KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Design and implementation of a time-optimal controller for model race cars

Robin Verschueren

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: wiskundige ingenieurstechnieken

Promotor:
Prof. dr. M. Diehl

Assessoren:
Prof. dr. ir. D. Huybrechs
Prof. dr. R. Vandebril

Begeleiders:
Dr. ir. S. De Bruyne
J. Frasch
M. Zanon

Academiejaar 2013 – 2014

Preface

Eerst en vooral zou ik professor Diehl willen bedanken om mij de uitgelezen kans te geven deze uitdagende masterproef uit te voeren. Mijn dank gaat ook uit naar LMS voor het ter beschikking stellen van de experimentele setup voor deze masterproef, alsook voor de unieke ervaring naar de European Vehicle Conference 2013 van LMS te mogen gaan. A big thank you goes out to my supervisors Stijn De Bruyne, Janick Frasch and Mario Zanon, for many hints, tips, and insights in optimization: the lengthy Skype sessions were enlightening. Tenslotte zou ik ook mijn familie willen bedanken voor de immer aanwezige morele en financiële steun.

Robin Verschueren

Contents

Abstract

In this master thesis different real-time control algorithms are developed that allow miniature race cars to drive autonomously around a predefined race track in a time-optimal fashion. The performance of the different methods is evaluated in simulation, as well as experimentally validated on a real-world test setup at LMS, A Siemens Business, our industrial cooperation partner. We propose to use advanced control methods for the complex task of time-optimal autonomous driving: both linear and nonlinear model predictive control are investigated as control methods. The linear methods presented are heuristic in nature, the nonlinear methods are an attempt to directly solve the time-optimal problem.

In a mechatronic system that operates in the millisecond range, such as the race cars we use, it is a computational challenge to calculate the next control action before the next data comes in, especially for the nonlinear control methods. In order to reach the computational deadline, we propose to use the ACADO Code Generation tool, which auto-generates tailored C-code that implements the real-time iteration scheme. A comparison is made between the different methods presented in this thesis. Using the developed methods, we are able to greatly reduce the lap time of the race car.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

ADAS	Advanced Driver Assistance System
MPC	Model Predictive Control
NMPC	Nonlinear MPC
CG	Center of Gravity
VPU	Vision Processing Unit
VCU	Vehicle Control Unit
RGB	Red Green Blue
EKF	Extended Kalman Filter
MHE	Moving Horizon Estimation
UDP	User Datagram Protocol
RT	Real-Time
DC	Direct Current
DAQ	Data Acquisition
ODE	Ordinary Differential Equation
QP	Quadratic Program
SQP	Sequential Quadratic Program
NLP	Nonlinear Program
RTI	Real-Time Iteration
CGT	ACADO Code Generation Tool

Symbols

X	X-position of the car's CG in the global reference frame
Y	Y-position in the global reference frame
x	x-position in the local reference frame attached to the car
y	y-position in the local reference frame
ψ	orientation
v	absolute velocity
v_X	X-component of the velocity in the global reference frame
v_Y	Y-component of the velocity in the global reference frame
v_x	x-component of the velocity in the local reference frame
v_y	y-component of the velocity in the local reference frame
t	time
ω	yaw rate
δ	steering angle
D	dutycycle of the DC motor
ξ	state vector
u	control vector
β	longitudinal slip coefficient
α	slip angle
F	force
m	mass
σ	reference trajectory
κ	local curvature of the track
ρ	local radius of curvature of the track
e^y	deviation from the centerline
e^ψ	orientation error
s	path advancement
s_s	spatial sampling of the time-optimal problem
N	number of control intervals

Chapter 1

Introduction

In the past few years, various Advanced Driver Assistance Systems (ADAS) have been introduced in commercial passenger vehicles, such as semi-autonomous parking systems, autonomous cruise control, or last-second crash prevention systems. Triggered by these technological advances, full automated driving of vehicles is becoming a reality.

In the following sections, we present a general framework for this master thesis, and briefly introduce the state-of-the-art.

1.1 Autonomous driving

According to a recent report on road safety by the World Health Organization (cf. [39]), 1.2 million people die in a traffic accident each year, which makes it the 10th leading cause of deaths overall. The main cause in these accidents is a human error, or human negligence (e.g. driving under influence).

Both the UN General Assembly and the European Union have set goals to vastly reduce traffic casualties by 2020. One of the most promising solutions is to take out the human aspect and switch to autonomous passenger vehicles.

Apart from the potentially increased safety, the transition to commercial autonomous passenger vehicles will have a tremendous impact on our daily lives. Instead of lining up in congestion, commuters can use their traveling time more usefully. Points of critique on self-driving cars are increased congestion, privacy (as the car needs to communicate with the grid) and legal issues (who is to blame when an accident does occur with an autonomous car? [9]).

Figure 1.1: An example of an Advanced Driver Assistance System: automated parking. Source: [1]

[1] http://images.thecarconnection.com/lrg/volvo-autonomous-parking_100430956_l.jpg

Autonomous and semi-autonomous ground vehicles are an interesting topic for both industry and academic research. The famous ARGO project [8] from Università di Parma was one of the first succesful academic project on autonomous driving, in the late 1990's. The car was instrumented with artificial vision cameras and a personal computer to automatically manage the steering wheel on routes along public highways. The Autonomos project [4] of Freie Universität Berlin is another example of the succesful development of autonomous cars in an academic setting with commercial cooperation partners.

Figure 1.2: Stanford Stanley, winner of the DARPA Grand Challenge in 2005.
Source: [2]

A lot of projects on autonomous driving are in development in industry. In 1980, Mercedes-Benz was the first to develop an autonomous road vehicle. It drove on empty highways at speeds around 100 km/h. In the 1990s, the Defense Advanced Research Projects Agency of the US Department of Defense (DARPA) created an autonomous land vehicle that was able to drive cross-country. Since then they organized their famous DARPA Grand Challenge, a competition for autonomous vehicles. In the last years, almost all car manufacturers are busy developing some sort of self-driving car that can maneuver in real-world traffic. However, the technical details are proprietary and thus rarely published. Examples of car manufacturers and high-tech companies working around autonomous ground vehicles are Volvo, BMW, Mercedes, Google, Hyundai, Toyota, and Nissan.

Developing autonomous cars and testing them are very cost-intensive activities. In an attempt to have an experimental testing environment that serves as a validation but is nevertheless feasible on a reduced budget, a small-scale setup is instructive in the comparison of different algorithms for autonomous driving. One of the pioneering groups adopting this strategy are the members of the ORCA (Optimal RC Autonomous racing, [31]) project at ETH in Zürich. They do research on optimal controlled race cars on a scaled setup, similar to what will be used in this thesis. For a clear overview, refer to [36] or [40].

A crucial part of any of these (semi-)autonomous systems is the control system. Often, a traditional control approach is used. In [8], the steering system was based on a classical proportional (P) controller. More recently, proportional-integral-derivative (PID) controllers are used in [29] for lateral steering in lane changing, in [3] for planning the velocity profile of a car, and in [28] for lane keeping.

The autonomous or semi-autonomous systems present in these cases work well in their particular use case, but more challenging situations may require more sophisticated control algorithms, such as model-predictive control (MPC) [34]. The

[2] `http://www.extremetech.com/`

approaches presented in [6, 15, 20] use various model simplification techniques to reduce the computational complexity of MPC. Particularly for the task of fully autonomous driving, (nonlinear) MPC permits the use of first-principle models (available from literature) that accurately predict the driving behavior even in extreme conditions. Consequently, MPC for autonomous driving has received growing attention in the research community over the past years.

1.2 Time-optimal Driving

In this context of autonomous driving, an interesting question is how to exploit available look-ahead information (for example from digital maps and GPS data) to operate a car optimally with respect to energy consumption, comfort, and time. Due to the natural antagonism of safety and speed in driving, a challenging question may be how to control the car at high velocities while satisfying limitations, such as the boundaries of a race track. Therefore, the target of this master thesis is time-optimal autonomous driving.

Time-optimal MPC approaches have received notable attention in the area of robotics for path following, e.g. [37] and the references therein. There however, the geometric path to be tracked in a minimum-time fashion is predetermined, which renders the problem significantly simpler than the driving problem, where the exact path to be taken by the vehicle is part of the optimization.

Previous attempts at minimum-time driving, based on advanced control strategies exist: in [35], a nonlinear vehicle model with decoupled lateral and longitudinal dynamics was used in minimum lap time problem formulation. In [26], a virtual environment was used to simulate minimum-time behavior. In these approaches, however, only offline open-loop solutions were computed. In [7] a cascading controller scheme was used, generating trajectory references from a simple geometric model and tracking these using a higher-detail model.

In this thesis, instead, we directly use a one-level approach, solving the nonlinear MPC problem with an approximated economic cost function in real-time. The great challenge is the tight real-time bounds imposed on computational times, which make it necessary to reformulate the problem so as to allow for the use of efficient algorithms.

1.3 Goals

The focus of this thesis lies on controlling model cars on time-optimal trajectories. The goal of the thesis is to develop a real-time feasible time-optimal controller based on existing control methods, software tools and vehicle models.

Developed algorithms will be evaluated experimentally on a 1:43 scaled race car setup of our cooperation partner LMS, A Siemens Business. Using such a downscaled setup fits into the paradigm of rapid prototyping: testing the performance of MPC controllers under more realistic conditions without the need for cost-intensive full-scale test vehicles and proving grounds. Furthermore, the small-scale setup can be

used as a demonstrator for LMS to show potential clients the current status of design of on-line methods for autonomous driving in particular, and real-time control in general.

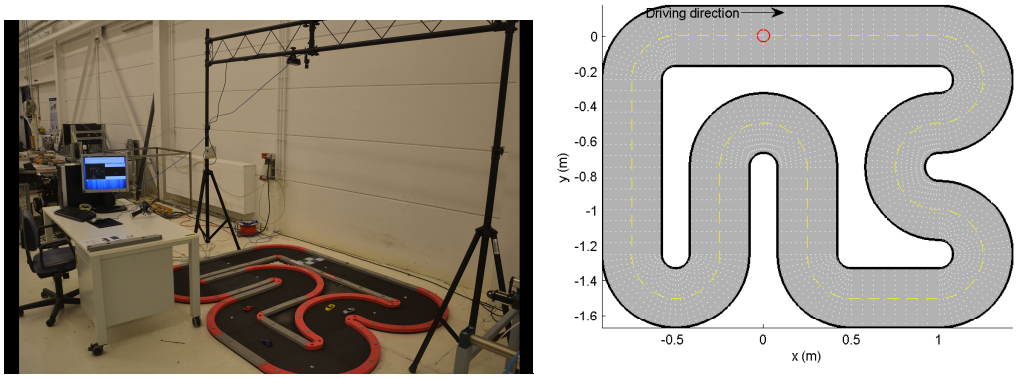
This master thesis text is structured as follows. In Chapter 2, the experimental setup is described in detail. The inner workings of every element of the closed control loop (vision system, state estimation, control calculation, actuation of the RC car) is explained, as well as the software project that implements these components. Next, the candidates for the vehicle and tire models used in this thesis are described in Chapter 3. The main computational challenge in a high-speed environment such as the setup that we intend to use, is to calculate the next control action before the new state measurement comes in. The available computational methods and corresponding software packages are listed in Chapter 4.

The second part of the text consists of a detailed explanation and analysis of the results of the three developed algorithms. In Chapter 5, we base ourselves on the existing linear MPC method and extend it with an on-line heuristic path planning optimization. Because we want to make use of a nonlinear vehicle model, the real-time feasibility of the nonlinear MPC methods used on this setup in particular, is assessed in Chapter 6. In Chapter 7, the nonlinear time-optimal MPC method is introduced. It employs a transformation of coordinates in order to render the calculations more simple. The results are compared to the methods mentioned above. Chapter 8 concludes the text.

Chapter 2

Experimental Setup

In cooperation with LMS, A Siemens Business, an experimental setup is made available for carrying out real-world experiments on miniature race cars (scale 1:43). There are two experimental setups. One is kept on site at LMS for development, the other is shown at conferences around the world. The small-scaled setup used in our experiments is shown in Fig. 2.1a. The race cars are 1:43 miniature race cars from the dNano series of Kyosho. The race track on which the model car drives features a chicane, a U-turn and a longer straight section (see Fig. 2.1b). In this chapter, the different components of the race car system will be described. At the start of the master thesis, the vision system is fully operational, the control algorithm is linear MPC tracking.



(a) The experimental setup at LMS.

(b) The test track used to experiment with the car control algorithm. The dimensions are approximately 2.5 m by 2 m

Figure 2.1: Details of the experimental setup

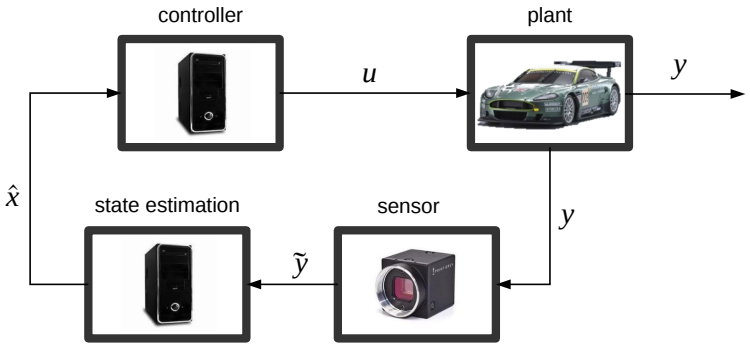


Figure 2.2: Schematical view of the feedback loop in the experimental setup.

2.1 Feedback loop

A feedback scheme of the setup is depicted in Fig. 2.2. At all times, infrared radiation is shed on the track by an infrared lamp mounted above the track. As the car drives on the track, the infrared sensor captures the reflection of the infrared radiation of the reflective markers on the model car. The vision processing unit (VPU) receives the camera images and estimates from them the current state vector (position and velocities) using a Kalman filter. The estimate $\hat{x}$ is used by the vehicle control unit (VCU) to calculate the next control action u. Each component of the experimental setup will be briefly described in greater detail in the following paragraphs. The technical details of the setup and its components can be found in appendix A.

2.2 Vision system

The vision system is in charge of capturing and processing images of the track, in order to come up with an estimate for the current state of the race car. This vehicle state consists of the position, velocity, orientation and angular velocity of the car.

2.2.1 Camera

There are two types of cameras that can be used to capture the vehicle driving on the track: an infrared camera and a color camera.

The infrared light comes from an infrared lamp (LED lamp) mounted on a horizontal bar approximately 2 m above the track. The infrared camera, which is placed next to the infrared lamp, captures images of the track at a chosen fixed rate between 50 Hz and 150 Hz and sends them to the computer over a USB 3.0 connection. If not noted otherwise, the fixed sampling rate of this camera will be 100 Hz, which results from a trade off between vision accuracy and computational effort for the image processing. The reflection of infrared light on three reflective markers placed on the car results in bright blobs on the camera images, which are processed in grayscale.

An RGB camera can also be used to capture the car's behavior. This camera captures three channels (red, green, blue) at 50 Hz. This approach has the advantage that the vision system is more robust, because the total area of the car is now captured instead of the three smaller reflective markers.

2.2.2 The VPU process

The vision processing unit (VPU) receives a camera image, thresholds it, and extracts the position and orientation of the car. In the case of the infrared camera, this is done by performing some geometrical calculations on the triangular placing of the reflective markers on the car. On the RGB images however, a bounding box is drawn around the brightest pixel. This rectangle represents the car, and the position and orientation can be extracted directly from this rectangle. In the next step, we use a Kalman filter to estimate the current state of the vehicle. The state vector is then sent over to the control process. After this is done, the VPU waits for another camera image, which guarantees that the vision system runs at a fixed rate (50 Hz or 100 Hz depending on which camera is used).

The VPU uses a Kalman filter (see [25] for a detailed description) for the filtering of noisy and missing measurements. These missing measurements result from the fact that the detection is not perfect: external factors such as sunlight, warm halogen spots or a person walking around the track can cause the car to go by unnoticed. In a Kalman filter, a model is included. In our case, it is that of a point mass moving at a constant speed. The possible accelerations of the vehicle are considered as noise on the constant motion. In future work, this model might be extended to a more sophisticated vehicle model (e.g. a bicycle model), or moving to an Extended Kalman filter. Another possibility would be to incorporate this filtering into the moving horizon paradigm and switch to Moving Horizon Estimation (MHE).

2.3 Control system

The vehicle control unit (VCU) takes the vehicle state coming from the VPU to determine the next control action. This is done by the methods presented in Chapters 5, 6 and 7. The vehicle states (position, orientation, velocity and angular velocity) are sent over from VPU to VCU via UDP communication between these two processes. The output of the Vehicle Control Unit is the steering angle and the throttle.

2.3.1 Real-time computer

Both the VPU and the VCU are two concurrently running processes on a real-time computer. This desktop computer runs Debian LINUX 7.0 'Wheezy' with its *RT Preempt* real-time kernel patch installed. This guarantees soft real-time properties: priorities of different processes can be assigned, but no hard timing constraints can be imposed. Tasks with higher priority are given an advantage over other tasks by the scheduler of the operating system. In our system, the vision processing system is given the highest priority, in order to ensure as fixed a sampling rate as possible.

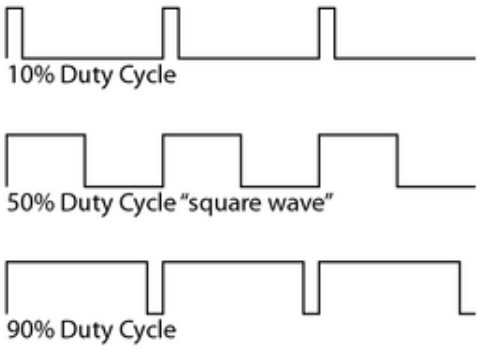


Figure 2.3: Dutycycle as input to the model car. Source:[1]

2.3.2 Communication and local control of the race car

The control inputs are sent to the car via a wireless connection. The steering angle is an action that can be immediately translated by the actuators on the car. However, we can not set the torque on the rear axle directly. Instead, we can control the *dutycycle* of the DC motor. This is the fraction of the time that the input voltage to the DC motor is 'high' (see Fig. 2.3). A negative value for D drives the rear axle backward. Two types of race car interfaces are available, which will be described subsequently.

The first is a classical Kyosho dNano model car which is steered by a manual controller. This manual controller's integrated circuit is modified to receive signals from the real-time computer. The conversion from digital signals (coming from the digital output card of the computer) to analog signals (going to the manual controller) is done by a data acquisition board (DAQ).

The other type of interface is a dNano race car that has been extended with a custom communication module, such that the control signals can be sent over an Asynchrounous Connection-Less Bluetooth (ACL) communication link. The real-time computer uses an off-the-shelf Bluetooth dongle to communicate with the onboard antenna.

The disadvantage of the first setup is the high cost of the DAQ board. However, this way of using the cars allows for quick replacement of broken race cars, whereas in the second setup, each new car has to be extended with a custom Bluetooth interface.

It is the Bluetooth communication interface that will be used in the remaining chapters of this thesis, as the second experimental setup is not currently present at LMS.

[1] `wiki.bildr.org/images/thumb/3/35/Duty_cycle.png/300px-Duty_cycle.png`

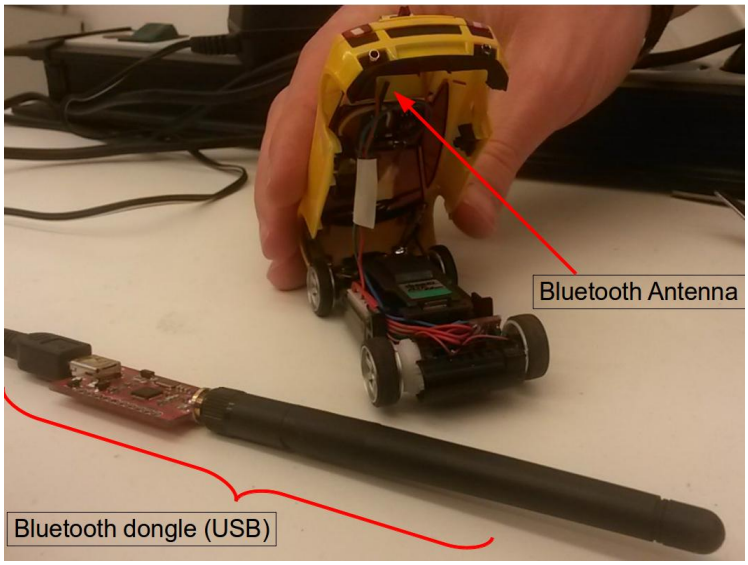


Figure 2.4: The Bluetooth race car interface.

2.4 Software architecture

The software that implements all of the above, is self-written C++ code, designed in a *modular* way. For example, the vison processing unit (VPU) and the vehicle control unit (VCU) are two different processes and thus completely independent. The VPU was already present at the start of the master thesis, the VCU was written from scratch. The one-way UDP communication from VPU to VCU, which consists of current estimate of the state vector, is the only interconnection between the two processes. This is an example of loose coupling in the architecture.

Inside of the VCU, the modularity principle is carried out further, with each class performing a single, coherent task. See Fig. 2.5 for a schematic description. The `main.cpp` code implements the main control loop:

1. Wait for current state estimate from the VPU,
2. Calculate the necessary track data starting from this estimate,
3. Compute the next control action and apply it to the miniature race car.

It makes use of the following classes:

- `trackParser.cpp`: Reads the geometrical track data from file and parses it.
- `controller.cpp`: Calculates the next control action.
- `RCdriver.cpp`: Transforms the calculated control actions into suitable signals to be sent to the car.

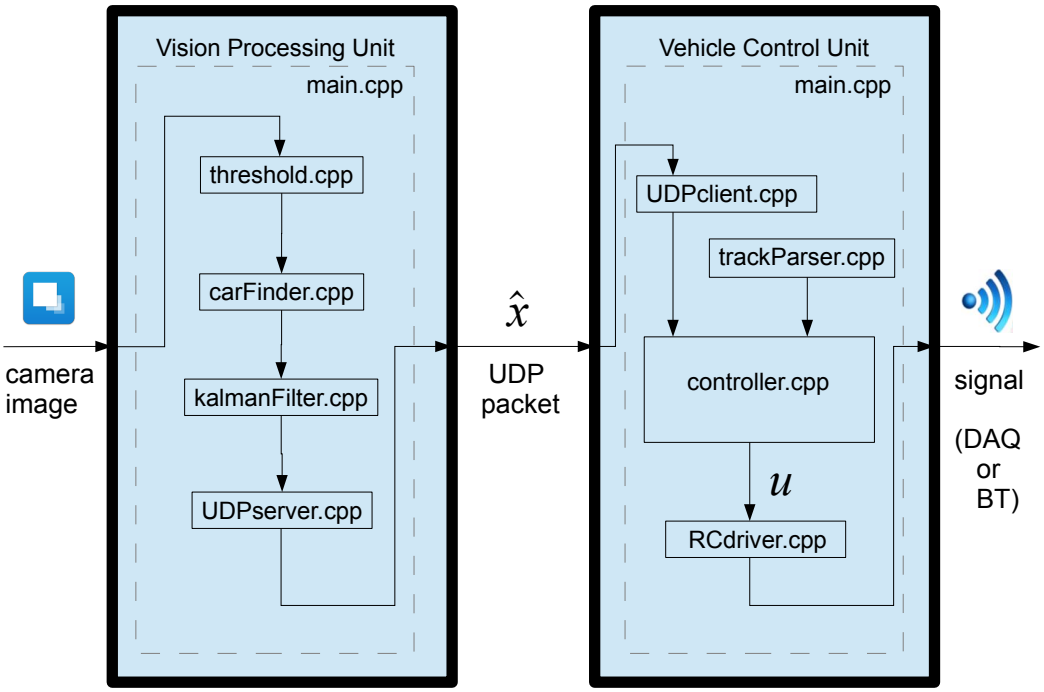


Figure 2.5: The architecture of the race car software.

- `UDPclient.cpp`: receives and parses the UDP packets coming from the vision system.

The modularity shows its advantages in the `controller.cpp` and `RCdriver.cpp` classes: the existing controller (e.g. MPC controller) can be replaced by a new controller without having to change anything to the main program; and the type of communication (via a DAQ card or via Bluetooth) is encapsulated in `RCdriver.cpp`. The software architecture and the information flow is depicted schematically in Fig. 2.5.

Chapter 3

Vehicle and Tire Models

Central to any form of model-based automotive control is the vehicle model. Different ground vehicle models exist, from very simple pure kinematic models to full-scale multibody vehicle models. In the following, a 'model' will always mean an ODE model. A challenging component in any vehicle model is the modeling of road-tire interactions which are dealt with by tire models. First, some common tire models will be described, followed by the different vehicle models relevant for this thesis.

For all models, we assume a rear-wheel driven car, with front-wheel steering, such as the dNano cars in the experimental setup. The global coordinates of the car (measured in the center of gravity, CG) at each point in time are given by (X, Y), and the orientation of the car with respect to the positive X-axis is ψ (see Fig. 3.1). Attached to the car is a local moving coordinate system (x, y), with the x-axis aligned with the longitudinal axis of the car. The rotational velocity is denoted by ω (both for the global and local coordinate systems).

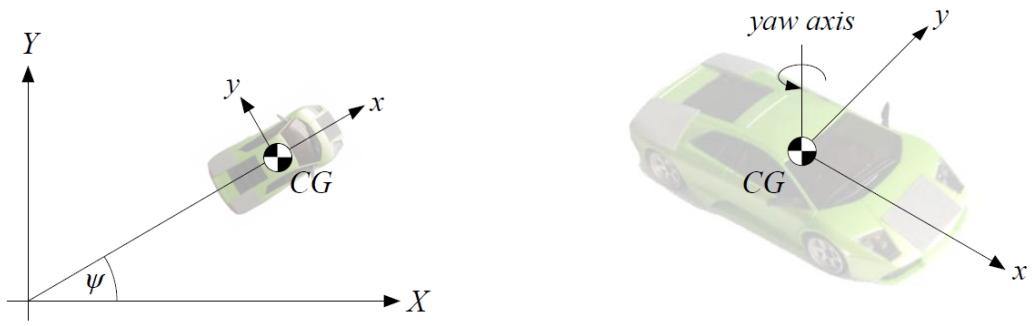


Figure 3.1: Global coordinate system (X, Y) and local coordinate system attached to the car's center of gravity (x, y). Source: [36].

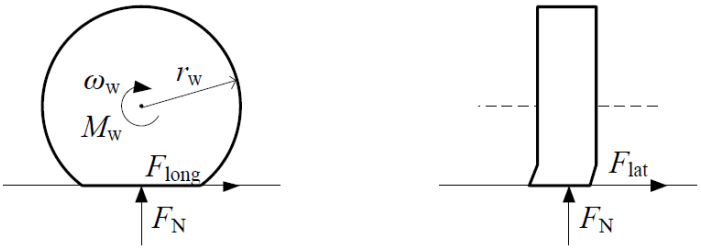


Figure 3.2: Decoupling of the tire forces. Side view (left) and front view (right). Source: [36].

3.1 Tire models

As a car drives, the tires almost always stay on the ground, resulting in a contact force. This interaction force between road surface and tire surface can be modeled in different ways. The decoupled tire force model will be treated first, the combined model is explained afterwards. The discussion of different tire models will be mainly based on [36], [27] and [32].

3.1.1 Decoupled tire models

For simplicity, it is common practice to decouple the tire force into a longitudinal and lateral force component (see Fig. 3.2).

Longitudinal forces The longitudinal slip coefficient is defined as

$$\beta = \frac{\omega_w r_w - v_x}{v_x}, \tag{3.1}$$

with ω_w, r_w as in Fig. 3.2 and v_x is the longitudinal component of the velocity of the car. In acceleration, the product $\omega_w r_w$ is slightly higher than v_x (a so-called burnout), and the resulting slip coefficient β is positive. When the car brakes, β becomes negative. The longitudinal tire force is modeled as a function of this slip coefficient:

$$F_{\text{long}} = f(\beta). \tag{3.2}$$

Lateral forces Also the lateral tire forces can be modeled as a function, but now the independent variable is the slip angle α (see Fig. 3.3). It is the angle between the direction of the tire and the velocity of the midpoint of the wheel. Thus, the lateral tire force becomes

$$F_{\text{lat}} = f(\alpha). \tag{3.3}$$

A first possible approximation to longitudinal tire forces is $f(\beta) = 0$. Although very simple, it is an often used assumption on longitudinal tire forces on front

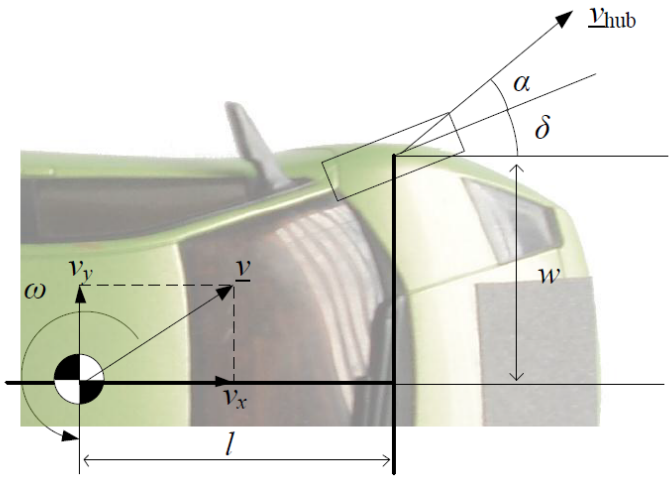


Figure 3.3: Detail of the car's geometry with steering angle δ and slip angle α. The velocity v_{hub} denotes the velocity of the middle of the tire. Source: [36].

wheels, in cars that are rear-wheel driven. Another possible approximation is a linear function, with a saturation:

$$F_{\text{long}}(\beta) = \begin{cases} C_\beta \beta & \text{for } |C_\beta \beta| < F_{\text{long,max}} \\ \text{sign}(\beta) F_{\text{long,max}} & \text{for } |C_\beta \beta| \geq F_{\text{long,max}} \end{cases} \tag{3.4}$$

As above, the lateral tire force can be neglected ($F_{\text{lat}} = 0$), or it can be modeled as a linear function with a saturation:

$$F_{\text{lat}}(\beta) = \begin{cases} -C_\alpha \alpha & \text{for } |C_\alpha \alpha| < F_{\text{lat,max}} \\ -\text{sign}(\alpha) F_{\text{lat,max}} & \text{for } |C_\alpha \alpha| \geq F_{\text{lat,max}} \end{cases}, \tag{3.5}$$

where the minus sign comes from the definition of α in Fig. 3.3.

Another very well known model for tire forces is Pacejka's so-called *Magic formula* [32]: it is a semi-empirical law which works well in a variety of tires and operating conditions. The Magic formula is as follows:

$$-F_{\text{lat}}(\alpha) = D \sin(C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))), \tag{3.6}$$

where B is the stiffness factor, C and E are shape factors and D is the peak factor. Its characteristic shape is depicted in Fig. 3.4. Note again the minus sign in the lateral tire force because of the definition of α.

3.1.2 Combined tire models

A more realistic approach to tire forces is to realize that tire force is limited, so longitudinal and lateral forces have to 'share' grip, they become dependent on each

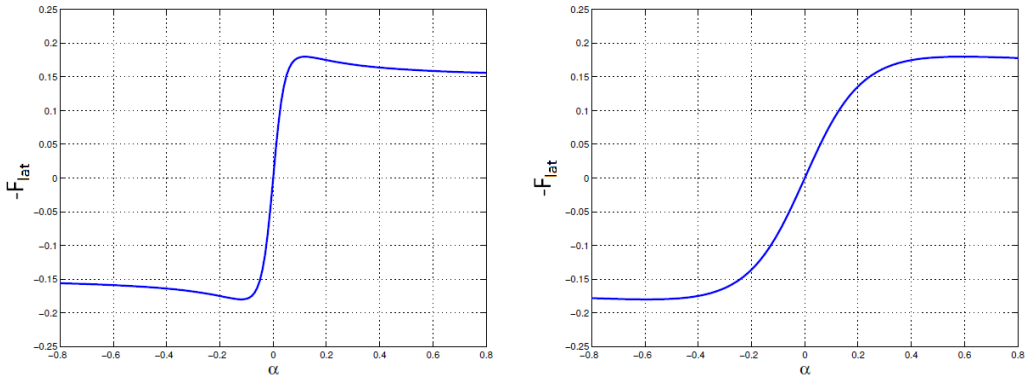


Figure 3.4: Left: Pacejka's original Magic formula for tire contact forces. Right: Simplified formula with $E = 0$. Source: [27]

other. The tire forces lie inside the ellipse

$$\left(\frac{F_{\text{long}}}{F_{\text{long,max}}}\right)^2 + \left(\frac{F_{\text{lat}}}{F_{\text{lat,max}}}\right)^2 \leq 1. \tag{3.7}$$

3.2 Vehicle models

In the following paragraphs, we consider three vehicle models. Note that this discussion does not seek to be complete, but presents some vehicle models that are of special interest to this particular master thesis.

3.2.1 Four wheel vehicle model

In [18], a four-wheel vehicle model is used for real-time obstacle avoidance. A more detailed discussion of this model can be found in [41]. The four wheels of the vehicle are modeled as independent bodies. Furthermore, the heave (vertical) motion of the car is neglected, but the load transfer on the different wheels is taken into account. A detailed depiction of the geometry can be found in Fig. 3.5.

Besides the three degrees of freedom of the rigid vehicle body, the rotational dynamics of each wheel are also included in the vehicle model. A Pacejka tire model (cf. section 3.1) is employed to model the road-tire interaction, taking into account the load transfer during handling, under the assumption of an infinitely stiff suspension.

Due to the limited availability of model parameters for miniature cars (particularly regarding road-tire interaction) and the slightly different underlying physics, we use a reduced model for control in this context. In particular, road-tire interaction and the load transfer are neglected in this thesis, yielding a rather standard slip-free bicycle model, which we briefly describe in the following.

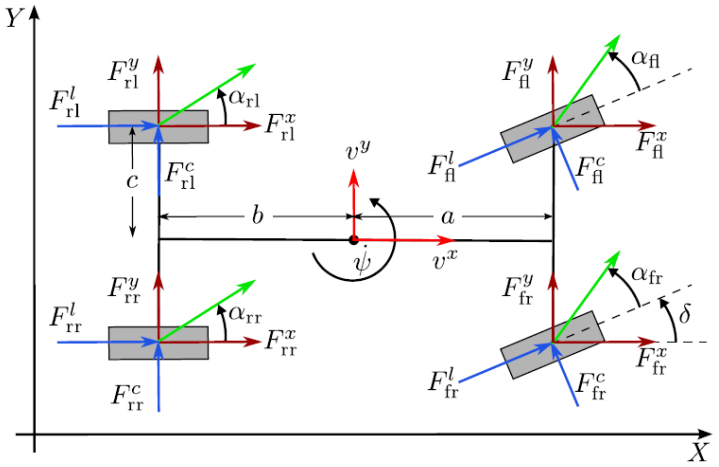


Figure 3.5: Four-wheel vehicle model with load transfer. Source: [18].

3.2.2 Bicycle model

A bicycle model simplifies a four-wheel car's geometry to a two-wheel geometry, as can be seen in Fig. 3.6. The symmetry of the appearing forces is inferred from the lateral symmetry of the geometry of most cars.

Let us write the dynamics of the car in terms of velocities, and let us project them onto the moving (x, y) coordinate system attached to the car (see Fig. 3.1):

$$m\dot{v}_x = F_x - mv_y\omega \tag{3.8a}$$
$$m\dot{v}_y = F_y + mv_x\omega \tag{3.8b}$$
$$I_z\dot{\omega} = M_z, \tag{3.8c}$$

where ω is the yaw rate of the car around the z-axis. Note that the z-axis remains the same in both the local and global coordinate systems. The forces F_x and F_y, aligned with the longitudinal and lateral directions of the car respectively, come from a summation of tire forces (front and rear), air drag and roll resistance. Since the dNano cars are rear-wheel driven, we assume that $F_{\mathrm{f,long}} = 0$.

We can combine these equations of motion with the kinematic model of the point mass, projected on the (x, y) coordinate system, to obtain a complete set of equations

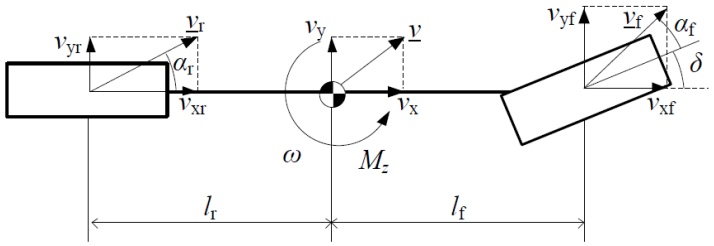


Figure 3.6: Geometry for the bicycle model. Source: [36].

for the bicycle model:

$$\dot{X} = v_x \cos(\psi) - v_y \sin(\psi) \tag{3.9a}$$
$$\dot{Y} = v_x \sin(\psi) + v_y \cos(\psi) \tag{3.9b}$$
$$\dot{\psi} = \omega \tag{3.9c}$$
$$\dot{v}_x = \frac{1}{m}(F_{r,x} + F_{\text{drag}} - F_{f,\text{lat}} \sin(\delta) - m v_y \omega) \tag{3.9d}$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} + F_{f,\text{lat}} \cos(\delta) + m v_x \omega) \tag{3.9e}$$
$$\dot{\omega} = l_f F_{f,\text{lat}} \cos(\delta) - l_r F_{r,y}, \tag{3.9f}$$

where f, r means front and rear respectively, δ is the steering angle, and F_{drag} is the driving resistance, a combined effect where air drag and roll resistance are the dominating contributions. A common approximation to this resistive force can be found in the book of Giancarlo Genta [22]:

$$F_{\text{drag}} = -C_{r0} - C_{r2} v_x^2. \tag{3.10}$$

The coefficients C_{r0} and C_{r2} are to be determined experimentally. Note that the lateral forces on the front tires can be modeled with any of the tire models discussed above.

3.2.3 Slip-free bicycle model

As both the race track and the tires of the model race cars are made of rubber (rubber on rubber results in high value for the friction coefficient), we assume a slip-free vehicle model in this master's thesis. It is valid under the following assumptions, as is explained in [36]:

- The tire force limit is never exceeded,
- The slip angles α_f and α_r are small (see Fig. 3.6),
- The steering angle δ is small.

Table 3.1: Bicycle Model Parameters

Parameter	Unit	Physical meaning	Value
C_1	$-$	geometrical (l_r/l)	0.5
C_2	m^{-1}	geometrical $(1/l)$	17.06
C_{m_1}	$\mathrm{m/s}^2$	motor parameter	12.0
C_{m_2}	$1/\mathrm{s}$	motor parameter	2.17
C_{r_2}	$1/\mathrm{m}$	second order friction parameter	0.1
C_{r_0}	$\mathrm{m/s}^2$	zero order friction parameter	0.6

Under these assumptions, we can use two approximations. The first is steady state turning, which means that the lateral component of the velocity at the rear wheels is zero. The second approximation is

$$v_x \approx v \tag{3.11}$$

$$v_y \approx v\frac{l_r}{l}\delta \quad , \tag{3.12}$$

see Fig. 3.6.

Using these approximations, we come to the slip-free bicycle model central in this thesis (see also [36]):

$$\dot{X} = v\cos(\psi + C_1\delta) \tag{3.13a}$$

$$\dot{Y} = v\sin(\psi + C_1\delta) \tag{3.13b}$$

$$\dot{\psi} = v\,\delta\,C_2 \tag{3.13c}$$

$$\dot{v} = C_{m_1}D - C_{m_2}\,D\,v - C_{r2}\,v^2 - C_{r_0} - (v\,\delta)^2\,C_2\,C_1. \tag{3.13d}$$

By δ and D we denote the steering angle and the dutycycle applied to the DC motor, respectively. A summary of model parameters $C_\cdot$ and their respective interpretation can be found in Table 3.1.

In this chapter, different physical vehicle and tire models available from literature were introduced. On-line data like slip angle is often hard to estimate. Furthermore, with computational complexity in mind, we favor simple vehicle models. Therefore, we choose to employ a slip-free bicycle model (cf. Eq. (3.13)) in the rest of this master thesis.

Chapter 4

Computational methods for real-time optimization

4.1 Model Predictive Control

As of today, PID control remains the most widely used control technique in the majority of application areas for automatic control. However, more advanced control algorithms such as model predictive control (MPC) have been developed since the 1960s. MPC is already succesfully applied in the chemical process industry, where rather slow dynamics (sampling times of seconds or even minutes) allow for complex control calculations. Recently, a number of contributions to real-time nonlinear model predictive control (NMPC) theory and applications have led to an increasing interest in this technique for fast mechatronic systems [11, 16, 37]. In the area of autonomous ground vehicles, MPC has become an attractive method for the tracking of feasible trajectories. All this serves as a motivation to use the MPC paradigm for our purposes. The following introduction to MPC is based on the book by Jim Rawlings and David Mayne [34].

4.1.1 Problem formulation

Model predictive control has its roots in the field of optimal control. The basic idea of MPC is to use a dynamic model to predict the system's behavior, and optimize this behavior to obtain the best control decision at the current time. Usually, MPC is carried out with a receding horizon: at each time step, the behavior up until some time horizon is considered. The dynamic models, which are central to any form of MPC, we will use are the ordinary differential equation (ODE) models

$$\frac{d\xi}{dt} = f(\xi, u, t) \tag{4.1}$$

$$\xi(t_0) = \xi_0, \tag{4.2}$$

in which $\xi \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, the output $y \in \mathbb{R}^p$, and $t \in \mathbb{R}$ is time. The state equations and the output equations are denoted by f and h,

respectively. As can be seen in the equations above, the control actions of the MPC controller depend on the current state ξ_0 of the system.

This being said, we can proceed to the MPC problem formulation. MPC tries to minimize a certain cost function, defined as

$$V_T(\xi(\cdot), u(\cdot)) = \int_0^T l(\xi(t)), u(t))dt + V_f(\xi(T)) \ , \tag{4.3}$$

with $l(\cdot)$ denoting the running cost and V_f the final cost. The optimal control problem formulation of MPC to be solved at each sampling time is then

$$\begin{aligned} \underset{\xi(\cdot),u(\cdot)}{\text{minimize}} \quad & V_T(\xi(\cdot), u(\cdot)) \\ \text{subject to} \quad & \dot{\xi} = f(\xi, u), \quad t \in [0, T], \\ & \xi(0) = \xi_0 \end{aligned} \tag{4.4}$$

4.1.2 Motivation

A first contrast that MPC makes with more traditional controllers is the fact that the control decisions are optimized with respect to a model. For instance, in PID, there are no optimization features whatsoever (except maybe for the tuning of the gain parameters). Another big advantage of MPC over conventional feedback controllers (such as PID controllers) is the ability to handle *constraints*, by introducing them in the control problem formulation. In a linear quadratic regulator (LQR) for example, it is possible to put constraints on the control variables only (not the state variables), but then the linearity of the problem is affected. A third beneficial property of MPC is that it is *model-based*, allowing the prediction of future behavior of the system model, where in other controllers the control action is often based on the tracking error

$$e(t) = y_{\text{ref}}(t) - y(t), \tag{4.5}$$

resulting in no lookahead features whatsoever.

4.1.3 Variants

There are different flavors of MPC: the easiest form is without doubt *linear* MPC. Here, the system equations are assumed to be linear, and only a quadratic cost function is allowed with linear constraints. It can be shown that linear MPC problems can be transformed into a dense Quadratic Program (QP) [5], which can be solved exactly at each time step. However, in this thesis, we choose to investigate nonlinear MPC (NMPC) methods as well, in order to obtain more accurate predictions. In this case, both the constraints and the cost function can be an arbitrary nonlinear function in C^2. Of course, this problem is much harder to solve numerically, in particular in real-time applications in the microsecond range.

Often, linear or nonlinear MPC is used with a tracking formulation. An alternative for this is *Economic MPC* [33]. This control technique has roots in the process control of chemical plants. Usually, the most profitable setpoint or trajectory in the

objective function is set externally (eg. some information management system or prior data). However, with economic MPC, the controller directly optimizes the economic performance of the process, rather than tracking to a setpoint or trajectory. In the time-optimal method formulated in this master thesis, 'economic perfomance' is translated to time: the higher the economic performance, the lower the time.

4.2 Linear MPC

While many real-world models are not linear, they can be locally approximated by a linear model. Historically, linear models were used by MPC practicioners up until a few years ago for the simple reason that nonlinear methods were computationally infeasible.

Another important property of linear MPC is that, in principle, all linear MPC formulations are *convex* optimization problems, for which very efficient methods exist. To be convex, a linear MPC problem must have a convex cost function, as well as convex constraints on states and control variables. This is not uncommon in practice: a least-squares cost function

$$l(\xi(t)), u(t)) = \left\|\xi(t) - \xi_{\text{ref}}(t)\right\|_2 + \left\|u(t) - u_{\text{ref}}(t)\right\|_2, \tag{4.6}$$

combined with a time-invariant linear model

$$\dot{\xi}(t) = A\xi(t) + Bu(t) \tag{4.7}$$

and simple state and control bounds

$$\xi_L \leq \xi(t) \leq \xi_U, \quad \xi_L, \xi_U \in \mathbb{R}^n \tag{4.8a}$$
$$u_L \leq u(t) \leq u_U, \quad u_L, u_U \in \mathbb{R}^m \tag{4.8b}$$

is an example of a convex linear MPC problem. Such an MPC problem can be approximated as a QP (see section 4.3) and can be solved efficiently.

For a computable solution to be possible, the above problem needs to be reduced to a finite number of optimization variables. The discrete-time linear MPC optimization problem amounts to

$$\underset{\substack{u_0 \ldots u_{N-1} \\ \xi_1 \ldots \xi_N}}{\text{minimize}} \quad \sum_{k=0}^{N-1} l_k(\xi_k, u_k) + V_N(\xi_N) \tag{4.9a}$$

$$\text{subject to} \quad \xi_{k+1} = A_k\xi_k + B_k u_k, \qquad k = 0, \ldots, N-1, \tag{4.9b}$$
$$E_{\xi,k}\xi_k \leq F_{\xi,k} \qquad k = 0, \ldots, N, \tag{4.9c}$$
$$E_{u,k}u_k \leq F_{u,k} \qquad k = 0, \ldots, N-1, \tag{4.9d}$$
$$\xi_0 = \xi_{\text{init}}. \tag{4.9e}$$

In this formulation, N is the horizon length, $l_k(\cdot)$ is the running cost and V_N the final cost. Note that the system equations and the constraints are linear. If l_k and V_N are chosen convex, then the optimization problem becomes convex. Moreover, if

the cost function is *quadratic*, the optimization problem is a quadratic programming (QP) problem.

The linear MPC scheme then consists of repeating the following steps in one sampling time:

1. Measure or estimate the current state ξ_{init},
2. Solve optimization problem (4.9e),
3. Apply the first control u_0 to the system.

4.3 QP solution method

A QP problem is a optimization problem with a quadratic cost function and linear constraints (any number of p equality constraints and q inequality constraints). The vector of optimization variables, also called decision variables, is denoted by $w \in \mathbb{R}^n$. A QP problem formulated in it's most basic form is

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2} w^T H w + f^T w && \text{(4.10a)} \\ \text{subject to} \quad & E_{\text{eq}} w = F_{\text{eq}} && \text{(4.10b)} \\ & E w \leq F. && \text{(4.10c)} \end{aligned}$$

In the formulation, the matrix H is a $n \times n$ semi-positive definite matricx, or $H \in \mathbb{S}_+^{n \times n}$. The other matrices are $E_{\text{eq}} \in \mathbb{R}^{p \times n}$, $E \in \mathbb{R}^{q \times n}$ and the vectors $f \in \mathbb{R}^n$, $F_{\text{eq}} \in \mathbb{R}^p$, and $F_{\text{eq}} \in \mathbb{R}^q$.

Several algorithms exist for solving convex QP problems; commonly used are the interior point methods and the active set methods. In this master's thesis, we will use the qpOASES solver [1], which is an active set method and especially designed for usage in MPC methods (both linear and nonlinear). Other available QP solvers are FORCES [14] and qpDUNES [19].

qpOASES uses a so-called online active set strategy. An active set method solves the QP problem by reducing the full set of constraints to a subset (the active set), and selects constraints to be added to or removed from the active set in each iteration. A detailed description of the algorithm can be found in [17].

4.4 Nonlinear MPC

A disadvantage of linear MPC when used in a nonlinear setting is the need for an adequate linearization. This linearized model is only valid close to the linearization point. However, when changing to a nonlinear MPC formulation, instead of solving one QP in each iteration, a nonlinear program (NLP) has to be solved. In general, this NLP is non-convex, so we have no guarantee that the solution found by the solver is a global optimum.

We directly jump to the formulation of a generic NMPC problem:

$$\begin{aligned}
\underset{\substack{u_0 \dots u_{N-1} \\ \xi_1 \dots \xi_N}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} l_k(\xi_k, u_k) + V_N(\xi_N) && && \text{(4.11a)} \\
\text{subject to} \quad & \xi_{k+1} = f(\xi_k, u_k), && k = 0, \dots, N-1, && \text{(4.11b)} \\
& h(\xi_k, u_k) \leq 0, && k = 0, \dots, N-1, && \text{(4.11c)} \\
& \xi_0 = \xi_{\text{init}}. && && \text{(4.11d)}
\end{aligned}$$

Now, the functions l_k, V_N, f, h can be arbitrary nonlinear functions in C^2, which poses a computational challenge.

4.5 NMPC solution method

A succesful strategy for solving NMPC sufficiently fast and accurately is the RTI scheme ([11]) which is based on sequential quadratic programming (SQP). This scheme is presented briefly in the following. An overview of numerical methods for nonlinear MPC can be found in [12].

4.5.1 Sequential Quadratic Programming

Consider the nonlinear optimization problem

$$\begin{aligned}
\min_{\zeta} \quad & V(\zeta) \\
\text{s.t.} \quad & h(\zeta) \leq 0.
\end{aligned} \tag{4.12}$$

The idea of Sequential Quadratic Programming (SQP) is to solve by advancing in the following way:

$$\zeta_{k+1} = \zeta_k + \beta p, \tag{4.13}$$

until some converge criterion is met. The parameter $\beta \in [0, 1]$ determines the step size. This can be calculated with a globalisation strategy like line search or the trust-region method. The step p is the solution of the QP

$$\begin{aligned}
\min_{p} \quad & \nabla V(\zeta_k)^T p + \frac{1}{2} p^T B p \\
\text{s.t.} \quad & h(\zeta_k) + \tfrac{\partial h}{\partial \zeta}(\zeta_k) p \leq 0,
\end{aligned} \tag{4.14}$$

where B is the Hessian of (4.12) at ζ_k or an approximation thereof. An ill-conditioned or indefinite Hessian can be regularized by adding Levenberg-Marquardt regularization:

$$B = J^T J + \alpha * I,$$

where J is the Jacobian of (4.12), and the regularization parameter $\alpha > 0$. For more algorithmic details, consult [30].

4.5.2 The real-time iteration scheme

The real-time iteration (RTI) scheme for nonlinear model predictive control (NMPC) was first proposed in [11]. A very simplified schematical view of this method is shown in scheme 1, see [11] for the details. The idea is to do as many calculations as possible before obtaining the new data in each sampling time. From the moment the data is available, the next control action is determined. In contrast to a regular SQP scheme, the RTI scheme performs only one iteration per sampling time.

Scheme 1 RTI scheme (simplified)

repeat online:

1. Preparation Step
   - Linearization & Condensing
   - Wait for data (inital value)
2. Feedback Step
   - Calculate first control value by solving a QP
   - Apply control to plant
   - Shift variables

until

The 'condensing' in step 1 is the procedure that reduces the size of the QP by eliminating intermediary states. The computations in step 2 involve solving only one QP, because the solution from the previous iteration is already a good approximation to the solution of the NLP. The RTI scheme has been applied succesfully in many practical applications, for example a high purity distillation column [13] or an overhead crane [38].

4.5.3 ACADO toolkit

In this master's thesis, we will make use of the ACADO software package [24, 23, 2]. This package is an algorithm collection for dynamic optimization; it contains solvers for moving horizon estimation (MHE), nonlinear optimal control and nonlinear model predictive control. The ACADO code generation tool (CGT) is of most interest to us, as it produces self-contained C++ code which solves NMPC problems fast. The code that the ACADO CGT produces is an implementation of the above described real-time iteration (RTI) scheme.

Chapter 5

Two-level linear MPC: trajectory planning and tracking

At the start of the master thesis, the algorithm running on the experimental setup is linear model predictive control (MPC) with a tracking formulation. The reference trajectory tracked by the car is computed beforehand. The first algorithm we present on our way towards time-optimality is an extension of this, namely linear MPC tracking with on-line trajectory planning.

The resulting algorithm consists of two levels. The trajectory that is to be tracked is determined by the upper level of the algorithm. On the upper level, the path that will be tracked can be chosen as the shortest path or the path of least curvature. On the lower level, linear tracking MPC is carried out. A similar algorithm was used in [21] to simulate a passenger vehicle driving on an icy road.

This on-line computed path is a heuristic approach to the time-optimal problem in the following way: if the path has minimum curvature, then the car needs to brake as little as possible, resulting in a higher average velocity. Tracking the shortest path, on the other hand, results in a shorter total distance driven by the car, resulting in a lower time per lap.

The motivation to start with a heuristic method is the following: the linear tracking algorithm proved to work on the experimental setup, so a natural extension to this algorithm is to provide other reference trajectories, by on-line computation.

5.1 Linear tracking MPC

The existing method, prior to the start of the master's thesis, is linear tracking model predictive control. For completeness, this algorithm is explained briefly. For a detailed description, consult [40].

The tracking problem is solved by reformulating it as a *regulation* problem, which means that we intend to drive the state vector to zero. To this end, we introduce

the error state

$$\xi = \begin{bmatrix} x_e \\ y_e \\ \psi_e \\ v_e \end{bmatrix} = \begin{bmatrix} \cos(\psi^\sigma) & \sin(\psi^\sigma) & 0 & 0 \\ -\sin(\psi^\sigma) & \cos(\psi^\sigma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X - X^\sigma \\ Y - Y^\sigma \\ \psi - \psi^\sigma \\ v - v^\sigma \end{bmatrix}, \tag{5.1}$$

where the superscript $^\sigma$ denotes a state belonging to the reference trajectory (in this particular case, the centerline). Note that we switch from a global reference frame (X, Y) to a local frame (x, y) attached to the car. The error state will be used to formulate the error dynamics corresponding to the system in Eq. (3.13). This will replace the original system dynamics. We can do so by introducing the system dynamics into the error state formulation, which gives

$$\dot{\xi} = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\psi}_e \\ \dot{v}_e \end{bmatrix} = \begin{bmatrix} v\cos(\psi_e + C_1\delta) - v^\sigma(1 - \kappa y_e) \\ v\sin(\psi_e + C_1\delta) - v^\sigma(\kappa x_e) \\ v\delta C_2 - v^\sigma\kappa \\ C_{m_1}D - C_{m_2}\,D\,v - C_{r2}\,v^2 - C_{r_0} - (v\,\delta)^2\,C_2\,C_1 - a^\sigma \end{bmatrix} \tag{5.2}$$

with $\kappa = \omega^\sigma / v^\sigma$ the curvature of the track. We collect the control inputs in the vector $u = [\delta, D]^T$.

Constraints The constraints on the input vector $u = [\delta, D]^T$ are the following:

$$\frac{-25 \cdot \pi}{180} \leq \delta_k \leq \frac{25 \cdot \pi}{180}, \tag{5.3}$$

$$-1 \leq D_k \leq 1 \tag{5.4}$$

The state constraints are given by

$$\Delta w - w/2 \leq e_{y,k} \leq \Delta w + w/2, \tag{5.5}$$

where w is the width of the track, and Δw is the distance between the current position on the reference trajectory and the middle of the road (see Fig. 5.1).

After linearization and discretization, we obtain the following linear MPC problem:

$$\min_{\xi \in \mathbb{R}^{n_\xi \times N}, u \in \mathbb{R}^{n_u \times N}} \quad \sum_{k=1}^{N} \xi_k^T Q_k \xi_k + \sum_{k=0}^{N-1} u_k^T R_k u_k \tag{5.6a}$$

$$\text{subject to} \quad \xi_{k+1} = A_k \xi_k + B_k u_k, \qquad k = 0 \dots N-1 \tag{5.6b}$$

$$e_{y,k} \in [\Delta w - w/2, \Delta w + w/2], \qquad k = 1 \dots N \tag{5.6c}$$

$$u_k \in [\frac{-25 \cdot \pi}{180}, \frac{+25 \cdot \pi}{180}] \times [-1, 1], \quad k = 0 \dots N-1. \tag{5.6d}$$

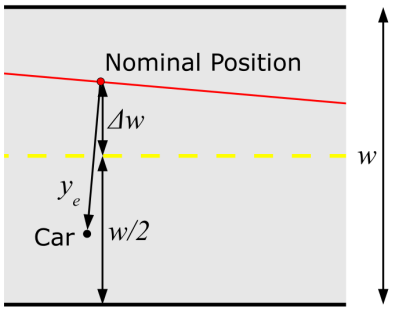


Figure 5.1: Wall constraints on the car. Source: [40]

Weighting matrices The symmetric weighting matrices Q, R allow for a distinction in tracking emphasis. State and control errors with a high weight are tracked more closely. Often, these matrices are chosen diagonally, because weights on the crossterms are not always physically meaningful. High weights on the position error are selected to achieve the desired accurate tracking. A relatively high weight on the velocity ensures a smooth driving. Following weighting matrices result in a satisfactory behavior:

$$Q_k = \texttt{diag}([1, 1, 10^{-5}, 1])$$
$$R_k = \texttt{diag}([8 \cdot 10^{-5}, 8 \cdot 10^{-5}]).$$

5.2 Linear MPC tracking: Results

Some results of linear MPC tracking are presented here to serve as a point of comparison of newly developed methods.

5.2.1 Simulation results

In tracking of the centerline, the goal is to keep the longitudinal and lateral position error x_e, y_e small. We employ a linearization of the bicycle model presented earlier (Eq. 3.13) as the model in the controller. The sampling time is chosen to be 0.01 s. In Fig. 5.3, the tracking performance of the linear MPC tracking method is shown. The resulting trajectories can be seen in Fig. 5.2.

From the figures, it is clear that the tracking performance deteriorates with increasing reference velocity v_{ref}. For even higher speeds, the underlying QP problem becomes infeasible.

5.2.2 Experimental results

The linear MPC tracking formulation is also tested on the experimental setup: the results are shown in Figs. 5.4 and 5.5. Note that the tracking of the trajectory is not very effective, even at a modest speed of 0.5 m/s. A comparison with nonlinear

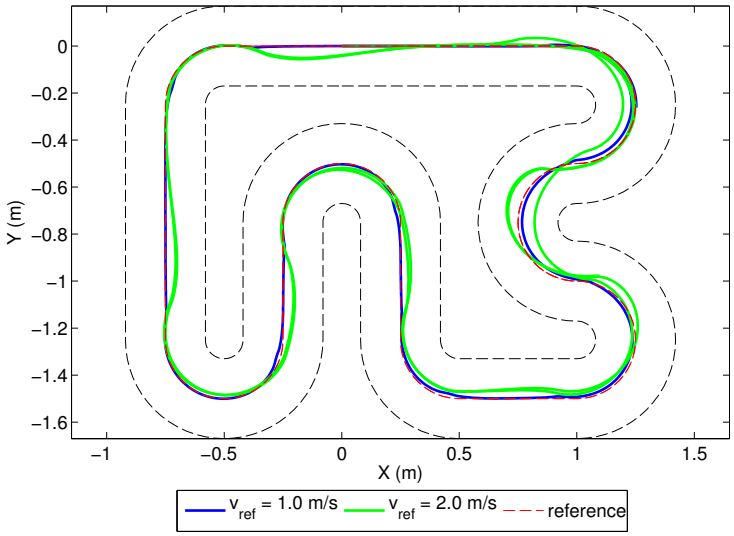


Figure 5.2: Comparison of linear MPC tracking the centerline at different reference velocities.

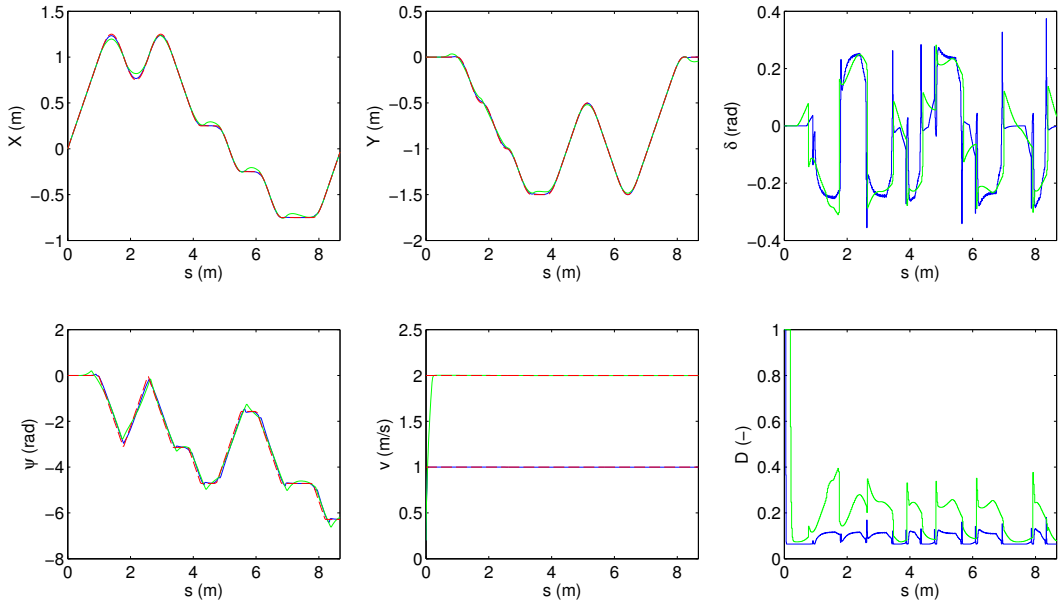


Figure 5.3: Comparison of the vehicle states and controls of linear MPC tracking of the centerline at different reference velocities. The horizontal axis describes the path advancement s.

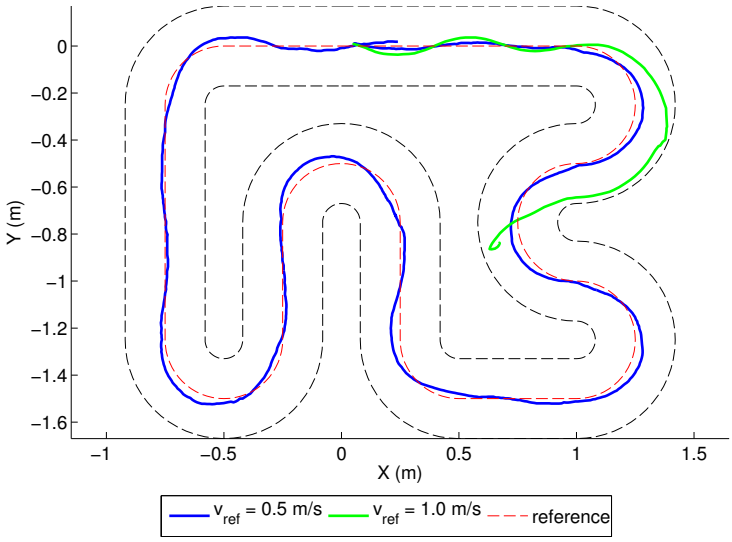


Figure 5.4: Linear MPC tracking at two different reference velocities. The centerline is the reference trajectory.

methods will be given in Chapter 6, where the benefits of nonlinearities will be demonstrated. At a higher reference velocity of 1.0 m/s, the car is not able to turn quickly enough to avoid violating the path constraints. This is due to the model nonlinearities that are not captured in a linear model. Also, there is a mismatch between the simplified slip-less bicycle model and the real miniature race car.

The resulting states can be seen in Fig. 5.5. The velocity is tracked closely; the large peak is a measurement error. The position where this occurs is exactly perpendicularly under the camera. It is often the case that there is a wrong measurement at this particular position, due to reflection of light above the camera on the shiny body of the car. Note also that steering angle in the experiment resembles that of the simulation.

5.3 Trajectory tracking

Tracking of the centerline is never time-optimal. Instead of feeding the centerline to the MPC tracking algorithm, we want to calculate a reference trajectory that differs from the centerline in order to decrease the lap time of the car. This calculation should be carried out on-line, resulting in a two-level MPC trajectory tracking algorithm: at each time step, a trajectory is calculated, which is then tracked inside an MPC tracking algorithm as explained in Eq. 5.6d. The planning algorithm makes use of a longer prediction horizon than the tracking algorithm. The two-level approach is depicted schematically in Fig. 5.6.

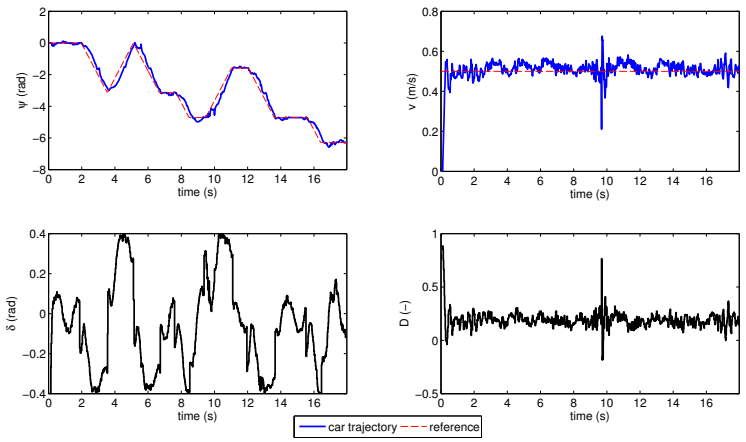


Figure 5.5: States and controls of linear MPC tracking. Also see Fig. 5.4.

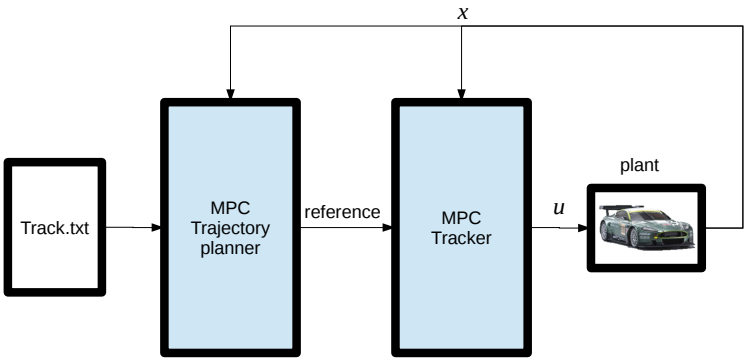


Figure 5.6: The two-level MPC trajectory planning algorithm.

5.3.1 Calculation of the trajectory

For the reference trajectory two choices are possible:

1. The path of least curvature: the car is able drive as 'smooth' as possible. In this way, the car does not need to slow down much in the corners.

2. The shortest path: the total length of the traveled path is shorter, so we can expect the total time needed to be smaller as well.

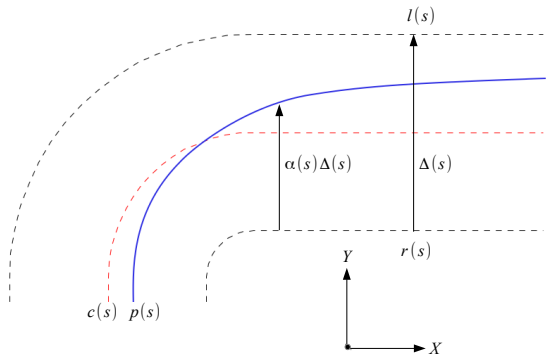


Figure 5.7: Model of the track and the vehicle trajectory.

Both paths described above are the result of an optimization problem. The elements needed in the optimization problem formulation are

$$c(s) = [x_c(s), y_c(s)]^T \in \mathbb{R}^2, \qquad \text{Track centerline,} \tag{5.7a}$$
$$r(s) \in \mathbb{R}^2, \qquad \text{Right hand border,} \tag{5.7b}$$
$$l(s) \in \mathbb{R}^2, \qquad \text{Left hand border,} \tag{5.7c}$$
$$\Delta(s) = l(s) - r(s) \qquad \text{Lateral deviation,} \tag{5.7d}$$
$$w(s) \in \mathbb{R}, \qquad \text{Track width,} \tag{5.7e}$$
$$p(s) = r(s) + \alpha(s)\Delta(s)\ ,\ \alpha(s) \in [0,1] \qquad \text{Vehicle trajectory.} \tag{5.7f}$$

These quantities are plotted in Fig. 5.7. In the following paragraphs, the computation of the reference trajectories is described.

Least curvature trajectory The local curvature of a smooth trajectory is defined as follows (see [10] for the details):

$$\kappa(s) \equiv \frac{||p''(s) \times p'(s)||}{||p'(s)||^3}. \tag{5.8}$$

Accordingly, the problem of minimizing the aggregate trajectory curvature can be formulated as:

$$\underset{\alpha(\cdot)}{\text{minimize}} \quad \mathcal{C} = \int_0^L \kappa(s)\,\mathrm{d}s \tag{5.9a}$$
$$\text{subject to} \quad 0 \le \alpha(s) \le 1 \quad , \quad \forall s \in [0, L] \tag{5.9b}$$
$$p(s) = r(s) + \alpha(s)\Delta(s). \tag{5.9c}$$

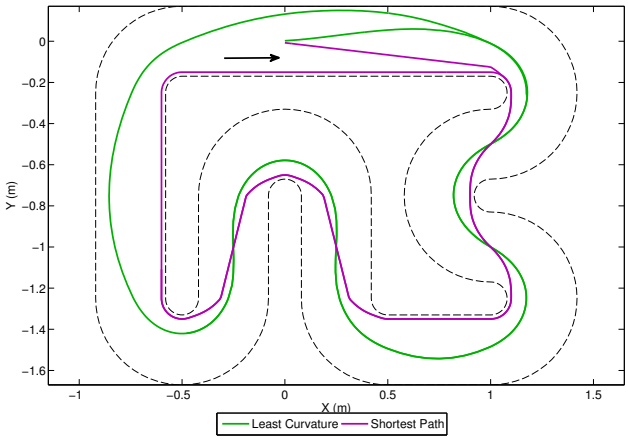


Figure 5.8: Two trajectories planned on-line by the trajectory planner.

Shortest path trajectory Using the notation presented in (5.7), the problem of finding the minimum length trajectory can be formulated in continuous space as follows:

$$\underset{\alpha(\cdot)}{\text{minimize}} \quad \int_0^L \sqrt{x_p'(s)^2 + y_p'(s)^2}\, \mathrm{d}s \tag{5.10a}$$

$$\text{subject to} \quad 0 \leq \alpha(s) \leq 1 \quad , \quad \forall s \in [0, L] \tag{5.10b}$$

$$p(s) = r(s) + \alpha(s)\Delta(s), \tag{5.10c}$$

where $p(s) = [x_p(s), y_p(s)]^T$, $x_p'(s) \equiv \frac{\partial x_p(s)}{\partial s}$ and similarly $y_p'(s) \equiv \frac{\partial y_p(s)}{\partial s}$ and L is the length of the track along the centerline.

The continuous problem formulation can be discretized, resulting in a QP problem. The details of this method can be found in [10].

5.4 Two-level algorithm: Results

The two-level method explained above has been implemented in MATLAB/Simulink. All the simulations were carried out in this environment. The relevant MPC solving code is exported to `C++`-code with the Simulink Coder tool, and inserted in the existing software running on the race car setup.

5.4.1 Simulation Results

The reference trajectories computed in problems (5.9) and (5.10) are displayed in Fig. 5.8. Note that the solution of the shortest path problem is generally close to the borders and touches it at some points. The least curvature solution has a much rounder profile, and touches the outside boundary at the top of the track.

The MPC tracking performance will be compared to the linear MPC tracking algorithm.

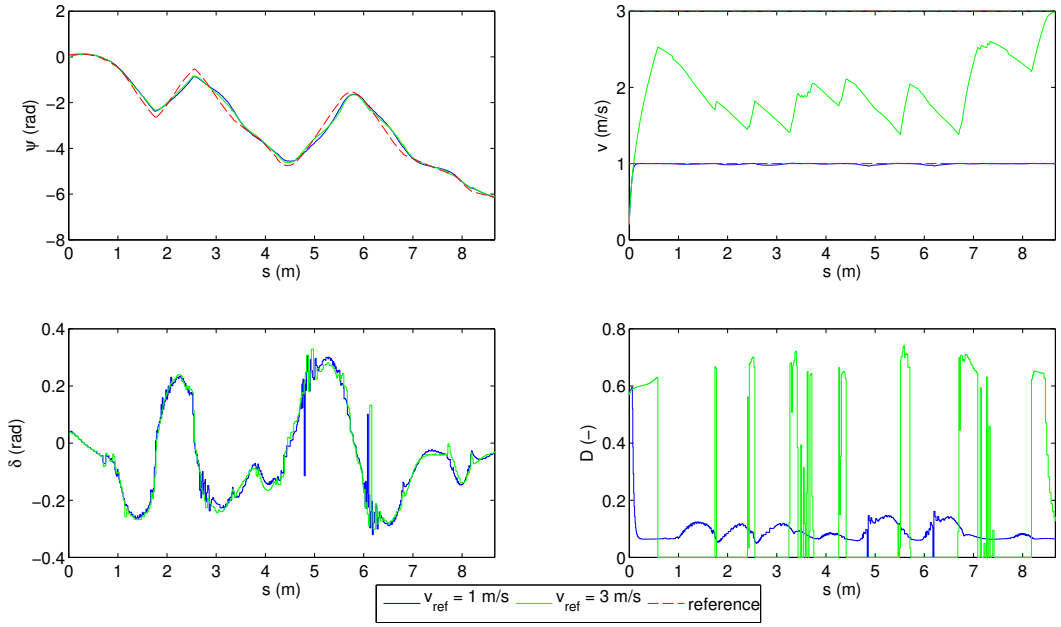


Figure 5.9: The vehicle states and control inputs of the car tracking the least curvature path.

Tracking of least curvature path

As we can see in Fig. 5.9, the velocity can be substantially higher than for centerline tracking. At the end of the trajectory, the velocity can be as high as 3 m/s.

The resulting trajectory is plotted in Fig. 5.10. From this plot it is apparent that the car gains the most speed in the longer straight section at the top of the track, whereas it needs to slow down a lot in the chicane (on the right).

Tracking the shortest path

We show the results of the shortest path tracking in Fig. 5.12. As can be seen, the car needs to turn more swiftly: the steering angle δ varies much faster than when tracking the least curvature path. This has an impact on the velocity of the race car, the velocity can not be tracked closely at all times. Setting the reference velocity even higher results in infeasibilities in the optimization routine.

Lap times

In Tab. 5.1, we can see that the trajectory planning methods outperform tracking of the centerline. In conclusion, we can say that our heuristic methods resulted in faster race performance.

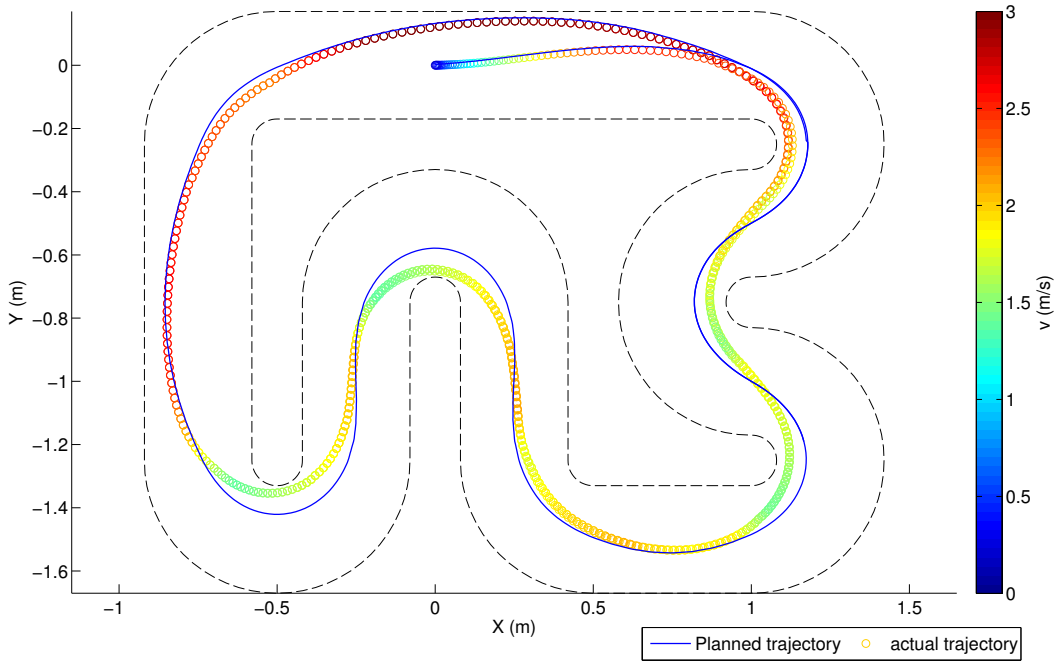


Figure 5.10: The resulting least curvature trajectory at $v_{\text{ref}} = 3.0$ m/s. The velocity of the car is encoded with colors.

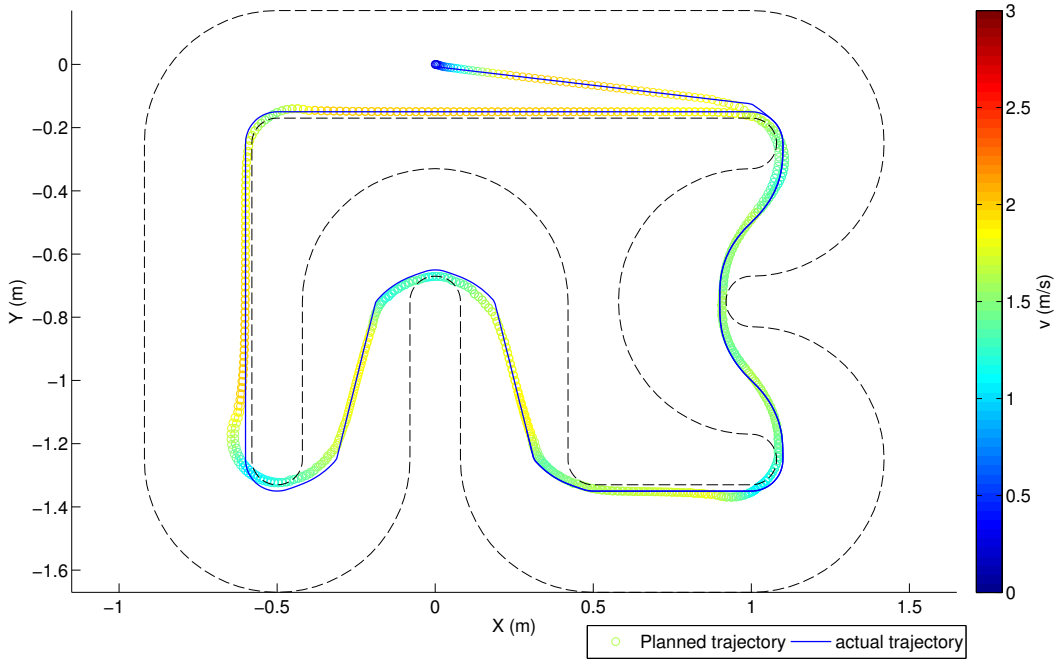


Figure 5.11: The resulting trajectory by tracking the shortest path around the track. The reference velocity is $v_{\text{ref}} = 2.0$ m/s.

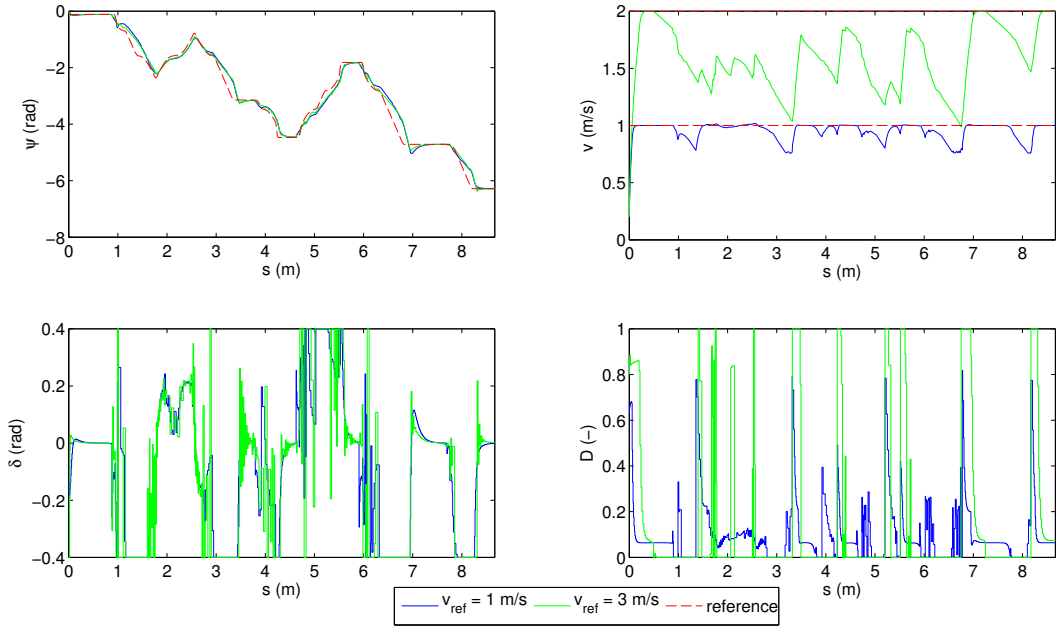


Figure 5.12: The vehicle states ψ and v and control inputs of the car tracking the shortest path around the track. See Fig. 5.11 for the trajectory.

Table 5.1: Comparing lap times between the different linear methods in simulation.
LC=least curvature, SP=shortest path

Algorithm	Lap time
Linear MPC tracking ($v_{\text{ref}} = 2.0$ m/s)	4.3 s
Two-level MPC (LC, $v_{\text{ref}} = 3.0$ m/s)	3.9 s
Two-level MPC (SP, $v_{\text{ref}} = 2.0$ m/s)	4.0 s

5.4.2 Experimental results

The experimental results for the two heuristic methods will be presented in this section.

Tracking of least curvature path

In Fig. 5.13, the resulting trajectory and reference trajectory for the path with the least curvature is shown with a reference velocity of 2.2 m/s. However, as can be seen from the remaining states and controls in Fig. 5.14, this velocity is never attained.

Tracking of the shortest path

The problem of tracking the shortest path around the track has to be altered slightly in proceeding from simulation to experiment: the width of the car has to be taken in

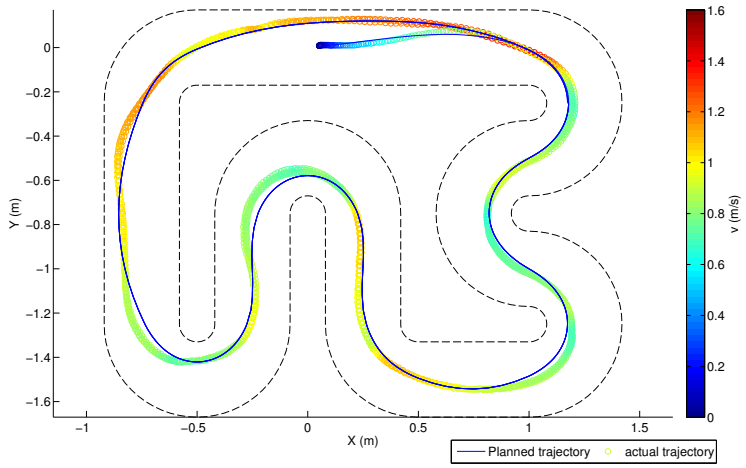


Figure 5.13: Experimental results of the MPC planning and tracking algorithm. The least curvature path is shown, together with the actual car trajectory, encoded with the car velocity in color.

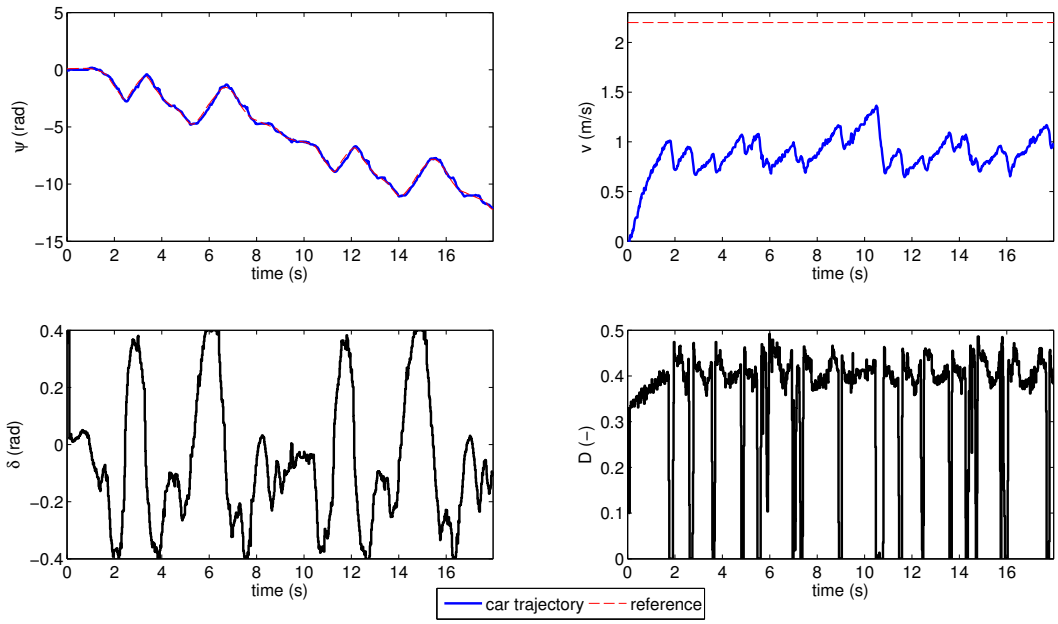


Figure 5.14: Experimental results of the MPC planning and tracking algorithm with the least curvature path as reference. The states ψ and v and the control inputs are shown. See Fig. 5.13 for the resulting trajectory.

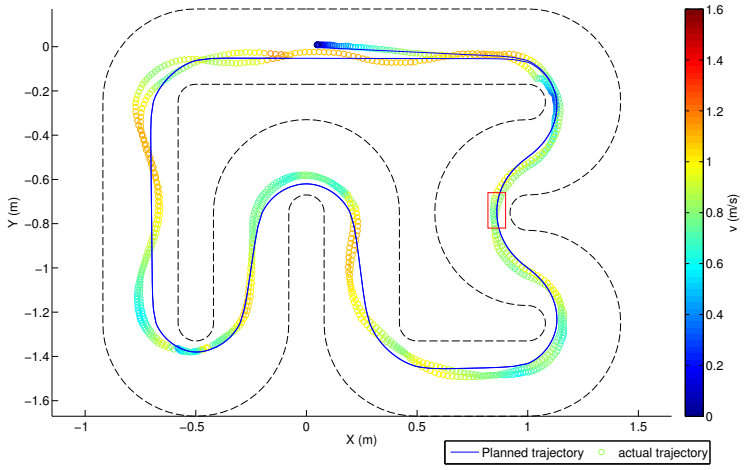


Figure 5.15: Experimental results of the MPC planning and tracking algorithm. The shortest path around the track is shown, together with the actual car trajectory, encoded with the car velocity in color. Note that a margin had to be introduced due to the width of the car (shown as a red rectangle).

Table 5.2: Comparing lap times between the different linear methods in simulation.
LC=least curvature, SP=shortest path

Algorithm	first lap time	average lap time
Linear MPC tracking ($v_{\text{ref}} = 0.5$ m/s)	17.6 s	17.5 s
Two-level MPC (LC, $v_{\text{ref}} = 2.2$ m/s)	8.8 s	8.0 s
Two-level MPC (SP, $v_{\text{ref}} = 2.0$ m/s)	9.1 s	8.95 s

account, so a small margin of 4.0 cm is added to the track. In Fig. 5.15 and Fig. 5.16, the experimental results of the planning-tracking MPC algorithm can be seen. From Fig. 5.15, we can see that the velocity is somewhat lower than in tracking the least curvature path, (maximum velocity 1.2 m/s vs. 1.6 m/s). Moreover, the car needs to turn sharper, resulting in a steering angle bound that is active longer than in the least curvature tracking (compare Fig. 5.16 to Fig. 5.14).

Lap times

We make the same comparison of the lap times as in simulation, in Tab. 5.2. In contrast to the simulations, the shortest path planning algorithm results in the fastest lap times, both in the first lap and on average. Note also the much larger difference between centerline tracking and the more advanced methods. This is due to the low reference velocity we were bound to set in the experiment as higher reference velocities resulted in violations of the road boundaries.

We can conclude this chapter by observing that the heuristic linear MPC methods are indeed faster than pure linear MPC tracking of the centerline. This holds both

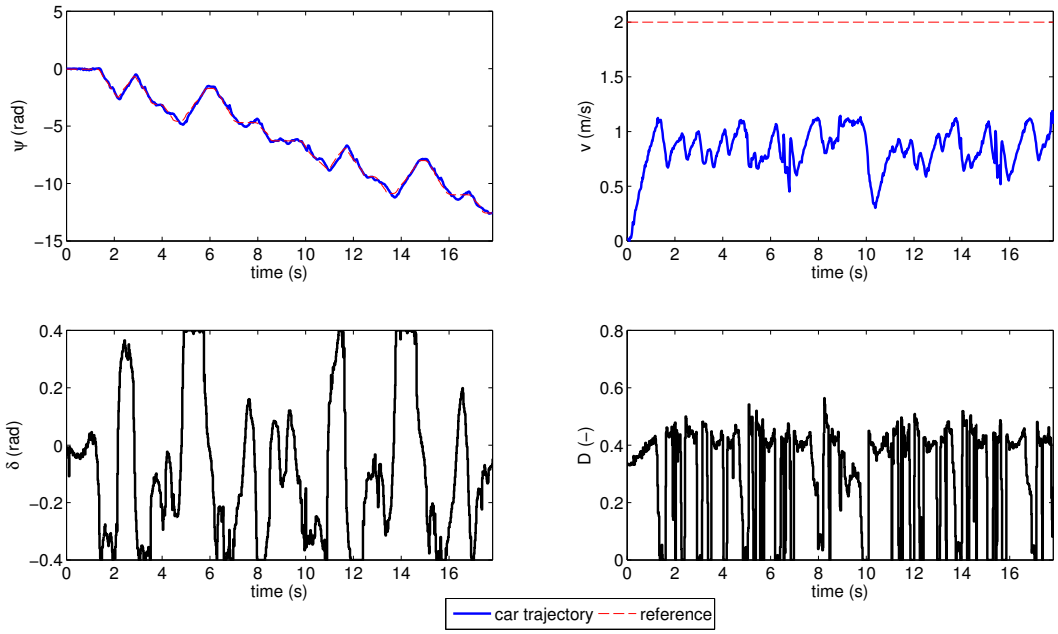


Figure 5.16: xperimental results of the MPC planning and tracking algorithm with the shortest path as reference. The states ψ and v and the control inputs are shown. See Fig. 5.16 for the resulting trajectory.

in simulation and experiment.

Chapter 6

Nonlinear MPC tracking

All methods presented in the previous chapter are *linear* methods. However, linear MPC has the disadvantage that we have to choose a point of linearization, usually the steady state of a system, in our case, the centerline of the track. Around this linearization point, the linearized model accurately describes the nonlinear model. However, the farther we move away from the linearization point, the less the linear model is an accurate description of the underlying nonlinear model.

Therefore, nonlinear methods are investigated in this chapter. These nonlinear methods allow for more accurate solutions, but pose a bigger computational challenge than linear methods. To assess the feasibility of a NMPC solver, we design a simple tracking model-predictive controller and analyse its behavior both in simulation and on the real-world setup.

6.1 NMPC Tracking

As a control objective, the distance of the state and control input vectors to the state and input reference is penalized. The most common choice is a least-squares penalty, because efficient methods for such a cost exist (see section 4.5.3). The nonlinear tracking problem then becomes

$$\min_{\xi(\cdot),u(\cdot)} \quad \int_{t_0}^{t_f} \|\xi(\tau) - \xi_{\text{ref}}(\tau)\|_Q^2 + \|u(\tau) - u_{\text{ref}}(\tau)\|_R^2 \, d\tau \tag{6.1a}$$

$$+ \left\|\xi(t_f) - \xi_{\text{ref}}(t_f)\right\|_P^2 \tag{6.1b}$$

$$\text{s.t.} \quad \dot{\xi}(t) = f(\xi(t), u(t)) \tag{6.1c}$$

$$u(t) \in [\delta_L, \delta_U] \times [-1, 1] \tag{6.1d}$$

$$\xi(0) = \xi_0. \tag{6.1e}$$

Recall that our model is (see section 3.2.3)

$$\begin{aligned}
\dot{X} &= v\cos(\psi + C_1\delta) \\
\dot{Y} &= v\sin(\psi + C_1\delta) \\
\dot{\psi} &= v\,\delta\,C_2 \\
\dot{v} &= C_{m_1}D - C_{m_2}\,D\,v - C_{r2}\,v^2 - C_{r_0} - (v\,\delta)^2\,C_2\,C_1,
\end{aligned}$$

and consists of four states and two inputs

$$\xi = \begin{bmatrix} X \\ Y \\ \psi \\ v \end{bmatrix}, u = \begin{bmatrix} \delta \\ D \end{bmatrix}. \tag{6.2}$$

The steering angle range $[\delta_L, \delta_U]$ is

$$[\delta_L, \delta_U] = [-\frac{25\pi}{180}, \frac{25\pi}{180}],$$

as before.

The symmetric weighing matrices Q, R and P allow for a distinction in tracking importance. As we are trying to track the centerline at a constant speed, the largest elements will be $Q_{1,1}, Q_{2,2}$ and $Q_{4,4}$.

A time-dependent reference for all states and controls is inserted in the optimization routine. This reference is calculated offline, and is read from file.

The $X_{\text{ref}}, Y_{\text{ref}}$ and ψ_{ref} references are fixed by the track geometry. The reference velocity v_{ref} was chosen at 1 m/s, which is a reasonable velocity for the miniature race cars.

It is not directly clear which values to choose for the control references $\delta_{\text{ref}}, D_{\text{ref}}$. The most simple choice would be $\delta_{\text{ref}} = 0, D_{\text{ref}} = 0$. For the steering angle δ, the value 0rad is reasonable because the steering angle range is symmetric around this value. However, for the duty cycle D, a reference value of 0 is equivalent to the car being at rest. Inserting a zero reference at all times, the control actions enter the cost function as $\|\delta\|$ and $\|D\|$ and large values for these quantities are thus penalized. Adding such a control regularization is common because large control actions are often considered harmful as they might damage the system.

Another possibility is to simulate the behavior of the car offline and insert the resulting δ and D of this simulation as a reference into the on-line control problem.

6.2 NMPC Tracking with input rates

In the trajectory tracking form of NMPC, it is not entirely clear which values for the reference of the control input to get the best tracking behavior. It is better to penalize the *change* in input, as the control inputs become smoother. Therefore, we have to include the derivatives of the control inputs into our problem formulation.

This model reformulates the original controls δ and D as states, resulting in the following optimization variables:

$$\xi = \begin{bmatrix} X \\ Y \\ \psi \\ v \\ \delta \\ D \end{bmatrix}, u = \begin{bmatrix} \Delta\delta \\ \Delta D \end{bmatrix}. \tag{6.3}$$

We restate the model as

$$\dot{X} = v\cos(\psi + C_1\delta) \tag{6.4a}$$
$$\dot{Y} = v\sin(\psi + C_1\delta) \tag{6.4b}$$
$$\dot{\psi} = v\,\delta\,C_2 \tag{6.4c}$$
$$\dot{v} = C_{m_1}D - C_{m_2}\,D\,v - C_{r2}\,v^2 - C_{r_0} - (v\,\delta)^2\,C_2\,C_1 \tag{6.4d}$$
$$\dot{\delta} = \Delta\delta \tag{6.4e}$$
$$\dot{D} = \Delta D. \tag{6.4f}$$

Introducing the input rates in the objective function results in the following NMPC problem formulation:

$$\begin{aligned} \min_{\xi(\cdot),u(\cdot)} \quad & \int_{t_0}^{t_f} \|\xi(\tau) - \xi_{\text{ref}}(\tau)\|_Q^2 + \left\|[\Delta\delta(\tau), \Delta D(\tau)]^T\right\|_R^2 \, d\tau \\ & + \left\|\xi(t_f) - \xi_{\text{ref}}(t_f)\right\|_P^2 \\ \text{s.t.} \quad & \dot{\xi}(t) = f(\xi(t), u(t)) \\ & [\delta, D]^T \in [\delta_L, \delta_U] \times [-1, 1] \\ & \xi(0) = \xi_0. \end{aligned} \tag{6.5}$$

6.3 Results

The methods described above have been implemented in the ACADO interface for MATLAB. To run the code on the real-time computer at the experimental setup, ACADO allows for the automatic generation of source code to `C++`, which is called from software routines based on earlier code as used in Chapter 5.

First, we will make a comparison between linear MPC tracking and nonlinear MPC tracking. Next, we will consider the performance of the formulation with input rates. The following results are obtained with a sampling time of $T_s = 0.02$ s, and a prediction horizon $N = 20$ (see eq. (4.11d)).

6.3.1 Simulation results

The need for control regularization, is depicted in Fig. 6.1, where we used the weighing matrix $Q = \text{diag}([1, 1, 10^{-5}, 1])$ and $R = \text{diag}([0, 0])$. As can be seen, the

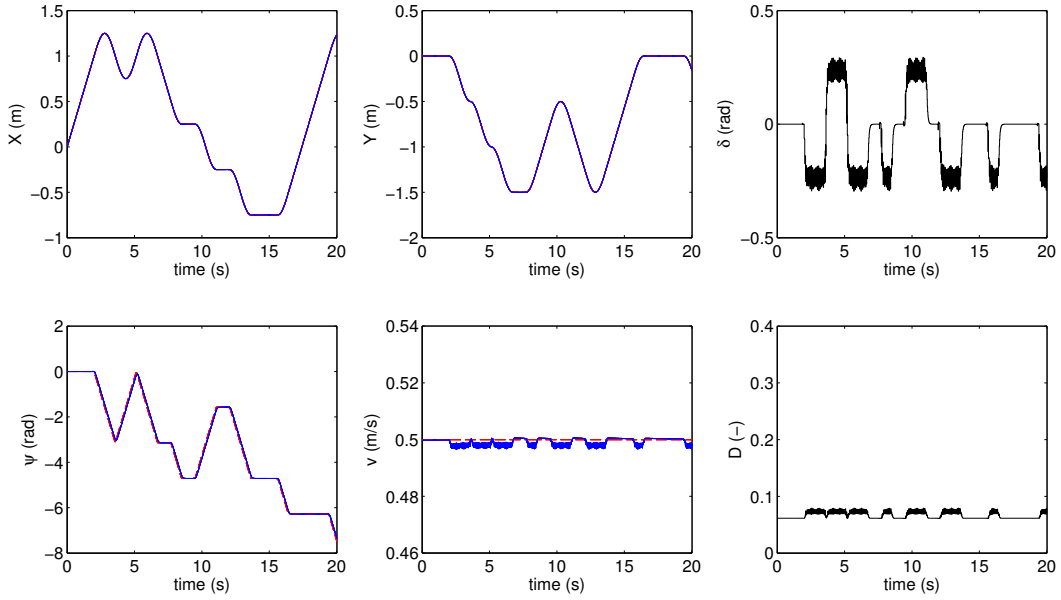


Figure 6.1: Tracking performance of a formulation without control regularization.

NMPC tracking performance is satisfactory, as the trajectory almost coincides with the reference. However, the control inputs oscillate at high frequency. This is often an undesirable feature.

Therefore, we introduce some weights on the control inputs as well. A set of well-performing weights is

$$Q = \mathrm{diag}([1, 1, 10^{-5}, 1]), \;\; R = \mathrm{diag}([10^{-4}, 10^{-4}]). \tag{6.6}$$

Inserting the control *rates* into the control problem results in very similar behavior as in the results without input rates. The results can be seen in Fig. 6.2, where we used the weighing matrices

$$Q = \mathrm{diag}([1, 1, 10^{-5}, 1, 10^{-10}, 10^{-10}]), \;\; R = \mathrm{diag}([10^{-4}, 10^{-6}]). \tag{6.7}$$

Indeed, the control inputs δ, D appear to be smoother when we weight their derivatives, respectively (see Fig. 6.3).

Although weighing the control input rates lowers the variability of the corresponding controls, the tracking performance is very similar, except for the velocity v.

6.3.2 Experimental results

The simulation results are helpful in tuning the weights and other parameters in the MPC problem. Nevertheless, the main emphasis of this thesis remains the real-world setup. Next, the experimental results will be analyzed.

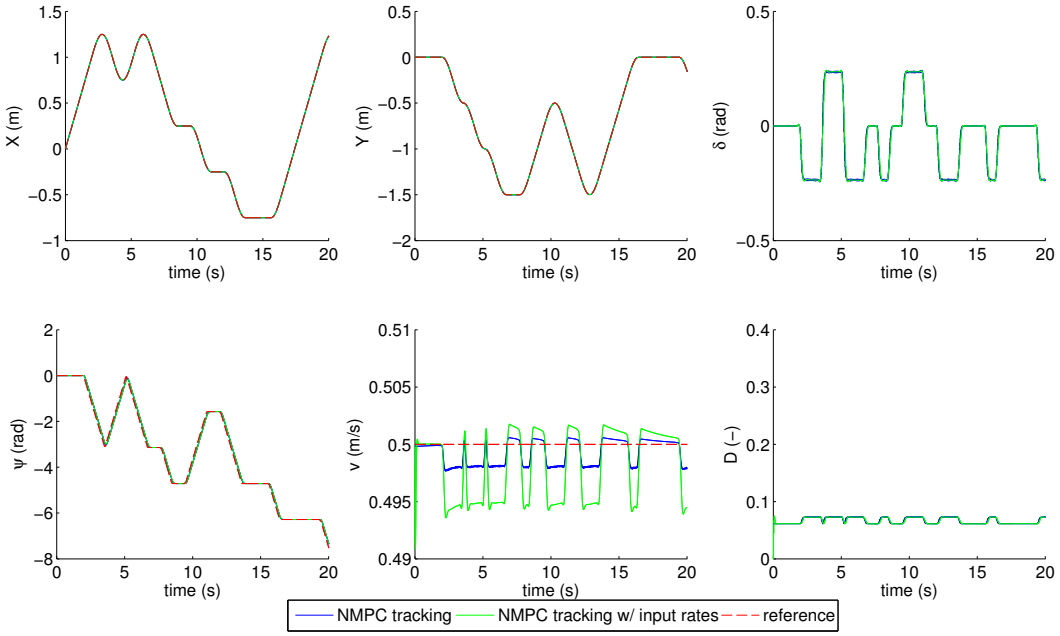


Figure 6.2: Comparison of the tracking performance with or without the input rates taken into consideration.

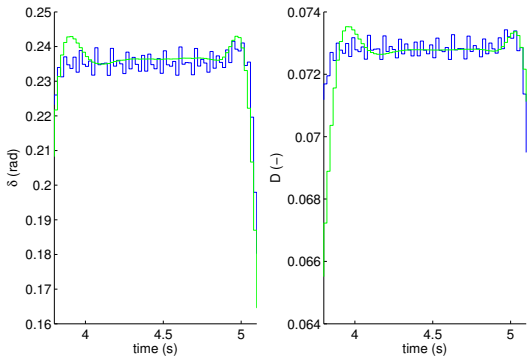


Figure 6.3: Close-up of the control inputs of Fig. 6.2.

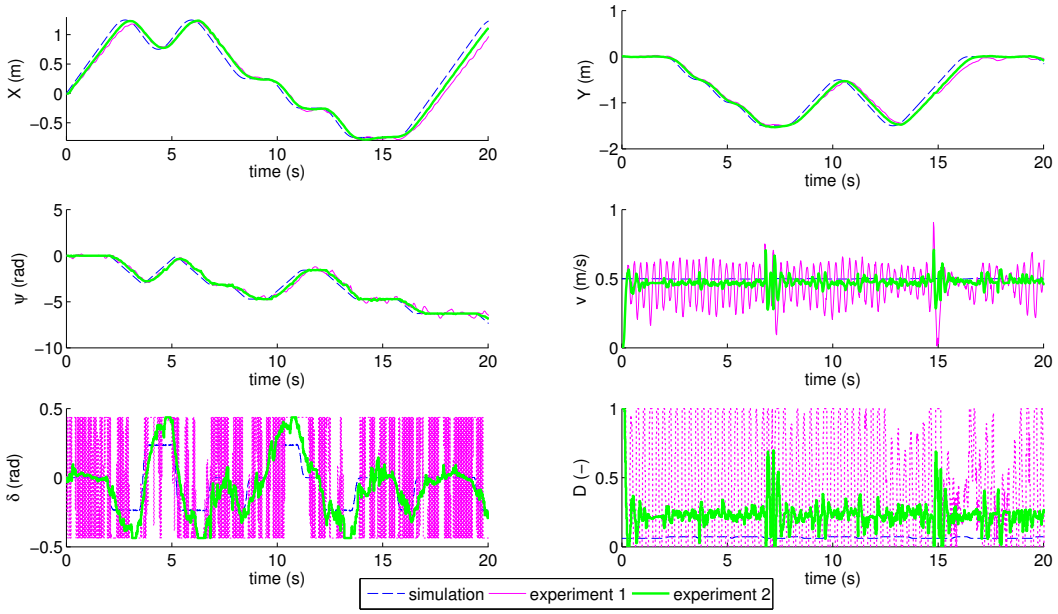


Figure 6.4: Comparison between simulation of the NMPC tracking problem and experimental results. In 'experiment 1', we used the same weights as in simulation, with resulting unsatisfactory performance. In 'experiment 2', the weights are adapted in order to obtain more satisfactory results.

NMPC tracking First, NMPC has been run on the experimental setup with exactly the same NMPC tracking problem formulation and parameters as in simulation and compare the experimental results with the simulation. The experimental results are given in Fig. 6.4 together with the simulation results for comparison. As can be seen from this plot, the control inputs are not weighted enough, resulting in an unstable behavior. Therefore, a different controller with weights

$$Q = \mathrm{diag}([1, 1, 10^{-5}, 1]),\ R = \mathrm{diag}([10^{-2}, 10^{-1}]) \tag{6.8}$$

is proposed (also shown in Fig. 6.4).

The above findings are even clearer from looking at the resulting trajectories (Fig. 6.5): the nervous behavior of the experimental controller with the simulation weights is deregatory for the tracking performance. The trajectory of the controller with the adapted weights is delivering a trajectory much closer to the centerline.

NMPC tracking with input rates We repeat the same exercise for the NMPC tracking formulation with inclusion of the control input rates in the objective function. In this case, the algorithm with the weights on the input rates used in simulation were not able to steer the car around the track: the car crashes into the border of the track before the first corner. Also here, the input rates have to be weighted more in order to obtain satisfactory results. The new weights are

$$Q = \mathrm{diag}([1, 1, 10^{-5}, 1, 10^{-10}, 10^{-10}]),\ R = \mathrm{diag}([10^{-3}, 10^{-4}]). \tag{6.9}$$

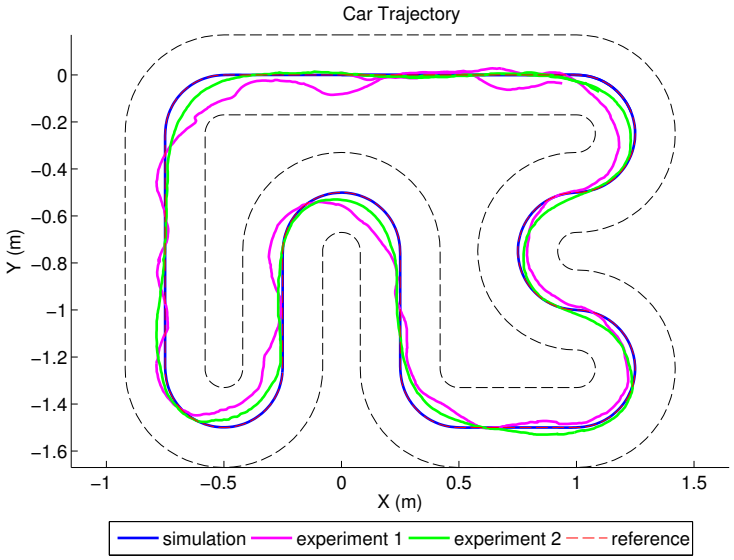


Figure 6.5: Comparison of the trajectories between simulation of the NMPC tracking problem and experimental results. Experiment 2 is the one with the adapted weights.

As can be seen from Fig. 6.7, the car trajectory is even closer to the reference than in the NMPC tracking without input rates. A reason for this is that weighing the 2-norm of the steering angle δ penalizes large values relatively more than small values, as it is a quadratic penalty. However, in the input rate formulation, norm of the steering angle is not penalized, only its rate of change.

At the bottom left of Fig. 6.7, the trajectory moves a bit to the outside. The cause of this is not entirely clear. It may well be due to a detection error in the vision system, hence the oscillation in the steering angle around $t = 14$ s (Fig. 6.6).

As a conclusion, we can state that the NMPC tracking with input rates performs best on the experimental setup. In going from simulation to experiment, the controller had to be modified only slightly (mostly increasing the input weights). In simulation, there is not much of a difference between both methods.

If we compare the nonlinear MPC tracking (without weights on the derivative of the inputs) presented in this chapter to the linear MPC tracking of the centerline (see section 5.2), we can state that the nonlinear methods performs in a much better way in experiment (compare Fig. 6.5 to Fig. 5.2). As this nonlinear method is feasible in real-time, we can now extend this method and proceed to time-optimal control.

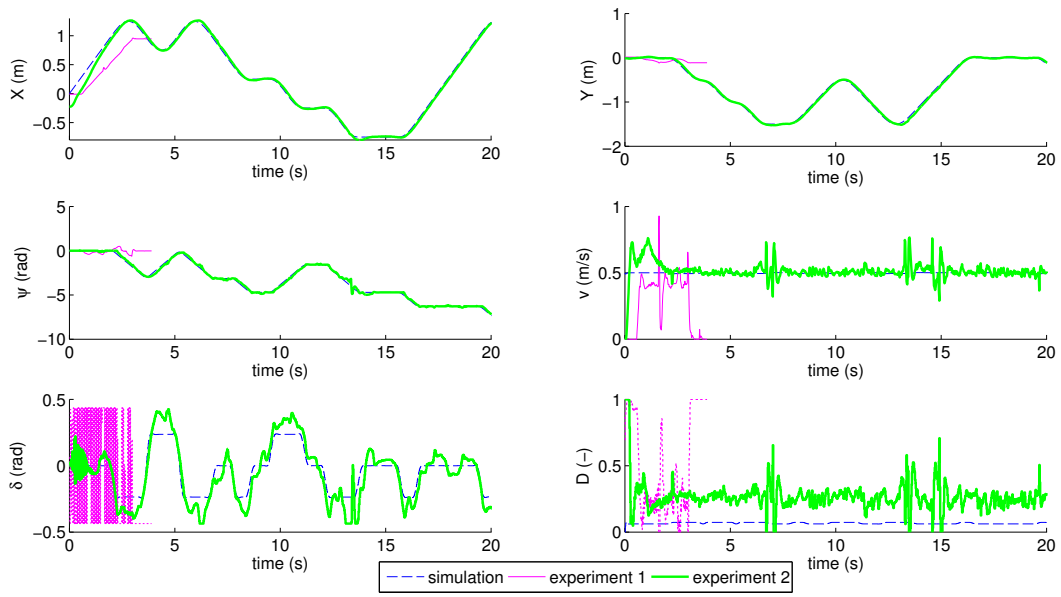


Figure 6.6: Comparison between simulation of the NMPC tracking problem with input rates and corresponding experimental results. In 'experiment 1', we used the same weights as in simulation, with resulting unsatisfactory performance. In 'experiment 2', the weights are adapted in order to obtain more satisfactory results.

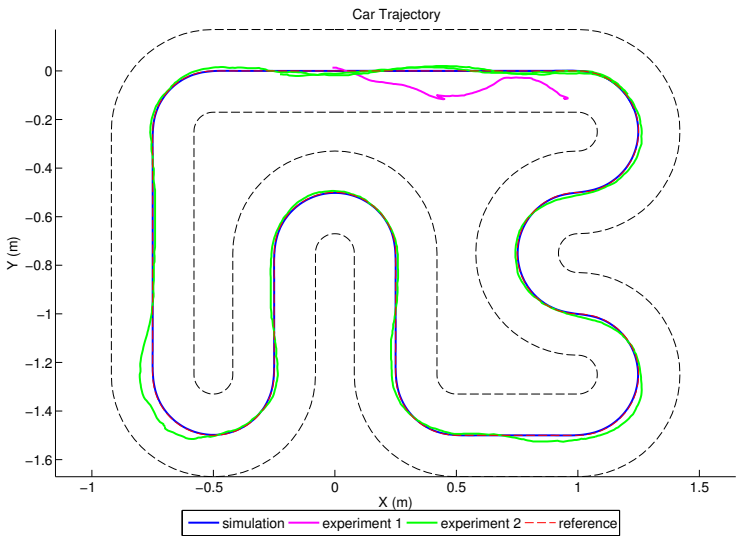


Figure 6.7: Comparison of the trajectories between simulation of the NMPC tracking problem with input rates and experimental results. Experiment 2 is the one with the adapted weights.

Chapter 7

Time-optimal MPC

The main goal of this thesis is to design a time-optimal control algorithm, validated on an experimental setup. Time-optimal MPC is a more difficult problem than trajectory tracking, because minimum time objectives do not naturally fit into the class of least-squares objectives for which highly efficient algorithms exist, such as ACADO, as used in [18, 42]. In this chapter, this algorithm will be introduced and described in detail.

7.1 Spatial reformulation

Model (3.13) is a dynamic system with *time* being the independent variable. A reparameterization is therefore required to render time an optimization variable. This allows us to introduce time in the objective function of the optimization algorithm. We propose to employ the transformation to spatial coordinates from [18] for this purpose. Track limitations become thus simple (convex) state bounds, which are independent of the vehicle speed. Furthermore, if the track limitations were constraining the track to a single curve, the time-optimal driving problem would even be a convex optimization problem, cf. [37].

Using a spatial reformulation of the dynamics, the horizon can be specified as a function of space instead of time. This mimics the behavior of real-world driving: drivers usually look ahead a certain distance instead of looking some time interval into the future.

We state the spatial transformation for completeness. Vehicle coordinates in the global frame are denoted by $[X, Y]^T$. We project these $X - Y$ coordinates on a curve σ, which is the reference trajectory, parameterized by its arc length $\sigma(s)$. Instead of states X, Y and ψ we obtain

$$\begin{aligned} e^y &= \cos(\psi^\sigma)(Y - Y^\sigma) - \sin(\psi^\sigma)(X - X^\sigma), \text{ and} \\ e^\psi &= \psi - \psi^\sigma \end{aligned}$$

in the spatial dynamic system, where $[X^\sigma, Y^\sigma]^T$ and ψ^σ are the position and orientation of the reference point on the path given by s, see Fig. 7.1.

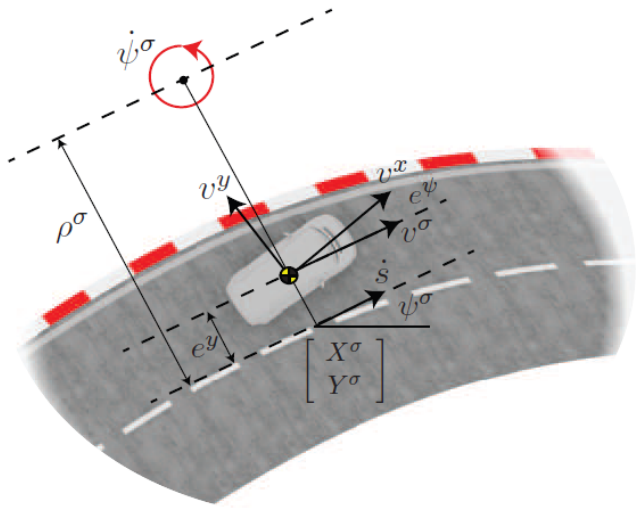


Figure 7.1: Definition of the coordinate system used in the spatial reformulation of the vehicle dynamics. The s coordinate denotes the arc-length along the track. Source: [18].

If we assume that the car is not at rest at any time instant ($\dot{s} > 0$), then it holds for the state vector $\xi = [\ e^y\ e^\psi\ v\ t\ ]$ that

$$\xi' = \frac{d\xi}{ds} = \frac{d\xi}{dt}\frac{dt}{ds} = \frac{1}{\dot{s}}\dot{\xi}, \tag{7.1}$$

where $\dot{\xi}$ is defined in (3.13).

In order to compute the vehicle speed with respect to the reference, $\dot{s}$, note that from Fig. 7.1 we have

$$v^\sigma = (\rho^\sigma - e^y)\,\dot{\psi}^\sigma, \text{ and}$$
$$v^\sigma = v^x \cos e^\psi - v^y \sin e^\psi,$$

where ρ^σ is the radius of local curvature of σ. The velocity along the path, $\dot{s}$, is then given by

$$\dot{s} = \rho^\sigma\,\dot{\psi}^s = \frac{\rho^\sigma}{\rho^\sigma - e^y}\,(v^x \cos(e^\psi) - v^y \sin(e^\psi)).$$

We obtain the spatial dynamic system as:

$$e^{y\prime}(s) = (v^x \sin(e^\psi) + v^y \cos(e^\psi))/\dot{s} \tag{7.2a}$$
$$e^{\psi\prime}(s) = \dot{\psi}/\dot{s} - \kappa^\sigma(s) \tag{7.2b}$$
$$v'(s) = \dot{v}/\dot{s} \tag{7.2c}$$
$$t'(s) = 1/\dot{s}, \tag{7.2d}$$

Table 7.1: States and control inputs of the spatial vehicle model

State	Unit	Description
e^y	m	Deviation from centerline
e^ψ	rad	Yaw angle relative to path
v	m/s	Absolute velocity
t	s	Time

Control	Range	Unit	Description
δ	[-0.44, 0.44]	rad	Steering Angle
D	[-1,1]	-	Dutycycle of DC motor

where $\kappa^\sigma(s) = 1/\rho^\sigma(s)$ is the local curvature of the track. Note that the states in the global coordinate system can always be recovered by

$$\begin{aligned} X &= X^\sigma - e^y \sin(\psi^\sigma) \\ Y &= Y^\sigma - e^y \cos(\psi^\sigma) \\ \psi &= \psi^\sigma + e^\sigma. \end{aligned}$$

A summarizing list of the states and inputs of the spatial model can be found in Table 7.1.

7.2 Offline solution

In order to have a reference to compare the online computed solutions with, we would like to have an offline computed time-optimal trajectory. To this end, we use the `fmincon` optimization routine in the MATLAB Optimization Toolbox.

The time-optimal trajectory is the curve that is driven by the car in the least amount of time on one lap of the race track. The optimal trajectory is continuous and periodic. The driven trajectory will converge to a trajectory that stays the same each lap: the trajectory converges to a stable limit cycle. The solution is computed in the following way.

The trajectory on one lap with length L is discretized in N intervals, resulting in $N + 1$ nodes. On each node, we define a state vector ξ, and on each interval, we define a control action u, which is constant throughout the interval. Each state and control input looks as follows:

$$\xi = \begin{bmatrix} e^y \\ e^\psi \\ v \\ t \end{bmatrix}, u = \begin{bmatrix} \delta \\ D \end{bmatrix} \tag{7.3}$$

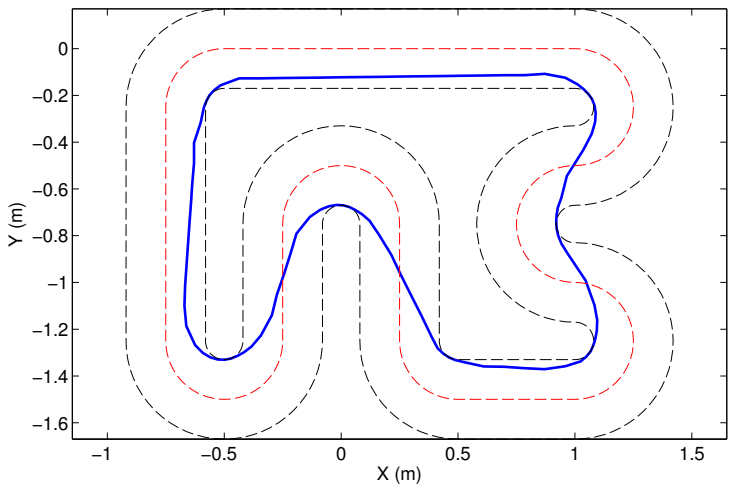


Figure 7.2: Periodic solution that minimizes the total lap time. See Eq. (7.5).

We introduce additional periodicity constraints

$$\begin{bmatrix} e_0^y \\ e_0^\psi \\ v_0 \end{bmatrix} = \begin{bmatrix} e_N^y \\ e_N^\psi \\ v_N \end{bmatrix}. \tag{7.4}$$

Because time always moves forward, it is not included in these periodic constraints.

The time-optimal trajectory is the solution to the following optimization problem:

$$\begin{aligned} \min_{\xi,u} \quad & t_N \\ \text{s.t.} \quad & \xi'(s) = f(s,\xi(s),u(s)), && s \in [0,L] \\ & e^y(s) \in [e_L^y(s), e_U^y(s)], && s \in [0,L] \\ & u(s) \in [\delta_L, \delta_U] \times [-1,1], && s \in [0,L] \\ & \begin{bmatrix} e_0^y \\ e_0^\psi \\ v_0 \end{bmatrix} = \begin{bmatrix} e_N^y \\ e_N^\psi \\ v_N \end{bmatrix}. \end{aligned} \tag{7.5}$$

The solution is shown in Fig. 7.2. Note that this minimum-time solution resembles the shortest path solution of Chapter 5. The algorithm wins a lot of time by cutting corners, eg. in the chicane. Here, the length of the path along the centerline is 2.36 m, whereas the time-optimal path of the car has a length of 0.70 m. In Fig. 7.3, the states and controls of the time-optimal MPC problem are shown with their bounds. Note that the angle error state e^ψ can become quite large, but this is not a problem: this occurs in places where the car crosses the centerline almost perpendicularly.

In Fig. 7.4, the trajectory is plotted with the positions where the states and controls meet their limit (the bounds become active). The bounds for e_y are active at

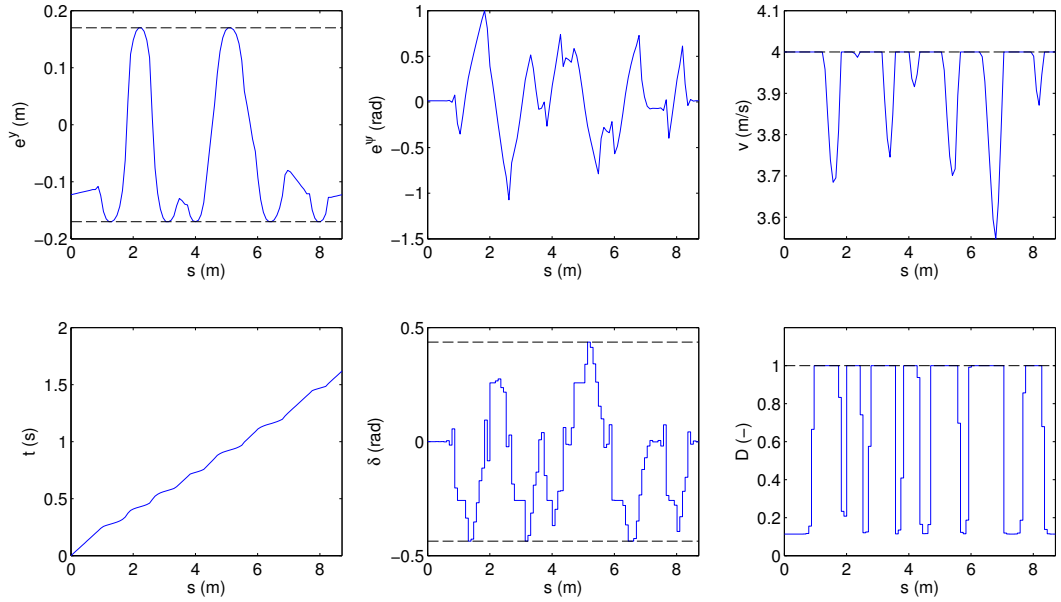


Figure 7.3: States and controls of the periodic solution that minimizes the total lap time. See eq. (7.5).

every corner, as can be seen from the trajectory. The remaining bounds are plotted on top of the trajectory.

Almost at every point of the track, a bound is active. This is to be expected, because of time-optimality. The velocity bound is active almost everywhere. The lower and upper bounds on the steering angle are active while turning sharply, as in the first corner (top right). The dutycycle is at its limit in each curve.

7.3 Online solution

In a real-time context, sampling times are too short to calculate the solution to an optimization problem such as 7.5 at every time step. Consequently, the solution of the time-optimal problem on an entire lap is brought down to the solution of a receding horizon control problem. In such a problem, the scope of the optimization problem is reduced to a fixed horizon. In the context of time-dependent models, the horizon length becomes an optimization variable. For a solution of the time-optimal driving problem, we aim at minimizing the time required for the race car to reach the end of the fixed-length spatial prediction horizon,

$$T = \int_{t_0}^{t_f} 1 \, \mathrm{d}t = \int_{s_0}^{s_f} \frac{1}{\dot{s}(\tau)} \, \mathrm{d}\tau \ . \tag{7.6}$$

For sufficiently long prediction horizons, this finite horizon objective approximates our goal of driving time-optimally.

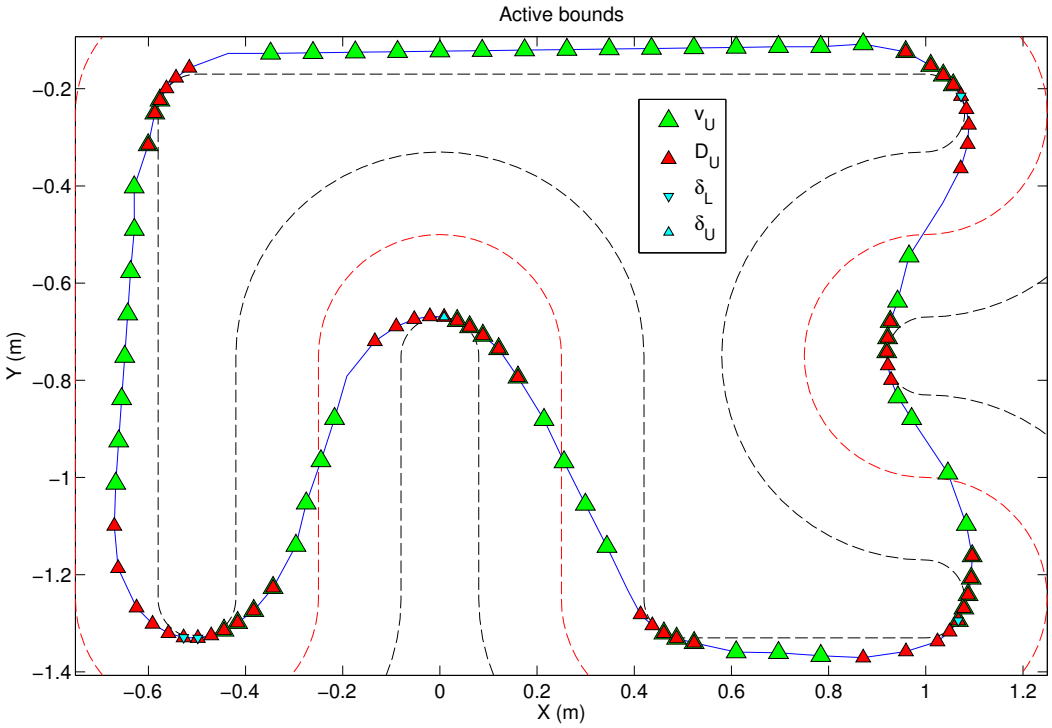


Figure 7.4: Plot of the active bounds at different positions on the trajectory.

The receding horizon optimal control problem then becomes:

$$\begin{aligned} \min_{\xi(\cdot),u(\cdot)} \quad & T = \int_{s_0}^{s_f} t(\tau)\mathrm{d}\tau \\ \text{s.t.} \quad & \xi'(s) = f(s,\xi(s),u(s)), \qquad s \in [s_0, s_f] \\ & e^y(s) \in [e^y_L(s), e^y_U(s)], \qquad s \in [s_0, s_f] \\ & u(s) \in [\delta_L, \delta_U] \times [-1, 1], \quad s \in [s_0, s_f] \\ & \xi(0) = \xi_0. \end{aligned} \tag{7.7}$$

For an efficient implementation (note that ACADO Code Generation only allows a least-squares type objective), we further use the following objective formulation. Let T^* be the minimum time required by a vehicle that satisfies the dynamics and constraints of Problem (5.6d). Then, for any $0 < T_{\text{ref}} < T^*$, the global optimum of the optimization problem

$$\min_{\xi(\cdot),u(\cdot),T} \quad \|T - T_{\text{ref}}\|^2_{q_t} \tag{7.8a}$$

$$\text{s.t.} \quad \xi'(s) = f(s,\xi(s),u(s)), \qquad s \in [s_0, s_f] \tag{7.8b}$$

$$e^y(s) \in [e^y_L(s), e^y_U(s)], \qquad s \in [s_0, s_f] \tag{7.8c}$$

$$u(s) \in [\delta_L, \delta_U] \times [-1, 1], \qquad s \in [s_0, s_f] \tag{7.8d}$$

$$\xi(0) = \xi_0, \tag{7.8e}$$

$(\hat{\xi}, \hat{u})$, satisfies $\hat{T} = T^*$.

By providing a suffciently small (i.e. infeasible) "target time" T_{ref} we can therefore have an approximate time-optimal MPC formulation in least-squares form.

7.4 Results

Using ACADO Code Generation, a solver for the NMPC problem described above (eq. (7.8)) is developed and tested both in simulation (Matlab) and on the experimental setup (custom C++ code).

7.4.1 Simulation results

On the model race cars, there is a speed limit, i.e. when the car drives with full power supplied to the DC motor. In simulation, this is enforced by adding an extra constraint

$$v(s) \in [0, v_{\max}], \quad s \in [s_0, s_f], \tag{7.9}$$

with $v_{\max} = 4.0$ m/s, for the 1:43 model race cars used in the experiments.

Numerous simulations were tried where the weight of all other states and controls than time was set to (almost) zero. However, unstable trajectories were the outcome. A solution might be to introduce Levenberg-Marquardt regularization (see section 4.5.1). We choose however to re-introduce all states and controls into the objective, weighting the time variable at the end of the prediction horizon t_N the most, and weighting the other states and controls only slightly.

In Figs. 7.5 and 7.6, the resulting trajectory and optimization variables are shown, respectively (blue curve). The samples are spaced $s_s = 0.05$ m apart along the centerline, the number of intervals over the horizon is set to $N = 20$, resulting in a horizon length of 1 m. The weights used are

$$\begin{aligned} Q &= \text{diag}([5 \cdot 10^{-4}, 10^{-10}, 10^{-10}, 10^{-10}]), \\ R &= \text{diag}([10^{-3}, 10^{-10}]), \\ q_{t_N} &= 1. \end{aligned} \tag{7.10}$$

The trajectory differs from the offline computed solution in the following ways: in the chicane, the car does not cut the corners as sharp as one would like, it swings a great deal to the side. Also, the steering behavior is less aggressive. This is due to the regularization on the steering angle and the short prediction horizon.

In both figures 7.5 and 7.6, the green curve plots the result of the same algorithm, but with a longer prediction horizon of 2.0 m (number of intervals $N = 40$). As we can see in Fig. 7.6, the lap time of the algorithm with the longer prediction horizon is slightly lower. Also the maximum velocity can be attained for longer than before. The control inputs have a similar behavior in the two simulations.

Looking at the resulting trajectories in Fig. 7.5, we see that the car drives a path that is closer to the offline optimization algorithm, that used a prediction horizon of one entire lap (cf. Fig 7.2). This is most apparent in the chicane (on the right) and the U-turn (bottom-middle).

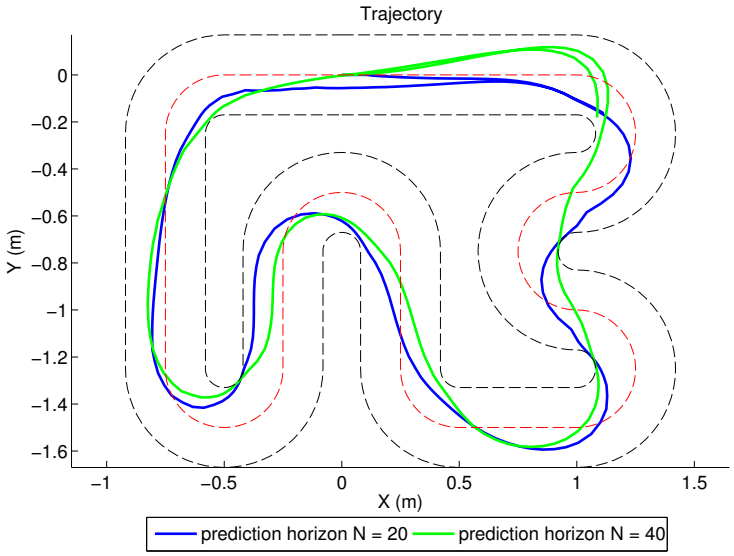


Figure 7.5: Comparison of the influence of the prediction horizon on the trajectories of the simulation of the on-line time-optimal problem.

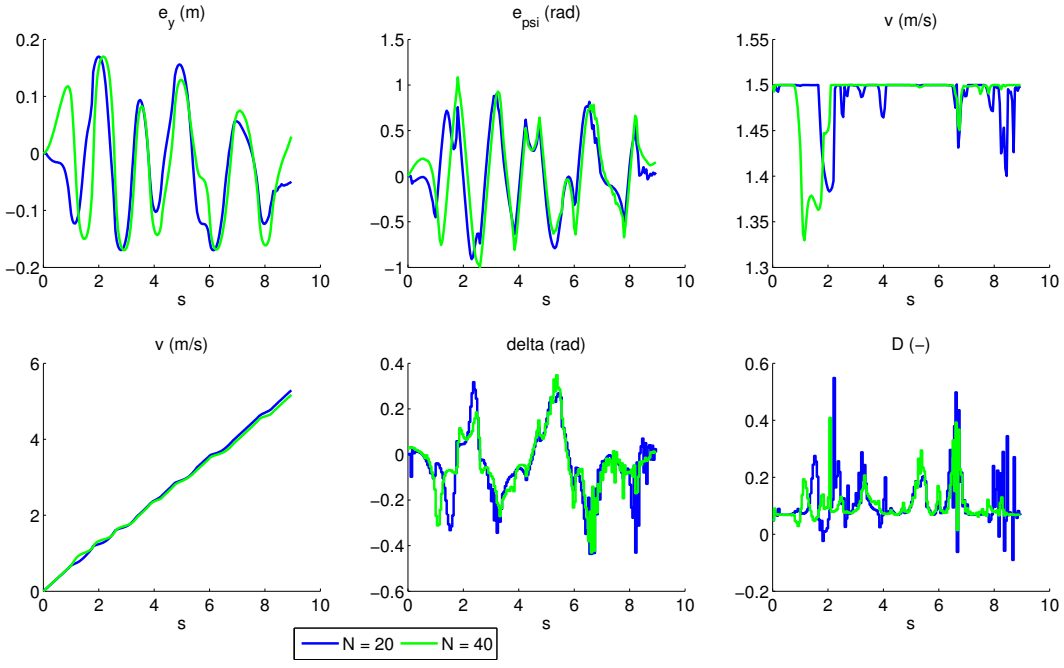


Figure 7.6: Comparison of the influence of the prediction horizon on the states and controls of the simulation of the on-line time-optimal problem.

7.4.2 Experimental results

Using the spatial reformulation, the sampling of the control intervals is done in space instead of time. In experiment however, we encounter the problem that the vision system runs at a fixed sampling in time (50 Hz). Consequently, the spacing of the control intervals need to be spaced closer to each other than 0.05 m, as in simulation, because the car will not advance 0.05 m each sampling time (this would imply a speed of 2.5 m/s). Therefore, samples of length 0.02 m are chosen. The number of prediction intervals is chosen to be $N = 30$ to obtain a reasonable horizon length of 0.60 m.

As seen in the experimental results in Chapter 6, in order to obtain satisfactory results, the weight matrices Q and R need to be tuned accordingly. The end time q_{t_N} remains the most heavily weighted, although for good performance, the weights are chosen as such:

$$\begin{aligned} Q &= \text{diag}([q_{e^y}, q_{e^\psi}, q_v, q_t]) = \text{diag}([10^{-4}, 10^{-10}, 10^{-10}, 5 \cdot 10^{-3}]), \\ R &= \text{diag}([r_\delta, r_D]) = \text{diag}([8 \cdot 10^{-5}, 10^{-4}]), \\ q_{t_N} &= 10^2. \end{aligned} \tag{7.11}$$

The results of the time-optimal MPC formulation with the above parameters are shown in Fig. 7.7. If we compare with the simulation results (Fig. 7.5), we see that the trajectories are quite similar. The car drives as close as possible to the insides of most curves, as in simulation (the car width is not shown here). By looking at the states and controls in Fig. 7.8, we can see that the steering behavior is aggressive. However, increasing the weight on the steering angle results in a crash of the car, because large values for the steering angle are then penalized more relatively, causing the car not to turn fast enough.

In Fig. 7.9, the predicted trajectories and the actual trajectory of the car are compared in the chicane, as this is the most challenging part of the race track. At the start of the first corner of the chicane, the actual trajectory of the car is more curved than the actual trajectory. However, going into the second corner, the predicted trajectory is flatter. Going into the third corner, the predicted one is curvier once again. This predicted vs. actual trajectory mismatch is due to the short prediction horizon.

Therefore, we increase the length of the prediction horizon. This can be done in two ways: either we increase the sampling of the control intervals s_s, or we increase the number of intervals N. As from the discussion before, it is not an option to increase s_s much, so instead we choose to extend the horizon by switching to $N = 50$ control intervals along the prediction horizon.

In Fig. 7.10, the actual trajectory is indeed closer to the predicted trajectory, because of the longer horizon. The resulting trajectory is plotted in 7.11 and the system states and controls are compared with the shorter horizon solution in Fig. 7.8. It can be seen that the longer horizon results in superior performance: compare the moderate deviation to the left-bottom border in Fig. 7.11 with the larger deviation on the same spot in Fig. 7.7. With the longer horizon, the coming straight section is visible at an earlier stage and therefore the turn can be taken more sharply.

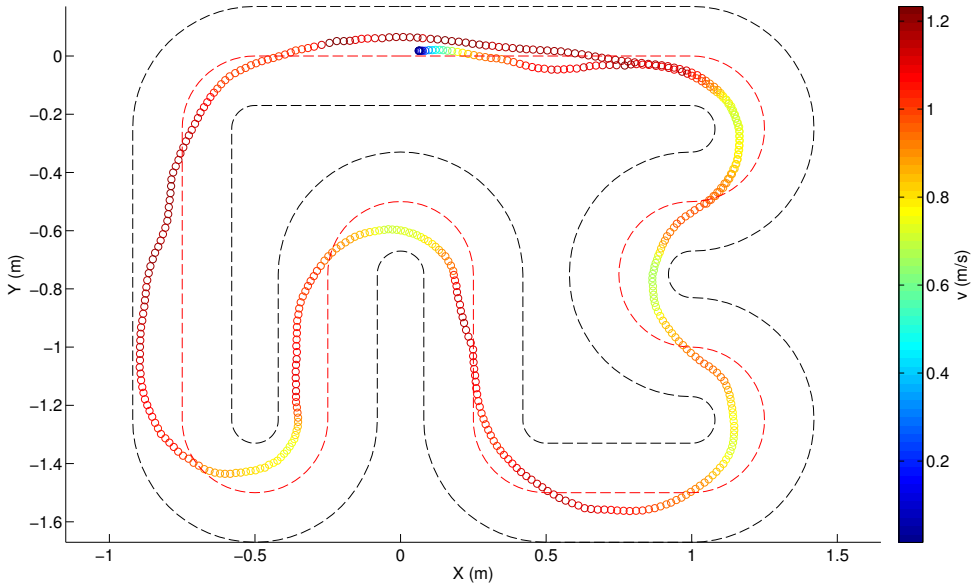


Figure 7.7: Trajectory driven by the real car using time-optimal MPC. The velocity of the vehicle is encoded with colors.

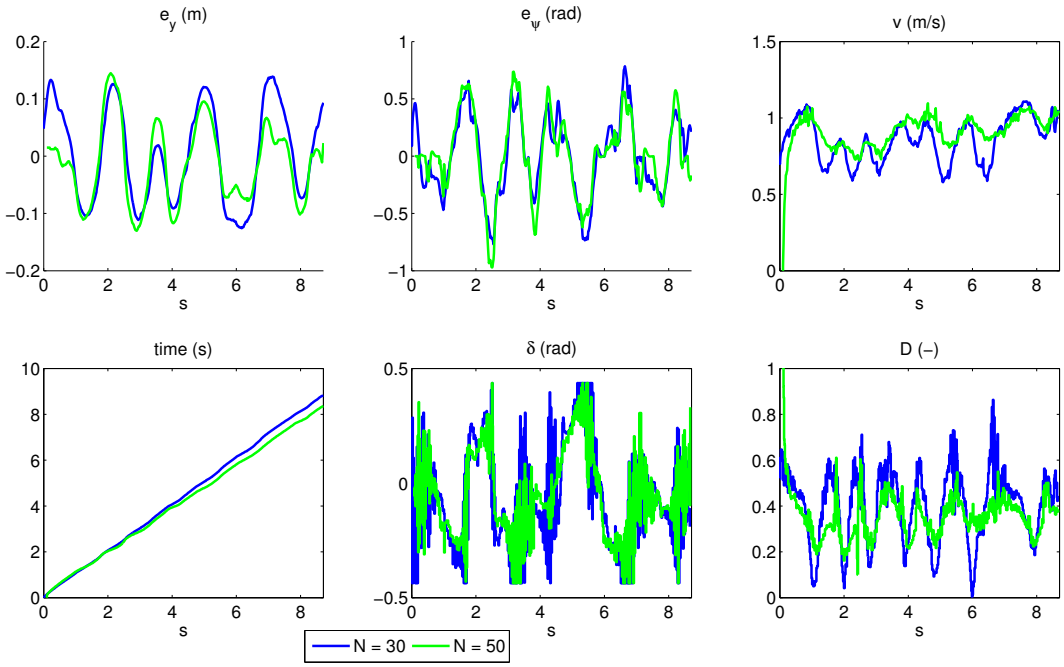


Figure 7.8: States and controls while running the experimental setup using time-optimal MPC.

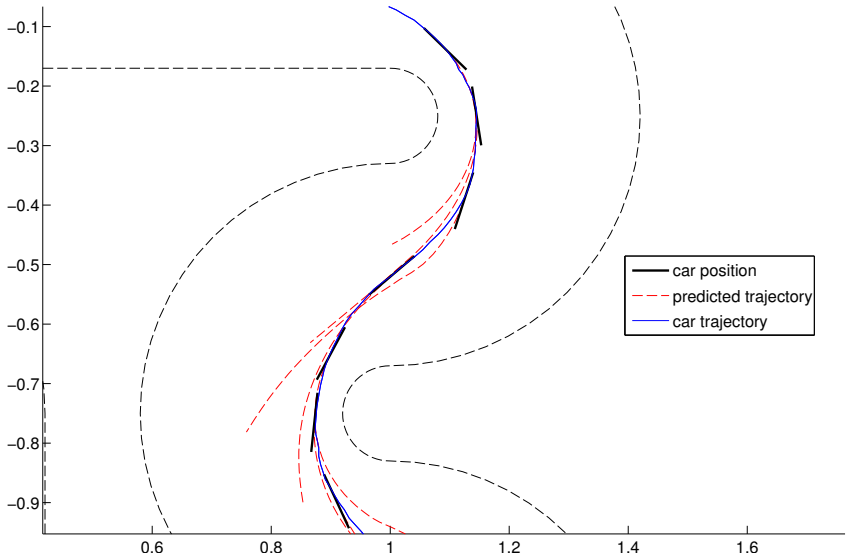


Figure 7.9: Predicted vs. resulting trajectory of the car.

Comparing the velocity in Fig. 7.8, the time-optimal MPC algorithm with longer horizon results in a higher velocity in most parts on the track. This, in combination with the shorter total length of the path driven with the longer horizon, results in a lower lap time (Fig. 7.8, bottom left).

7.4.3 Computational timings

As the vision system of the car runs at a fixed rate (50 m/s or 100 Hz), it is important to keep the computation time of our control loop below this limit. We therefore measure the computation time of ACADO using the built-in ACADO functions `tic(*timer)` and `toc(*timer)`. The results are shown in the boxplots in Fig. 7.12. As can be seen, for the shorter horizon algorithm, the computation time is well below the 0.01 s limit. For the longer horizon however, the median computation time is close to 0.01 s. Furthermore, there are quite a lot of outliers beyond 0.02 s. In the most extreme case, almost three measurement updates are missed. Therefore, a larger number of intervals than $N = 50$ is dangerous, as the car is not able to act on the fresh data until the computations are done.

7.4.4 Lap times

The most important performance criterium is the lap time of the car. The lap times of all the methods presented in this thesis are shown in Tab. 7.2. As can be seen, the time-optimal method with the longer horizon is faster than the variant with the shorter prediction horizon. The fastest algorithm to drive the car remains the two-level algorithm with the least curvature path.

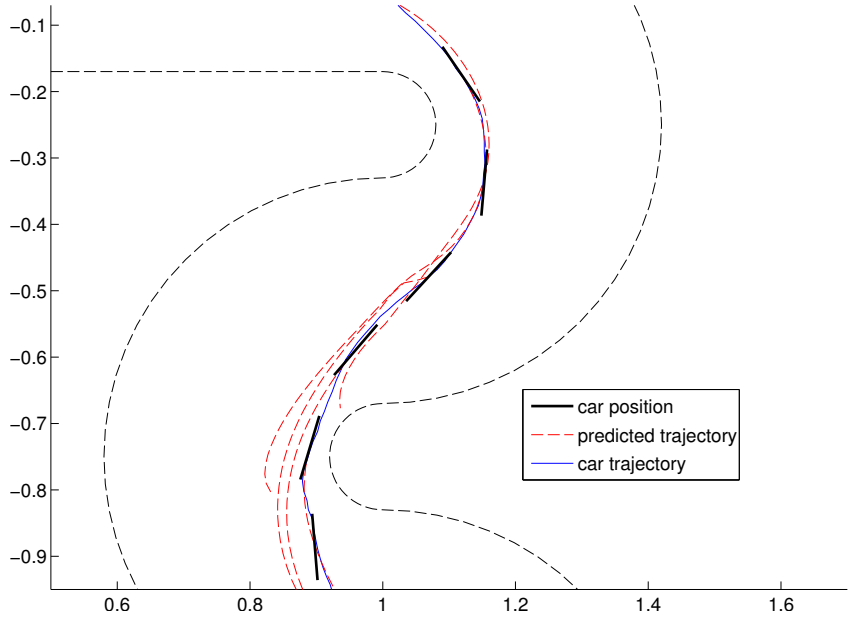


Figure 7.10: Predicted vs. resulting trajectory of the car for an time-optimal MPC formulation with a longer horizon of 1.0 m. For comparison, look at Fig. 7.9.

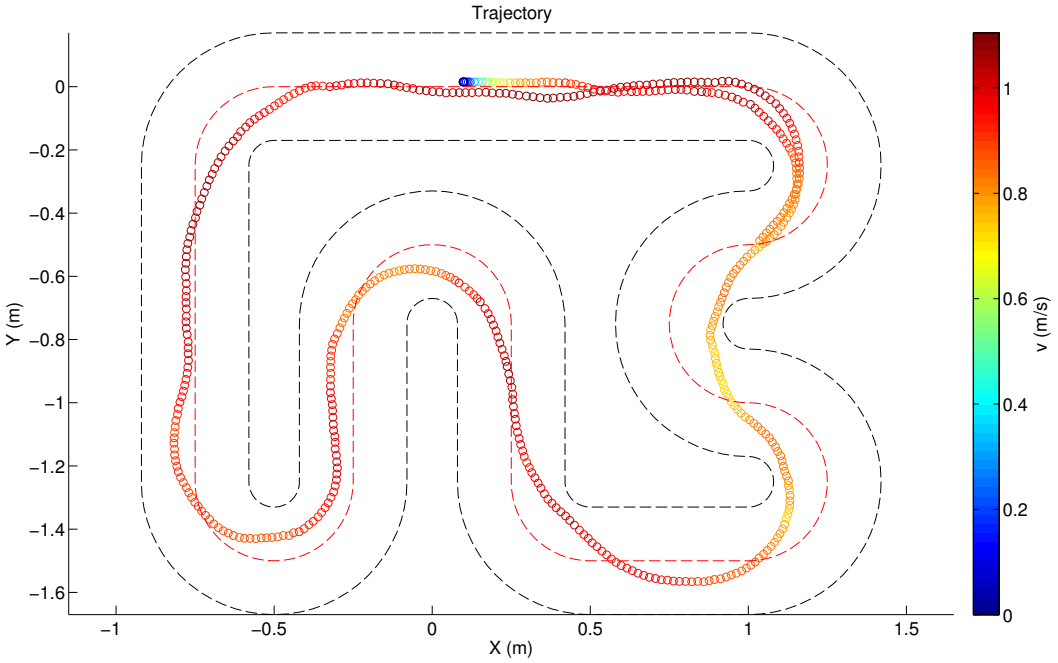


Figure 7.11: The trajectory driven by the car for a time-optimal MPC formulation with a longer horizon of 1.0 m.

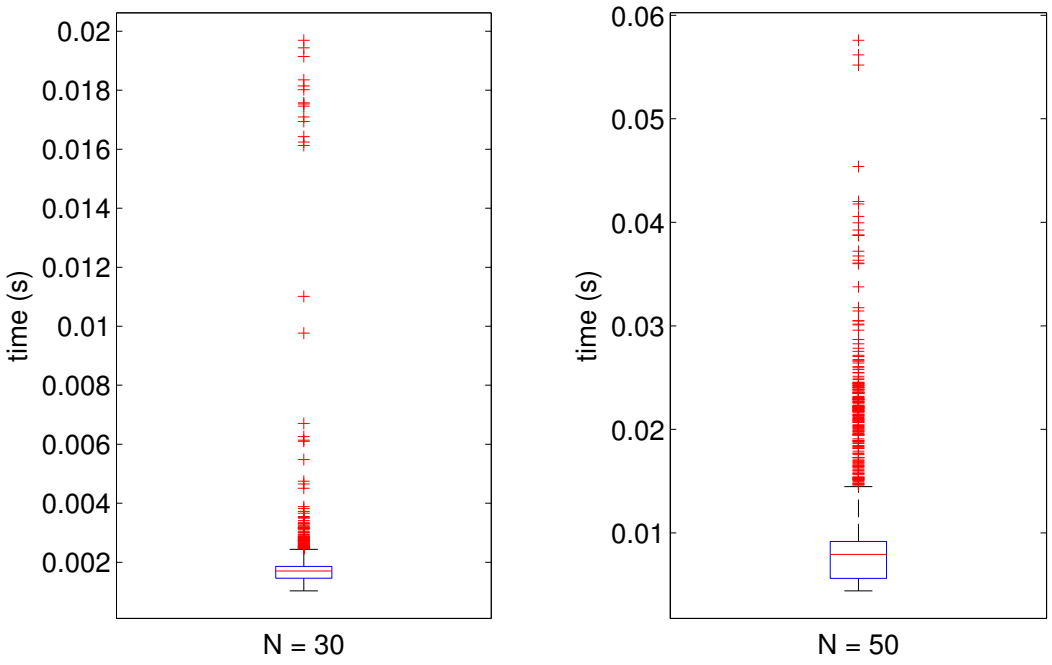


Figure 7.12: Computational times for one call of the `feedbackStep()` function of ACADO. The red horizontal line denotes the median, the + markers denote outliers.

Table 7.2: Comparing lap times between the different methods in simulation.
LC=least curvature, SP=shortest path

Algorithm	first lap time	average lap time
Linear MPC tracking ($v_{\text{ref}} = 0.5$ m/s)	17.6 s	17.5 s
Two-level MPC (LC, $v_{\text{ref}} = 2.2$ m/s)	8.8 s	8.0 s
Two-level MPC (SP, $v_{\text{ref}} = 2.0$ m/s)	9.1 s	8.95 s
Time-optimal MPC (horizon 0.6 m	8.4 s	8.3 s
Time-optimal MPC (horizon 1.0 m	8.3 s	8.2 s

Chapter 8

Conclusion

This master thesis developed different methods for autonomous driving for small-scale radio-controlled race cars. The time-optimal problem was tackled in two ways. First, a set of heuristic linear MPC algorithms structured in two levels was presented. Afterwards, a more advanced one-level nonlinear MPC method for time-optimal control using a nonlinear bicycle model was developed. The real-time feasibility of the resulting algorithm was confirmed.

The main contribution of this thesis is the development of new methods that resulted in a significant decrease in lap time in comparison with linear MPC centerline tracking existing formerly at the setup, and the implementation of a real-time controller that can be used on an experimental setup. Furthermore, the use of nonlinear MPC and the ACADO software on a setup of this kind is, to our knowledge, a new result.

Outlook

Starting from here, different paths for further research can be investigated.

Improvements Several improvements to the developed methods can be proposed. As the plant-model mismatch is the highest in the curves (the car is not able to turn sharply enough), the time-optimal MPC formulation can be extended with a velocity constraint depending on the maximum lateral acceleration of the vehicle, and the curvature of the track. Also, the objective function of the time-optimal formulation can be made to contain only the time at the end of the horizon.

Model The physical model remains a crucial part of any model-based control method. Often, the unsatisfactory performance of the car in this thesis, was due to the inability to very precisely predict the future behavior of the car. Extensions to more involved vehicle models (such as those presented in Chapter 3) are surely worth an investigation, because a more accurate model results in a more accurate prediction of the car's behavior.

Other solvers ACADO, in combination with qpOASES, uses a condensing technique to reduce the problem dimension of the MPC problem. However, other solvers, like qpDUNES or FORCES, do not condense the problem but instead exploit the structure resulting from the full MPC problem. In this way, a longer horizon can be attained. This will make it possible to reduce the lap times.

State estimation An accurate state estimation is needed for an accurate prediction of the vehicle's state. Thus, switching from a Kalman filter to an extended Kalman filter (EKF) or moving horizon estimation (MHE) can be the subject of further research. In combination with a more advanced vehicle model, also the estimation of the vehicle model or tire model parameters could be a promising option for performance increase.

Appendices

Appendix A

Technical details of the experimental setup

Infrared camera: PointGrey Flea 3 Infrared Camera
Imaging: 1280x1024 at 100 FPS
Connection with computer: USB 3.0

RGB camera: xiQ MQ013CG-ON Camera
Imaging: 1280x1024 (1.3 MPixel) at 50 FPS
Connection with computer: USB 3.0

Computer: Real-time Debian system
Intel Core i3-3220 @ 3.3 GHz
OS: Debian 7.0 'Wheezy'
Kernel: Linux 3.8.13 with Preempt RT Patch

Bluetooth link: Asynchrounous Connection-Less Bluetooth (ACL) communication link

Race Car: Kyosho dNaNo FX-101 ASF2.4GHz System

RC-controller: PERFEX KT-18 Transmitter 2.4GHz

Appendix B

Master thesis paper

Below, the master thesis paper is included.

Ontwerp van een Tijdsoptimale Regelaar voor Modelracewagens

Robin Verschueren

***Abstract*— Dit artikel bespreekt een real-time regelaar voor het tijdsoptimaal controleren van autonome voertuigen. De controlesignalen worden berekend met een niet-lineaire, modelgebaseerde predictieve regelaar (model predictive control, MPC). Een aangepaste herformulering van de tijdsoptimale doelfunctie is ontwikkeld om het gebruik van bestaande, efficiente algoritmes voor niet-lineaire MPC mogelijk te maken. De validatie van deze aanpak gebeurt zowel in een numerieke simulatie-omgeving, als aan de hand van een experimentele proefopstelling met racewagens op schaal.**

I. Inleiding

In de voorbije jaren hebben verschillende Advanced Driver Assistance Systems (ADAS) hun intrede gedaan in personenvoertuigen. Voorbeelden hiervan zijn het Anti-lock Braking System (ABS) en Electronic Stability Control (ESC) die te vinden zijn in de voertuigen van bijna alle autoconstructeurs. Meer recent nog kwam Toyota in 2010 met een autonoom parkeersysteem [16] en introduceerde Volvo in 2012 'platooning' [14], waarbij auto's op automatische wijze een leidersvoertuig volgen. De bovengenoemde semi-autonome systemen zetten de technologie op weg naar compleet autonome personenvoertuigen, die binnenkort een realiteit zullen zijn.

Een centraal onderdeel in elk autonoom systeem is het regelsysteem. De regelaar zorgt ervoor dat het systeem zich gedraagt zoals verwacht. In het geval van autonome voertuigen is dit zelfstandig rondrijden, de grenzen van de weg niet overschrijden, zich aan een bepaalde snelheid houden, obstakels ontwijken, enz. De meest in de praktijk gebruikte regeltechniek is de proportionele-integrerende-differentiërende regelaar (PID). Bij deze techniek introduceert men een foutensignaal

$$e(t) = y(t) - y_{\mathrm{ref}}(t) \tag{1}$$

in functie van de uitgang $y(t)$ van het systeem en het referentiesignaal $y_{\mathrm{ref}}(t)$. De regelactie $u(t)$ wordt berekend als volgt:

$$u(t) = K_p e(t) + K_i \int_{-\infty}^{t} e(\tau)\,\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}. \tag{2}$$

Deze techniek vertoont echter een aantal gebreken: ten eerste is een PID regelaar niet modelgebaseerd, er wordt enkel naar de uitgang $y(t)$ gekeken zonder rekening te houden met de interne toestand van het model. Een ander nadeel is dat er geen rekening wordt gehouden met beperkingen op controlesignalen en systeemrespons. Ten slotte is er bij PID geen sprake van optimalisatie: het is niet duidelijk of de berekende controle-actie $u(t)$ een goede keuze is of niet.

Model Predictive Control (MPC) is een regeltechniek die tegemoetkomt aan deze tekortkomingen. Deze techniek baseert zich op een dynamisch model en maakt op basis daarvan een voorspelling van het verloop van de interne toestand $x(t)$ van dit model afhankelijk van de gekozen controle-actie $u(t)$. De systeemrespons kan vervolgens geoptimaliseerd worden om een gegeven doelfunctie V te minimaliseren. De eerste actie van de optimale inputsequentie u wordt vervolgens aangelegd aan het systeem, waarna de hele procedure zich herhaalt in de volgende tijdsstap. Echter, in real-time toepassingen (zoals autonoom rijden) is het een uitdaging om MPC aan hoge bemonsteringsfrequentie uit te voeren.

Enkele van bovengenoemde ADAS zijn gebaseerd op traditionele regeltechnieken, zoals PID. Daarnaast heeft MPC sterk aan populariteit gewonnen in het onderzoek naar autonome voertuigen, voornamelijk door de nood aan een betrouwbare voorspelling van het rijgedrag. In [4], [8], [9] werd MPC al succesvol toegepast op autonoom rijden. Hier werden modelvereenvoudigingen gebruikt om de computationele complexiteit van MPC te verlagen. In [6], [18] echter, wordt een meer geavanceerd voertuigmodel gebruikt, waarbij het toch mogelijk blijft om in real-time de controle-acties te berekenen. Daarvoor werd op maat gemaakte, hoogperformante `C`-code voorzien, maar enkel simulatieresultaten zijn beschikbaar om de regelperformantie na te gaan.

In de eerdergenoemde publicaties was het doel telkens het volgen van een referentie. Dit artikel, daarentegen, beschouwt *tijdsoptimaal* rijden. Doordat er een duidelijk conflict is tussen veiligheid en snelheid bij het besturen van wagens, lijkt het ontwerp van een tijdsoptimale regelaar die voldoet aan de veiligheidsbeperkingen (bv. de rand van de weg), een bijzondere uitdaging.

Tijdsoptimale formuleringen in MPC hebben aandacht gekregen in trajectvolgen in de robotica, zie [17]. Daar echter, is het pad dat de robotarm moet volgen op voorhand gekend, en kan het probleem geschreven worden als een convex optimalisatieprobleem. Dit is aanzienlijk eenvoudiger op te lossen dan het probleem van autonoom rijden, omdat het pad daar online moet berekend worden.

Bestaande resultaten in tijdsoptimaal rijden met een geavanceerde regeltechniek zijn te vinden in [11], [13], maar hier werden enkel open-lus oplossingen berekend. In [2] werd een gelaagde regelarchitectuur van twee regelaars voorgesteld, waar een eerste regelaar het traject plant, dat gevolgd moet worden door de tweede regelaar.

Robin Verschueren, Tweede Master Wiskundige Ingenieurstechnieken `robin.verschueren@student.kuleuven.be`

Promotor: Prof. Dr. Moritz Diehl. Begeleiders: Stijn De Bruyne, Janick Frasch, Mario Zanon

Thesis in samenwerking met LMS, A Siemens Business, Haasrode.

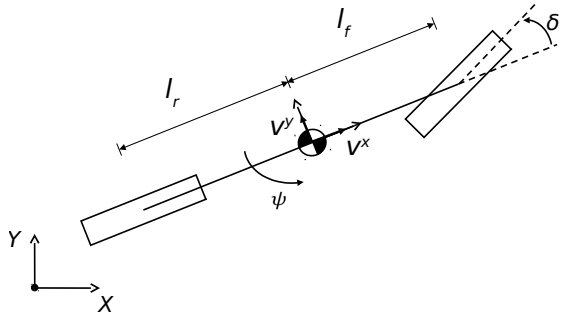


Fig. 1. Tweewielig voertuigmodel.

In deze paper stellen we integendeel een ééntrapsregelaar voor, waarbij we een niet-lineair MPC (NMPC) probleem oplossen in real-time. Niet enkel tonen we simulatieresultaten, de ontwikkelde algoritmes zullen geëvalueerd worden op een experimentele proefopstelling, bij LMS, A Siemens Business, in Haasrode. Deze aanpak past in het kader van 'rapid prototyping', waarbij een algoritme op kleine schaal getest wordt, zonder nood aan kostintensieve testmodellen en proefterreinen.

De rest van dit artikel is als volgt gestructureerd: eerst wordt er in sectie II ingegaan op het voertuigmodel dat werd gebruikt bij de experimenten. In sectie III en IV wordt het optimalisatieprobleem geformuleerd en oplossingsmethoden aangereikt. Sectie V beschrijft de experimentele setup, de resultaten van de experimenten worden besproken in VI en sectie VII concludeert het artikel.

II. VOERTUIGMODEL

In [6] werd een voertuigmodel met inbegrip van de wielen ophangingsdynamica voor de vier wielen gebruikt in autonoom rijden gebaseerd op niet-lineaire MPC in simulatie. In dit artikel beschouwen we experimenten uitgevoerd op een proefopstelling bestaande uit op afstand bestuurbare racewagens en een parcours op schaal. Vanwege de beperkte beschikbaarheid van modelparameters voor miniatuur racewagens, in het bijzonder de interactie tussen de banden en de weg, kiezen wij ervoor om een gereduceerd model te gebruiken in het regelsysteem. We verwaarlozen de band-weg interactie, en de lastoverbrenging, resulterend in een slipvrij tweewielig voertuigmodel (*bicycle* model), wat hierna in het kort besproken wordt.

A. Tijdsafhankelijk voertuigmodel

Het chassis van de wagen is gemodeleerd als een stijf lichaam, gekarakteriseerd door zijn positie in het globaal assenstelsel $X-Y$, zijn oriëntatie ψ en zijn absolute snelheid v, zie Fig. 1.

De inputs voor het model zijn de stuurhoek δ en de *dutycycle* D aangelegd aan de DC motor. De dutycycle is de fractie van de bemonsteringstijd van de motor dat het voltage over de motor hoog is. Een hogere dutycycle komt overeen met een groter moment geleverd op de achteras van de wagentjes.

Parameter	Eenheid	Fysische betekenis	Waarde
C_1	-	geometrisch (l_r/l)	0.5
C_2	m^{-1}	geometrisch ($1/l$)	17.06
C_{m_1}	m/s^2	motorparameter	12.0
C_{m_2}	1/s	motorparameter	2.17
C_{r_2}	1/m	tweede-orde wrijvingsparameter	0.1
C_{r_0}	m/s^2	nulde-orde wrijvingsparameter	0.6

TABLE I
MODELPARAMETERS VAN HET BICYCLE MODEL

De modelvergelijkingen zijn als volgt:

$$\dot{X} = v\cos(\psi + C_1\delta) \tag{3a}$$
$$\dot{Y} = v\sin(\psi + C_1\delta) \tag{3b}$$
$$\dot{\psi} = v\,\delta\,C_2 \tag{3c}$$
$$\dot{v} = C_{m_1}D - C_{m_2}\,D\,v - C_{r2}\,v^2 - C_{r_0} - (v\,\delta)^2\,C_2\,C_1^2. \tag{3d}$$

Hierin gebruikten we de volgende veronderstellingen, cf. [15], [12],

$$v^x \approx v$$
$$v^y \approx v\,C_1\,\delta$$
$$v^2 = (v^x)^2 + (v^y)^2$$

Een samenvatting van de modelparameters $C_\cdot$ is gegeven in Tabel I.

B. Spatiale herformulering

Model 3 is een dynamisch systeem met tijd als onafhankelijke veranderlijke. Omdat we van de tijd een optimalisatievariabele willen maken, is een transformatie van variabelen nodig. We gebruiken de transformatie naar spatiale coördinaten zoals die gebruikt werd in [6]. Een voordeel van deze transformatie is dat de beperkingen op de grenzen van de weg een constante worden, onafhankelijk van de toestand van de wagen op dat moment.

We vermelden hier de spatiale transformatie voor volledigheid. De voertuigcoördinaten in het wereldassenstelsel worden gegeven door $[X, Y]^T$. We projecteren deze coördinaten op een curve σ, het referentietraject, geparametriseerd door zijn booglengte $\sigma(s)$. We vervangen de toestanden X, Y en ψ door

$$e^y = \cos(\psi^\sigma)(Y - Y^\sigma) - \sin(\psi^\sigma)(X - X^\sigma), \text{ and} \tag{4a}$$
$$e^\psi = \psi - \psi^\sigma, \tag{4b}$$

waar $[X^\sigma, Y^\sigma]^T$ en ψ^σ de positie en de oriëntatie van het referentiepunt op het pad zijn (zie Fig. 2). De variabele s, gebruikt om het referentietraject te parametriseren, geeft de vooruitgang langs het pad weer. Als we aannemen dat de wagen altijd in beweging is op elk tijdstip ($\dot{s} > 0$), geldt er voor de toestandsvector $\xi = [\ e^y\ e^\psi\ v\ t\ ]$ dat

$$\xi' = \frac{d\xi}{ds} = \frac{d\xi}{dt}\frac{dt}{ds} = \frac{1}{\dot{s}}\dot{\xi}, \tag{5}$$

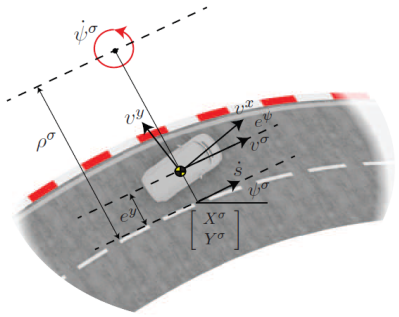


Fig. 2. Definitie van het coördinatenstelsel gebruik in de spatiale herformulering van het voertuigmodel. De s-coördinaat geeft de booglengte langs het parcours aan. Bron: [6].

TABLE II

TOESTANDEN EN INPUTS VAN HET SPATIALE VOERTUIGMODEL

Toestand	Eenheid	Beschrijving
e^y	m	Afwijking van de middenlijn
e^ψ	rad	Hoek ten opzichte van het pad
v	m/s	Absolute snelheid
t	s	Tijd

Input	Bereik	Eenheid	Beschrijving
δ	[-0.44, 0.44]	rad	Stuurhoek
D	[-1,1]	-	Dutycycle van de DC motor

waarbij $\dot{\xi}$ kan afgeleid worden uit (3) en (4).

Nu rest er ons nog een uitdrukking van $\dot{s}$ in functie van de toestanden van het voertuigmodel te bepalen. Uit de tekening in Fig. 2 halen we

$$v^\sigma = (\rho^\sigma - e^y)\,\dot{\psi}^\sigma, \text{ and}$$
$$v^\sigma = v^x \cos e^\psi - v^y \sin e^\psi,$$

mat ρ^σ de lokale kromtestraal van σ. De snelheid van de wagen langs het referentie pad, $\dot{s}$, is gegeven door

$$\dot{s} = \rho^\sigma\,\dot{\psi}^s = \frac{\rho^\sigma}{\rho^\sigma - e^y}\,(v^x \cos(e^\psi) - v^y \sin(e^\psi)).$$

Het dynamische systeem met spatiale herformulering is dan

$$e^{y\prime}(s) = (v^x \sin(e^\psi) + v^y \cos(e^\psi))/\dot{s} \quad (6a)$$
$$e^{\psi\prime}(s) = \dot{\psi}/\dot{s} - \kappa^\sigma(s) \quad (6b)$$
$$v'(s) = \dot{v}/\dot{s} \quad (6c)$$
$$t'(s) = 1/\dot{s}, \quad (6d)$$

waar $\kappa^\sigma(s) = 1/\rho^\sigma(s)$ de lokale kromtestraal van het parcours is. De toestanden in het globale assenstelsel kunnen ten allen tijde berekend worden als

$$X = X^\sigma - e^y \sin(\psi^\sigma)$$
$$Y = Y^\sigma - e^y \cos(\psi^\sigma)$$
$$\psi = \psi^\sigma + e^\sigma.$$

Een samenvattende tabel van toestanden en inputs van het spatiale model kan gevonden worden in Tab. II.

III. NIET-LINEAIRE MPC FORMULERING

A. Trajectvolgen

In [18] werd een niet-lineaire MPC formulering voor autonoom rijden geïntroduceerd voor het volgen van de middellijn van een circuit. Het terugtredende-horizonprobleem is dan:

$$\begin{aligned}
\min_{\xi(\cdot),u(\cdot)} \quad & \int_{s_0}^{s_f} \left\|\xi(\tau) - \xi_{\text{ref}}(\tau)\right\|_Q^2 + \left\|u(\tau) - u_{\text{ref}}(\tau)\right\|_R^2 \, d\tau \\
& + \left\|\xi(s_f) - \xi_{\text{ref}}(s_f)\right\|_P^2 \\
\text{s.t.} \quad & \xi'(s) = f(s, \xi(s), u(s)), \qquad s \in [s_0, s_f] \\
& e^y(s) \in [e_L^y(s), e_U^y(s)], \qquad s \in [s_0, s_f] \\
& u(s) \in [\delta_L, \delta_U] \times [-1, 1], \quad s \in [s_0, s_f] \\
& \xi(0) = \xi_0,
\end{aligned} \quad (7)$$

waar $\xi = [e^y, e^\psi, v, t]^T$ de toestandsvector voorstelt en $u = [\delta, D]^T$ de vector van regelinputs. Het interval $[s_0, s_f]$ is de predictiehorizon en $\|\cdot\|$ de Euclidische norm met gewichtsmatrices Q, R, P. De grenzen van de baan zijn aangeduidt met $e_L^y(s), e_U^y(s)$. Merk op dat het stelsel gewone differentiaalvergelijkingen geformuleerd is met spatiale afgeleiden $\xi' = \dfrac{d\xi}{ds}$ (zie (6)) in plaats van tijdsafgeleiden. Voorts gebeurt de bemonstering nu in de ruimte in plaats van in de tijd. Deze wordt aangegeven met monsters van lengte s_s.

B. Tijdsoptimale formulering

Voor een benaderende oplossing van het tijdsoptimaal probleem, trachten we de tijd te minimaliseren die de auto nodig heeft om het einde van de horizon

$$T = \int_{t_0}^{t_f} 1 \,\mathrm{d}t = \int_{s_0}^{s_f} \frac{1}{\dot{s}(\tau)} \,\mathrm{d}\tau \quad (8)$$

te bereiken. Voor een voldoende lange predictiehorizon, verwachten we dat deze aanpak een goede benadering is van tijdsoptimaal rijden. Omdat efficiënte methoden bestaan voor NMPC formuleringen met een kleinste-kwadraten kostfunctie (zie sectie IV), gebruiken we verder de volgende formulering:

$$\begin{aligned}
\min_{\xi(\cdot),u(\cdot),T} \quad & \|T - T_{\text{ref}}\|_{q_{t_N}}^2 \\
\text{s.t.} \quad & \xi'(s) = f(s, \xi(s), u(s)), \qquad s \in [s_0, s_f] \\
& e^y(s) \in [e_L^y(s), e_U^y(s)], \qquad s \in [s_0, s_f] \\
& u(s) \in [\delta_L, \delta_U] \times [-1, 1], \quad s \in [s_0, s_f] \\
& \xi(0) = \xi_0.
\end{aligned} \quad (9)$$

Door het kiezen van een voldoende lage referentietijd T_{ref} hebben we een benaderende methode voor tijdsoptimaal rijden in een NMPC formulering met een kleinste-kwadraten doelfunctie.

IV. OPLOSSINGSMETHODE

We maken gebruik van software gebaseerd op de ACADO Toolkit [1], dat eerder al gebruikt werd voor snelle niet-lineaire MPC in de context van autonoom rijden in [18]. Uit deze toolkit gebruiken we de ACADO Code Generation

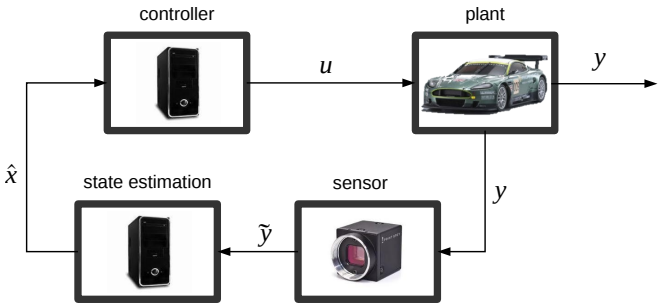


Fig. 3. Schematische weergave van de experimentele setup.

tool [10], dat toelaat om automatisch code te exporteren die een implementatie bevat van het *Real-time Iteration Scheme* (RTI), op maat gemaakt voor de dynamica van het model in het MPC probleem. Dit RTI-schema laat toe om NMPC problemen efficiënt op te lossen, gebruik makend van *multiple shooting*, zie [3] voor de algoritmische details. De geëxporteerde ANSI C-code bevat daarnaast ook een efficiënte Runge-Kutta integrator van orde 4 en een in geheugen beperkte versie van de QP solver qpOASES [5].

V. Experimentele setup

Eén van de belangrijkste bijdragen in dit artikel is de experimentele validatie van de numerieke methodes op een proefopstelling op schaal 1:43. In wat volgt wordt deze setup kort besproken. In de experimenten wordt gebruikt gemaakt van de Kyosho dNaNo modelwagens. Metingen van de huidige positie (X, Y), oriëntatie ψ, snelheden v_X, v_Y en hoeksnelheid $\dot{\psi}$ worden gedaan met behulp van een visiesysteem op basis van een RGB-camera met een frequentie van 50 Hz. De opstelling wordt schematisch voorgesteld in Fig. 3.

Omgevingsfactoren zoals zonlicht kunnen ertoe leiden dat er geen meting van de wagen plaatsvindt. Daarom worden de metingen gefilterd door een Kalman filter, zodat er toch steeds nuttige data voorhanden is. Het Kalman filter algoritme gebruikt een puntmassamodel van de wagen om de positie, oriëntatie en snelheid van de wagen te schatten.

De regelaar loopt op een dekstop computer met Debian LINUX 7.0 'Wheezy' met de *RT Preempt* real-time kernel patch geïnstalleerd dat een soft real-time uitvoering van het visiesysteem garandeert.

De controle-inputs worden met behulp van een Bluetooth-dongle naar de auto gezonden, waarop een Bluetooth-antenne op geplaatst is. Verdere technische details zijn aangegeven in appendix A.

Het gebruikte racecircuit bestaat uit rechte stukken en bochten van $90°$. Het parcours heeft o.a. een chicane, een U-bocht en een langer recht deel. Een overzicht van de setup is getoond in Fig. 4.

VI. Resultaten

In deze sectie worden de resultaten van trajectvolgen en tijdsoptimaal rijden geanalyseerd en vergeleken, zowel in simulatie als in experiment.

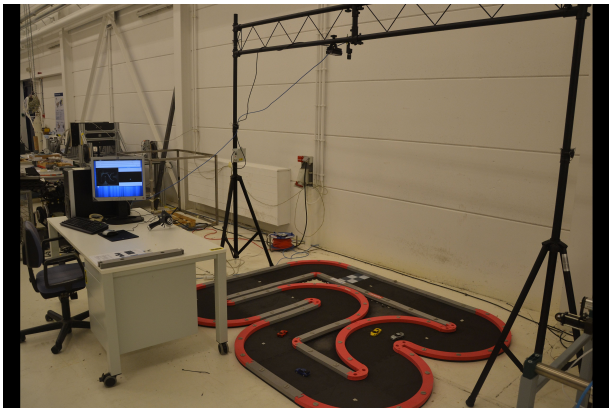


Fig. 4. De experimentele proefopstelling.

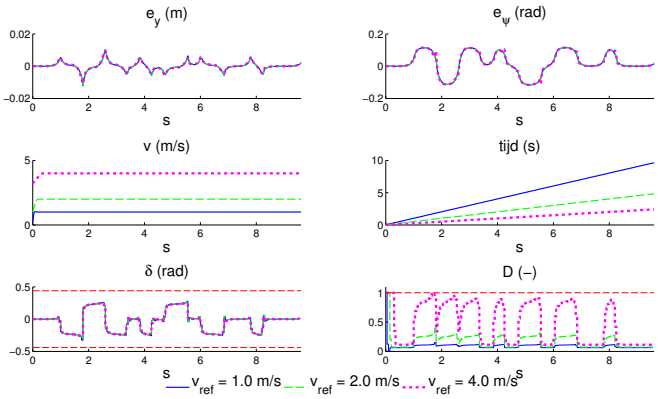


Fig. 5. Toestanden en inputs bij trajectvolgen in simulatie op drie verschillende referentiesnelheden.

A. Simulatie

Alle simulaties worden uitgevoerd in MATLAB, gebruik makend van de ACADO Code Generation Tool. Het model dat gebruikt wordt is voor trajectvolgen en tijdsoptimaal rijden hetzelfde, net als de spatiale predictiehorizon die is vastgelegd op 1.0 m, opgedeeld in $N = 20$ controle-intervallen.

Voorts simuleren we de topsnelheid van de wagentjes door een extra beperking aan te leggen op de snelheid:

$$v(s) \in [0, v_{\max}], \quad s \in [s_0, s_f].$$

Voor de miniatuurwagentjes ligt deze op ca. $v_{\max} = 4$ m/s.

1) Trajectvolgen: De middellijn van het parcours is gekozen als referentiepad. De diagonale gewichtsmatrices worden gekozen als $Q = \text{diag}([1, 0.01, 0.1, 0])$, $R = \text{diag}([10^{-4}, 10^{-4}])$, $P = Q$.

De resultaten voor simulatie op verschillende referentiesnelheden $v_{\text{ref}} = 1.0\,\text{m/s}$, $2.0\,\text{m/s}$, en 4.0 m/s zijn getoond in Fig. 5. Zoals blijkt uit de figuren is het resulterende traject zeer dicht bij de middellijn: de afwijking van het referentietraject is nooit meer dan 1 cm. Ook de snelheid wordt zeer accuraat gevolgd, op alle refentiesnelheden. Hoe hoger de refentiesnelheid, hoe hoger ook de resulterende dutycycle D.

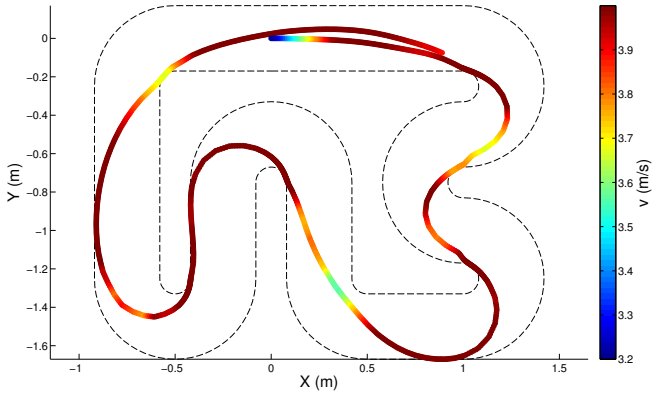


Fig. 6. Traject bij tijdsoptimaal rijden in simulatie. De snelheid is aangegeven in kleur.

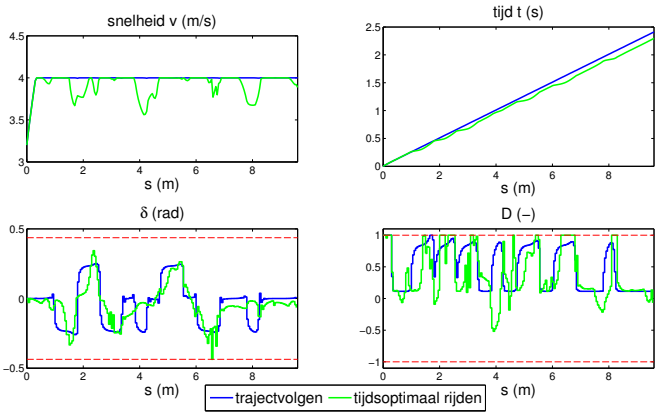


Fig. 7. Vergelijking tussen trajectvolgen en tijdsoptimaal rijden in simulatie.

2) Tijdsoptimaal rijden: Bij tijdsoptimaal rijden veranderen de gewichten substantieel: de tijd aan het eind van de predictiehorizon (met gewicht q_{t_N}) wordt het meest gewogen, terwijl de andere gewichten een pak lager liggen, echter niet nul, omdat regularisatie nodig is om een onstabiele oplossing te vermijden. Een adequaat stel gewichten hiervoor zijn

$$Q = \texttt{diag}([10^{-10}, 10^{-10}, 10^{-10}, 10^{-10}]),$$
$$R = \texttt{diag}([10^{-3}10^{-10}]),$$
$$q_{t_N} = 1.$$

De referentietijd aan het eind van de predictiehorizon is gekozen als 0.24 s

Het resulterende traject met snelheidsprofiel is weergegeven in Fig. 6 en de resterende variabelen zijn vergeleken met die van trajectvolgen in Fig. 7. Zoals men kan zien, is het traject erg verschillend van de centerlijn. Op sommige plaatsen rijdt de auto tegen de grenzen van de baan. Uit Fig. 7 blijkt inderdaad dat tijdsoptimaal rijden resulteert in een lagere eindtijd dan trajectvolgen aan maximum snelheid (grafiek rechtsboven): de tijd voor één rondje is 2.07 s tegenover 2.19 s voor trajectvolgen. Het tijdsoptimaal rijden volgt minder de maximale snelheid, maar haalt zijn tijdswinst echter uit bochten afsnijden. Dit is duidelijk uit de chicane (rechterdeel van het parcours). Ook de grenzen op de dutycycle D zijn actief op sommige posities langs het parcours.

B. Experimentele resultaten

We testen de twee methodes vervolgens uit op de experimentele setup. Omdat er sprake is van een discrepantie tussen systeem en model (slip wordt niet gemodeleerd) moeten we de referentiesnelheid voor trajectvolgen een flink stuk lager leggen dan in simulatie. Bij hoge snelheden kan de regelaar niet vermijden dat de auto tegen de zijkanten botst. We kiezen daarom een referentiesnelheid van 1.0 m/s.

Het visiesysteem werkt aan een vaste frequentie van 50 Hz. Dit komt bij een snelheid van 1.0 m/s overeen met een bemonstering in de ruimte van $s_s = 0.02$ m in de ruimte. Omdat we dezelfde predictiehorizon willen behouden als in simulatie (lengte 1.0 m), delen we de horizon op in meer

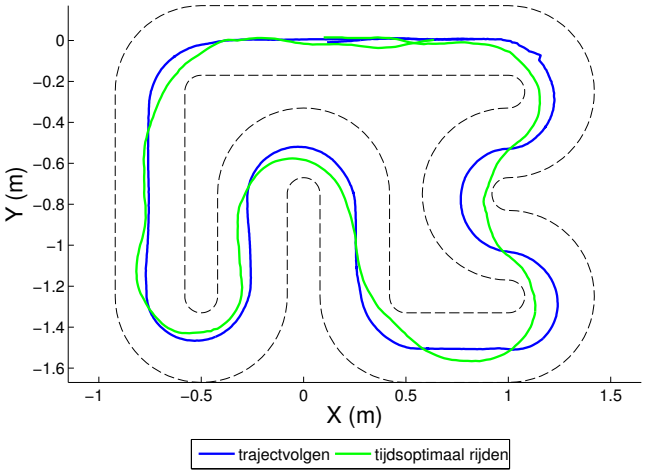


Fig. 8. Vergelijking tussen traject bij trajectvolgen en tijdsoptimaal rijden uitgevoerd op de experimentele proefopstelling.

controle-intervallen ($N = 50$). Merk op dat dit op de rand is van wat computationeel mogelijk is binnen de gegeven bemonsteringstijd: de mediaan van de berekeningstijden ligt op 0.01 s, met uitschieters tot 0.06 s. In dit geval betekent dat drie metingen verloren gaan terwijl de volgende controle-actie berekend wordt.

De resultaten van tijdsoptimaal rijden op de proefopstelling wordt vergeleken met experimentele resultaten van trajectvolgen in Fig. 8 en Fig. 9. Merk op dat bij tijdsoptimaal rijden de auto de bochten afsnijdt, het duidelijkste voorbeeld hiervan is in de chicane (rechterkant van Fig. 8). De totale eindtijd van de tijdsoptimale methode is ook lager dan die van trajectvolgen; de snelheid ligt op de meeste plaatsen langs het parcours ook hoger bij tijdsoptimaal rijden. Wel is de stuurhoek bij trajectvolgen minder nerveus. De resulterende tijd voor één ronde langsheen het parcours is 9.1 s voor trajectvolgen en 8.4 s voor tijdsoptimaal rijden.

VII. Conclusie

In dit artikel werd een tijdsoptimale niet-lineaire MPC regelaar voorgesteld voor autonome voertuigen. Bovendien werd de real-time haalbaarheid aangetoond op experimentele

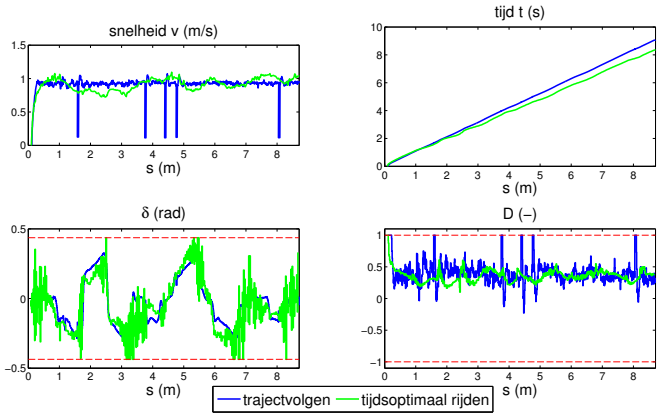


Fig. 9. Vergelijking tussen de toestanden bij trajectvolgen en tijdsoptimaal rijden in experiment.

wijze op een proefopstelling op schaal. We gebruikten een niet-lineair tweewielig voertuigmodel in spatiale coördinaten, die een eenvoudige tijdsoptimale formulering toeliet. Op deze manier zijn we in staat het tijdsoptimaal probleem te formuleren met een kleinste-kwadraten doelfunctie. De experimenten tonen aan dat deze aanpak adequaat werkt in real-time, hoewel de afwezigheid van slip in ons model een onvoldoende hoge snelheid toelaat. Verder onderzoek omvat dan ook de overgang naar meer geavanceerde voertuigmodellen, alsook het gebruik van spaarse QP solvers die een langere predictiehorizon mogelijk maken [7].

Appendix A: Technische details over de experimentele setup

[RGB camera:] xiQ MQ013CG-ON Camera
Imaging: 1280x1024 (1.3 MPixel) at 50 FPS
Connection with computer: USB 3.0
[Computer:] Real-time Debian system
Intel Core i3-3220 @ 3.3 GHz
OS: Debian 7.0 'Wheezy'
Kernel: Linux 3.8.13 with Preempt RT Patch
Bluetooth connection: Asynchrounous Connection-Less Bluetooth (ACL) communication link
[Race Car:] Kyosho dNaNo FX-101 ASF2.4GHz System

Referenties

[1] ACADO Toolkit. `http://www.acadotoolkit.org/`, 2009–2013. [Online].

[2] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. Race driver model. *Computers & Structures*, 2008.

[3] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[4] Paolo Falcone, Francesco Borrelli, H Eric Tseng, Jahan Asgari, and Davor Hrovat. Integrated braking and steering model predictive control approach in autonomous vehicles. In *Advances in Automotive Control*, volume 5, pages 273–278, 2007.

[5] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[6] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An Auto-generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles. In *Proceedings of the European Control Conference*, 2013.

[7] J. V. Frasch, M. Vukov, H.J. Ferreau, and M. Diehl. A new Quadratic Programming Strategy for Efficient Sparsity Exploitation in SQP-based Nonlinear MPC and MHE. In *Proceedings of the 19th IFAC World Congress*, 2013. accepted for publication.

[8] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME 2010 Dynamic Systems and Control Conference*, pages 265–272. American Society of Mechanical Engineers, 2010.

[9] A. Gray, Y. Gao, T. Lin, J.K. Hedrick, E. Tseng, and F. Borrelli. Predictive control for agile semi-autonomous ground vehicles using motion primitves. *Proceedings American Control Conference*, 2012.

[10] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011.

[11] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager. Optimal control of formula 1 race cars in a VDrift based virtual environment. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proceedings of the 18th IFAC World Congress*, pages 11907–11912, 2011.

[12] K. Popp and W.O. Schiehlen. *Ground Vehicle Dynamics*. Springer, 2010.

[13] A. Rucco, G. Notarstefano, and J. Hauser. Computing minimum lap-time trajectories for a single-track car with load transfer. In *51st IEEE Conference on Decision and Control*, 2012.

[14] SARTRE. The sartre project. `http://www.sartre-project.eu/en`.

[15] P. Spengler and C. Gammeter. Modeling of 1:43 scale race cars. Master's thesis, ETH Zürich, 2010.

[16] Toyota. Intelligent parking assist (ipa) supports parking maneuvers. `http://www.toyota-global.com/innovation/safety_technology/safety_technology/parking/`.

[17] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-Optimal Path Tracking for Robots: a Convex Optimization Approach. *IEEE Transactions on Automatic Control*, 54:2318–2327, 2009.

[18] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. Model Predictive Control of Autonomous Vehicles. In *Proceedings of the Workshop on Optimization and Optimal Control of Automotive Systems*. 2014.

Appendix C

Poster

Below, the master thesis poster is included.

KU LEUVEN

LMS®
A Siemens Business

Time-optimal race car driving

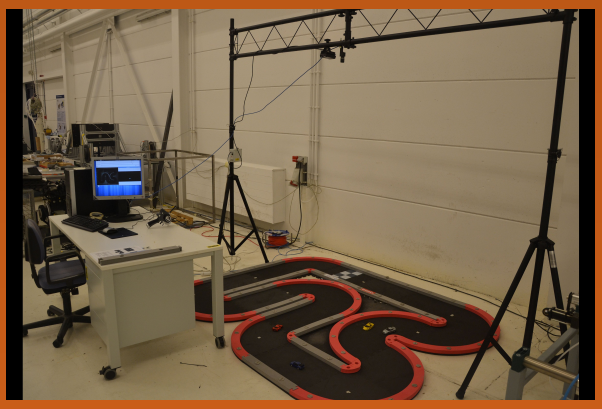
Motivation

Advanced Driver Assistance Systems (ADAS) and full autonomous driving are a reality in passenger vehicles (cf. Volvo, Toyota, Google).

Properties:

- Constrained system
- Highly nonlinear
- Fast dynamics

→ need for advanced control technique: Model Predictive Control



Goals

- Time-optimal MPC
- In real-time
- On real-world setup

Technology

Model:

- Nonlinear dynamics
- 6 DOF physical model
- 2 inputs: throttle and steering

Experimental setup:

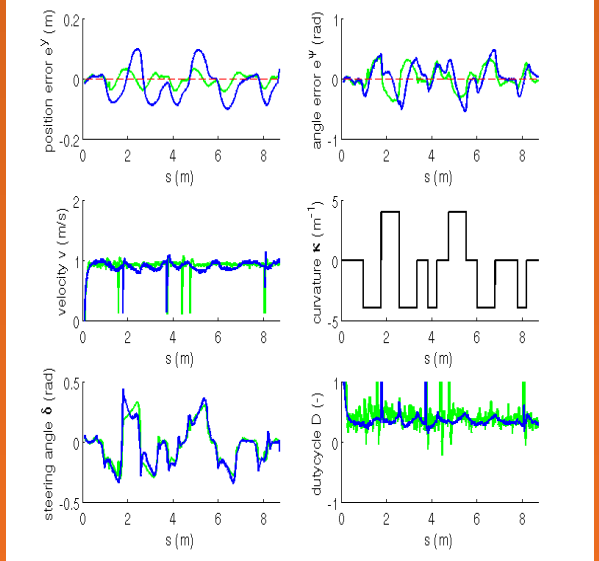
- 1:43 miniature race cars
- 8.7 m car track
- RT infrared vision system

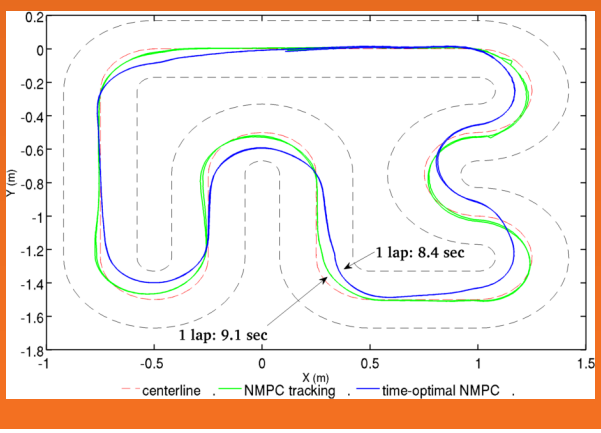
Software: ACADO (Matlab/C++)

Results

Three algorithms have been developed:

- Two-level linear MPC (trajectory planning + tracking)
- Nonlinear tracking MPC
- Nonlinear time-optimal MPC with spatial reformulation





Keywords

- Automotive control
- Real-time optimization
- Nonlinear predictive control

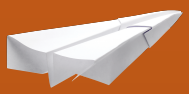
Master Wiskundige Ingenieurs-technieken

Masterproef
Robin Verschueren

Promotor
Moritz Diehl

Begeleiders
Stijn De Bruyne
Janick Frasch
Mario Zanon

Academiejaar 2013-2014



Bibliography

[1] qpOASES. http://www.qpOASES.org, 2007–2011. [Online].

[2] ACADO Toolkit. http://www.acadotoolkit.org/, 2009–2013. [Online].

[3] T. Au, M. Quinlan, and P. Stone. Setpoint scheduling for autonomous vehicle controllers. 2012.

[4] AutoNOMOS Labs, Freie Universität Berlin. http://autonomos.inf.fu-berlin.de, 2014.

[5] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 242–247. Pergamon Press, 1984.

[6] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, and Jahan Asgari. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2):265–291, 2005.

[7] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. Race driver model. *Computers & Structures*, 2008.

[8] A. Broggi, M. Bertozzi, A. Fascioli, C. Lo Bianco, and A. Piazzi. The argo autonomous vehicle's vision and control systems. *International Journal of Intelligent Control and Systems*, 3(4):409–441, 1999.

[9] A. Cohen. Will self-driving cars change the rules of the road?, 2013.

[10] Stijn De Bruyne. *Model-based control of mechatronic systems - Bridging between advanced methods and industrial applications.* PhD thesis, KU Leuven, Leuven, Belgium, December 2013.

[11] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[12] M. Diehl, H. J. Ferreau, and N. Haverbeke. *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, chapter Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pages 391–417. Springer, 2009.

[13] M. Diehl, I. Uslu, R. Findeisen, S. Schwarzkopf, F. Allgöwer, H.G. Bock, T. Bürner, E.D. Gilles, A. Kienle, J.P. Schlöder, and E. Stein. Real-Time Optimization for Large Scale Processes: Nonlinear Model Predictive Control of a High Purity Distillation Column. In M. Grötschel, S. O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 363–384. Springer, 2001. download at: http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-01-16.html.

[14] A. Domahidi, A. Zgraggen, M.N. Zeilinger, M. Morari, and C.N. Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *IEEE Conference on Decision and Control (CDC)*, pages 668 – 674, Maui, HI, USA, December 2012.

[15] P. Falcone, F. Borrelli, J. Asgari, H.E. Tseng, and D. Hrovat. Low complexity MPC schemes for integrated vehicle dynamics control problems. *9^{th} International Symposium on Advanced Vehicle Control*, 2008.

[16] H. J. Ferreau, P. Ortner, P. Langthaler, L. del Re, and M. Diehl. Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control*, 31(2):293–301, 2007.

[17] H.J. Ferreau, H.G. Bock, and M. Diehl. An Online Active Set Strategy for Fast Parametric Quadratic Programming in MPC Applications. In *Proceedings of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, Grenoble*, pages 21–30, 2006.

[18] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An Auto-generated Nonlinear MPC Algorithm for Real-Time Obstacle Avoidance of Ground Vehicles. In *Proceedings of the European Control Conference*, 2013.

[19] J. V. Frasch, M. Vukov, H.J. Ferreau, and M. Diehl. A new Quadratic Programming Strategy for Efficient Sparsity Exploitation in SQP-based Nonlinear MPC and MHE. In *Proceedings of the 19th IFAC World Congress*, 2013. accepted for publication.

[20] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J.K. Hedrick, and F. Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[21] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME 2010 Dynamic Systems and Control Conference*, pages 265–272. American Society of Mechanical Engineers, 2010.

[22] G. Genta. *Motor vehicle dynamics: modeling and simulation*. World Scientific, 1997.

[23] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011.

[24] B. Houska, F. Logist, J. Van Impe, and M. Diehl. Approximate robust optimization of time-periodic stationary states with application to biochemical processes. In *Proceedings of the 28th Conference on Decision and Control*, pages 6280–6285, Shanghai, China, 2009.

[25] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[26] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager. Optimal control of formula 1 race cars in a VDrift based virtual environment. In S. Bittanti, A. Cenedese, and S. Zampieri, editors, *Proceedings of the 18th IFAC World Congress*, pages 11907–11912, 2011.

[27] K. Kuchera. Autonomous racing using model predictive contouring control. Master's thesis, KU Leuven, 2013.

[28] R. Marino, S. Scalzi, and M. Netto. Nested pid steering control for lane keeping in autonomous vehicles. *Control Engineering Practice*, 19:1459–1467, 2011.

[29] J. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro. Lane-Change Fuzzy Control in Autonomous Vehicles for the Overtaking Maneuver. *IEEE Transactions on Intelligent Trasportation Systems*, 9, 2008.

[30] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[31] ORCA. Orca racer homepage. `https://sites.google.com/site/orcaracer/home`.

[32] H. B. Pacejka. *Tyre and Vehicle Dynamics*. Elsevier, 2006.

[33] J. B. Rawlings, D. Angeli, and C. N. Bates. Fundamentals of economic model predictive control. In *51st IEEE Conference on Decision and Control*, 2012.

[34] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[35] A. Rucco, G. Notarstefano, and J. Hauser. Computing minimum lap-time trajectories for a single-track car with load transfer. In *51st IEEE Conference on Decision and Control*, 2012.

[36] P. Spengler and C. Gammeter. Modeling of 1:43 scale race cars. Master's thesis, ETH Zürich, 2010.

[37] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl. Time-Optimal Path Tracking for Robots: a Convex Optimization Approach. *IEEE Transactions on Automatic Control*, 54:2318–2327, 2009.

[38] M. Vukov, W. Van Loock, B. Houska, H.J. Ferreau, J. Swevers, and M. Diehl. Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In *The 2012 American Control Conference, Montreal, Canada.*, 2012.

[39] World Health Organization. Global status report on road safety 2013. `http://www.who.int/violence_injury_prevention/road_safety_status/2013/en/`, 2013.

[40] L. Wunderli. Mpc based trajectory tracking for 1:43 scale race cars. Master's thesis, ETH Zürich, 2011.

[41] M. Zanon, J. Frasch, and M. Diehl. Nonlinear Moving Horizon Estimation for Combined State and Friction Coefficient Estimation in Autonomous Driving. In *Proceedings of the European Control Conference*, 2013.

[42] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. Model Predictive Control of Autonomous Vehicles. In *Proceedings of the Workshop on Optimization and Optimal Control of Automotive Systems*. 2014.

Fiche masterproef

Student: Robin Verschueren

Titel: Design and implementation of a time-optimal controller for model race cars

Nederlandse titel: Ontwerp en implementatie van een tijdsoptimale regelaar voor modelracewagens

UDC: 51-7

Korte inhoud:

In deze thesis worden verschillende real-time regeltechnieken voor tijdsoptimaal autonoom rijden van modelracewagens ontwikkeld. De methodes worden geëvalueerd in simulatie, alsook getest op een experimentele proefopstelling op schaal, bij coöperatiepartner LMS, A Siemens Business. Vanwege de complexiteit van tijdsoptimaal autonoom rijden geavanceerde regelmethodieken gebruikt: zowel lineaire en niet-lineaire modelgebaseerde predictieve regelaars (model predictive control, MPC) worden onderzocht. In een real-time context met bemonsteringstijden in de ordegrootte van milliseconden, is het een computationele uitdaging de controle-acties tijdelijk te berekenen. Er wordt daarom gebruik gemaakt van de ACADO Code Generation toolkit, die automatisch een hoogperformante implementatie van het real-time iteratie schema (RTI scheme) genereert, om het niet-lineaire real-time optimalisatieprobleem op te lossen. Gebruik makende van de ontwikkelde methoden, zijn we in staat om de rondetijd van de wagen drastisch te verlagen.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: wiskundige ingenieurstechnieken

Promotor: Prof. dr. M. Diehl

Assessoren: Prof. dr. ir. D. Huybrechs
Prof. dr. R. Vandebril

Begeleiders: Dr. ir. S. De Bruyne
J. Frasch
M. Zanon