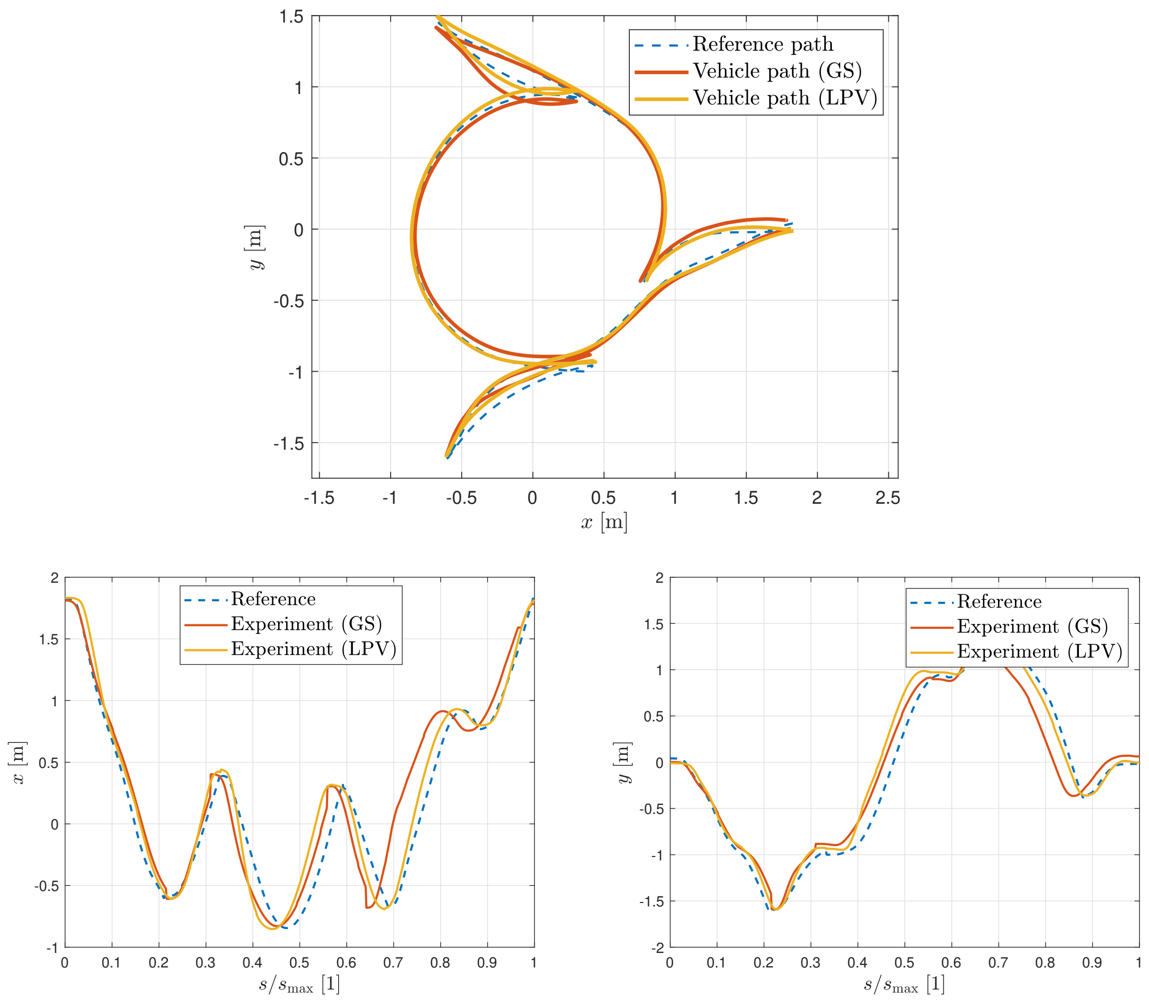


Figure 6.2: Comparison between the reference the actual path realized by the vehicle.

The speed profile tracking can also be examined with both of the implemented controllers. A comparison between the reference and the actual realized velocities of the vehicle is shown in Figure 6.3. As it is displayed, after the initial overshoot, the speed profile of the path was tracked accurately by the longitudinal controllers. The overshoot is a result of the steps in the reference signal, as the vehicle not only tries to track the speed profile, but also its integral s^{ref}. The overshoot of the LPV controller is also noticeably smaller, but this means that the controller also reacts slower to changes in the reference. While fine tuning can further reduce the effect of the overshoot, it increases the lag between s and s^{ref}, and the agile maneuvering capability of the vehicle weakens.

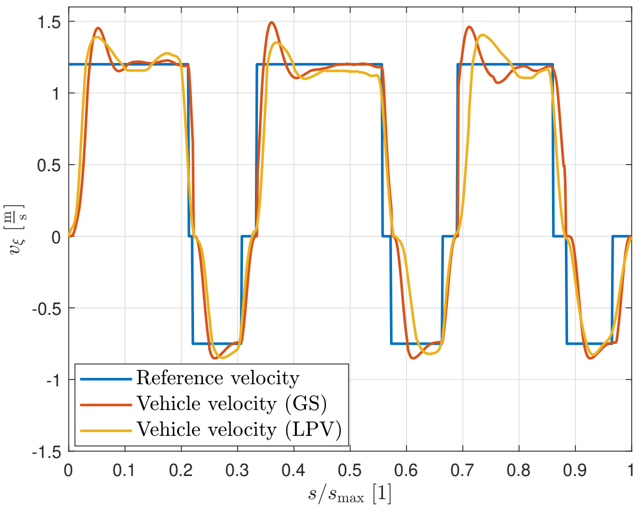


Figure 6.3: Comparison between the reference velocity and the actual velocity of the vehicle.

The results of this experiments show that the controller design was successful, and the vehicle is capable of tracking the reference path and speed profile accurately. Considering the speed profile references the trajectory execution was accomplished with up to 1.5 m/s in forward and −0.9 m/s in backward motion. As the curvature values of the designed path are quite high, larger velocity values result in significant side-slip and the tracking performance decreases. This issue can be countered with a more complex speed profile reference, where the prescribed velocity is decreased near the high curvature values. On the other hand, in the current limited space of the lab, driving much faster also has safety concerns.

The experiments were also recorded to showcase the performance of the designed control algorithms. A composite image of it is displayed in Figure 6.4, and the full video is available at `https://youtu.be/X_GDFcN_bQE`. As it is shown, the vehicle can precisely dock into the predefined parking spaces, both in case of forward and backward driving. The final position error of the vehicle was below 5 cm for every section of the trajectory.

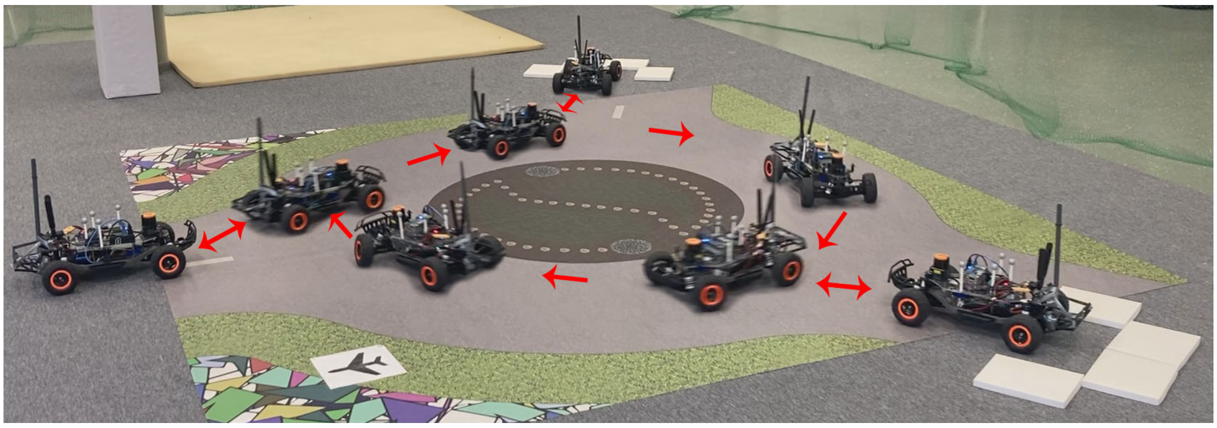


Figure 6.4: Composite images of the performed experiment.

7 Conclusions and future work

The aim of this work has been to develop a ground vehicle research framework with precise low-level control methods to provide a test environment for high-level algorithms. After a thorough literature survey about the available small scale ground vehicle platforms, the F1TENTH has been deemed the most fitting to our tasks. For the development of a motion-control solution that enables agile maneuvering and high performance reference tracking, an accurate mathematical model of the chosen vehicle is necessary. Therefore, a nonlinear, dynamic single track model with linear tires and a custom drivetrain model has been obtained and identified.

After the control problem formulation, two independent SISO LPV subsystems, that describe the longitudinal and lateral behavior of the vehicle, have been derived from the nonlinear model, for control purposes. Then, gain-scheduled feedback controllers have been developed with LQR technique. To present the performance of the designed controllers, a simulation framework has been designed, that utilizes the previously obtained nonlinear vehicle model.

After successful simulations, the designed control algorithms have been implemented in the F1TENTH vehicle. With real world experiments the control algorithms have been proven to work effectively. To aid future research, a complete control and management framework have been also been developed, that integrates the the F1TENTH cars into the *AIMotionLab* autonomous vehicle test environment.

The future goal of this project is to develop more versatile control methods for fast and safe maneuvering, in an obstacle rich environment. This includes the further improvement of the low-level path-following algorithms, with learning-based or model predictive methods, as well as the high-level path-planning and navigation tasks. In addition, to support the continuous research and development, the designed software framework is planned to extended with more advanced simulation tools, such as HIL simulations or the integration of a high fidelity physical simulator like MuJoCo[1].

[1]https://mujoco.org/

Reference

[1] Abhijeet Agnihotri et al. “Teaching Autonomous Systems at 1/10th-Scale: Design of the F1/10 Racecar, Simulators and Curriculum”. In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. SIGCSE ’20. Association for Computing Machinery, 2020, 657–663.

[2] Humberto Pessoa Almeida et al. “Autonomous Navigation of a Small-Scale Ground Vehicle Using Low-Cost IMU/GPS Integration for Outdoor Applications”. In: *Proc. of 2019 IEEE International Systems Conference (SysCon)*. 2019, pp. 1–8.

[3] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. “CommonRoad: Composable benchmarks for motion planning on roads”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 719–726.

[4] Egbert Bakker, Lars Nyborg, and Hans B. Pacejka. “Tyre Modelling for Use in Vehicle Dynamics Studies”. In: *SAE Transactions* 96 (1987), pp. 190–204.

[5] R. Craig Conlter. *Implementation of the Pursuit Path Tracking Algorithm*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, 1992.

[6] Raymond A. DeCarlo. *Linear systems : a state variable approach with numerical implementation*. Englewood Cliffs (N.J.) : Prentice-Hall international, 1989.

[7] Pascal den Boef, Pepijn B. Cox, and Roland Tóth. “LPVcore: MATLAB Toolbox for LPV Modelling, Identification and Control”. In: *IFAC-PapersOnLine* 54.7 (2021). 19th IFAC Symposium on System Identification SYSID 2021, pp. 385–390.

[8] Howard Dugoff, P. S. Fancher, and Leonard Segel. “An Analysis of Tire Traction Properties and Their Influence on Vehicle Dynamic Performance”. In: *SAE Technical Paper Series*. SAE International, Feb. 1970.

[9] F1TENTH Foundation. *F1TENTH*. URL: `https://f1tenth.org/about.html` (visited on 08/18/2022).

[10] T. Faulwasser, B. Kern, and R. 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*. 2009, pp. 8642–8647.

[11] E. Fiala. “Seitenkraften am rollenden Luftreifen”. In: *V. D. I.* 96 (1954), pp. 973–979.

[12] J.D Gergonne. “The application of the method of least squares to the interpolation of sequences”. In: *Historia Mathematica* 1.4 (1974), pp. 439–447.

[13] Thomas D Gillespie. *Fundamentals of vehicle dynamics*. Tech. rep. SAE Technical Paper, 1992.

[14] Gabriel M. Hoffmann et al. “Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing”. In: *Proc. of 2007 American Control Conference*. 2007, pp. 2296–2301.

[15] Xuewu Ji et al. “Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits”. In: *Control Engineering Practice* 76 (Apr. 2018).

[16] Minsung Kim, Seho Shin, and Jaeheung Park. “Study on vehicle lateral control for backward driving”. In: *Prof. of 2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2016, pp. 191–193.

[17] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*. New York: Wiley Interscience, 1972.

[18] Douglas Leith and W.E. Leithead. “Survey of gain-scheduling analysis and design”. In: *Int. J. Control* 73 (Jan. 2000), pp. 1001–1025.

[19] Alexander Liniger, Alexander Domahidi, and Manfred Morari. “Optimization-based autonomous racing of 1:43 scale RC cars”. In: *Optimal Control Applications and Methods* 36.5 (July 2014), pp. 628–647.

[20] J. Löfberg. “YALMIP : A Toolbox for Modeling and Optimization in MATLAB”. In: *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[21] W. F. Milliken and D. L. Milliken. *Race Car Vehicle Dynamics*. Society of Automotive Engineers, 1995, pp. 713–715.

[22] MIT. *The MIT RACECAR*. 2016. URL: `https://mit-racecar.github.io/` (visited on 08/18/2022).

[23] Matthew O’Kelly et al. “F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning”. In: *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. Ed. by Hugo Jair Escalante and Raia Hadsell. Vol. 123. Proceedings of Machine Learning Research. PMLR, Dec. 2020, pp. 77–89.

[24] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*. Kobe, Japan, May 2009.

[25] Carlo Ravizzoli. “Identification and control of an RC car for drifting purposes”. MA thesis. Politecnico di Milano, 2017.

[26] Carsten Scherer and Siep Weiland. “Linear matrix inequalities in control”. In: *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands* 3.2 (2000).

[27] Jarrod M. Snider. *Automatic Steering Methods for Autonomous Automobile Path Tracking*. Tech. rep. Pittsburgh, PA: Carnegie Mellon University, Feb. 2009.

[28] D. Soetanto, L. Lapierre, and A. Pascoal. “Adaptive, non-singular path-following control of dynamic wheeled robots”. In: *Proc. of 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Vol. 2. 2003, 1765–1770 Vol.2.

[29] Siddhartha S. Srinivasa et al. “MuSHR: A Low-Cost, Open-Source Robotic Racecar for Education and Research”. In: *CoRR* abs/1908.08031 (2019).

[30] The MathWorks. *MATLAB Optimization Toolbox*. Natick, MA, USA. 2022.

[31] R. Toth, P.S.C. Heuberger, and P.M.J. Hof, Van den. “Discretization of linear parameter-varying state-space representations”. In: *IET Control Theory & Applications* 4.10 (2010), pp. 2082–2096.

[32] Traxxas. *Slash 4x4 Ultimate Owner’s Manual*. URL: `https://traxxas.com/sites/default/files/68077-4-OM-EN-R07.pdf` (visited on 06/28/2022).

[33] Benjamin Vedder. *The VESC Project*. URL: `https://vesc-project.com/` (visited on 08/18/2022).

[34] A. Veltman, D.W.J. Pulle, and R.W. de Doncker. *Fundamentals of Electrical Drives*. Power Systems. Springer International Publishing, 2018, pp. 279–288.

[35] Christoph Voser, Rami Y. Hindiyeh, and J. Christian Gerdes. “Analysis and control of high sideslip manoeuvres”. In: *Vehicle System Dynamics* 48.sup1 (2010), pp. 317–336.

[36] Qiangqiang Yao et al. “Control Strategies on Path Tracking for Autonomous Vehicle: State of the Art and Future Challenges”. In: *IEEE Access* 8 (2020), pp. 161211–161222.

[37] David Zahradka. “Optimization-based control of the F1/10 autonomous racing car”. MA thesis. Czech Technical University in Prague, 2020.

Összefoglaló

Az F1TENTH egy valós gépjármű 1/10-ed méretarányú, elektromos hajtású modellje. Fejlesztői egy olyan nyílt járműplatformot szerettek volna megalkotni, amely hatékonyan tudja támogatni az autonóm járműirányítással kapcsolatos kutatási és oktatási feladatokat. Jelen szakdolgozat célja, az F1TENTH nemlineáris, dinamikus mozgás modelljének megalkotása, majd annak felhasználásával agilis manőverezésre alkalmas, pályakövető szabályozási algoritmus tervezése volt.

Első lépésként, a szakirodalmi áttekintést követően, az F1TENTH jármű dinamikus modellje került bevezetésre. Ez a modell magában foglalta a nemlineáris járműdinamikát, a motorbemenet és a kerekeken ható nyomaték között kapcsolatot teremtő hajtásláncot, valamint a kerekek laterális, súrlódási viselkedését leíró egyenleteket is. A modellstruktúra megválasztását után modellparaméterek meghatározása követte. A közvetlenül mérhető fizikai jellemzők mellett az empirikus paraméterek becslésének folyamata is részletezésre került, ami magában foglalta az identifikációhoz alkalmazott kísérletek megtervezését és kivitelezését, valamint a mérési adatgyűjtés megvalósítását is. Ezt követte a gyűjtött adatok feldolgozása és az optimalizáció alapú paraméterbecslés. A kapott modellt a valós rendszerrel összehasonlítva kísérletek validálták.

A dolgozat második része a modell alapú pályakövető irányítási algoritmus megtervezésével foglalkozott. A tervezéshez a teljes nemlineáris rendszert először szétcsatolásra került külön laterális, illetve longitudinális részrendszerre, majd külön-külön, mindkét alrendszerre, az egyszerűsített dinamikus modellt felhasználva, állapotvisszacsatolás alapú szabályozási alogitmusok fejlesztése valósult meg. Ezután a tervezett szabályzók robusztusságát és performancia tulajdonságait, mind szimulációs környezetben, mind pedig az F1TENTH platformon implementálva is vizsgálta a dolgozat.

Mindezek mellett megvalósítás elengedhetetlen része volt egy, az F1TENTH platformot kezelő, szoftveres keretrendszer kialakítása, a modellezési és irányítási feladatok megvalósításához. Ez a keretrendszer felelős a jármű beépített és külső szenzorjainak vezérléséért, a mérési adatok gyűjtéséért, valamint az autó irányításához szükséges aktuátorok működtetéséért. A szoftvercsomag fő előnye, hogy egy tisztán Python interfészt ad, amelyen a jármű magas szintű vezérlése és a platformon futó — ROS alapú — szoftverek egyszerű kezelése megvalósítható. A szoftverkörnyezet lehetővé teszi több jármű szimultán vezérlését is, ezzel járműcsoportok irányítására is alkalmas. Így az F1TENTH autóra építve könnyedén létrehozható egy olyan laboratóriumi tesztkörnyezet, amely további kutatási és fejlesztési projektek alapjául szolgálhat.

Kulcsszavak: modellezés, identifikáció, mozgássszabályozás, pályakövetés, gain-scheduling, LQR, LPV

A Derivation of the single track model

Single tracks models are commonly used in motion planning and motion control tasks as they can describe the behavior of car-like vehicles accurately, despite the obvious simplifications. As the authors present in [3], the single track model uses only two wheels to model the vehicle. This is achieved by lumping together the front and rear wheel pairs of the car. This simplification neglects the roll and pitch dynamics and constrains the motion into a 2 dimensional space. In [3], there are multiple models introduced, based on the single track principle. The most simple one is the kinematic single track model, which describes the vehicle motion only based on the kinematic relations. While this model can also be sufficient for basic motion control tasks, agile maneuvering — where the speed is significantly increased – cannot be achieved with such simple model. Therefore, a more complex, dynamic single track model is introduced to consider the effects of side slip behavior.

A.1 Kinematic single track model

The kinematic single track model is one of the most simple vehicle models used for describing the behavior of car-like robots as it incorporates the non-holonomic constraints of their motion. As this model completely neglects the side slip behavior, it is assumed that the velocity vector v is always parallel with the link between the front and rear tires [3]. The proposed model is depicted in Figure A.1a. To derive the equations of motion, the first step is finding the *instantaneous center of rotation* (ICR). From the geometric relations the following equations hold:

$$\dot{x} = v\cos(\varphi), \tag{A.1a}$$
$$\dot{y} = v\sin(\varphi), \tag{A.1b}$$
$$\dot{\varphi} = \frac{v}{R}, \tag{A.1c}$$
$$\tan(\delta) = \frac{l}{R}, \tag{A.1d}$$

where $(x,\ y)$ is the global position of the vehicle, φ is the heading angle measured from the X axis, δ is the steering angle and l is the wheelbase of the vehicle. By substituting (A.1d) into (A.1c) and considering $a = \dot{v}$ acceleration and δ steering angle as the inputs of the system, the kinematic single track model can be obtained:

$$\dot{x} = v\cos(\varphi), \tag{A.2a}$$
$$\dot{y} = v\sin(\varphi), \tag{A.2b}$$
$$\dot{\varphi} = \frac{v}{l}\tan(\delta), \tag{A.2c}$$
$$\dot{v} = a. \tag{A.2d}$$

A.2 Dynamic single track model

Unlike the kinematic model from the previous section, the dynamic single track model also considers the forces acting on the tires. As a result, a more complex model, that incorporates the side-slip behavior of the vehicle, is introduced. First, define a local coordinate frame $\mathcal{V}$ with its axes denoted as (ξ, η), so that the origin of this system is in the *center of mass* (CoM) of the vehicle and the ξ axis is parallel to the link between the two tires of the model. Based on this coordinate frame, the velocity of the CoM, denoted as $v_{\mathcal{V}}$, can either be expressed with a longitudinal and a lateral part or with the absolute value $v_{\mathcal{V}}$, and the angle between the velocity vector and the ξ axis. This angle is the so-called side slip angle, represented as γ.

$$v_{\mathcal{V}} = \begin{bmatrix} v_\xi \\ v_\eta \\ 0 \end{bmatrix} = \begin{bmatrix} v\cos(\gamma) \\ v\sin(\gamma) \\ 0 \end{bmatrix} \tag{A.3}$$

Next, the tire forces $F_{\mathrm{f},*}$ and $F_{\mathrm{r},*}$ are introduced for the front and rear tires respectively. These forces can also be slit into a lateral (denoted with subscript η) and longitudinal (denoted with subscript ξ) part, as shown in Figure A.1b. Summing up the lateral and longitudinal forces in Figure A.1b yields the following equations:

$$ma_\xi = F_{\mathrm{r},\xi} + F_{\mathrm{f},\xi}\cos(\delta) - F_{\mathrm{f},\eta}\sin(\delta), \tag{A.4a}$$

$$ma_\eta = F_{\mathrm{r},\eta} + F_{\mathrm{f},\xi}\sin(\delta) + F_{\mathrm{f},\eta}\cos(\delta). \tag{A.4b}$$

where δ is the steering angle, as shown in Figure A.1b. Substituting $a_\xi = \dot{v}_\xi - v_\eta\omega$ and $a_\eta = \dot{v}_\eta + v_\xi\omega$ into (A.4), $\dot{v}_\xi$ and $\dot{v}_\eta$ can be obtained as

$$\dot{v}_\xi = \frac{1}{m}\left(F_{\mathrm{r},\xi} + F_{\mathrm{f},\xi}\cos(\delta) - F_{\mathrm{f},\eta}\sin(\delta) + mv_\eta\omega\right), \tag{A.5a}$$

$$\dot{v}_\eta = \frac{1}{m}\left(F_{\mathrm{r},\eta} + F_{\mathrm{f},\xi}\sin(\delta) + F_{\mathrm{f},\eta}\cos(\delta) - mv_\xi\omega\right), \tag{A.5b}$$

where $\omega = \dot{\varphi}$ is the angular rate about the yaw axis. The yaw moment balance equation is expressed as

$$I_z\dot{\omega} = F_{\mathrm{f},\eta}l_\mathrm{f}\cos(\delta) + F_{\mathrm{f},\xi}l_\mathrm{f}\sin(\delta) - F_{\mathrm{r},\eta}l_\mathrm{r}, \tag{A.6}$$

where l_f and l_r are the distance of the front and rear wheels from the CoM, respectively. Using (A.6), $\dot{\omega}$ can be calculated as

$$\dot{\omega} = \frac{1}{I_z}\left(F_{\mathrm{f},\eta}l_\mathrm{f}\cos(\delta) + F_{\mathrm{f},\xi}l_\mathrm{f}\sin(\delta) - F_{\mathrm{r},\eta}l_\mathrm{r}\right) \tag{A.7}$$

With the kinematics discussed in Section A.1, the motion in the global (X, Y) coordinate frame is obtained from

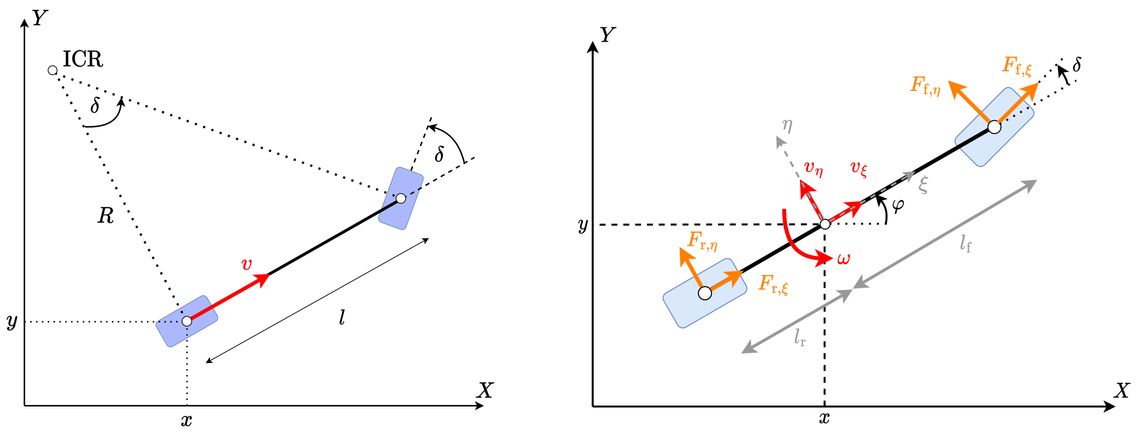
$$\dot{x} = v_\xi\cos(\varphi) - v_\eta\sin(\varphi), \tag{A.8a}$$

$$\dot{y} = v_\xi\sin(\varphi) + v_\eta\cos(\varphi). \tag{A.8b}$$

Combining the derived equations, the motion of the dynamic single track model is described by the following equations:

$$\dot{x} = v_\xi \cos(\varphi) - v_\eta \sin(\varphi), \tag{A.9a}$$
$$\dot{y} = v_\xi \sin(\varphi) + v_\eta \cos(\varphi), \tag{A.9b}$$
$$\dot{\varphi} = \omega, \tag{A.9c}$$
$$\dot{v}_\xi = \frac{1}{m}\left(F_{\mathrm{r},\xi} + F_{\mathrm{f},\xi}\cos(\delta) - F_{\mathrm{f},\eta}\sin(\delta) + m v_\eta \omega\right), \tag{A.9d}$$
$$\dot{v}_\eta = \frac{1}{m}\left(F_{\mathrm{r},\eta} + F_{\mathrm{f},\xi}\sin(\delta) + F_{\mathrm{f},\eta}\cos(\delta) - m v_\xi \omega\right), \tag{A.9e}$$
$$\dot{\omega} = \frac{1}{I_z}\left(F_{\mathrm{f},\eta} l_\mathrm{f}\cos(\delta) + F_{\mathrm{f},\xi} l_\mathrm{f}\sin(\delta) - F_{\mathrm{r},\eta} l_\mathrm{r}\right). \tag{A.9f}$$

The physical parameters of the model such as m, l_r, l_f and I_z can be measured directly or estimated from other parameters. The other unknown variables are the lateral and longitudinal tire forces and δ steering angle. In most applications, the steering angle can be directly controlled, so it is considered a control input. Tire forces are computed with the help of different tire models that are not discussed in this section, as they are not strictly connected the vehicle dynamics.



(a) Kinematic single track model **(b)** Dynamic single track model

Figure A.1: Single track models with different complexity

B LQR optimal control

This section summarizes the theoretical background of the control design steps used in this thesis. First, Section B.1 describes the handling of *linear time invariant* (LTI) systems. Then, Section B.2 extends the introduced concepts to *linear parameter-varying* (LPV) [7] systems.

B.1 Linear time invariant systems

This section provides an overview of the LTI control techniques that this work relies on. First, Section B.1.1 details the *zero order hold* (ZOH) method, which is used to obtain the discrete time state space representation of continuous systems. Next, Section B.1.2 introduces a discrete time infinite horizon *linear quadratic regulator* (LQR) [17], which is an optimal control method, used for state feedback control. Finally, Section B.1.3 proposes a *linear matrix inequality* (LMI) based solution for the LQR optimization problem that will be later extended to LPV systems.

B.1.1 Discretization

The models used in this work describe the behaviors of real-world physical systems in continuous time with the help of differential equations. However, the control algorithms used for the manipulation of these systems are predominantly implemented on micro-controllers. Like most modern computers, these also work in discrete time. In order to successfully develop well-functioning and accurate control algorithms, the derived continuous models need to be converted into discrete time for the control design procedure.

The continuous time state equation of a linear time invariant system is expressed as

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{B.1}$$

where $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$ are the state and input matrices and $x(t) \in \mathbb{R}^{n_x}$, $u(t)\mathbb{R}^{n_u}$ are the state and input vectors of the system, respectively. The integers n_x and n_u denote the number of system states and the number of inputs, respectively. The equivalent discrete time system can be obtained by discretizing the continuous system with the zero order hold method. The obtained discrete time system can be expressed as

$$x[k+1] = A_\mathrm{d}x[k] + B_\mathrm{d}u[k], \tag{B.2}$$

where A_d and B_d are the discrete time state and input matrices respectively. Their values can be computed analytically [6] as

$$\begin{bmatrix} A_\mathrm{d} & B_\mathrm{d} \\ 0 & I \end{bmatrix} = \exp\left(\tau \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}\right) \tag{B.3}$$

where τ is the sampling time of the discretization.

B.1.2 Linear Quadratic Regulator

Consider a discrete time, linear, time invariant system in state space representation:

$$x[k+1] = A_{\mathrm{d}}x[k] + B_{\mathrm{d}}u[k]. \tag{B.4}$$

The asymptotic stability of this system can be checked by calculating the eigenvalues (denoted as μ_i) of the system matrix. If $|\mu_i| < 1$ holds for all μ_i eigenvalues the system is considered asymptotically unstable. Therefore, to stabilize the system, full state feedback control is applied.

Before further steps, it is important to check if the state-space representation of the system is controllable. The controllability of a linear time invariant system can be examined by calculating the rank of the controllability matrix, which has the following form:

$$M_{\mathrm{c}} = \begin{bmatrix} B_{\mathrm{d}} & \cdots A_{\mathrm{d}}^{i} B_{\mathrm{d}} \cdots A_{\mathrm{d}}^{n-1} B_{\mathrm{d}} \end{bmatrix}, \tag{B.5}$$

where n is the size of A_{d} system matrix. If M_{c} has full rank, the system is controllable and full state feedback van be applied. The control law is defined as

$$u[k] = -Kx[k], \tag{B.6}$$

where K is the feedback gain vector. Combining (B.2) and (B.6), the closed loop can be expressed as

$$x[k+1] = \left(A_{\mathrm{d}} - B_{\mathrm{d}}K\right)x[k]. \tag{B.7}$$

Equation (B.7) shows that if the system is controllable, the eigenvalues of the closed loop system matrix can be replaced with the proper choice of K. In this work, rather than calculating the feedback gains manually, a discrete time, infinite-horizon LQR is utilized [17], which was used in [27] for the development of steering control algorithms. This approach forms and optimization problem to obtain the optimal feedback gains of the controller. For the optimization, the performance requirements are expressed by a quadratic cost function:

$$J = \sum_{k=0}^{\infty} \left(x[k]^{\top} Q x[k] + u[k]^{\top} R u[k] \right), \tag{B.8}$$

where $Q \succeq 0$ and $R \succ 0$ are manually selected weighting matrices. The goal is to minimize J by calculating the optimal control sequence. This results in the following optimization problem:

$$\begin{aligned} \min_{u[\cdot]} \quad & \sum_{k=0}^{\infty} \left(x[k]^{\top} Q k[k] + u[k]^{\top} R u[k] \right) \\ \text{subject to} \quad & x[k+1] = A_{\mathrm{d}}x[k] + B_{\mathrm{d}}u[k] \qquad k = 0, 1, 2, ..., \infty \end{aligned} \tag{B.9}$$

The optimal control input is given as

$$u[k] = -Kx[k], \tag{B.10}$$

where K feedback vector is calculated as

$$K = (R + B_\mathrm{d}^\top P B_\mathrm{d})^{-1} B_\mathrm{d}^\top P A_\mathrm{d}, \tag{B.11}$$

and P is the solution to the algebraic Riccati equation:

$$P = A_\mathrm{d}^\top P A - A_\mathrm{d}^\top P B_\mathrm{d} (R + B_\mathrm{d}^\top P B_\mathrm{d})^{-1} B_\mathrm{d}^\top P A_\mathrm{d} + Q \tag{B.12}$$

In the cost function, Q is a diagonal weighting matrix which can be used to tune how much each state contributes to the overall performance. Similarly, R is used to weight the control effort corresponding to the cost function.

B.1.3 LMI-based LQR

The LQR method is a powerful tool to find the optimal feedback gains of a system and it can also provide mathematical guarantees for the stability of the closed loop of the controlled system. However, the Ricatti equation based solution, introduced in Section B.1.2, only considered LTI systems and cannot be applied directly to LPV systems, with the mathematical guarantees preserved. Therefore, this section proposes an LMI based solution for the LQR problem, which can be easily extended to the LPV framework.

Consider the following discrete time LTI system, and the previously introduced LQR optimization problem:

$$x[k+1] = A_\mathrm{d} x[k] + B_\mathrm{d} u[k], \tag{B.13a}$$

$$\begin{aligned} \min_{u[\cdot]} \quad & \sum_{k=0}^{\infty} \Big(x[k]^\top Q k[k] + u[k]^\top R u[k] \Big) \\ \text{subject to} \quad & x[k+1] = A_\mathrm{d} x[k] + B_\mathrm{d} u[k] \qquad k = 0, 1, 2, ..., \infty. \end{aligned} \tag{B.13b}$$

If full state feedback is applied ($u[k] = -Kx[k]$), the closed loop system is asymptotically stable if a Lyapunov function $V(x[k])$ exists such that

$$1, \qquad V(x[k]) > 0 \quad \forall k = 0, 1, 2, ..., \infty \text{ and } x \neq 0; \tag{B.14a}$$

$$2, \qquad V(x[k+1]) - V(x[k]) < 0. \tag{B.14b}$$

If a quadratic Lyapunov function is chosen in the form of

$$V(x[k]) = x[k]^\top P x[k], \tag{B.15}$$

the matrix $P \succeq 0$ must be positive semi-definite to satisfy (B.14a). Next, the LQR cost is substituted into the second condition to obtain:

$$V(x[k+1]) - V(x[k]) < -x[k]^\top Q x[k] - u[k]^\top R u[k]. \tag{B.16}$$

The right side of the inequality can also be expressed as

$$-x[k]^\top Q x[k] - u[k]^\top R u[k] = -x[k]^\top (C_1 + D_{12} K)^\top (C_1 + D_{12} K) x[k], \tag{B.17}$$

where

$$C_1 = \begin{bmatrix} Q^{\frac{1}{2}} \\ 0 \end{bmatrix}, \qquad\qquad D_{12} = \begin{bmatrix} 0 \\ R^{\frac{1}{2}} \end{bmatrix}. \tag{B.18a}$$

Substituting (B.17) into (B.16) yields the following inequality:

$$V(x[k+1]) - V(x[k]) \le -x[k]^\top (C_1 + D_{12}K)^\top (C_1 + D_{12}K)x[k] - x[k]^\top \epsilon P x[k], \tag{B.19}$$

where the $x[k]^\top \epsilon P X[k]$ term is introduced with $\epsilon = 10^{-12}$ to relax the strict inequality, as the numerical methods of optimization solvers can only handle non-strict inequalities. Considering the iteration $k = 0, 1, ...\infty$, the following inequalities need to be satisfied:

$$\begin{aligned}
V(x[1]) - V(x[0]) &\le -x[0]^\top (C_1 + D_{12}K)^\top (C_1 + D_{12}K)x[0] - x[0]^\top \epsilon P x[0], && \text{(B.20a)}\\
V(x[2]) - V(x[1]) &\le -x[1]^\top (C_1 + D_{12}K)^\top (C_1 + D_{12}K)x[1] - x[1]^\top \epsilon P x[1], && \text{(B.20b)}\\
V(x[3]) - V(x[2]) &\le -x[2]^\top (C_1 + D_{12}K)^\top (C_1 + D_{12}K)x[2] - x[2]^\top \epsilon P x[2], && \text{(B.20c)}\\
&\vdots
\end{aligned}$$

By summing up the inequalities, the higher index Lyapunov functions cancel out, and the following expression can be obtained

$$-V(x[0]) \le -\sum_{k=0}^{\infty} \left(x[k]^\top Q k[k] + u[k]^\top R u[k] \right) - \epsilon \sum_{k=0}^{\infty} \left(x[k]^\top P k[k] \right), \tag{B.21}$$

which can also be expressed as

$$\sum_{k=0}^{\infty} \left(x[k]^\top Q k[k] + u[k]^\top R u[k] \right) + \epsilon \sum_{k=0}^{\infty} \left(x[k]^\top P k[k] \right) \le x[0]^\top P x[0]. \tag{B.22}$$

This inequality shows that $x[0]^\top P x[0]$ is an upper bound to the LQR cost. Therefore, by minimizing this upper bound, a suboptimal solution for the LQR problem can be obtained. To minimize $x[0]^\top P x[0]$ for all $x[0]$, we minimize the trace of P which leads to the following optimization problem

$$\begin{aligned}
\min_{K} \quad & \text{trace}(P) \\
\text{subject to} \quad & P \succeq 0, \\
& \text{(B.19)}, \\
& x[k+1] = A_\text{d} x[k] + B_\text{d} u[k] \qquad\qquad k = 0, 1, 2, ..., \infty.
\end{aligned} \tag{B.23}$$

The constraint in (B.19) is a nonlinear matrix inequality, but it can be converted to an LMI by performing the following steps: substituting the quadratic Lyapunov function (B.15) and system dynamics (B.13a) into the inequality in (B.19) leads to

$$\begin{aligned}
x[k]^\top (A_\text{d} - B_\text{d}K)^\top P (A_\text{d} - B_\text{d}K) x[k] - x[k]^\top P x[k] \le \\
-x[k]^\top (C_1 + D_{12}K)^\top (C_1 + D_{12}K) x[k] - x[k]^\top \epsilon P x[k].
\end{aligned} \tag{B.24}$$

Multiplying the inequality with $x[k]^\top$ from the left side and with $x[k]$ from the right side leads to

$$(A_\mathrm{d} - B_\mathrm{d}K)^\top P(A_\mathrm{d} - B_\mathrm{d}K) - P \leq -(C_1 + D_{12}K)^\top (C_1 + D_{12}K) - \epsilon P. \tag{B.25}$$

Next, the inequality is multiplied with $X = P^{-1}$ from the right and the left side to obtain

$$(A_\mathrm{d}X - B_\mathrm{d}Y)^\top P(A_\mathrm{d}X - B_\mathrm{d}Y) - X \leq -(C_1X + D_{12}Y)^\top (C_1X + D_{12}Y) - \epsilon X, \tag{B.26}$$

where $Y = KX$. After reorganizing the terms in the inequality, it can also be expressed in matrix form:

$$X - \epsilon X - \begin{bmatrix}(A_\mathrm{d}X - B_\mathrm{d}Y)^\top & (C_1X + D_{12}Y)^\top\end{bmatrix} \begin{bmatrix} P & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A_\mathrm{d}X - B_\mathrm{d}Y & C_1X + D_{12}Y \end{bmatrix} \geq 0, \tag{B.27}$$

by Schur decomposition theorem [26], this is equivalent to

$$\begin{bmatrix} X - \epsilon X & (A_\mathrm{d}X - B_\mathrm{d}Y)^\top & (C_1X + D_{12}Y)^\top \\ (A_\mathrm{d}X - B_\mathrm{d}Y) & X & 0 \\ (C_1X + D_{12}Y) & 0 & I \end{bmatrix} \geq 0. \tag{B.28}$$

Finally, the optimization problem that needs to be solved to obtain the optimal feedback gains can be expressed as

$$\begin{aligned} \min_{X,Y} \quad & -\operatorname{trace}(X) \\ \text{subject to} \quad & X \succeq 0, \\ & \text{(B.28)}. \end{aligned} \tag{B.29}$$

This optimization problem can be easily formulated with YALMIP [20] a high-level modelling language, and can be solved by MOSEK[1], a semidefinite solver program.

From the optimization results, the feedback gains are computed as

$$K = YX^{-1}. \tag{B.30}$$

B.2 Linear parameter-varying systems

In the recent years, the LPV framework has become popular, as it can represent complex, nonlinear behavior, while preserving the advantageous properties of the linear model structure. The LPV state space representation can be expressed as

$$\dot{x} = A(\rho)x + B(\rho)u, \tag{B.31}$$

where $A(\rho)$, $B(\rho)$ system and input matrices are all a matrix functions of a measurable, time-varying signal ρ, which is called the scheduling variable, as it is introduced in [7].

[1]`https://www.mosek.com/`

This section extends the previously introduced LTI control concepts to LPV systems. First, Section B.2.1 describes the ZOH discretization of LPV systems. Then, Section B.2.2, details the gain-scheduled control design method. While the gain-scheduling is a convenient way to control parameter varying systems, the prescribed performance and robustness can only be guaranteed at the design points of the controller. Therefore, Section B.2.3 introduces an LPV-LQR controller, which can provide mathematically guaranteed stability and performance properties for the whole operating range of the parameter varying systems.

B.2.1 LPV discretization

The ZOH discretization from Section B.1.1 can be extended to LPV systems (B.31), resulting in the discrete time LPV representation in the form of

$$x[k+1] = A_{\mathrm{d}}(\rho[k])x[k] + B_{\mathrm{d}}(\rho[k])u[k], \tag{B.32}$$

where $A_{\mathrm{d}}(\rho[k])$, $B_{\mathrm{d}}(\rho[k])$ are the discrete time system and input matrices at k.

Under the assumption that the scheduling variable ρ and the control input u are constant signals inside the sampling intervals, denoted by $\rho[k]$ and $u[k]$, the discrete time LPV system can be calculated with the following equations [31]:

$$A_{\mathrm{d}}(\rho[k]) = \mathrm{e}^{A(\rho[k])\tau} \tag{B.33a}$$
$$B_{\mathrm{d}}(\rho[k]) = A^{-1}(\rho[k])\left(\mathrm{e}^{A(\rho[k])\tau} - I\right)B(\rho[k]), \tag{B.33b}$$

where τ is the sampling time of the discretization.

B.2.2 Gain-scheduled control

The first method introduced for the control of parameter varying systems is a gain-scheduled LQR controller. This control technique can be split into three main steps.

Assuming scalar valued scheduling parameter (i.e. $\rho(t) \in \mathbb{R}\ \forall t$), the first step is the selection of an appropriate grid over the parameter domain:

$$\Gamma = [\rho_0 = \rho_{\min} \leq \rho_1 \leq ... \leq \rho_{n-1} = \rho_{\max}], \tag{B.34}$$

where the ρ_i grid points correspond to local LTI systems. These LTI systems are first discretized by ZOH, as it is described in Section B.1.1. Then, at each grid point, LQR feedback controllers are designed, which is introduced in Section B.1.2. After the local LTI controllers have been successfully designed for each $\rho_i \in \Gamma$, the final step is the polynomial interpolation of the feedback gains $k_{i,j}$, $i = 1, ..., n_x$, $j = 1, .., n_u$ of the feedback matrix K. For this a least-squares polynomial regression [12] can be used. This type of interpolation can easily be carried out with the `polyfit` MATLAB function.

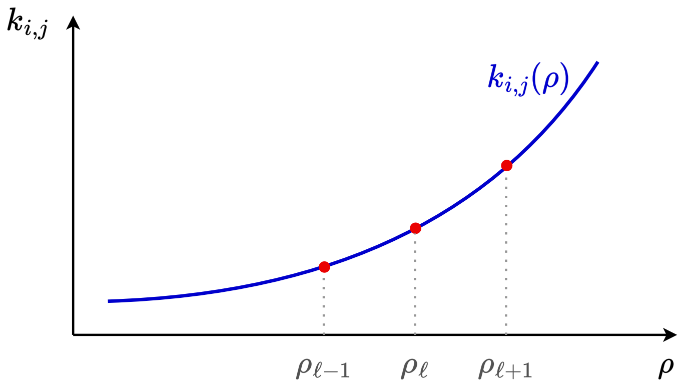


Figure B.1: Polynomial interpolation of the feedback gains.

B.2.3 LPV-LQR

In this section, the LMI-based controller, derived for LTI systems in Section B.1.3, is extended to the linear parameter-varying systems. The LPV systems have the following representation:

$$x[k+1] = A_{\mathrm{d}}(\rho[k])x[k] + B_{\mathrm{d}}(\rho[k])u[k], \tag{B.35}$$

where A_{d} and B_{d} are the discrete time state and input matrices respectively, which are all depending on the scheduling parameter $\rho[k]$. The feedback law used can be expressed as

$$u[k] = -K(\rho[k])x[k], \tag{B.36}$$

so the feedback gain is also dependent on the scheduling variable. The cost function of the LQR, that needs to be minimized is still

$$J = \sum_{k=0}^{\infty} \left(x[k]^{\top} Qk[k] + u[k]^{\top} Ru[k] \right) x[k], \tag{B.37}$$

and the chosen Lyapunov function is the same quadratic function from Section B.1.3:

$$V(x[k]) = x[k]^{\top} Px[k]. \tag{B.38}$$

As in this case P is not dependent on the scheduling variable $\rho[k]$, the derivation steps introduced in Section B.1.3 can also be carried out with the parameter dependent state and input matrices to obtain the following LMI:

$$\begin{bmatrix} X - \epsilon X & (A_{\mathrm{d}}(\rho[k])X - B_{\mathrm{d}}(\rho[k])Y(\rho[k]))^{\top} & (C_1 X + D_{12} Y(\rho[k]))^{\top} \\ (A_{\mathrm{d}}(\rho[k])X - B_{\mathrm{d}}(\rho[k])Y(\rho[k])) & X & 0 \\ (C_1 X + D_{12} Y(\rho[k])) & 0 & I \end{bmatrix} \succeq 0. \tag{B.39}$$

It is important to note, that by choosing a non parameter dependent P, the designed controller will be more conservative than it would be with $P(\rho[k])$, as this solution does not consider the change rate of ρ. Next, the following parameterization is chosen, as it is linear in the optimization variables:

$$Y(\rho[k]) = Y_0 + \rho[k]Y_1 + \rho[k]^2 Y_2 + \ldots + \rho[k]^n Y_n, \tag{B.40}$$

where n is the order of the polynomial. A grid is also introduced with n grid points on the range of the varying parameter:

$$\Gamma = [\rho_0 = \rho_{\min} \le \rho_1 \le ... \le \rho_{n-1} = \rho_{\max}]. \tag{B.41}$$

Using Γ, the LMIs can be evaluated at the grid points only, relaxing the infinite set of LMIs to a finite set, that can be used as constraints for the optimization problem:

$$\begin{bmatrix} X - \epsilon X & (A_{\mathrm{d}}(\rho_i)X - B_{\mathrm{d}}(\rho_i)Y(\rho_i))^\top & (C_1 X + D_{12}Y(\rho_i))^\top \\ (A_{\mathrm{d}}(\rho_i)X - B_{\mathrm{d}}(\rho_i)Y(\rho_i)) & X & 0 \\ (C_1 X + D_{12}Y(\rho_i)) & 0 & I \end{bmatrix} \succeq 0. \tag{B.42}$$

The final optimization can be formulated as follows:

$$\begin{aligned} \min \quad & -\operatorname{trace}(X) \\ \text{subject to} \quad & X \succeq 0, \\ & \text{(B.42)} \qquad \forall \rho_i \in \Gamma. \end{aligned} \tag{B.43}$$

This optimization problem can also be easily formulated in YALMIP and solved by MOSEK.

From the optimization results, the parameter dependent feedback gain is computed at each time instant k as

$$K(\rho[k]) = Y(\rho[k])X^{-1} = \left(Y_0 + \rho[k]Y_1 + \rho[k]^2 Y_2 + ... + \rho[k]^n Y_n\right) X^{-1}, \tag{B.44}$$

which means that the elements of the resulting feedback matrix are n order polynomials.

C AIMotion – Fleet1tenth framework

C.1 General overview

AIMotion – Fleet1tenth is the control and simulation framework, developed for the integration of the F1TENTH vehicles into the *AIMotionLab* autonomous vehicle test environment. It completely replaces the official onboard software stack, developed by the F1TENTH Community [9], to provide a more versatile control solution. The main advantages of this framework compared to the previous developments:

- Multiple vehicles can be controlled from a single Python script compared to the ROS based command line approach;
- Implemented easy manual remote control solutions;
- Implemented motion-capture based state estimation module;
- Implemented trajectory tracking controller module for precise navigation;
- Implemented logging functionalities for measurements;
- Implemented on-the-fly software and parameter update solutions.

The framework consists of three main subpackages, each responsible for different tasks. The overall software architecture is presented in Figure C.1.

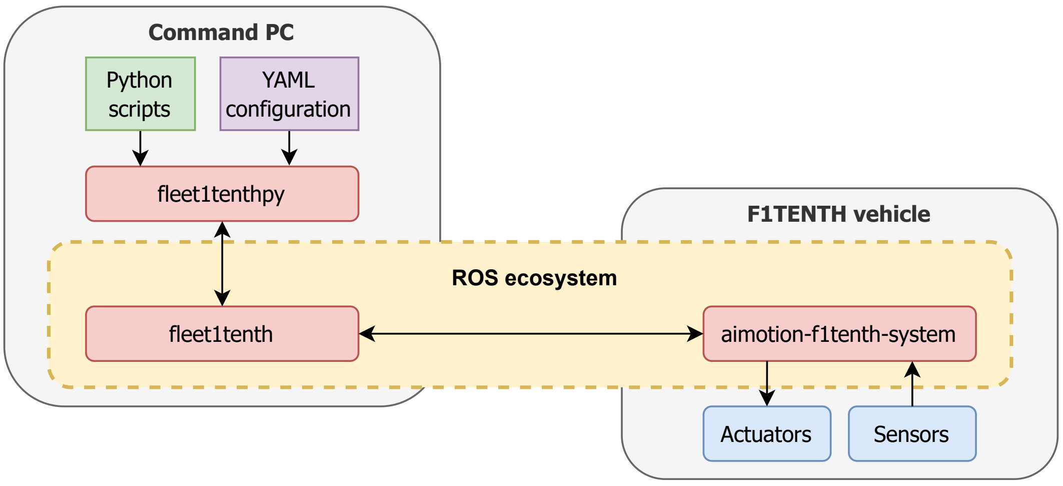


Figure C.1: The main software components of the AIMotion – Fleet1tenth framework.

The `aimotion-f1tenth-system` package is running onboard the F1TENTH vehicles. It includes the ROS based implementation of the trajectory tracking controller, supported with complementary functions such as the state estimator module and the different low-level hardware APIs responsible for the serial communication between the Jetson embedded computer and the VESC board.

The `fleet1tenth` is the high-level interface of the onboard stack. It is also a self-developed and ROS based package, to provide convenient communication with the vehicles. It runs on separate computer, called the Command PC, and oversees the different processes running in `aimotion-f1tenth-system`. This is a useful solution, as the official F1TENTH software stack would require manual remote connection and command execution which is really slow and unpractical, especially if there are multiple vehicles present.

Finally, the `fleet1tenthpy` package is the high-level Python API of the framework, that provides a set of helpful tools for easier usage and implementation, such as source file management and online parameter update options. The main advantage of this API is the fact that it enables the whole system to be controlled from Python scripts only.

In the following sections, the introduced subpackages are described in details. While this project is under continuous development, the framework is expected to be extended in the future with additional functionalities, such as a visualization environment for the simulator or path planning algorithms. The current version of the package is available at the *AIMotionLab* Github page[1].

C.2 aimotion-f1tenth-system

As it was already introduced, the `aimotion-f1tenth-system` is the onboard software stack of the F1TENTH vehicles. The overall structure of the package is depicted in Figure C.2.

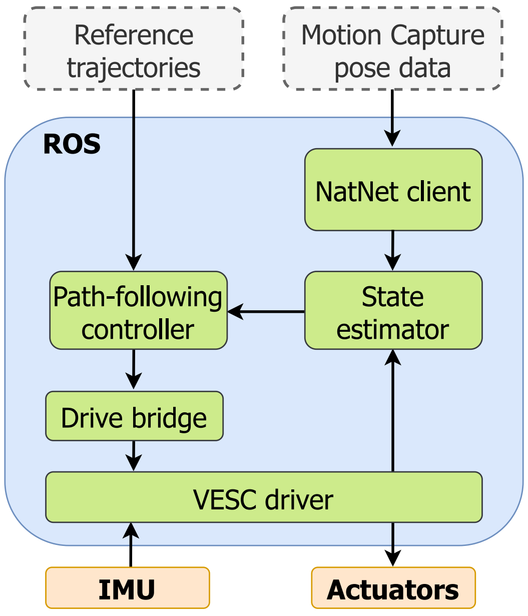


Figure C.2: The structure of the `aimotion-f1tenth-system` package.

The **VESC driver** node is the ROS based interface to the VESC board. It communicates

[1]`https://github.com/AIMotionLab-SZTAKI/aimotion-fleet1tenth`

with other processes through the topic based Publisher/Subscriber model of ROS, and transfers data to and from the VESC, using a serial communication. It is important to note that this package is already made available in the Github page[1] of the F1TENTH community [9].

It was already detailed in Section 4.3, that the physical steering angle control input, determined in radians, cannot be applied directly on the steering. Therefore, the **Drive bridge** node is responsible for shaping the control inputs of the system into the signal format of the VESC, using (4.4).

During autonomous navigation, the control inputs are determined by the **Path-following controller**. This package contains the implementation of the trajectory tracking control algorithms introduced in Section 5.

As the implemented control algorithms rely on full state feedback, it is important to have the accurate state data of the vehicle. The **State estimator** module of the system collects the vehicle pose data, coming from the motion capture system, and the IMU measurements of the VESC and calculates the current state of the car.

The OptiTrack motion capture data is accessed via the **NatNet client** node. This package communicates with the motion capture server and forwards the data to the internal nodes, using ROS based communication. The implementation of the client application was provided by Matthew Edwards[2].

C.3 fleet1tenth

The main task of this package is to provide the low-level ROS software components, that are needed to communicate with the F1TENTH vehicles. On top of that, there is a simulation application implemented in the package that uses the identified nonlinear dynamic model of the F1TENTH cars. With the help of this simulator, modifications of the onboard control stack or the different motion scenarios can be evaluated safely in a virtual environment, prior to the real hardware implementation.

C.4 fleet1tenthpy

Very early during the software development, it became clear that the everyday usage of the ROS ecosystem can be cumbersome. With multiple cars running, management of different processes and timing issues raise important concerns. Therefore, to simplify the general operation, this package provides a Python API to the developed system.

The main task of the `fleet1tenthpy` package is the high-level control, of the F1TENTH vehicles. It can be used to remotely drive the cars, but it can also provide reference

[1]`https://github.com/f1tenth`
[2]`https://github.com/mje-nz`

trajectories for the them to follow. To ensure proper surveillance there are also logger functionalities implemented, that can track the state of the running vehicles.

Another important aspect is the management of the onboard stack and parameter tuning. With the help of this API, the `aimotion-f1tenth-system` package running on the F1TENTH vehicles can be easily started remotely and all its parameters can be modified without accessing the source code. Moreover, there are also other utility tools integrated, that can remotely install and update the onboard software, to aid the development process.

Although, most of the introduced tasks can be accomplished through the ROS ecosystem, the everyday usage of such a complex system quickly becomes overcomplicated. The introduced framework not only tackles these issues, but as a pure Python API, it provides an intuitive interface that can be used to integrate the F1TENTH cars into larger projects, running at the *AIMotionLab* autonomous vehicle test area of SZTAKI.