Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Alexander Liniger

Autonomous Drift Control

Master Thesis

Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Nikolaos Kariotoglou
Prof. Dr. John Lygeros

December 2012

Contents

Abstract

In this Master Thesis a non-linear car model which can represent drift is identified and successfully used to control a miniature R/C car within and beyond its handling limit. Two different controllers are implemented, a PID path tracking controller as well as a predictive controller augmented with an online path planning. Both controllers are very fast and can reduce the lap time by more than 20% compared to all existing controllers.

Nomenclature

Symbols

α	Slip angle	[-]
β	Side slip angle	[deg]
v_x	Longitudinal velocity	[m/s]
v_y	Lateral velocity	[m/s]
$\dot{\varphi}$	Yaw rate	[rad/s]
δ	Steering angle	[rad]
D	Duty cycle	[-]

Indicies

x	Longitudinal direction
y	Lateral direction
f	Front wheel
r	Rear wheel
k	Time step k

Acronyms and Abbreviations

MPC	Model Predictive Controller
MPCC	Model Predictive Contouring Controller
RHC	Receding Horizon Controller
PID	Proportional Integral Derivative Controller
QP	Quadratic Program
NLP	Non Linear Program
ORCA	Optimal RC Autonomous Racing
DA	Digital Analog Converter
CG	Center of Gravity
PWA	Piece Wise Affine
CAD	Computer Added Design
LQR	Linear Quadratic Regulator
ZOH	Zero Order Hold

LMI	Linear Matrix Inequality
ODE	Ordinary Differential Equation
ETH	Eidgenössische Technische Hochschule

Chapter 1

Introduction

This thesis is part of the Optimal RC Autonomous Racing (ORCA) project, which has to goal to investigate high speed, online control of 1:43 remote controlled (R/C) cars. The project does have a working control test bed, which allows control the cars at real time.

1.1 Goals

Previous projects showed that if the controller is pushing the car to the limits of it handling abilities the car starts drifting. This leads often to the failure of the controller, because the controller cannot react correctly which leads to the spinning of the car. The idea in order to solve this problem is to investigate drift control, where a controller is designed which can steer the car also at these extreme conditions.
For this it is necessary to model the behavior of a drifting car, as well as to detect the car drifting. If this is possible the automatic drift controller can be used to steer the car if drift is detected.
This idea can be interpreted differently, so it is by itself an interesting question to design a controller which can bring the car into drift and is again able to stop the car from drifting. This is still a problem only few people solved, in simulation, and to the authors best knowledge only one publication shows a drift controller on a real test bed (the Fast Toy Lab at EPFL also has a drift controller).
The goal of this thesis is to demonstrate how a drift controller can be used to control a car so as to be faster than non-drifting controllers. This is achieved by creating a driving cycle during which the car is forced to drive at high speeds, thus enforcing drifting. Only a drift controller has the capability to control the car under such driving conditions, without risking a crash.

1.2 Previous/Related Work

The relevant literature consist of work done in previous ORCA projects and work done by other research groups in the field of drift control.
In the ORCA project, all control done until now uses a no slip model, which uses an infinite grip assumption. Such models are behave very predictive and

are simple to control, however there are only valid as long as the car is not starts slipping. Based on this model different PID tracking controllers are designed, which can follow a given path as well as a velocity reference, [22]. The controller uses a pre calculated trajectory and velocity profile, which can a least curvature, minimal distance or combination of such trajectories, to drive around the track. Beside PID controllers a Model Predictive Controller (MPC) tracking controller is also implemented, [22], which uses an error model derived from the no slip model. This controller also needs a given trajectory and velocity profile. The main advantage is that border constraints can be included and that the controller has not have to be tuned such that the car does not hit the borders.
The last controller implemented is a special form of an MPC, the so called Model Predictive Contouring Controller (MPCC). The MPCC's objective function is formulated so the controller can be used to follow a trajectory. However it follows the trajectory as fast as possible, and can, if needed, deviate from the reference. This allows the controller to plan its trajectory and velocity online, based on a simple reference trajectory as the center line. The implementation of this controller is still in progress and can be found in several previous ORCA projects, as for example [18]
However, none of the controllers are designed to drift and none of them can handle a drifting car. Therefore, all information for drift modeling and control is found in literature. A previous ORCA project on drift modeling, used different model and identification assumptions, such that the model could not be used as a basis for the given task.
In literature only few publications can be found on the topic of drift control. The basic models to describe cars and tires which can represent drift exist since a long time, but haven not been used to control a drifting car. The most probable reason for this is, that most researchers and car manufacturers work on control techniques which prevent drifting, as for example Electronic Stability Programs (ESP).
The few research results in the topic of drift control can be separated into three topics. In the first kind of papers the model itself is investigated and data from drift maneuvers generated by skilled drivers are analyzed, [3], [10] and [12]. All of these papers conclude that a combination of steering and motor inputs are necessary to stabilize drifting. Edelmann et. al [12], also did a model analysis of the resulting model, and showed that the stationary motion corresponding to drifting is unstable, thus the inputs are also necessary from a model point of view. Further non linear model discussion can be found in [16] and [7] where mainly the bifurcation analysis of the model is of interest. This shows that the car model has a bifurcation, with the steering angle as its bifurcation parameter.
The second research topic of interest is that of optimal trajectories, which are generally generated using non linear optimization routines. Velenis and Tsiotras [20], investigated the time optimal solutions as well as maximum exit velocity trajectories for a 90° curve for different road conditions and car conditions. By forcing the final orientation at the end of the curve, they were able to generate trajectories including drift. In [21] and [11], they were able to generate other rally typical maneuvers such as tail break and pendulum turns, using a similar non linear optimization set up. Also here the key point was to force the car to have the same orientation as the track at the end of the curve.
Jeon et. al [13] used a Rapidly-exploring Random Tree algorithm to generate time optimal maneuvers on loss surface and were also able to generate tail break

behavior.
However, the previous results for the optimal optimal trajectories are only derived from algorithms tested in simulations, and the optimization problems are not real time compatible.
All of the resulting trajectories using non linear optimization tools result in instationary trajectories, which is the main difference to the last kind of papers. These papers analyze steady state cornering equilibrium, where the main goal is to find a drifting equilibrium point in the model and design a controller which can maintain the drift. This is done for different drive train configurations, for example forward or rear wheel drive [19], [9], where both the steering and the motors are used to control the car. Other control concepts are also used, where only the motor torques are used to control the drift [8]. These papers therefore only design a drift controller which can maintain the drift in simulation.
The only paper, known to the author, presenting a working drift controller on a real testbed is [4]. The work described in this paper covers the identification of the car, as well as an in depth non linear model analysis. Finally a drift controller only using the steering input is designed, which can maintain the drift in simulation and in reality.
However, there are still a lot of open questions, mainly how can a car be driven around a track using a drift controller if it is necessary or wanted. In this thesis mainly this question is being answered, using the work done by Voser et. al [4] as an inspiration for the model building and control strategy.
In this report first an introduction to the existing hardware and the testbed is given. In a next step the used model is derived and identified. The non linear identified model of the car is used for a basic non linear model analysis, which gives the base for the next developed controller.
Two different controllers are derived and explained in the following chapters, first a PID path tracking controller, and secondly a receding horizon drift controller. During the work on the PID controller some changes on the existing MPCC simulation environment were necessary to generate reference trajectories. The work on the MPCC controller is also documented.

Chapter 2

Hardware

2.1 Existing Testbed

The testbed of the ORCA project has been build in previous projects and here only the basic set up is explained and the major problems are highlighted.

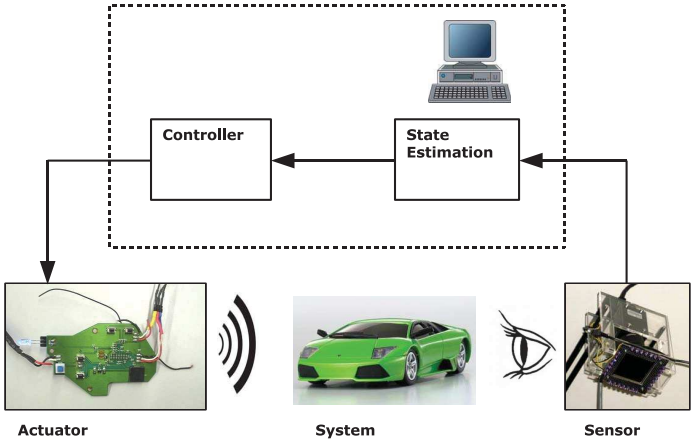

Figure 2.1: Existing control loop of the ORCA project, [22]

The control loop consists of different parts, see Figure 2.1, The used R/C car, are 1:43 scale models, which are real wheel driven by a DC motor, and have front wheel steering. The controls of the car are sent using a 2.4 GHz remote control, which is wired to a Digital Analog (DA) converter. This DA converter is emulating the real potentiometers of the original remote control. This allows to send calculated control inputs to the car.
The second part is a desktop computer which is used as controller. The controller computer uses Windows Real Time Target, which allows to use Matlab Simulink Real Time Toolbox. Using this toolbox it is possible to rapid prototype a controller out of a simple Simulink file, by automatically converting it to C code. The computer is able to run computationally expensive algorithms, and the Simulink interface allows easy generation and modification of controllers.
The controller needs a feedback from the car, which is given by two high speed

infrared cameras, the cameras are able to run up to 200 Hz. The cars are highlighted using reflective markers such that the cameras can easily see the reflecting infrared light, which is emitted by LEDs around the camera. The raw vision data is filtered using a Kalman filter, which estimates the velocities of the car.

2.2 Problems

The system as presented in the previous section has some problems. The used camera work generally well, however in the overlapping area, where both cameras see the car, the measurement is problematic. Because the position measured by the two cameras is not the same, the Kalman filter has a problem to estimate the correct position and velocity. This leads to ripples in the measured position and to jumps in the velocities.
The Kalman filter also does not estimate the velocities in direction of the car but in an inertial X-Y plane. Thus the velocities have to be further transformed, to get the wanted velocities the car orientation, which can add further inaccuracies.
Lastly, the camera returns the point at the center of the used pattern, which is in general not the center of gravity. On the used car the measured position is the center of gravity, but for new pattern car this has to kept in mind.
The biggest weakness of this testbed is the interaction between the control PC, the remote controller and the car. The first problem is that the physical controls generated using the controller, need to be transformed such that they approximately correspond to the values of the emulated potentiometers. This is already critical and leads to an unknown change of the control inputs. It is also problematic for the identification, because the values of the steering, for example, are not known exactly. The second problem is the communication from the remote control to the car, which has an unknown sampling time, and there is no information about whether the communication is successful. There is also no feedback from the car, which could be very useful for an identification, if the real measured steering angle could be logged for the identification and not only the approximate reference value of the controller.

Chapter 3

Drift Model

To successfully control the car while drifting, a mathematical model which can represent the dNano car in all driving situations has to be derived. The derivation of this model is separated into multiple steps. First the generic car model which can represent drift is derived. The derivation has two main parts, the mechanical model and the tire model. In a next step the model is identified such that it represents the used dNano car, and lastly the resulting model is analyzed.

3.1 Mechanical Model

The mechanical model of a car can be derived by simplifying the real car to one rigid body. Using this approximation most of the dynamics of a real car can be captured and only the torsion of the car is neglected. The dynamics of the rigid body can be separated into in-plane and out-of-plane motion. The in-plane motion describes the movement of the car on a flat surface (x and y direction as well as the yaw angle) and the out-of-plane motions are the movement in z direction as well as the pitch and roll angles.
For drift and general motion control the in-plane motion is of interest. However the out-of-plane motion can also be of interest because it can influence the in-plane motion by changing the normal loads of the tires and by this the friction force between the tire and the ground.
Nevertheless, the in-plane motion can be formulated without considering the out-of-plane motion, because the normal load changes influence the tire model and not directly the in-plane dynamics.

3.1.1 In-Plane Motion

The in-plane motion describes the rigid body moving on a flat surface, where the wheels are described using general forces. This forces generally have a lateral and a longitudinal component, relative to the wheel position.
The used dNano cars only have front wheel steering and only the back wheels are actuated in longitudinal direction. Thus the front wheels have a steering angle input which changes the orientation of the wheel relative to the car, but there is no longitudinal force component. Or in other words the wheels are

assumed to roll freely. The rear wheels on the other hand always have the same orientation as the car. The actuation of the rear wheels by the motor is modeled using a longitudinal force on each rear wheel.
The resulting model can be seen in Figure 3.1, and the analysis of such a model can be found in [12].

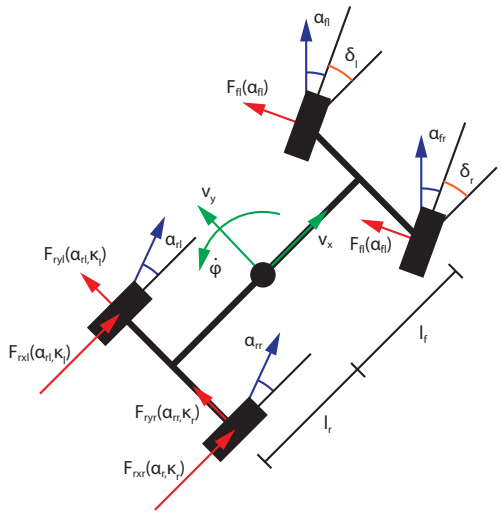

Figure 3.1: Schematic four wheel model

The mechanical model can be further simplified by the assumption that the car is symmetric, and thus the right and the left part of the car are lumped together. Such that the resulting "bicycle model" or "singe track model" just has one front and one rear wheel, see Figure 3.2. The assumption is valid if the velocities at the inside and the outside wheel are approximately the same and if the roll dynamics of the car are negligible.
The first assumption only depends o the geometry of the car, more specifically the width and the length of the car, which determined how important the indicated moments are. The second assumption depends on the suspension and the height of the center of gravity (CG) and will be discussed in more detail in the next section. For the used cars both assumptions are valid because the car is symmetric, has standard relations between width and length and the car has a low CG and a firm suspension.
The motion of the bicycle is described in a body fixed frame which is placed in the CG of the car. The frame is rotated with a angle of φ compared to the fixed initial frame. By using the body fixed frame the dynamics of the car is described by the longitudinal and the lateral velocity as well as the yaw rate of the CG.
The equation of motion can be derived using Newton's and Euler's law, where the fact that the rigid body is described in a moving coordinate system has to be considered by using Euler derivation. This leads to the following accelerations

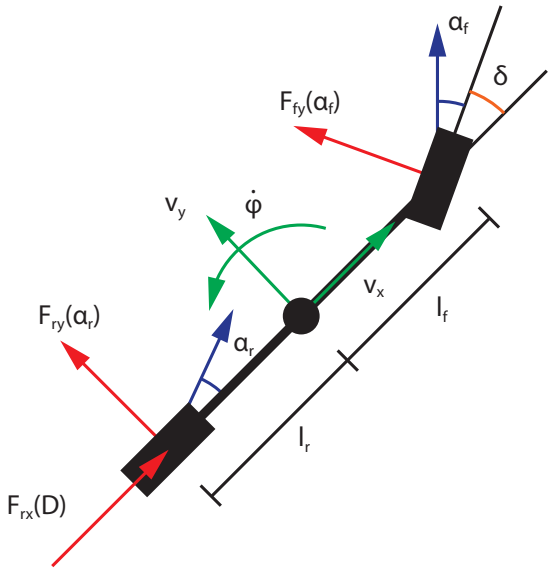

Figure 3.2: Schematic bicycle model

of the CG in the body fixed frame

$$a_{CG,x} = \dot{v}_x - \dot{\varphi} v_y \tag{3.1}$$
$$a_{CG,y} = \dot{v}_y + \dot{\varphi} v_x \tag{3.2}$$

Thus the equations of motion are

$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y} \sin\delta + m v_y \dot{\varphi}) \tag{3.3}$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} - F_{f,x} \cos\delta - m v_x \dot{\varphi}) \tag{3.4}$$
$$\ddot{\varphi} = \frac{1}{\Theta_z}(F_{f,y} l_f \cos\delta - F_{r,y} l_r) \tag{3.5}$$

Where the indices x and y correspond to longitudinal and lateral direction respectively r and f corresponds to rear and front wheel. m is the mass of the car, Θ_z the moment of inertia around the z axis, l_r and l_f the distance from the CG to the front and back wheel and δ the steering angle.

The steering angle δ is one of the two control inputs. The second control input is related to the longitudinal driving force, $F_{r,x}$, acting on the rear wheel and depends on the chosen tire model.

The open question is still how the three forces $F_{r,x}$, $F_{r,y}$ and $F_{f,y}$ can be modeled, which will be answered in the section concerning the tire model.

The position and orientation of the car in the fixed initial frame can be described

by integrating the transformed velocities

$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi) \tag{3.6}$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi) \tag{3.7}$$
$$\dot{\varphi} = \dot{\varphi} \tag{3.8}$$

Where X and Y is the position in the initial frame and φ the orientation relative to this frame.
The in-plane motion is fully described by the last six differential equations, governing the three velocities, position and orientation of the car.

3.1.2 Out of Plane Motion

The out of plane motion is mainly of interest due to its influence on the normal load of the tires, which affect the friction forces of the in-plane model.
A general friction force is given by:

$$F_f = \mu_f F_N \tag{3.9}$$

Thus the friction force is described by the friction force curve μ_f multiplied with the normal load, F_N.
If the car is not moving this is not of interest, because the normal load is constant and can thus be included into the friction force curve to one friction law. On the other hand, if the car is moving this load can change based on the dynamics of the system.
In the following subsection the different possible models for such load changes are discussed. There are two possible ways to model the motion, first using dynamic models and secondly using algebraic models.

Dynamic Suspension Model

One out-of-plane motion which can be of interest is the suspension dynamics which is normally modeled as a dynamic spring damper system, this can be done in pitch and roll direction.
The roll suspension dynamics of the car are modeled as an inverted pendulum, attached to a spring damper system, rotating around a fix axis. The movement around the axis moves the CG of the car and so changes the normal load of each wheel. This model of roll can be used only when using a four-wheel model.
The suspension dynamics in pitch direction are modeled allowing the rigid body, which represents the car, to move up and down in z direction and to rotate around the CG. The body and the ground are connected using a spring damper element at the position of each wheel. The modeling of such a suspension can be found in [20].

Algebraic Load Change

The second approach to model the load changes uses an algebraic model and not a dynamic model. The approach uses a force and moment balance in pitch

direction. The force and moment balance leads to the following normal forces

$$F_{f,N} = \frac{l_r mg - hmg\mu_{r,x}}{(l_f + l_r) - h(\mu_{f,y}\sin(\delta) + \mu_{r,x}} \tag{3.10}$$

$$F_{r,N} = mg - F_{f,N} \tag{3.11}$$

where the normal forces affect the force acting on the bicycle model by

$$F_{i,j} = F_{i,N}\mu_{ij} \quad i = r, f \quad j = x, y \tag{3.12}$$

and h the height from the ground to the CG. The rest of the notations are used as in Equation 3.3.
Thus the normal normal loads depend on the control inputs δ and $\mu_{r,x}$, as well as on velocities, because the force laws $\mu_{i,j}$ are velocity dependent. This results in the fact that the normal loads can be influenced by the control inputs.

Discussion

The dynamic suspension model adds more information to the model and helps to more accurately describe the real car, but the pitch model alone adds four states to the model. This might be useful in a simulation, however for control it is problematic. Moreover, the suspension dynamics are not of interest for the used car, as the CG is very low and the suspension very stiff. The algebraic load changes, which would not lead to more states, but increasing the complexity of the model and additional even more non linearities are also neglected.
A model including algebraic load changes could be interesting for offline trajectory calculation, like in [11] and [19] where such models are successfully used in trajectory optimization problems.

3.1.3 Summary

In summary an in-plane dynamic bicycle model is sufficient for the car used in the project and helps keeping the needed states to a minimum. The out-of-plane effects can be neglected because the CG of the car is very low and the suspension very stiff.

3.2 Tire Models

The dominant part of the resulting car model is the tire force laws. The shape of the force law is a determining factor for whether and when the car drifts and is therefore the main focus of interest.
There are three types of models which are investigated in more depth. The first tire model is a no slip model which has been used until now for control in the ORCA project. This model is investigated to compare the new drift model with the existing model. The second model is a lateral slip model which can represent drift. However the model neglects the spinning of the wheels. The spinning, which is a relative velocity between the ground and the tire, affects the lateral force. To include the effect of spinning, a combined slip model has to be used, which is the third kind of model which is investigated.

3.2.1 No Slip Model

In the no slip model two assumptions have to be made to simplify the model

- $v_{r,y} = 0$
- $\delta \approx 0$

The first assumption sets the lateral velocity at the back wheel to zero, which is the non slip assumption of the tire. The small steering angle assumption is used to simplify the trigonometric functions. The first assumption is only valid for slow driving, because it assumes a non saturating friction force.
The assumptions can be used to simplify the model to a pure kinematics model (only position and orientation), which is of favor, because it leads to a simple model for control. The model directly describes the influence of the controls to the position and orientation, which helps in the control design.
Using the first assumption, the yaw rate can be simplified to

$$v_{r,y} = v_y - \dot{\varphi} l_r = 0 \tag{3.13}$$

$$\Rightarrow \dot{\varphi} = \frac{v_y}{l_r} \tag{3.14}$$

Using the small steering angle assumption the model can be further simplified

$$\frac{v_y}{v_x} = \frac{l_r}{L}\tan(\delta) \approx \frac{l_r}{L}\delta \quad \text{with: } L = l_r + l_f \tag{3.15}$$

$$\Rightarrow v_y = \frac{l_r}{L}\delta v_x \tag{3.16}$$

$$\Rightarrow \dot{\varphi} = \frac{v_x \delta}{L} \tag{3.17}$$

By moving the origin of the body fixed frame to the rear wheel hub, the lateral velocity becomes zero by the first assumption, and the position and orientation of the car are

$$\dot{X}_r = v_x \cos(\varphi) \tag{3.18}$$

$$\dot{Y}_r = v_x \sin(\varphi) \tag{3.19}$$

$$\dot{\varphi} = \frac{v_x \delta}{L} \tag{3.20}$$

The forward velocity is modeled separately using Newton's law, where the forces acting on the car are the force of the motor, rolling friction forces as well as the breaking of a steering angle

$$\dot{v}_x = (C_{m1} - C_{m2} v_x) D - C_d v_x^2 - C_r - C_\delta (v_x \delta)^2 \tag{3.21}$$

Where C_{m1} and C_{m2} are the model parameters of the DC motor of the car and D is the duty cycle which is the second control input. C_d and C_r are the parameters of the rolling friction and the drag. Lastly C_δ models the additional friction caused by the steering input.
A more in depth identification and discussion can be found in [17] and [22].
By its no slip assumption which leads to an infinite grip, this model is not suited to model drift. Thus the model is of no interest for drift control.

3.2.2 Slip Models

Instead of simplifying the model to get rid of the force laws, the idea in slip models is to use generally non linear functions which can represent the behavior of a real tire. A lot of different tire models have been derived over the years, in this thesis just the magic formula is discussed, which was first published in 1987 [6] and is discussed in more detail in [16] where other models can also be found.

Magic Formula

The magic formula is a semi-empirical steady state tire friction law. To calculate the forces acting on the model the magic formula uses the slip angle of each wheel. In this section first the slip angles of the given bicycle model are derived and then the magic formula and its possible simplification are described.
By the simplification that the car is a rigid body, the angular velocity is the same for every point on it. Thus the velocities at a point P can be easily calculated if the velocities at the CG are known

$$(\vec{v})_P = \vec{v}_{CG} + \vec{\omega} \times r_{P,CG} \tag{3.22}$$

This way the side slip angle α, which is the angle between the lateral and the longitudinal velocity at each wheel, and the normalized longitudinal relative velocity, κ, can be calculated.
For the rear wheel this is given by

$$\alpha_r = \arctan\left(\frac{\dot{\varphi} l_r - v_y}{\omega_r r_r}\right) \tag{3.23}$$

$$\kappa_r = \frac{\omega_r r_r - v_x}{\omega_r r_r} \tag{3.24}$$

Where ω_r is the wheel speed of the rear wheel and r_r the radius of the rear wheel. For the front wheel the wheel speed is not of interest because the wheel is assumed to roll freely. By this assumption also $\kappa_f = 0$ and the slip angle is given by:

$$\alpha_f = -\arctan\left(\frac{\dot{\varphi} l_f + v_y}{v_x}\right) + \delta \tag{3.25}$$

$$\tag{3.26}$$

The magic formula is a combination of trigonometric functions, which can represent the usual shape of tire friction curves, and it is given by

$$F_{i,y} = D\sin(C\arctan(B\alpha_i - E(B\alpha_i - arctan(B\alpha_i)))) \quad \text{with: i = r, f} \tag{3.27}$$

For the longitudinal force the magic formula is the same by just replacing α with κ.
The resulting curve can be seen in Figure 3.3, where the main characteristic of the tire model is visible. Around zero the force law has a linear region, which is then saturating. Before the saturation there can be a peak, as often seen in the

friction force law where the peak friction is higher than the saturated friction. In Coulomb friction the peak is called static friction and the saturation dynamic friction. The exact shape of the curve depends on the four parameters, B, C, D and E, which do influence the shape differently.

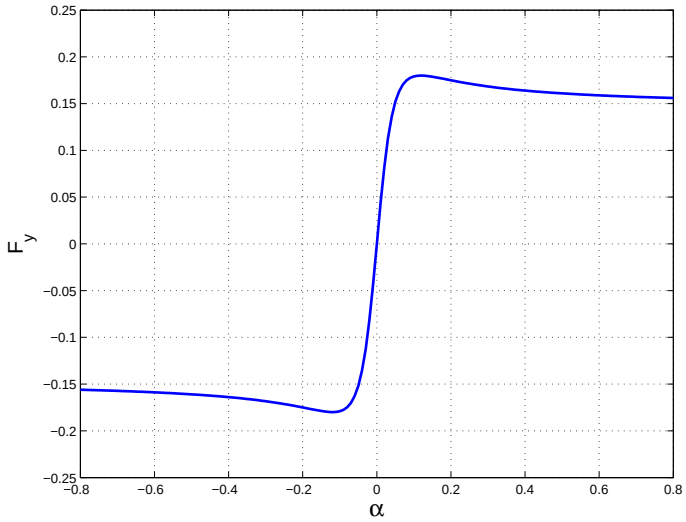

Figure 3.3: Magic Formula

- B is the stiffness factor
- D is the peak factor, the maximal force which can be transmitted
- BCD is the cornering stiffness, or in other words the gradient of the force law at zero, which characterizes the linear region of the function
- C and E are shape factors

The function is often simplified by $E = 0$ such that the magic formula is given by

$$F_{i,y} = D \sin(C \arctan(B\alpha_i)) \quad \text{with: i = r, f} \tag{3.28}$$

This simplification still represents the tire behavior well and a lot of non linearities are removed. The main difference is the steepness of the peak, where the complicated function allows more flexibility, compare Figure 3.4 and 3.3.
The simplification gets rid of some of the redundancy of the function, by having only one shape factor and, furthermore it reduces the number of parameters which have to be identified. This leads to a more easily identifiable model, which is important.
As discussed, there are two concepts how the magic formula can be used. The first uses combined slip, where the spinning of the wheel κ influences the lateral force. in the second only the lateral slip is modeled and the combined slip is ignored.

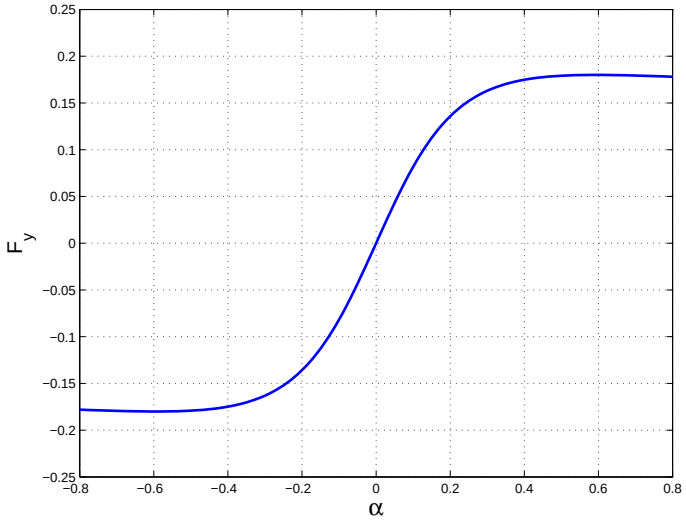

Figure 3.4: Simplified Magic Formula with $E = 0$

Combined Slip Model

There are several combined slip models which use the magic formula, and here two of them are explained. The first is based on the friction circle and the second uses an empirical formulation. The goal of both is to model how the lateral slip of a wheel influences the longitudinal slip and vice versa.
The friction circle links the lateral and the longitudinal force by assuming the resulting force moves on a circle or in a more general case on a ellipse. The resulting force can be calculated using the total slip s_r which is defined as

$$s_r = \sqrt{\alpha_r^2 + \kappa_r^2} \tag{3.29}$$

The resulting force is then calculated using the magic formula

$$F_r = D_r \sin(C_r \arctan(B_r s_r)) \tag{3.30}$$

which can be divided into the lateral and longitudinal force

$$F_{r,y} = k\frac{\alpha_r}{s_r} F_r \tag{3.31}$$

$$F_{r,x} = \frac{\kappa_r}{s_r} F_r \tag{3.32}$$

where $k \neq 1$ can be used, if the relationship between the longitudinal and lateral force is not moving on a circle but an ellipse.
This results in maps for the force law and the wanted interconnection between the lateral and longitudinal slip. This interconnection is important for drifting because a longitudinal slip lowers the lateral force, see Figure 3.5. Also if the car is slipping in lateral direction much less longitudinal force can be transmitted. In other words, the spinning of the wheels can be used to control the drift and to initialize the drift.
in this only one magic formula is used to describe the longitudinal and the lateral tire force. However, if the pure cornering ($\kappa_r = 0$, only lateral force) and the

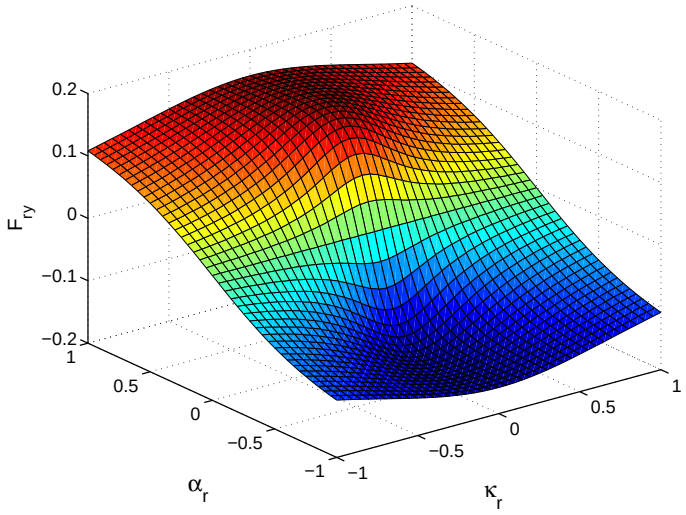

Figure 3.5: Friction force map, for a combined slip force law using the friction circle

pure acceleration ($\alpha_r = 0$, only longitudinal force) model, cannot be linked by the multiplicative parameter k, the model is not suited (This is easy to test by identifying the lateral as well as the longitudinal force laws independently). If the force laws can be linked with the parameter k this model is most probably suited and by its simplicity preferable.
However, if this is not the case the second approach to model the combined slip is more suited. The model is empirical and does not use a friction ellipse to model the interaction of the lateral and the longitudinal slip. This combined slip model uses a function very similar to the magic formula to capture the interaction, by weighting the pure slip values. The weighting function G is given as

$$G_x = D_{g,x} \cos(C_{g,x} \arctan(B_{g,x}\alpha)) \tag{3.33}$$
$$G_y = D_{g,y} \cos(C_{g,y} \arctan(B_{g,y}\kappa)) \tag{3.34}$$

Important to see, is that the weight factors for the lateral force using the longitudinal slip value and vice versa, which leads to the wanted interconnection between the lateral and the longitudinal slip, which is the characteristic of combined slip.
Multiplying the weights with the pure slip values leads to the wanted combined slip forces

$$F_{0,x} = D_{f,y} \sin(C_{f,y} \arctan(B_{f,y}\kappa)) \tag{3.35}$$
$$F_{0,y} = D_{f,y} \sin(C_{f,y} \arctan(B_{f,y}\alpha)) \tag{3.36}$$
$$F_x = G_x F_{0,x} \tag{3.37}$$
$$F_y = G_y F_{0,y} \tag{3.38}$$

This model needs more parameters which have to be identified, however it can

also represent more complicated combined slip relationships than the friction ellipse model.
For a longer discussion mainly about the second model refer to [16].
However, to efficiently use a combined slip model a wheel speed sensor is necessary, because the wheel speed information ω_r is essential.

Lateral Slip Model

The lateral slip model is only using pure cornering conditions. Under this assumption the lateral dynamics, v_y and $\dot{\varphi}$ are not affected by the spinning of the wheels. Such a model can be used without a wheel speed sensor by simplifying the slip angle at the rear wheel to

$$\alpha_r = \arctan(\frac{\dot{\varphi} l_r - v_y}{v_x}) \tag{3.39}$$

Thus the tire forces are given by

$$F_{f,y} = D_f \sin(C_f \arctan(B_f \alpha_f)) \tag{3.40}$$
$$F_{r,y} = D_r \sin(C_r \arctan(B_r \alpha_r)) \tag{3.41}$$

The force $F_{r,x}$ is modeled using, for example the same model as in the previous projects. Where $F_{r,x}$ represents the model of the motor and the rolling resistance.

$$F_{r,x} = m((C_{m1} - C_{m2} v_x) D - C_d v_x^2 - C_r) \tag{3.42}$$

The problem of the lateral slip model is that the spinning is completely neglected, but the combined slip is an important effect during drifting. Thus the same idea posted in [4] is used, which is to use identification techniques to capture as much of the combined slip with the lateral slip model as possible, by identifying the lateral slip model using maneuvers with significant combined slip.

Simplified Lateral Slip Models

Several simplifications of the lateral slip model are used, in this project. Firstly, the force laws are approximated using piecewise affine (PWA) functions such that for example the rear lateral force is given as

$$F_{r,y} = \begin{cases} C_{r,lin} \alpha_r & \text{if } |\alpha_r| < \text{threshold} \\ C_{r,sat} \alpha_r + \text{sign}(\alpha_r) D_{sat} & \text{else} \end{cases} \tag{3.43}$$

Where $C_{r,lin}$ is the cornering stiffness and the $C_{r,sat}\alpha_r + \text{sign}(\alpha_r) D_{sat}$ represents the saturated region, see Figure 3.6. Using this PWA approximation of the magic formula the model can be further simplified to a PWA model. Using the following assumptions:

- small steering angle
- small slip angles

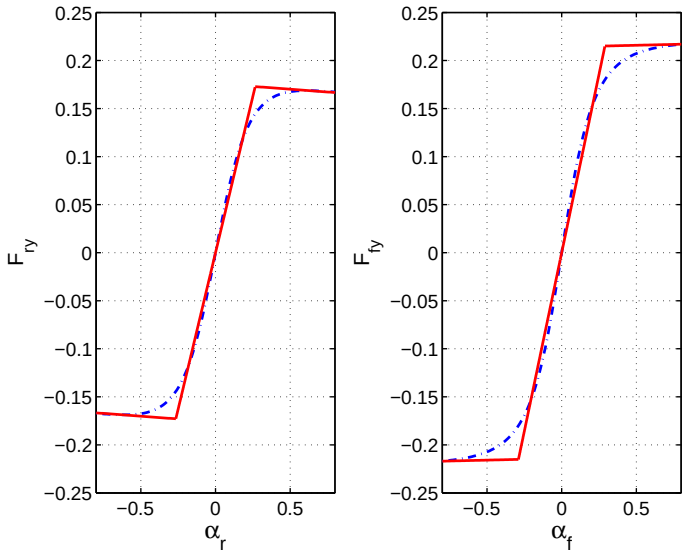

Figure 3.6: PWA approximation of the magic formula

The slip angles can be approximated as

$$\alpha_f = -\frac{\dot{\varphi} l_f + v_y}{v_x} + \delta \tag{3.44}$$

$$\alpha_r = \frac{\dot{\varphi} l_r - v_y}{v_x} \tag{3.45}$$

and the equation of motion of the bicycle model can be simplified to

$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y}\delta + m v_y \dot{\varphi}) \tag{3.46}$$

$$\dot{v}_y = \frac{1}{m}(F_{r,y} - F_{f,x} - m v_x \dot{\varphi}) \tag{3.47}$$

$$\ddot{\varphi} = \frac{1}{\Theta_z}(F_{f,y} l_f - F_{r,y} l_r) \tag{3.48}$$

For a constant longitudinal velocity $v_x = const$ the model is now PWA.

3.2.3 Summary

Several different ways to model the interaction of the car with the ground are possible. For the given task of drift control a combined slip formulation would be best fitted, this is however not possible, due to the lack of a wheel speed sensor. Thus a lateral slip model is used, which is able to represent the drifting. The combined slip is not neglected completely as combined slip effects are identified into the non linear force laws of the lateral model.
The final model falls under the category of simplified lateral slip models and is

given by

$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y}\sin\delta + mv_y\dot{\varphi}) \tag{3.49}$$

$$\dot{v}_y = \frac{1}{m}(F_{r,y} - F_{f,x}\cos\delta - mv_x\dot{\varphi}) \tag{3.50}$$

$$\ddot{\varphi} = \frac{1}{\Theta_z}(F_{f,y}l_f\cos\delta - F_{r,y}l_r) \tag{3.51}$$

with: (3.52)

$$F_{f,y} = D_f\sin(C_f\arctan(B_f\alpha_f)) \tag{3.53}$$

$$F_{r,y} = D_r\sin(C_r\arctan(B_r\alpha_r)) \tag{3.54}$$

$$\alpha_f = -\arctan(\frac{\dot{\varphi}l_f + v_y}{v_x}) + \delta \tag{3.55}$$

$$\alpha_r = \arctan(\frac{\dot{\varphi}l_r - v_y}{v_x}) \tag{3.56}$$

$$F_{r,x} = m((C_{m1} - C_{m2}v_x)D - C_dv_x^2 - C_r) \tag{3.57}$$

3.3 Identification

The chosen model needs to be identified to represent the dNano car on the given track material. As in the modeling part the identification can also be separated into the mechanical and the tire model.
The parameters of the mechanical model can be measured or calculated using a CAD program. The mass m and the distances l_f and l_r can be measured, and every one of the used cars has to be investigated separately. The moment of inertia can be calculated using the assumption that the mass of the car is uniformly spread in a cubic the size of the car.

$$\Theta_z = \frac{1}{12}m(w^2 + d^2) \tag{3.58}$$

Better results can be obtained by using a CAD program and using not just one body but multiple parts of the car. This is done using the four main parts of the car, the hull, the battery, the chassis with the front axle, and the rear axle with the motor. All parts are measured and built in a CAD program using the assumption that each of the parts has a uniformly spread mass. All the parts are then arranged correctly in an assembly and the resulting moment of inertia is calculated by the CAD program. The main advantage of using a CAD program is that now multiple parts of the car can be considered which can have an arbitrary shape. Thus heavy parts close to the CG do not effect the moment of inertia as strong as parts like the hull.
The difference between the two approximations is about 5%. The exact CAD model allows to set the moment of inertia fix, such that it does not have to be identified dynamically. Thus the dynamic identification is simpler and more robust.
The four parameters of the mechanical model are given in Table 3.2.
The six parameters of the tire model cannot be identified that simply. However, Voser et al. [4], use a systematic way to identify the tire model such that it is usable for drift control. The approach uses two steps: First steady state

Table 3.1: Moment of Inertia of the Lamborghini Murcielago

Moment of Inertia with one block	2.67 10^{-5} kgm^2
Mass hull	9.7 g
Mass chassis	17.3 g
Mass rear axle	7.5 g
Mass battery	5.5 g
Moment of inertia with CAD	2.78 10^{-5} kgm^2

Table 3.2: Mechanical model parameters for the Lamborghini Murcielago

Θ_z	2.78 10^{-5} kgm^2
m	40.1 g
l_f	0.029 m
l_f	0.033 m

experiments are used to identify the tire models and in a second step the rear wheel is further dynamically identified. A maneuver with significant combined slip is used, to identify this combined slip into the lateral model. The car used by [4], is very similar to the dNano cars, from a modeling point of view. The car uses rear wheel drive and does not use front wheel brakes, thus it is also assumed to have no combined slip at the front wheel.

Steady State Identification

The first identification step uses slow ramp steer maneuvers, where a slow ramp input is given to the steering angle with a slope of 0.1 rad/s, such that the steering angle does not introduce dynamic effects. Furthermore a PID controller is used to keep the forward velocity constant. Such experiment assume stationary conditions, so all accelerations are zero.

$$\dot{v}_x = 0 \tag{3.59}$$
$$\dot{v}_y = 0 \tag{3.60}$$
$$\ddot{\varphi} = 0 \tag{3.61}$$

Using there assumptions the equation of motion can be reformulated, such that the forces action in the model can be calculated as a function of the velocities

$$\dot{v}_y = \frac{1}{m}(F_{r,y} - F_{f,x}\cos(\delta) - mv_x\dot{\varphi}) = 0 \tag{3.62}$$
$$\ddot{\varphi} = \frac{1}{\Theta_z}(F_{f,y}l_f\cos(\delta) - F_{r,y}l_r) = 0 \tag{3.63}$$
$$\Rightarrow F_{f,y} = \frac{mv_x\dot{\varphi}l_r}{(l_r + l_f)\cos(\delta)} \tag{3.64}$$
$$\Rightarrow F_{r,y} = \frac{mv_x\dot{\varphi}l_f}{l_r + l_f} \tag{3.65}$$

The tire forces are by the model assumption a function of the slip angles, $F_{f,y} = f(\alpha_f)$ and $F_{r,y} = f(\alpha_r)$, thus the calculated forces using the steady-

state assumption can be plotted against the slip angle for each measurement point, see Figure 3.7. The velocities used to generate the plots are additionally filtered using a second order Butterworth filter with a cut off frequency of 30 Hz and a zero phase digital filtering algorithm. The additional filtering is used to further smooth the estimated velocities from the Kalman filter.

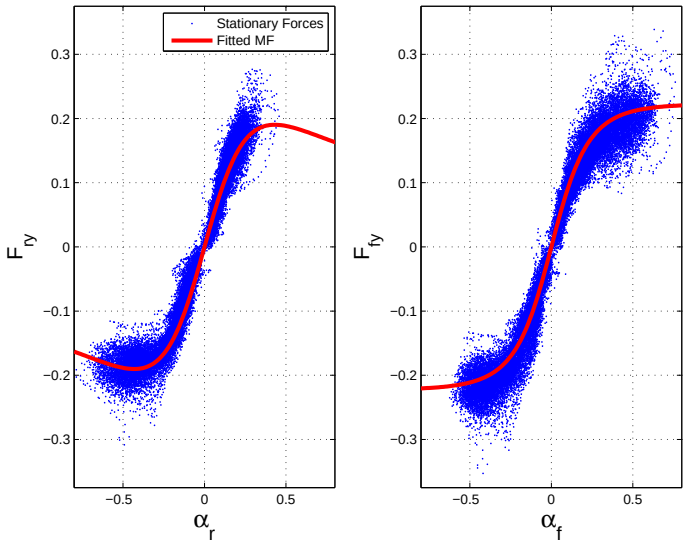

Figure 3.7: Force point cloud from steady state assumption and fitted magic formula

Using multiple experiments with different forward velocities, plots as shown in Figure 3.7 can be generated. This specific figure uses 31'000 measurement points and 31 different experiments. In the plot it is visible that the car has a slightly different behavior for right and left turns. In a left turn (positive slip angles) the car has an under-steer characteristic. Which leads to the characteristic that the rear tire does have no saturation, whereas the front tire completely saturates. In a right turn, the car has a neutral to over-steer characteristic where both tires saturate. This different behavior is most probably cause by the fact that the steering does not have the same maximal steering angle in both directions. Several causes can be excluded: The differential at the rear wheel does not influence the behavior, this is checked by testing the car with a looked and a normal differential, which does not change the behavior. The different characteristic can also be seen with all tested cars, thus it is a general dNano problem and not only one of the identified Lamborghini Murcielago.
Furthermore, all experiments are carried out using a locked rear wheel differential. This influences the driving by forcing both rear wheels to have the same wheel speed. Because of the different curve radius of the inner and the outer wheel, the two wheels should turn at different speeds. Locking of the differential introduces a longitudinal slip which makes the car react a bit slower. On the other hand the locking of the rear differential is favorable for drift because both wheels get the same amount of torque, whereas in an open differential if one wheel looses traction, so if the force law saturates, it has less resistance and thus gets more torque. The other wheel with grip on the other hand gets less torque,

which leads to the problem that the torque is not used any more to generate forward motion. In real cars the problem is solved using a so called limited slip differential, which only allows a certain difference of the wheel velocity between the right and the left wheel. To not deal with such effects the differential is locked during the identification.
The result of the slow ramp steer maneuvers is a force point cloud, which has the expected shape and can be well represented by the magic formula. The parameters of the magic formula for the rear and front wheel can be identified by fitting the non linear function using a least square problem to the stationary forces.

$$\min_{\beta} \left(\sum_{i=1}^{N} [y_i - f(x_i, \beta)]^2 \right) \tag{3.66}$$

y_i are the measured forces, x_i the measured inputs of f and β the to identified parameters of the magic formula. The optimization problem is solved using a Levenberg-Marquardt algorithm which solves the problem iteratively using an interpolation between the Gauss-Newton-Algorithm and the method of gradient descent to calculate each update step. This makes the optimization more robust than the Gauss-Newton-Algorithm. The fits obtained by the least squares can be seen in Figure 3.7.
The identification is sufficient for the front wheel, because the wheel is freely rolling and thus there is no combined slip and negligible dynamic effects. The back wheel on the other hand is driven and has so significant combined slip. Thus the identification preformed using the stationary assumption does not lead to sufficient parameters. The obtained parameters can only be used as an initial guess for the dynamic time domain identification, which is necessary in order to include the combined slip effects.
For the dynamic identification of the back wheel the parameters of the front wheel are fix to the value from the stationary identification and given in Table 3.3

Table 3.3: Front Wheel Magic Formula Parameter for the Lamborghini Murcielago

B_f	4.1
C_f	1.1
D_f	0.22

Time Domain Rear Wheel Identification

The identification f the back wheel under the steady state assumption is not sufficient due to the dynamic effects of the combined slip. Thus the dynamic response of the model is fitted with the measured velocities.
Slow ramp steer maneuvers are used again as an experiment to identify the combined slip effects. A high forward velocity is chosen, as it contributes to drifting. This can be seen in Figure 3.8, where the X-Y trajectory of such an identification maneuver is plotted. The car starts by driving a curve with

a closing radius, until the car starts slipping, which leads to the car spinning around. This can also be seen in the measured velocities, see Figure 3.9, which first increase linearly until the spinning starts and afterwards the velocities behave periodically. The periodic behavior can also be seen in the forward velocity, see Figure 3.10. This behavior is caused by the friction effects during drifting. The high friction, caused by drifting, slows down the car. This stops the drifting and the car can accelerate again, thus leading again to drifting. This behavior is clearly driven by combined slip.

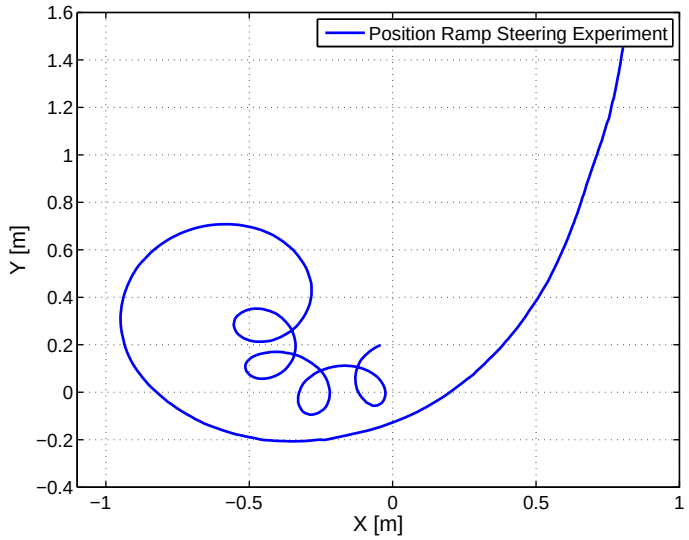

Figure 3.8: Trajectory of a ramp steer maneuver with drifting at the end

This experiment excites the dynamics which should be investigated, which are the large slip angles, see Figure 3.10, mainly at the back wheel and so suited to identify the combined slip effects. Furthermore, the steering angle is well known, which is generally a problem with the current remote control set up, where only the input to the remote control is known but not what the remote control and the car are doing with the control. The slow ramp steer input is due to its slow change no problem, compared to inputs necessary to maintain a drift, which contain fast changes.

The identification of the magic formula is done using a weighted least squares optimization, which is solved using a simplex search method, which does not use numerical or analytic gradients.

$$\min_{\beta} \left(\sum_{i=1}^{N} W[y_i - f(x_i, \beta)]^2 \right) \tag{3.67}$$

For the optimization the forward velocity and the steering angle are known and lateral model response is fitted with the measured lateral velocity and yaw rate. The error of the yaw rate and the lateral velocity are weighted differently to get

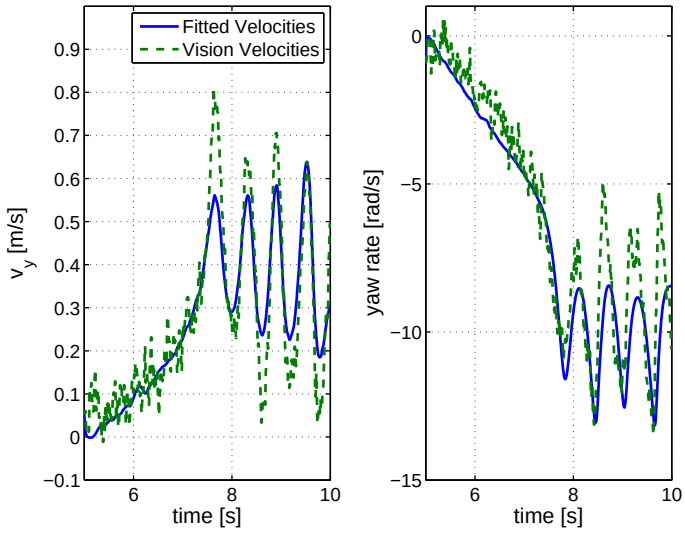

Figure 3.9: Measured velocities and fitted response of the model

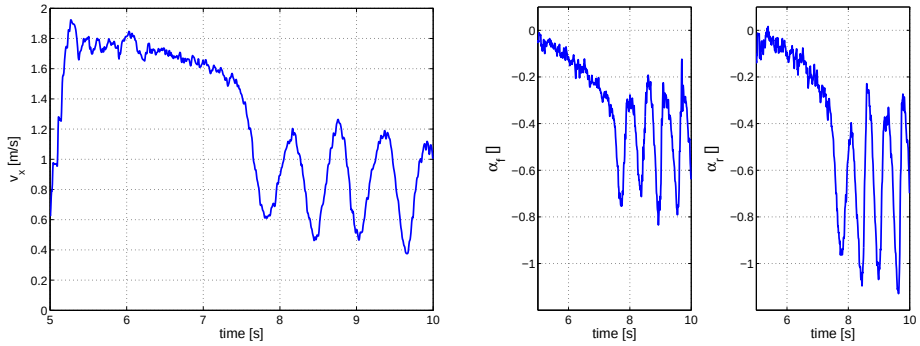

Figure 3.10: Left, velocity over a ramp steer maneuver and right the slip angle of the rear and front wheel

better identification results. The final objective function is given by

$$\min_{\beta}(\sum_{i=1}^{N}(W_{v_y}(v_{y,meas} - v_{y,model}(\beta, v_x, \delta))^2 + W_{\dot{\varphi}}(\dot{\varphi}_{meas} - \dot{\varphi}_{model}(\beta, v_x, \delta))^2) \tag{3.68}$$

where $W_{v_y} = 1000$ and $W_{\dot{\varphi}} = 1$. The weights are necessary to get sufficient fits over the whole experiment period, see Figure 3.11. To get better results, the velocities are filtered as in the steady state identification, however with a cut off frequency of 50 Hz. Furthermore, the first second of the measurements is not used for the identification, because the car is still accelerating and sometimes the camera does not see the car at the beginning.
A bigger problem for the identification lies in the magic formula itself, as the parameters of the formula do have some redundancies. The parameter C_r and B_r of Equation 3.28 both influence the shape in the linear region as well as

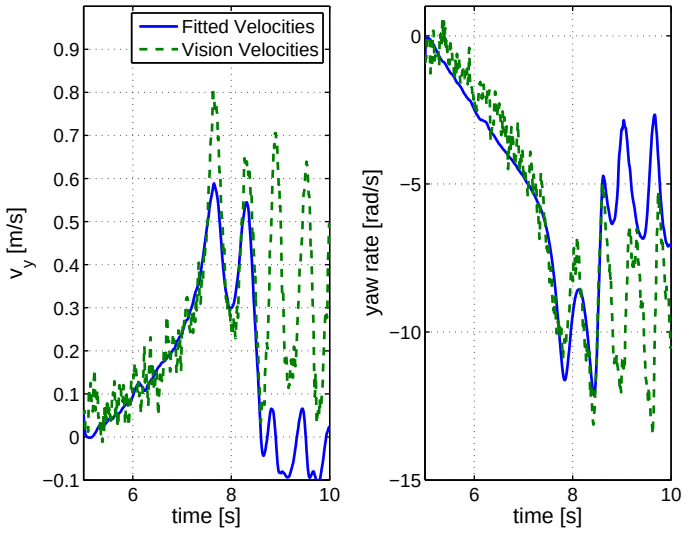

Figure 3.11: Measured velocities and fitted response of the model with not weighted least squares problem

the shape in the saturated region. D_r is only a multiplicative scaling factor and gives mainly the height of the peak. The redundancies lead to the problem that it is not possible to identify all three parameters at once. Thus different possible combinations are tested for which parameters can be best identified together while fixing the value of the other.
The best results are obtained by fixing C_r and optimizing for the other two. This has the advantage that the B_r can still shape the curve as necessary and that the peak force can be identified dynamically which is very important for the drifting. C_r is optimized by hand to get good fits for the model, which was the case for values around 1.3 - 1.4, where at the end the best results were obtained using $C_r = 1.4$.
The resulting fits can be seen in Figure 3.9, where the main importance is a good fit in the first normal driving linear region and a qualitative representation in the drifting region.
The identification is repeated for six different data sets with the results shown on Table 3.4. The six identification results are then averaged, see Table 3.5.

Table 3.4: Back Wheel Magic Formula Parameters for the Lamborghini Murcielago

B_r	D_r
3.812675484499625	0.166564193278013
3.935706799234405	0.163959277956922
3.814345269943725	0.165463233286617
3.900359625263297	0.161142204255270
3.766408063007261	0.166721831868024
3.936106061344378	0.162057321076370

Table 3.5: Averaged Back Wheel Magic Formula Parameters for the Lamborghini Murcielago

Parameter	mean	variance
B_r	3.8609	5.2 $10^{-}3$
C_r	1.4	-
D_r	0.1643	5.4921 10^{-6}

In the identification progress, averaged parameters are used, due to the problems obtained by cross validation. Normally identification results from one data set should be cross validated using another data set. However, this progress is not successful for the given problem, as seen in Figure 3.12, where the averaged parameters are used on one of the data sets. Even the cross validation using the identified parameters for one data set, and only deleting the last four digits, leads to an unsuccessful cross validation. However, the individual parameters of each data set are very similar, so the parameters are averaged and this parameters are used for the further model analysis and control synthesis.
There are more reasons suggesting that the averaged fits are sufficient. First the response of the model does have a good fit in the linear region which is very important, because this is the main driving region. Secondly it does follow the first peak, which indicates that the saturation height and position is correct. But the response of the model does not has the periodic behavior, instead it diverges, which is the main problem of the cross validation.

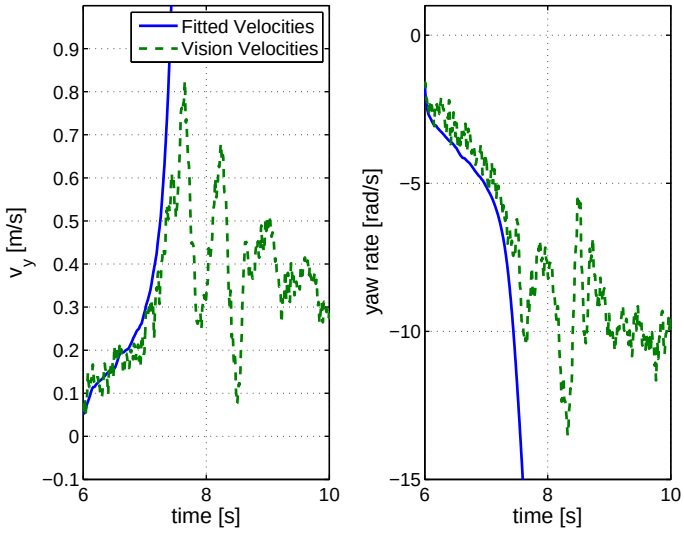

Figure 3.12: Cross validation of the model, by comparing measured velocities and response of the system

The most probable reason for the described problem is that when using the given experiment the steering angle can only increase, whereas in normal drifting counter steer would be used to control drifting. Therefore if the model diverges once from the real measurement there is no input which can bring the model

back.
Furthermore, is the cross validation is successful if just one velocity is simulated and the other one is used as an input to the model, see Figure 3.13. Which is also an indication that the model does give a reasonable accordance with the measurements.

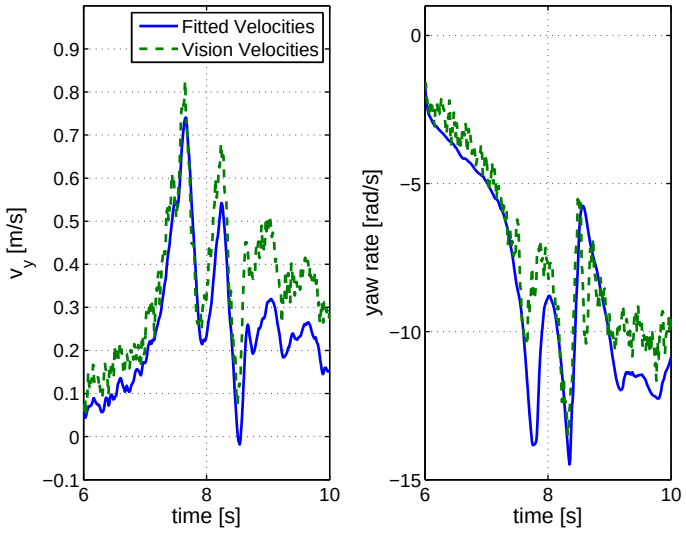

Figure 3.13: Cross validation of the individual model, by comparing measured velocities and response of the one velocity when the other is known

The main problem in the cross validation is that the simulated velocities diverge after the first peak and do not come back. In an attempt to solve the problem of the divergence of the model when drifting starts, artificial damping terms were introduced in the model. (It was expected that the damping terms would prevent the velocities from diverging to fare). However, the problem was not solved. Too little damping has not influence on the model and too strong damping always brings the system back to zero. Another approach to solve the cross validation problem is changing the C_r parameter of the magic formula so that the force law has a damping behavior in the saturation region. This approach, however, also does not yield the desirable results. Smaller C_r values have the same effect as artificial damping and instead of inducing the desired periodic behavior, the velocities decrease to zero.
Another issue is how the model changes over time when the quality of the tires changes. Driving and mainly the drifting wears the tires down. The biggest influence it has on the rear tires. The difference between new and old rear tires can also be seen in the system identification process. The results for new tires are given by another series of experiments, which leads to the results seen in Table 3.6.
The main difference between new and old rear wheel tires lies in the saturation force, as can be seen in Figure 3.14. As expected, the wear of the tires does not affect the behavior in the linear region, but mainly reduces grip.
A model with used tires is used for the model analysis and the control synthesis, as the tires are used when the car is racing and the qualitative identification

Table 3.6: Averaged Back Wheel Magic Formula Parameter for the Lamborghini Murcielago with new tires

Parameter	mean	variance
B_r	3.4177	$4.47\ 10^{-3}$
C_r	1.4	-
D_r	0.1643	$5.567\ 10^{-6}$

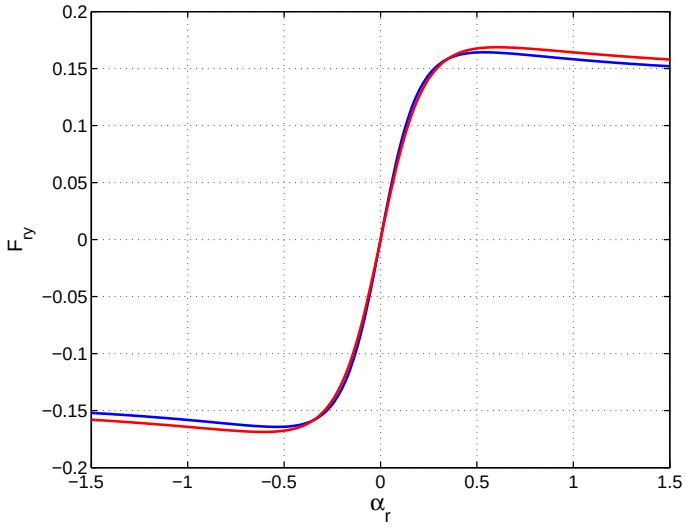

Figure 3.14: Difference in the shape of the magic formula between new (red) and old (blue) tires

results are better for the used tires.
This leads to a fully identified model which can represent all diving situations, and includes the saturation of the rear and the front tires.

3.4 Non Linear Model Analysis

To better understand the non linear model and the different driving conditions, for the subsequent control synthesis, the non linear model is analyzed. In this report just the stationary cornering conditions are investigated. In literature this is better known as steady-state cornering equilibria, however the model as presented in the previous section also consists of the position and orientation states, which are not in equilibrium if the car is in motion. An in depth study of the model including bifurcation analysis can be found in several papers, [12], [4], [7].
The stationary cornering conditions are points in the model where all accelerations are zero, which is an equilibrium of the velocity model. These points of the model can be found by assuming the forward velocity and steering angle to be constant. The lateral velocity and the yaw rate are calculated such that the accelerations are zero. For the calculation a numerical solver is used which

is based on a Levenberg-Marquardt optimizer. To find different points which fulfill the requirements, different initial conditions are used.

$$v_x = const \tag{3.69}$$
$$\delta = const \tag{3.70}$$
$$v_y \rightarrow \dot{v}_y = 0 \tag{3.71}$$
$$\dot{\varphi} \rightarrow \ddot{\varphi} = 0 \tag{3.72}$$

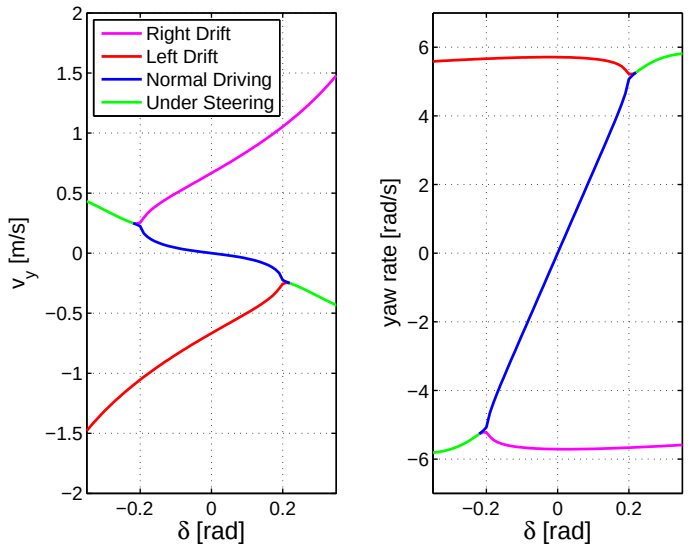

Figure 3.15: Stationary cornering conditions of the model with a $v_x = 1.5$ m/s, with the different regions of the model shown in different colors

By repeating this for many different steering angles, plots as in Figure 3.15 can be produced, which is an example with a forward speed of 1.5 m/s. Three different regions can be separated, whereas only two are of interest. The first is the normal driving region in the middle of the plots in blue. This region is characterized by a linear relation between the steering angle and the yaw rate. The yaw rate is linearly related to the curvature the car is driving, see Equation 3.73. In other words the bigger the steering angle the tighter the curve the car is driving.

$$\dot{\varphi} = \frac{v_x}{R} = \kappa v_x \tag{3.73}$$

Where R is the radius of the curve and κ the curvature. This region is also characterized by a small lateral velocity which is also linearly linked to the steering angle.
The second region of interest is the drifting region, or over-steering region shown in red and magenta, which is radically different from the normal driving region. The region is characterized by counter steer, the steering angle is of the opposite sign than the yaw rate of the car. Furthermore, the lateral velocity is high, or in other words the side slip angle, which as shown in Equation 3.74 is related to

the lateral velocity, is high.. The side slip angle can also be understood as the drift angle. Thess two points mainly characterize drift.

$$\beta = \arctan(\frac{v_y}{v_x}) \tag{3.74}$$

The third region is the under-steering region shown in green in Figure 3.15, which is not of interest in this work. This region is characterized by a slipping of the car over the front wheel, which makes it very hard to control the car in this region.
The stability of the non linear system is also different for the different regions. To investigate the stability just the lateral motion model, consisting of the v_y and $\dot{\varphi}$ states is considered. The stability is checked by an Eigenvalue analysis of the linearized system around each calculated stationary point, which gives the local stability property for each of this points. This is then plotted, in the same kind of plot as in Figure 3.15, however using blue for stable points and red for unstable points.
The analysis shows that the normal driving and the under-steer branches are stable, whereas the drifting branch is unstable, see Figure 3.16. However, the stability margin of the under-steer branch is very small. The Eigenvalues are nearly zero, and even a slightly different linearization point (change at around the fifth digit), for example obtained by an other initial condition of the numerical solver, leads to an unstable system.

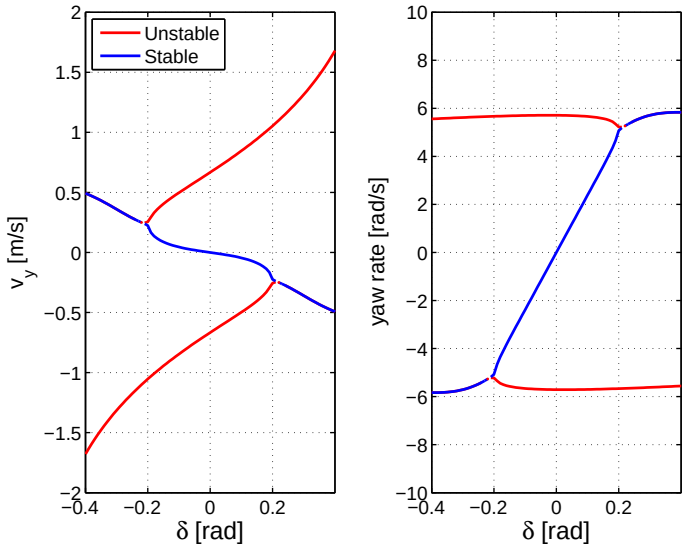

Figure 3.16: Stationary cornering conditions of the model with a $v_x = 1.5$ m/s. The stability properties of the different branches are shown in different colors.

Until now only 1.5 m/s forward speed is investigated. The behavior of the model for different forward velocities can be seen in Figure 3.17. For lower speeds the maximal yaw rate is higher, which means that the car can drive tighter curves. For example, for a forward velocity of 2 m/s, the maximal radius the car can drive in the normal driving region is about 0.5 m, whereas with 1 m/s the

maximal radius is limited not by the friction but by the maximal steering angle and is about 0.136 m.

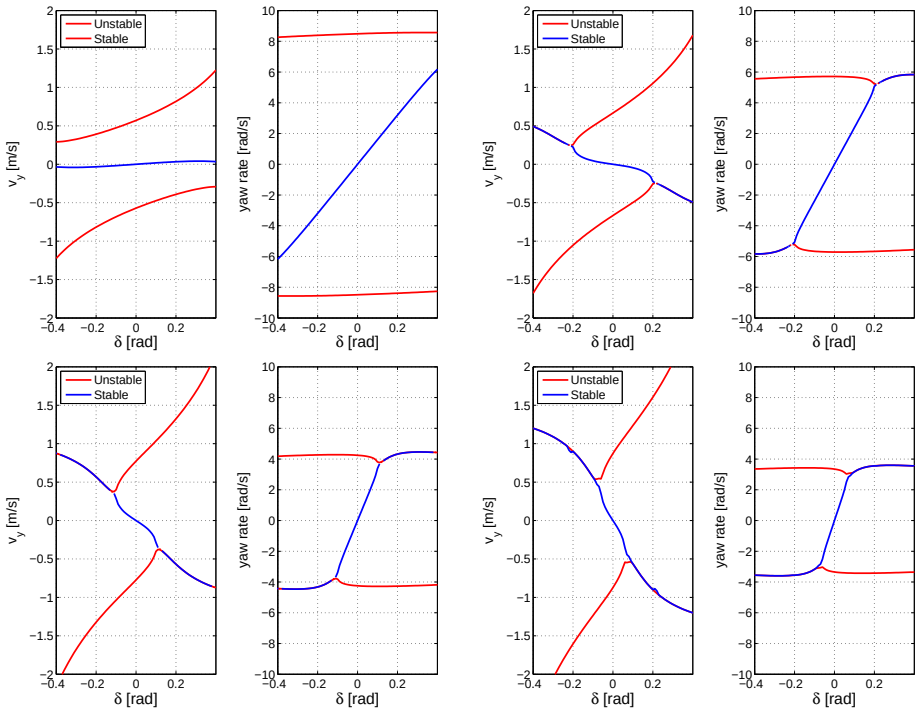

Figure 3.17: Stationary cornering conditions of the model with a $v_x = 1$, 1.5, 2, 2.5 m/s, with stability properties of the different branches

The second phenomenon is the behavior of the linear region, which slope gets flatter the slower the car is driving. Firstly, this changes the sensitivity of the steering angle input to the yaw rate. Secondly, the lower slope moves the connection between the normal driving region and the drifting region to steering angles which are not inside the steering angle limitations. This leads to the fact that for slow forward velocities the maximal curvature is not limited by the tire friction but by the maximal steering angle and secondly makes it problematic to reach the drifting points at low forward velocities.
However, the stability behavior of all four examined velocities is the same. The regions of normal driving and under-steering region are stable, whereas the region of drifting is unstable.
This analysis is very useful in order to choose regions which can be represented well by linear models. Firstly the normal driving region can be represented using a linear model generated by linearizing around

$$v_x = const \tag{3.75}$$
$$v_y = 0 \tag{3.76}$$
$$\dot{\varphi} = 0 \tag{3.77}$$
$$\delta = 0 \tag{3.78}$$

which is the same as assuming linear force laws ($F = C\alpha$), small slip angles

($\alpha \approx 0$) and small steering angles ($\delta \approx 0$). Similar to the assumptions made to create the PWA model, without including the a saturation of the force laws. These assumptions are evaluated for the normal driving region at each forward velocity, see Figure 3.18, where the stationary response of the linear model is plotted into the previously discussed non linear stationary cornering conditions. By using linear approximations at different forward velocities, the effect of the changing steepness of the normal driving region can be approximated.
The drifting region can also be approximated by a linear model at the different forward velocities of interest. The linearization point is chosen at counter steer angle of about 0.1 to 0.2 rad. The affine model approximates the drifting branch well, see Figure 3.18. The stationary velocities of the affine system are shown as an input of the steering angle

$$\dot{x} = Ax + Bu + g = 0 \tag{3.79}$$

$$\Rightarrow x_{ss} = -A^{-1}(Bu + g) \tag{3.80}$$

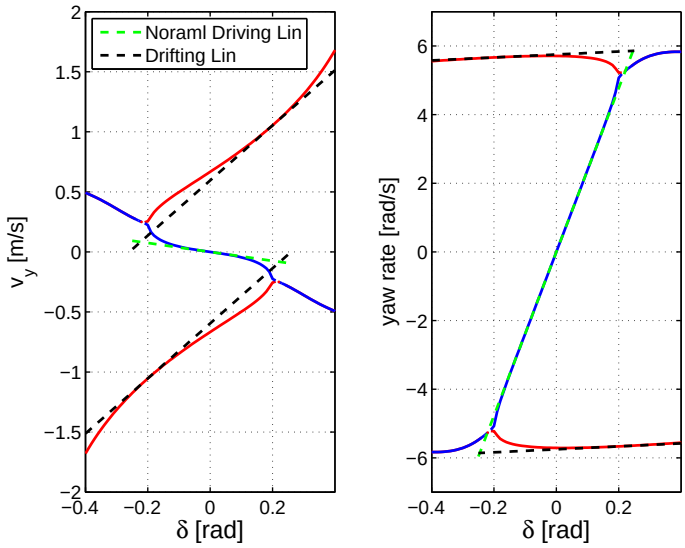

Figure 3.18: Stationary cornering conditions of the model with a $v_x = 1.5$ m/s, comparing linear models to the non linear model

These different linear models give a good starting point for control synthesis, which is discussed in the following chapter.

Chapter 4

PID Path Tracking Controller

In this chapter, a controller which can track a given path and velocity profile is derived. The controller can by itself detect drift and in this case react and maintain the drift until it is better to drive normally.

To realize this controller, first different feedback controllers which can steer the car are designed and secondly a PID structure which can track a known reference path is derived.

4.1 Steering State Feedback Controller

The idea of the presented controller is to use a state feedback controller, which can guarantee stability of the linear system around which it is designed. Such a steering controller can for example be used to maintain drift. The steering controller only needs the lateral velocity model, thus only the lateral velocity and the yaw rate are states of the linear model used to design the steering controllers.

To design the state feedback controller the linear system approximations introduced in the last chapter are used. This way both different velocities as well as drifting and normal driving are captured by these multiple linear systems approximation. However, such a steering controller can only maintain the equilibrium around which it is designed and in order to steer the car the ability to track a yaw rate reference is necessary. The tracking of the yaw rate allows to give the controller the radius the car should drive, given by $\dot{\varphi} = \frac{V_x}{R}$.

In this thesis two control synthesis methods are used to generate such a controller, firstly Linear Quadratic Regulators (LQR) and secondly H_2 controllers. Both are linear, optimization based feedback controllers. In this section the necessary optimization problem and how it is solved is explained and afterwards the results are discussed.

4.1.1 Control Synthesis

Linear Quadratic Regulator

The discrete LQR controller has the goal to find a feedback policy of the form $u = -Kx$ which minimizes a quadratic cost function, subject to the discrete linear time invariant system dynamics

$$\min_u \sum_{k=0}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \tag{4.1}$$

$$s.t \tag{4.2}$$

$$x_{k+1} = Ax_k + Bu_k \quad k = 0, 1, 2, ... \tag{4.3}$$

where Q and R are cost matrices and A and B are the system matrices of the discrete linear time invariant system. The discrete dynamics are generated by descretizing the linear continuous model using a zero order hold (ZOH) approach.In order to find a solution the model as well as the cost function need to fulfill some requirements. First the couple (A, B) needs to be stabilizable and furthermore the cost matrices need to fulfill the following properties

$$Q = Q^T \geq 0 \tag{4.4}$$

$$R = R^T > 0 \tag{4.5}$$

If the above requirements are met, the controller matrix K can be found by solving the optimization using a dynamic programming algorithm, which leads to a discrete algebraic Riccati equation

$$A^T SB[R + B^T SB]^{-1} B^T SA + S - B^T SB - Q = 0 \tag{4.6}$$

$$K = [R + B^T SB]^{-1} B^{-1} SA \tag{4.7}$$

The linearization point of the non linear system is generally not at the origin. Thus the state feedback has to be corrected by the linearization point x_{ss} such that the linear model is valid. The linearization point u_{ss} needs to be added to the resulting control output. By doing so the linear controller interacts correctly with the non linear model.

$$u = -K(x - x_{ss}) + u_{ss} \tag{4.8}$$

The reference tracking can be easily included in the controller, by substituting the linearization point x_{ss} by the reference point x_{ref}

$$u = -K(x - x_{ref}) + u_{ss} \tag{4.9}$$

In the present case only the yaw rate tracking x_{ref} is of interest, which leads to

$$x_{ref} = \begin{bmatrix} v_{y,ss} \\ \dot{\varphi}_{ref} \end{bmatrix} \tag{4.10}$$

H_2 Control Synthesis

Beside the LQR synthesis an H_2 synthesis is also used, which has the advantage that the reference tracking can be addressed using a frequency domain weighting

function. Noise and actuation limitations can also be considered using weighting functions.
In an H_2 control synthesis problem, the goal is to solve the following conditions, which can be rewritten as a Linear Matrix Inequality (LMI) condition

$$\begin{aligned}
&G(s) \quad \text{is stable and} \quad \|G(s)\|_{\mathcal{L}_2}^2 < \gamma \qquad \text{with:} \quad G(s) = \begin{bmatrix} A & B \\ C & 0 \end{bmatrix} \\
&\Leftrightarrow \qquad \text{there exist} \quad X = X^T \succ 0 \quad \text{such that:} \\
&AX + XA^T + BB^T \prec 0 \\
&\begin{bmatrix} W & CX \\ XC^T & X \end{bmatrix} \succ 0 \\
&\text{Trace}(W) < 0
\end{aligned} \tag{4.11}$$

Where $G(s)$ is the transfer function from w to e with the closed loop controller $K(s)$, see Figure 4.1.

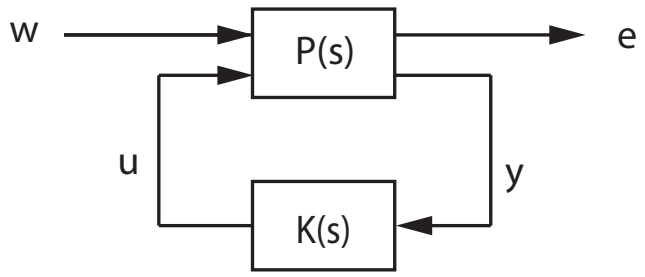

Figure 4.1: Interconnection structure of P(s) and K(s), which result in G(s)

The problem can be solved for an output feedback controller, if $P(s)$ and $K(s)$ have the following form

$$P(s) = \begin{bmatrix} A & B_w & B_u \\ C_e & 0 & D_{eu} \\ C_y & D_{yw} & 0 \end{bmatrix} \tag{4.12}$$

$$K(s) = \begin{bmatrix} A_k & B_k \\ C_k & 0 \end{bmatrix} \tag{4.13}$$

if (A, B_u) is stabilizable and (C_y, A) is detectable. For the given $P(s)$ and $K(s)$ the interconnection has the following form.

$$G(s) = \begin{bmatrix} A & B_u C_k & B_w \\ B_k C_y & A_k & B_k D_{yw} \\ C_e & D_{eu} C_k & 0 \end{bmatrix} \tag{4.14}$$

This problem can the be solved using a linearization transformation, as shown in [1], which leads to an convex Linear Matrix Inequality (LMI) optimization

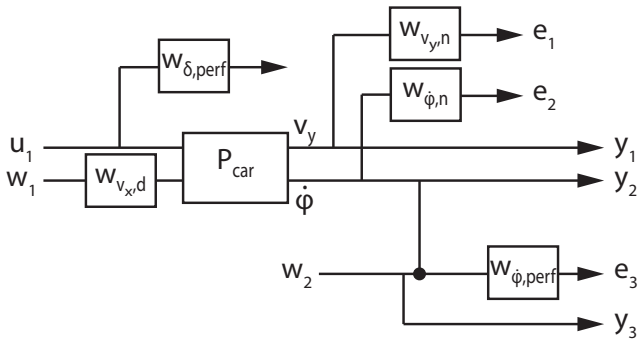

Figure 4.2: Definition of P(s) for the H_2 control synthesis

problem. The resulting optimization problem can then be solved using LMI solvers as CVX or by the robust control toolbox in Matlab.
For the H_2 synthesis the definition of $P(s)$ and the corresponding weights are very important. In Figure 4.2 the used structure can be seen, where P_{car} is the linearized car model. $W_{v_y,n}$ and $W_{\dot{\varphi},n}$ are the noise weights, which are designed to be constant over the whole frequency range, or in other words whit noise. $W_{v_x,d}$ is a disturbance of the forward velocity, which is there to deal with the fact that the controller is used at velocities different than the nominal velocity, the weight is also assumed to be flat over the whole frequency range. Additionally to the noise and disturbance weighting functions, performance weights are also used. $W_{\delta,perf}$ to limit the steering angle to 40° as done in [23], by a flat weight of 57.29/40. Secondly, a performance weight on the tracking error which is designed as a first order element.

$$W_{\dot{\varphi},perf} = \frac{a}{bs+1} \tag{4.15}$$

Where a and b are design variables which are different for the different linearization points and around a = 1 to 3 and b = 1 to 3. This allows to design a reference tracking which follows slow changes, but does not react to noise in the input. This is very important because the reference signal can be very noise, which is calculated by the later on explained PID path tracking controller.

Comparison

The two control synthesis methods are compared in simulation, using an interconnection of the controller and the non linear model. The comparison is done between the H_2 and the LQR controller, once for the controller designed around the linearization point in the normal driving region and once for the design around the drifting linearization point, both at a forward velocity of 2 m/s. Because noise in the reference input is a problem, a random uniform number of the range $[-0.1\ 0.1]$ is added to the reference to simulate noise in the signal.
For the normal driving region good tracking of the yaw rate reference and at the same time a good noise rejection in the reference input are essential. The H2 controller can combine these two tasks better than the LQR, by explicitly

including them in the optimization problem. The H_2 controller follows the reference signal with a smaller static offset and does reject the noise in the input signal better than the LQR controller, see Figure 4.3. Thus in the normal driving region the H_2 synthesis method is clearly preferable.

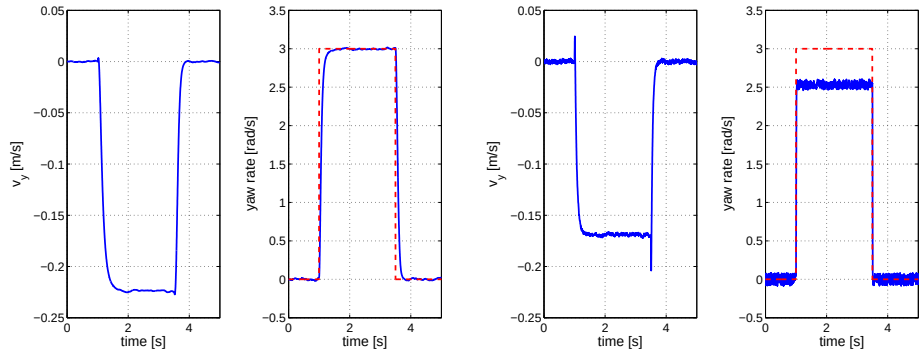

Figure 4.3: Comparison between H_2 (left) and LQR (right) in the normal driving region, using a yaw rate step input, red doted

In the drifting region the reference tracking is less important, because the car is already driving a curve and the goal is more to stabilize the drift. The different performance of the of the two controllers can be seen in Figure 4.4, without noise in the input signal. Whereas both controller are not able to follow the reference, the H_2 controller cannot maintain a constant lateral velocity when the reference yaw rate changes.

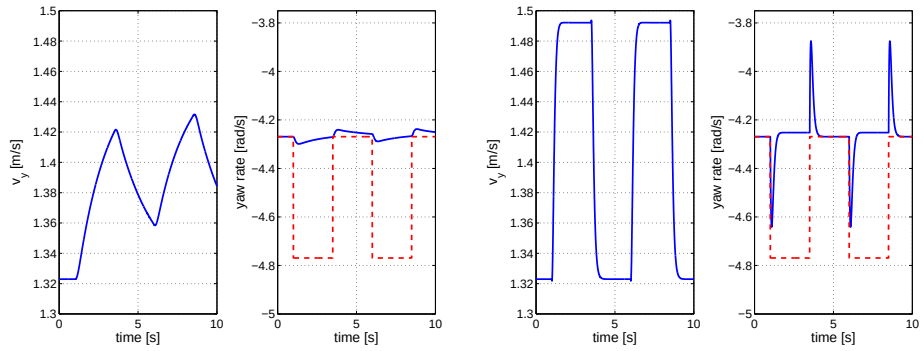

Figure 4.4: Comparison between H_2 (left) and LQR (right) in the drift region, using a yaw rate step input, red doted

Even more important than the tracking of the reference is the controller's ability to reach the stationary point around which the controller is designed. To compare the two controllers a phase portrait of both controllers is generated. On Figure 4.5 it is visible that both controllers can reach the stationary point. However, the LQR has a more favorable behavior than the H2 controller.
Thus the following combination of controllers is used for the linearization points in the normal driving region an H_2 synthesis is used do generate the controller, and for the drifting linearization point an LQR approach is used.

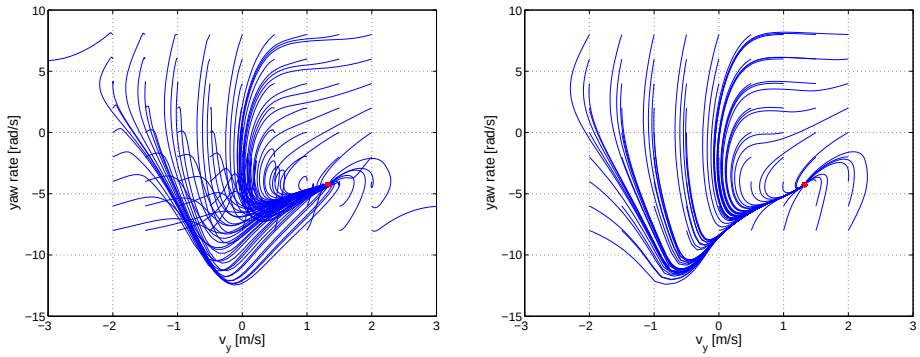

Figure 4.5: Comparison between H_2 (left) and LQR (right) in the drift region, using a phase portrait of the controller system

4.2 Path Tracking

In the last section controllers using different linearization points are designed, which can follow a yaw rate reference. Thus the steering feedback controller can drive curves with different radii based on the yaw rate reference input. However, in order to drive around a track it is also necessary to follow the trajectory. This is achieved by generating the yaw reference for the given track and using this as the input to the controller. The controlled car will therefore follow the desired trajectory.
The generation of such a reference is simple, by using the formula for the curvature, and the relation between the curvature and the yaw rate.

$$\kappa = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}} \tag{4.16}$$

$$\dot{\varphi} = \kappa v_x \tag{4.17}$$

The problem of this approach can be shown on a simple example, where the controller should follow a 180° curve. As the controller is not able to follow the reference signal perfectly the car does not follow the path, see Figure 4.6. The problem is that there is no correction between the position the car is driving and where the car should drive. The problem gets even worse if additionally model mismatch, an offset in the tracking, and noise is added. This corresponds to an open loop situation.
Thus the idea is to add an additional feedback loop which corrects the yaw rate reference to the steering controller, such that the car follows the reference path, see Figure 4.7. The correction controller uses two PID controllers, which change the input such that the car has no error between its position and the X-Y reference position and that the car and the reference trajectory have the same orientation.
This is done by using the orthogonal error between the reference and the car for the first PID controller (position PID) and the orientation error for the second PID controller (orientation PID), see Figure 4.8.
The position PID is essential in order to follow the reference, which is the most important part of the path tracking controller. The orientation PID is not necessary for the reference tracking, but does improve the tracking performance.

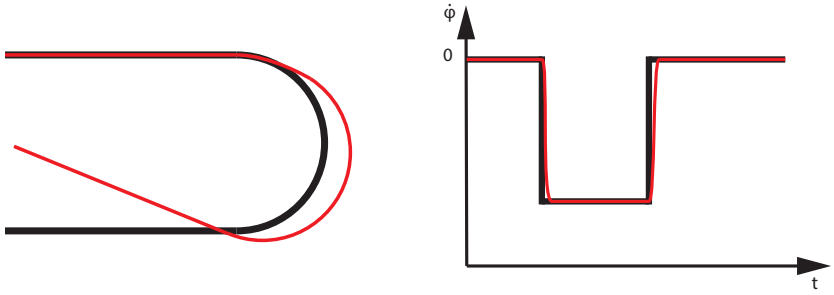

Figure 4.6: Graphical example of the open loop path tracking, where only the yaw rate reference is used

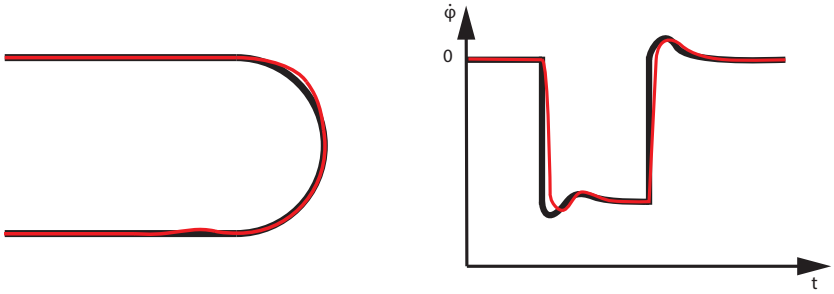

Figure 4.7: Graphical example of the closed loop path tracking, where the yaw rate reference is additionally changed

The performance can be improved because orientation of the car strongly influences its future position. Under the assumption of a smooth trajectory, the same orientation thus helps to improve the tracking in the close future, see Figure 4.9.
The final structure of the control can be seen in Figure 4.10, with the position and orientation PID in parallel. The addition of the two controller outputs and the curvature feed forward signal leads to the yaw rate reference for the steering controller. The steering controller still needs rules how the controllers using the different linearization points are chosen, such that the chosen controller is suited for the task.
This decision is based on a drift detection algorithm and further switching rules. The drift detection is done as simply as possible, using a model based approach based on the magic formula. Drift is characterized by the saturation of the rear wheel force law. The force law and the saturation point are known from the model identification. Thus based on the magic formula of the rear wheel, the car starts drifting if the absolute value of the slip angle of the rear wheel $|\alpha_r|$ is bigger than about $0.3 - 0.5$, where 0.3 is the point where the linear region ends and 0.5 is the point where the force law is completely saturated, see Figure 4.11. The exact point of the switching is a tuning variable of the controller.
To switch from the drift controller back to the normal driving controller another criteria is used, such that the controller switches back to the normal driving

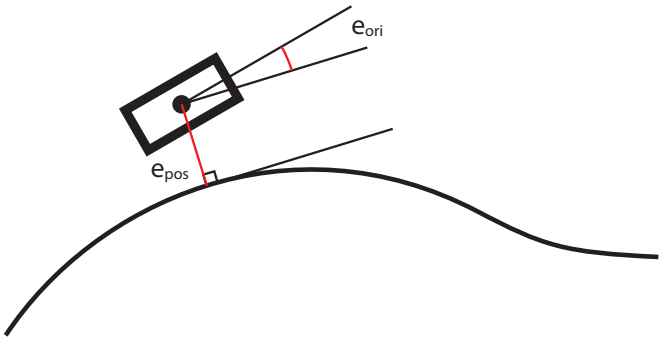

Figure 4.8: Visualization of the position and the orientation error

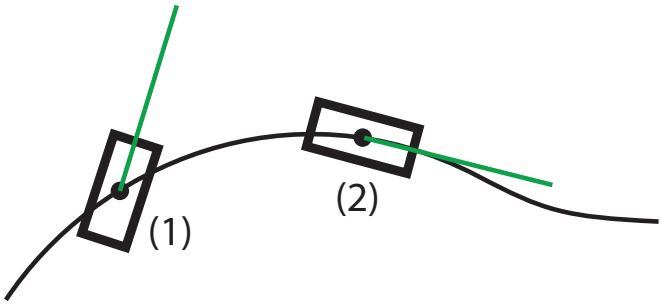

Figure 4.9: Advantage of orientation error tracking, (1) has a zero position error, but will not follow the trajectory in the future, (2) has a zero position and orientation error and will follow the trajectory in the near future

controller if the curve is over and the car is close to the reference path. This is done using the absolute value of the yaw rate reference, which gives a measure for the distance to the trajectory, by the position PID, and an indication if the curve is over, by the curvature part of the signal.
Beside the drift detection, the controller with the closest linearization point in terms of the forward velocity is used. The whole drift detection can be seen in Figure 4.12.
Beside the path tracking controller, a PID controller is used which tracks the forward velocity, with an additional feed forward signal based on an old path tracking implementation from [22].

$$D = \frac{PID(v_{x,ref} - v_x)}{C_{m1} - v_{x,ref}C_{m2}} + \frac{C_d v_{x,ref}^2 + C_{r0}}{C_{m1} - v_{x,ref}C_{m2}} \tag{4.18}$$

Where the parameters are also given from the previous project, [22].

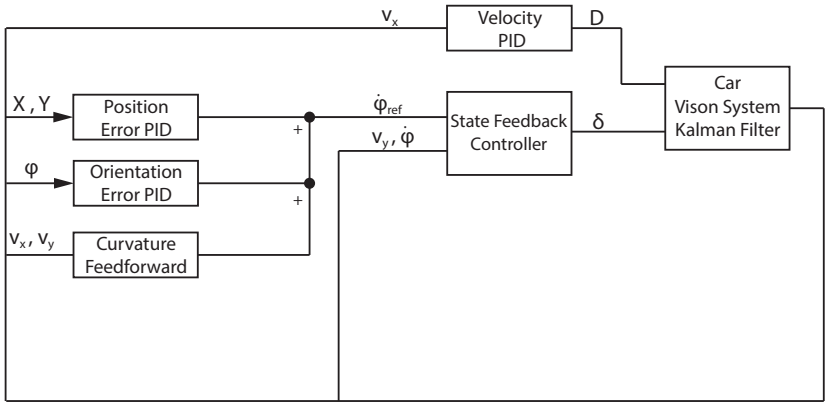

Figure 4.10: Control structure of the PID path tracking controller

4.3 Implementation

The complete controller is implemented using a sampling time of 5 ms on a Windows Real Time Target using Matlab Simulink Realtime Workshop.

In this section several implementation are described which are necessary to run the controller and are not yet explained.

4.3.1 Error Calculation

The main missing point is the calculation of the orthogonal and orientation error. The position and orientation of the car are known by the camera system, and the discrete version of the reference path and its orientation are known too.

The error calculation first calculates the closest point of the path compared to the measured position. To reduce the computation time of this search, just a small part of the track around the last closest point is used as a search space. This also solves the problem that the search algorithm finds a point which is on the other side of the border but closer to the measured position. Which can be the case if a twisty track is used.

Next the second closest point is searched such that the projection of the measured position lays between the two path points. Now the orthogonal error between the measured point and the line formed by the two points can be calculated. Lastly the sign of the error has to be calculated by finding on which side of the two points the car is located.

Based on the known closest point also the reference orientation and the corresponding point in the velocity profile can be found.

This set up allows arbitrary reference paths as well as velocity profiles, and is equivalent to linear interpolating between the points of the reference path. This makes the algorithm quite independent of the spacing between the reference path points, which simplifies the design of such references.

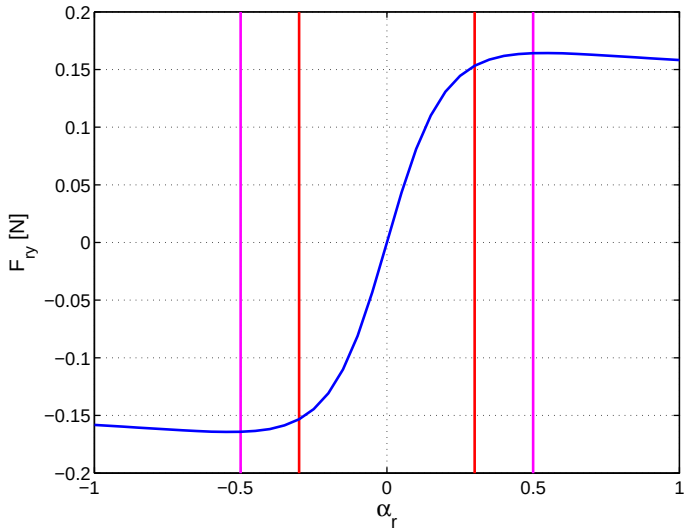

Figure 4.11: Visualization of the saturation of the magic formula, with red line at ± 0.3 and magenta line at ± 0.5

4.3.2 Linearization Points

In the implementation, four different linearization points are used to generate the normal driving controllers, at the forward velocities of 1, 1.5, 2, 2.5 m/s. For the drift controllers just two linearization points are used both at $v_x = 2$ m/s and a steering angle of 0.2 rad and -0.2 rad. This is sufficient for the given task. However, it could be possible that more linearization points could increase the performance of the overall controller. Lastly was the switching between the linearization points never a problem.
To check if this, a stability analysis of the different interconnections of the used controllers and liner models is used, where the main interest is the interconnection of the drift controller with the normal driving linearization points.
The interconnection is given by

$$x_{k+1} = Ax_k + Bu_k \tag{4.19}$$
$$u_k = -K(x_k - x_{ss}) + u_{ss} = -Kx_k + (Kx_{ss} + u_{ss}) = -Kx_k + g \tag{4.20}$$
$$\Rightarrow x_{k+1} = Ax_k - BKx_k + Bg = (A - BK)x_k + Bg \tag{4.21}$$

Thus as long as all Eigenvalues of $A - BK$ are inside the unit circle the interconnection is stable.
It is easy to see in Table 4.1, that all Eigenvalues are inside the unit circle and thus also the interconnection of the drift controller with the normal driving linear model is stable.

4.3.3 Controller Tuning

The tuning of the overall controller requires the tuning of two types of controllers, first the steering controller and secondly the PID controller.

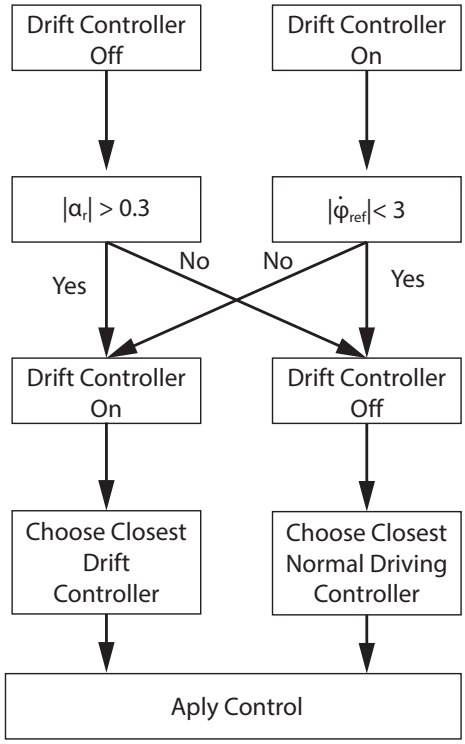

Figure 4.12: Drift detection algorithm

The major tuning of the steering controller can be done in simulation, including noise and disturbance to test the performance. The final test can be preformed using experiments where the car drives in a circle, with changing yaw rate references.

The tuning of the PID controller is more time intensive The controller is first tuned in simulation, and afterwards on the real system. Already the simulation showed that the position PID needs a significant D part, otherwise the controller always overshoots and has a periodic behavior around the reference path. The large D part makes the tuning on the real system quite hard, due to the noise amplification.

The tuning is done in three steps. First the controller is tested on a oval track, where the goal was more a prove of concept and the derivation of some basic parameters for the controller. The second step is to tune the controller on a part of the track, however without physical borders on the overlaying identification track. By having just lines as borders the gains of the controller can be easily tuned such that the car always stays inside the borders and also follows the reference path also at high speeds. In the third step the controller can be implemented directly on the real track, where only a fine adjustment of the gains to the new velocity profile is necessary. The velocity profile is also part of the tuning and is discussed next.

Table 4.1: Eigenvalues of the closed control loop, of the normal driving region model with the drift controller, for different forward velocities

v_x	1. Eigenvalue	2. Eigenvalue
1 m/s	0.779955	0.319392
1.5 m/s	0.823465	0.394757
2 m/s	0.836283	0.444016
2.5 m/s	0.834792	0.482707

4.3.4 Reference Path and Velocity Profile

The reference path is generated using a Model Predictive Contouring Controller (MPCC), using the full six state drift model, the implementation of the MPCC and the changes to make it stable are discussed in the next section. The advantage of the MPCC in generate an offline reference path is the fact that it is a model based optimization, which leads to nice and smooth paths, and include the velocity in generating the path. So the path is a variable mixture of a minimal curvature and minimal distance path.
The MPCC would also give a velocity profile, however it is too aggressive for the presented real controller. Thus a hand tuned velocity profile is used, which consists of different forward velocity levels, see Figure 4.13. This set up is used because it is easy to tune, mainly it has to be decided, when the car starts breaking each for a curve and what the highest possible speed is for this curve. Secondly the position when the car can start accelerating again and how fast it can drive on each straight track segment has to be decided, resulting in the profile as seen in Figure 4.13.

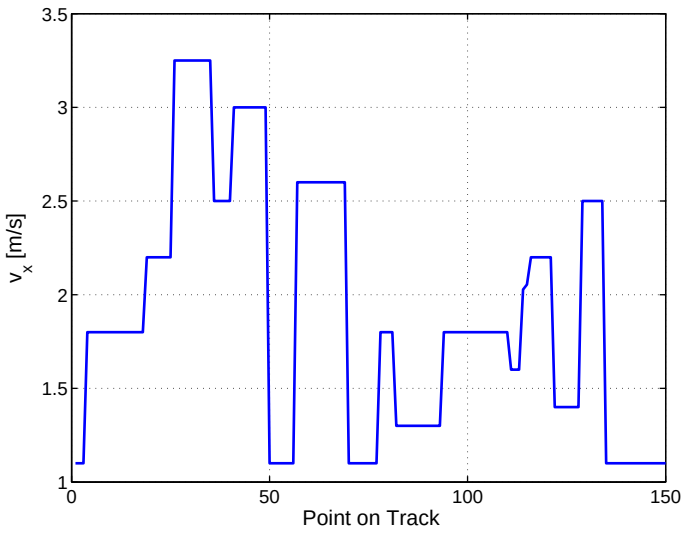

Figure 4.13: Velocity profile for PID path following controller

Such a velocity profile can also be used to initialize drifting if it is desired. By having a break point at the beginning of the curve, the drift can be initialized,

using the load change effect and the blocking of the rear wheel.

4.4 Results

Using this implementation on the real track it is possible to drive with a forward velocity above 3.0 m/s and the car has a lap time of around 9.4 seconds. The tracking of the reference is not perfect, see Figure 4.14. However, this is caused by the aggressive velocity profile which does not have the goal to allow a perfect tracking but is tuned for minimal lap times. The tracking of the path is also worse if the car is drifting as can be seen in several curves. This tracking inaccuracy is just a velocity profile problem. This is visible if a slow velocity profile is used, generated using the algorithm presented in previous projects [22]. Using the slow velocity profile, the velocity tracking as well as the path tracking are much better, see Figures 4.15. This is on the other hand paid by a very slow lap time of around 14 seconds.

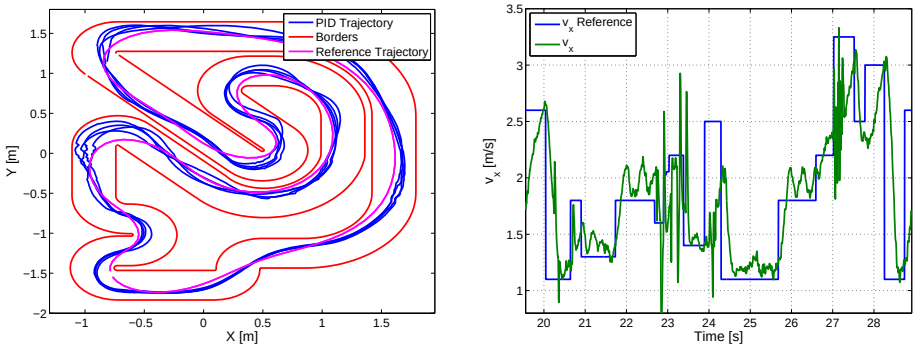

Figure 4.14: Right, four different trajectories of the PID path tracking controller and left velocity tracking over one lap

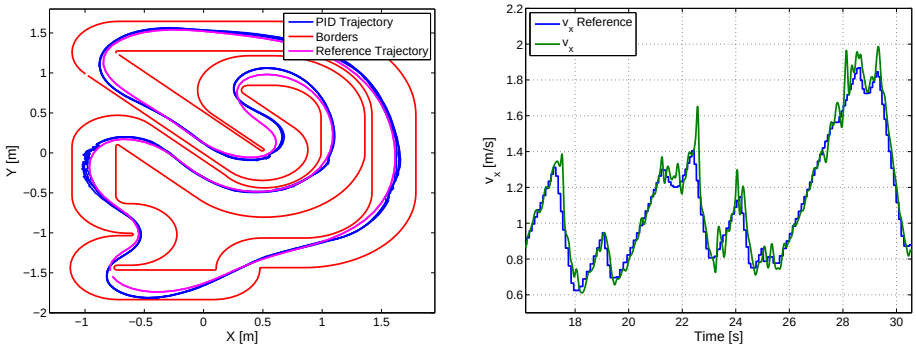

Figure 4.15: Right, four different trajectories of the PID path tracking controller and left velocity tracking over one lap, for old slow velocity profile

The drift controller is also essential for the stability at high cornering speeds, which can be seen if the control set up is tested with and without the drift controller, during the testing on the partial track without physical borders.

The drift controller is only necessary in the one corner, however without the drift controller the car is not always able to drive around the corner inside the borders, see Figure 4.16.

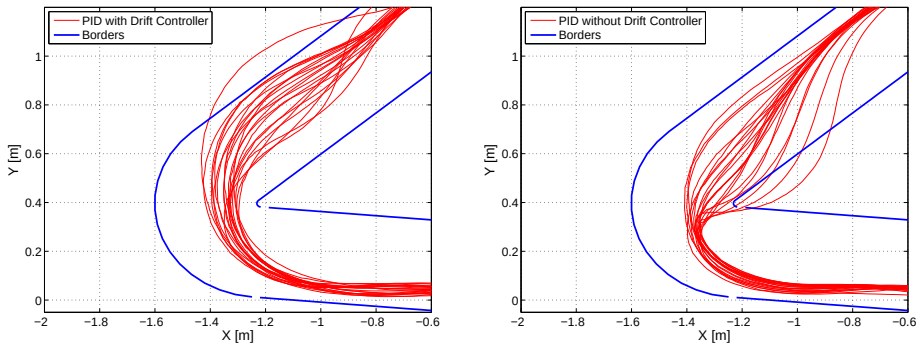

Figure 4.16: Comparison between PID path tracking controller with (left) and without (right) drift controller, in a drifting curve of the track. There are no physical borders, so the car can leave the track.

4.5 Conclusion

This controller gives an effective set up to race the cars at high speed and to use drifting as a security to not spin and to not hit borders and thus be able to drive more aggressively. The controller is not computationally expensive and also suited for less powerful micro controller systems.

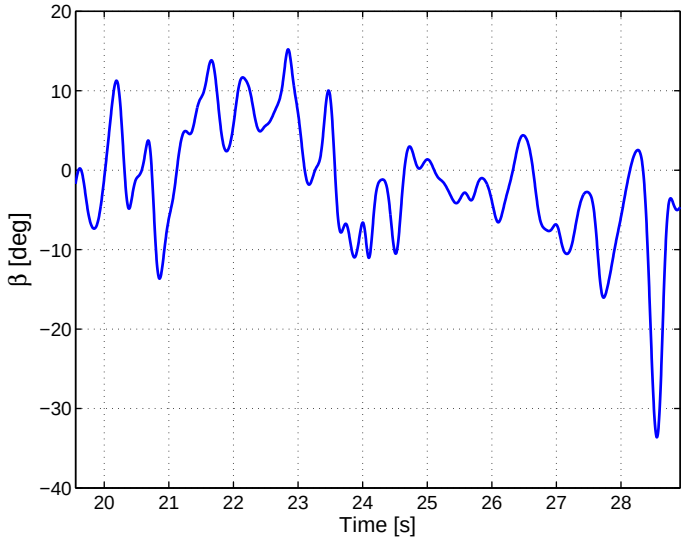

Figure 4.17: Side slip angle over one lap

However, the controller needs a pre defined reference path, which makes it hard to use the controller in competitive environment.

When multiple cars race against each other the path the car has to drive can change dynamically, which is not possible with the current set up. If this controller is used in competitive racing the controller has to be augmented with a path planner to fulfill this requirement.
Secondly, the tuning of the velocity profile is very time consuming, and a critical point of the whole controller tuning process.
Because of these problems in the next chapters different online path planning controllers are investigated.

Chapter 5

Model Predictive Contouring Control

In literature several MPC approaches can be found for controlling cars to follow a given trajectory, often only for lane cross maneuvers, [15]. More complicated trajectories can be found for air planes, [14] , which are similar from a model point of view, such that the control only affects the velocities and the position and orientation are given by integration. However, such MPC controllers are only reference tracking controllers, so the main disadvantage of the given PID path tracking controller cannot be solved.

In several previous ORCA thesis, an adapted MPC, the MPCC, which solves the explained problem of any standard tracking controller is implemented in simulation. The MPCC is implemented using the no slip model. In this chapter the necessary changes to the controller to run the simulation using the drift model are explained.

5.1 Introduction

The MPCC optimize its path and velocity online using a reference line and the projection of the predicted positions onto this reference line. The main idea is to approximate the projection using a virtual state and error terms which minimize the deviation between the approximated and the real projection, and so force the virtual state to be feasible with the model dynamics. Using the virtual state the progress of the car respective to the reference trajectory can be maximized and the orthogonal deviation from the reference can be minimized. The maximization of the virtual state, is the optimization of the progress over the whole horizon, which is identical to a time optimal trajectory. Using multiple linearization steps the generally non linear problem can be transformed into a convex optimization problem, which can be solved using a Quadratic Program

(QP), with the corresponding optimal control problem

$$
\begin{aligned}
&\min_{u} \sum_{k=1}^{N} e_{C,k}^T q_C e_{C,k} + e_{L,k}^T q_L e_{L,k} - q_\Theta \Theta_k \\
&+ \sum_{k=1}^{N-1} \Delta u_k^T R_u \Delta u_k + \Delta v_k^T R_v \Delta v_k \\
&s.t. \\
&x_0 = x \\
&\Theta_{k+1} = \Theta_k + v_k \\
&x_{k+1} = A_k x_k + B_k u_k + g_k \\
&A_{ineq,k} x_k \leq B_{ineq,k} \\
&x_k \in \mathcal{X} \quad u_k \in \mathcal{U} \quad v_k \in \mathcal{V}
\end{aligned}
\tag{5.1}
$$

where e_C is the contouring error, e_L is the lag error, Θ_k is the virtual position and v_k is the virtual velocity. Δ is indicating the difference, so only the changes in the controls are penalized and not there absolute deviation from zero.

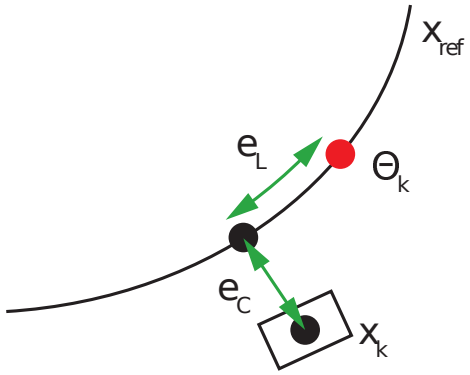

Figure 5.1: Graphical visualization of the different MPCC errors and the virtual state

The whole derivation of the problem can be found in [5] and specific for the given project in [18].

Compared to the previous projects the drift model is used instead of the no slip model. To use this model a linearization has to be calculated anaytically, such that it can be used in the existing simulation environment. Beside the standard model using the magic formula also the simplified model using a PWA approximation of the magic formula and the PWA model are implemented. To check how much the model can be simplified until the controller is not able to control the non linear drift model using the magic formula.

5.2 Model Linearization

One of the key points of the MPCC is the re-linearization of the whole problem, which is based on the last iteration of the controller. Using the controls and the states of the whole horizon the non linear model equation as well as the lag and the contouring error are linearized. The re-linearization and solving the new problem in every time step leads to a Sequential Quadratic Programming (SQP) approach, which leads to the ability to solve the non linear problem.

Therefore, the way the points for the re-linearization are generated is essential. Until now mainly two approaches are used, the first using the controls of the whole horizon to propagate the non linear model using an Ordinary Differential Equation (ODE) solver as for example an 4th order Runga Kutta method with adaptive step size (ODE45). The second approach uses an high order ODE solver for the first time step to simulate the real car, and the measurement update. However, for the rest of the horizon a first order method with the same step size as the controller is used.

These two approaches work fine for the no slip model which is used until now, because the model is well behaved and stable. However, the drift model is more complicated and has a strongly changing behavior due to the saturation of the force laws. So applying the control in an open loop way to generate the new states is very risky and leads to problems, because the non linear model does not behave as expected by the QP solver.

Thus a third way to generate the state sequence for the next linearization is used. For the first step an high order ODE solver with the non linear model is used to simulate the real car. But for the rest of the horizon just the solution from the QP solver is used. This solution is guaranteed to stay inside to bounds and to be optimal with respect to the last linearization points. The QP output has to be augmented with the last time step, where the first order approach with the same controls as in the last control output is used.

It is important that the non linear model is linearized around each old time step separately. This is especially necessary because the position and the orientation over the horizon are very different and also the velocities change strongly. This is a big difference to the most MPC controller found in literature which deal with line cross maneuvers, where it is often sufficient to linearize around the measured point.

5.3 Soft Constraints

Additionally to the re-linearization around the QP output, it was also necessary to use additional soft constraints to limit the lateral velocity and the yaw rate. The soft constraints allow the QP solver to have prediction states which are outside the soft constraints if it is necessary. This works by not constraining the states to a fix value, but changing the constraint, such that, if the states exceed the limit they generate an additional cost, using an additional optimization

variable. For a general example, this is done as follows

$$
\begin{aligned}
&\min \sum_{k=0}^{N} J(x_k, u_k) + q_\eta \eta_k \\
&x_0 = x \\
&x_{k+1} = A_k x_k + B_k u_k + g_k \\
&A_{ineq,k} x_k \leq b_{ineq,k} \\
&x_k \leq x_{max,k} + \eta_k \\
&x_k \geq x_{min,k} - \eta_k \\
&\eta_k \geq 0 \\
&x_k \in \mathcal{X} \quad u_k \in \mathcal{U}
\end{aligned}
\tag{5.2}
$$

Soft constraints have the additional advantage that the measured states can by outside the bounds, which in a hard constraint set up would lead to an infeasible problem. This enables the use of narrow constraints in a real implementation without having feasibility problems.
Soft constraints are only used for the yaw rate and the lateral velocity. By limiting the yaw rate to tight bounds the model behaves well, and narrow turns the optimizer can violate the constraints. Thus it is not completely unwanted that the controller has big yaw rates, the weight of the soft constraints is chosen small. The lateral velocity is also limited, but it is mostly necessary if the PWA model is used.

5.4 Parameter Variation

The control problem has mainly three tuning parameters, the contouring error e_C, the weight on the progress maximization q_Θ and the maximal virtual velocity $v_{\Theta,max}$. The contouring error penalizes the deviation from the center line, which is the reference line. However, due to the border constraints the contouring error can be chosen very small and the car can still not leave the track. The weight for the progress maximization or better the weight on the maximization of the virtual position influences how important the progress maximization is. The last tuning parameter is the maximal virtual velocity, which is the approximation of the projected velocity onto the center line. Higher bounds allow the optimization to be more aggressive. However, too aggressive changes in the virtual velocity are not feasible with the model which leads to instabilities.
Because the MPCC is used to generate a reference for the real controller, the track is shrunk by 6 cm, to take care of the car width, which is 5 cm and the track is 47 cm wide. This is necessary because the border points are at the position of the real borders, but the model equation describes the CG, which is in the middle of the car.
The bound for the yaw rate used in the soft constraint formulation is set to ± 6 rad/s, and the lateral velocity is bounded to ± 1 m/s and for the PWA model to ± 0.5 m/s.
To judge the parameter variation 4 different objectives are compared, the lap time, the time during which the rear force law is starting to saturated, the time

during which it is fully saturated and the time during which counter steer is used.
In the parameter variation three different maximal virtual velocities, 2.4, 3 and 3.5 m/s are compered, by investigating for each maximal virtual velocity different contouring error weights with fixed q_Θ and different q_Θ for a fixed q_C. Choosing the maximal velocity to the same as the real maximal velocity of the model, which is the case with 2.4 m/s, limits the potential of the controller. This is clear because the projected velocity can be higher than the real forward velocity of the car. However, the influence of higher possible virtual velocities is getting smaller the higher the limit is. This is due to the limitations of the model, which has a maximal stationary forward velocity. The higher $v_{\Theta,max}$ limitation mainly makes the controller more aggressive, which can be seen by the fact at the higher percentage of the lap time the rear wheel force law is saturated, but this does not decrease the lap time significantly.

Table 5.1: Comparison of lap time and time the force law of the rear wheel is saturated for different maximal virtual velocities, all other weights are fixed to $q_C = 20$ and $q_\Theta = 1.5$ and the standard drift model is used

$v_{\Theta,max}$	$\alpha_r > 0.3$ [s]	% of lap time $\alpha_r > 0.3$	
2.4 m/s	8.85	0.75	8.47%
3.0 m/s	8.55	0.95	11.11%
3.5 m/s	8.50	1.05	12.35 %

The influence of the contouring error can be seen in Table 5.2. Higher contouring error force the controller to track the reference line, in this case the center line more accurately. This leads to a trade of between tracking and optimizing the tracking error. Thus with a contouring error higher than 30 the lap time starts to increase. The percentage of the lap time during which the rear tire force law is saturated increases for larger contouring error. This is most probably due to the more accurate tracking of the center line which has tight turns.
For contouring errors smaller than 30 it is hard to recognize any tendencies. The lap time is nearly the same, as well as the percentage the rear tire force is saturated. This indicates that the contouring error has only a minor influence for such low values, so the car has no tendencies to follow the center line and thus the car can drive as needed to be fast.
The influence of the progress optimization weighted q_Θ, also has a strong influence on the lap times and the drifting, see Table 5.3. For low values of q_Θ the car drives quite slowly, and for values above 1 the influence reduces. The reduced influence is due the maximal speed limitation of the model, where 8.4 seconds is the lowest lap time obtained with the MPCC and this model. The lowest lap time is not obtained with the highest weight on the progress optimization, but with an intermediate value. Moreover the lowest lap time is also the simulation with the longest saturation of the real tire force law.
Comparing the two controller, one using the magic formula model an the other using the model with PWA approximation of the force laws, the controller using the magic formula model preformed best in terms of lap times. This is as expected because the controller uses the exact same model as the simulation. If the model complexity is reduced and the model using PWA approximation of

Table 5.2: Comparison of lap time and time the force law of the rear wheel is saturated for different contouring errors, all other weights are fixed to $v_{\Theta,max} = 3$ m/s and $q_\Theta = 1.5$ and the standard drift model is used

q_C	Lap Timer	$\alpha_r > 0.3$ [s]	% of lap time $\alpha_r > 0.3$
0.1	8.55	0.80	9.36%
1	8.50	0.90	10.59%
10	8.50	1.00	11.76%
20	8.55	0.95	11.11%
30	8.60	1.10	12.79%
50	8.70	1.10	12.64%
100	8.90	1.25	14.04%
200	9.25	1.50	16.22%

Table 5.3: Comparison of lap time and time the force law of the rear wheel is saturated for different q_Θ, all other weights are fixed to $v_{\Theta,max} = 3$ m/s and $q_C = 20$ and the standard drift model is used

q_Θ	Lap Timer	$\alpha_r > 0.3$ [s]	% of lap time $\alpha_r > 0.3$
0.1	12.05	0	0%
0.5	9.10	0.90	9.89%
1	8.70	0.95	10.92%
1.5	8.55	0.95	11.11%
2	8.45	1.05	12.43%
2.5	8.40	1.20	14.29%
3	8.50	1.15	13.53%
3.5	8.45	1.05	12.43%

the force laws is used, the performance is still good and the lap times are on a similar level. However, the controller has more tendency to drift. The PWA model on the other hand is not as stable, as for a lot of the tested weights the controller using the model crashed. The robustness of the controller could most probably be improved by better tuned soft constraints.
In Table 5.4 an example where all there models work can be seen. It is interesting that all three lap times are only 50 ms apart which is the minimal resolution of the used post processing and the sampling time of the controller. Furthermore it is clearly visible that the second model with the simplified force laws has the most drifting characteristics, where compared to all previous examples driving time is spend in counter steer and strongly saturated force laws.
Simplifying the PWA model even further and getting ride of the tire saturation lead to a controller which is not able to control the non linear model using the standard drift model with the magic formula. Also the no slip model is not successful in controlling the drift model.
An interesting but not tested combination would be an MPCC with soft constraints limiting the slip angles instead of the states. Such a controller could work good with the PWA model or even with a model using only linear tire forces.

Table 5.4: Comparison of lap time and time the force law of the rear wheel is saturated for different maximal models in the MPCC, all weights are fixed to $q_C = 20$, $q_\Theta = 1.5$ and $v_{\Theta,max} = 2.4$ m/s

$v_{\Theta,max}$	Lap Timer	$\alpha_r > 0.3$ [s]	$\alpha_r > 0.5$ [s]	Counter steer [s]
Standard Drift Model m/s	8.55	0.95	0	0
PWA Force Law Model	8.6	1.45	0.2	0.1
PWA Model m/s	8.6	1.1	0	0

Chapter 6

Receding Horizon Drift Controller

The MPCC controller using the drift model is not a controller which efficiently drifts, but more a controller which reacts correctly if the tires saturate. Furthermore the implementation of the MPCC controller is still in progress and therefore cannot be tested on the real platform.
The idea is to derive a new Receding Horizon Controller (RHC) which solves the problems of the existing PID path tracking controller, and fulfills the following goals

- Online trajectory and velocity profile optimization
- Staying inside given borders (static (track) or dynamic (opposing car))
- Deciding when it is better to drift

The controller should be able to plan its trajectory online such that the controlled car can stay inside given borders, which in this case are the normal static borders. The controller should also be able to deal with dynamically changing borders, which are generated so that the car does not hit other cars. Finally, the controller should be able to decide when it is better to drift and not just react if the car stats drifting.
The problem is solved by separating it into two steps. First an online path planner based is used on stationary cornering conditions and then an MPC reference tracking controller is used to track the output of the path planner.

6.1 Path Planner

6.1.1 Principle

The path planner is based on stationary cornering conditions. These are points in the model where all three velocities are stationary and all accelerations are zero. Because all velocities are constant, the resulting trajectories of such points all correspond to circles.

By choosing a finite number of such points, in the normal driving as well as in the drifting region for different forward velocities, all stationary cornering conditions of interest can be captured.
This is realized by including normal driving points at forward velocities of 0.75, 1, 1.5, 2, 2.4, 2.75, 3 and 3.25 m/s, as well as drifting points at forward velocities of 1.5 and 2 m/s. At slower velocities the transient from normal driving to drifting is not possible. On the other hand at higher speeds drifting is undesirable, because such speeds are only possible on straights and light turns. Altogether 52 such stationary points are used, where only 8 of them are drifting points.
Using these stationary points it is easy to generate the corresponding trajectories, by using the assumption that the car can reach the velocities within on time step. Under this assumption the acceleration in the whole horizon is zero and only the position and orientation have to be integrated starting from the measured point. This is done using an euler backward integration method.

$$y_{k+1} = y_k + \frac{1}{Ts} f(t_{k+1}, y_{k+1}) \tag{6.1}$$

The assumption of reaching the stationary velocities within one time step, makes it very easy to deal with the unstable drifting trajectories. Other approaches like using the corresponding controls to propagate the whole system would not lead to drifting trajectories.

$$\varphi_0 = \varphi \tag{6.2}$$
$$X_0 = X \tag{6.3}$$
$$Y_0 = Y \tag{6.4}$$
$$\varphi_{k+i} = \varphi_k + \frac{1}{Ts}\dot{\varphi}_i \tag{6.5}$$
$$X_{k+1} = X_k + v_{x,i}\cos(\varphi_{k+1}) - v_{y,i}\sin(\varphi_{k+1}) \tag{6.6}$$
$$Y_{k+1} = Y_k + v_{x,i}\sin(\varphi_{k+1}) + v_{y,i}\cos(\varphi_{k+1}) \tag{6.7}$$
$$\text{with: } v_{x,i},\ v_{y,i} \text{ and } \varphi_i \quad \in \quad \text{stationary velocities} \tag{6.8}$$

The propagation of all trajectories leads to plots as seen in Figure 6.1. The difference between drifting and non drifting trajectories can be seen if only one drifting trajectory is compared to a normal driving trajectory at the same forward speed. To additionally visualize the difference a straight driving trajectory is added as a reference, see Figure 6.2. Due to the high lateral velocity the drifting car is driving more sideways, therefore the trajectory has a completely different shape than the normal driving trajectory, which behaves as expected.
To simplify the explanation of the following algorithm, a pure graphical example is used, which only contains normal driving trajectories for two different forward velocities, see Figure 6.3.

6.1.2 Optimization Problem

After such a set of possible trajectories is generated, the problem is to find the optimal trajectory. For racing the goal is to choose the time optimal trajectory, which stays inside the track.

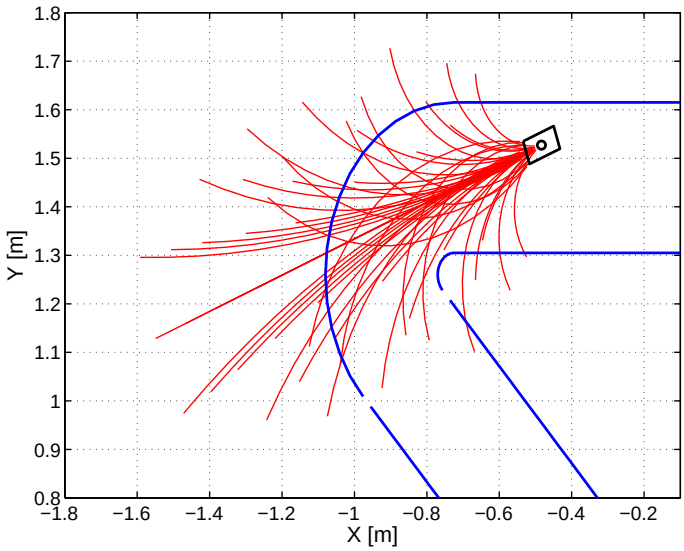

Figure 6.1: All 52 possible trajectories from path planner

The time optimal trajectory is the one with the biggest progress on the track. This is measured using the orthogonal projection of the last point onto the center line of the track.
Beside choosing the trajectory with the biggest progress on the track, which stays inside the borders of the track, the difference between the measured velocities and the stationary velocities in the horizon should be small. This is necessary, because the trajectories are generated under the assumption that the car can reach the velocity within one time step. This assumption is not valid if the measured velocities and the planed velocities are too different. If on the other hand the bounds are too tight, the flexibility of the path planner is limited too strongly to achieve low lap times.
This leads to the following optimization problem, which can be solved using a brute force algorithm, which is here not too computationally intensive due to the small search space.

- Objective: Choosing the trajectory with the biggest progress on the track
- Such that: 1. Trajectory stays inside borders of the track
  2. Stationary velocities of the trajectory are close to the measured velocities

A visualization of the optimization problem and its solution can be seen in Figure 6.4.
To solve the optimization problem, it is necessary to calculate the progress on the track and an algorithm has to be designed to check if a trajectory stays inside the track. Lastly an appropriate definition of close has to be found for the last optimization point. This is united in the following algorithm.

1. Generate trajectories

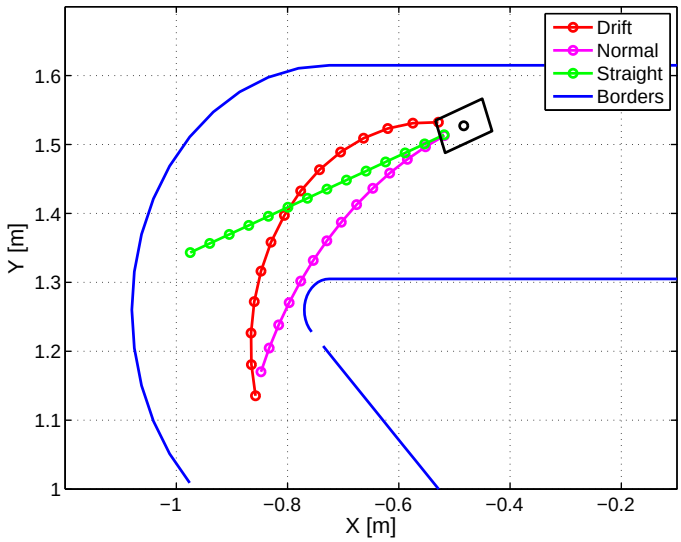

Figure 6.2: Difference between drifting and normal driving trajectories

2. Calculate progress on track for each trajectory

3. Sort the trajectories by their progress on the track

4. Repeat until optimal trajectory is found: start with the best trajectory

4.1 Check if stationary velocities are close to measured
→ if No go back to 4.

4.2 Check if trajectory stays inside borders
→ if No: go back to 4.
→ if Yes: optimal trajectory is found

The progress of a trajectory is measured by the projection of the last point of the trajectory onto the center line. The projection is calculated onto the piece wise linear center line, given by a finite number of points. As in the error calculation of the PID path tracking controller, the two closest points on the center line are searched, within a limited search region. The orthogonal projection onto the linear vector, formed by the two closest points, can be easily calculated using the dot product, see Figure 6.5

$$L_p = A\cos(\Theta) = A\frac{AB}{\|A\|\|B\|} \tag{6.9}$$

To compare the progress of the different trajectories, the arc length from the start of the center line until the orthogonal projection is calculated, by adding L_p to the arc length until the point x_1.
This leads to the progress on the track of each trajectory. The progress of all trajectories can then easily be sorted for the next steps.
To check if the stationary velocity is close enough to the measured velocity some rules have to be found. These rules are different for drifting and normal driving

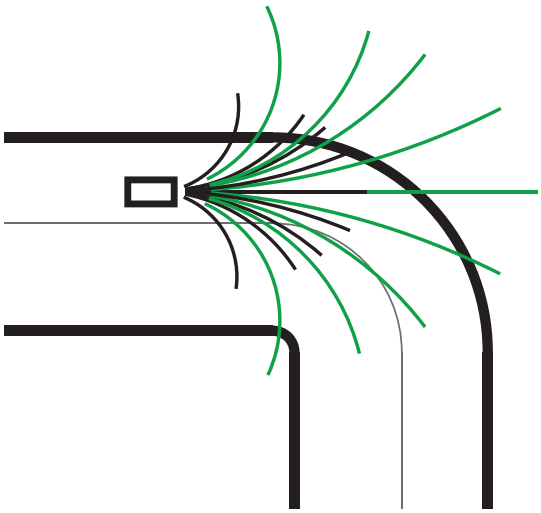

Figure 6.3: Graphical example of the path planner, with two forward velocity levels

points. For normal driving trajectories only the planed forward velocity has to be close to the measured forward velocity. To allow the path planner to change the velocity the difference between the measured and the planed forward velocity has to be at least ± 0.5 m/s, given by the maximal spacing of the used stationary points of 0.5 m/s. To add more flexibility to the path planner, slightly larger differences are allowed, of $+0.75$ and -1.1 m/s, which allows harder breaking than accelerating.
Drifting trajectories on the other hand are only allowed if the measured velocities are already indicating sufficiently large lateral velocity and yaw rate, that is if $|v_y| > 0.2$ and $|\dot{\varphi}| > 3$.
The exact values of all limitations are a tuning parameter which strongly effects the robustness of the path planner and the overall controller. Using these parameters the level of how aggressive the path planer is can be adjusted.
A slightly different approach to solve the problem would be to limit the distance between the measured and the planed velocities.

$$\|v_{x,m} - v_{x,p}\| < B_{v_x} \tag{6.10}$$
$$\|v_{y,m} - v_{y,p}\| < B_{v_y} \tag{6.11}$$
$$\|\dot{\varphi}_m - \dot{\varphi}_p\| < B_{\dot{\varphi}} \tag{6.12}$$

The set in which the measured velocity can be so that each stationary velocity is in the permitted range, can be determined by a reachability analysis, for each linearized model.
If the planed velocity fulfills the requirements, the trajectory is checked to stay inside the track. This is done using a discrete version of the borders, without interpolating between the points. Each point of the trajectory is tested individually to check if it lies inside the borders. This does not guarantee that

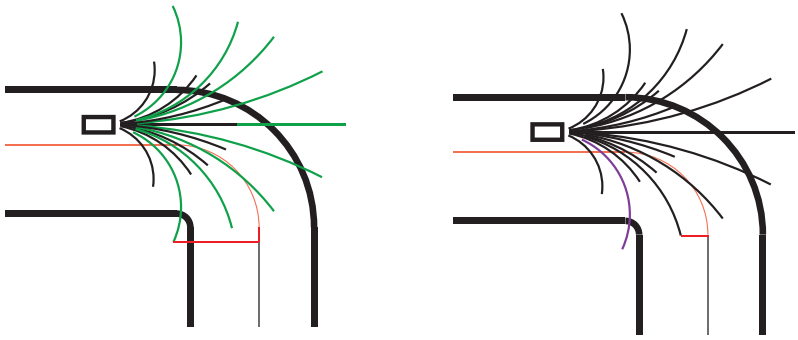

Figure 6.4: Graphical example of the optimization problem. Left the trajectory with the biggest progress, which leaves the track and right the optimal trajectory

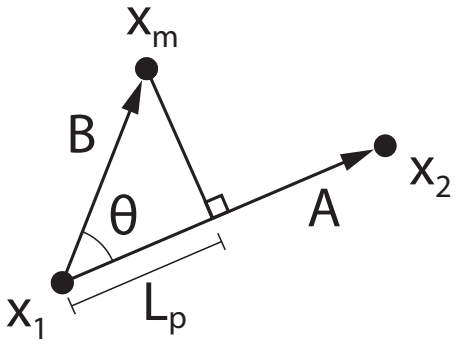

Figure 6.5: Calculation of the orthogonal projection using the dot product

the continuous trajectory stays inside the borders, however, with the used sampling time the trajectory points are close enough together, such that this is no problem.
Each point is checked by the following algorithm, for which it is necessary that the border points are orthogonal to the points of the center line.

1. Search closest point on the center line, within search region

2. Calculate the distance from the corresponding border points to the measured point

$$d_{inner} = \|x_{border,inner} - x_m\| \tag{6.13}$$
$$d_{outer} = \|x_{border,outer} - x_m\| \tag{6.14}$$

3. Calculate the width of the track as the distance between the two border points

$$w_{track} = \|x_{border,outer} - x_{border,inner}\| \tag{6.15}$$

4. Check that both distance from the measured point to the border are smaller than the track width

$$d_{inner} \leq w_{track} \quad \text{and} \quad d_{outer} \leq w_{track} \tag{6.16}$$

5. If both inequalities are true the trajectory point is inside the track

The algorithm is graphically illustrated in Figure 6.6 and gives a conservative test if the trajectory is inside the track. The online calculation of the track width allows to have a track with changing width, which is necessary if the borders can be changed dynamically.

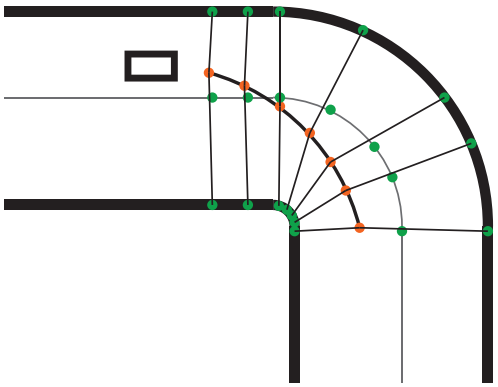

Figure 6.6: Graphical example for the calculation to check if a trajectory stays inside the track

The algorithm starts by checking the last point of the trajectory and goes back to the first one. This order is used because often the last points are outside the borders, thus the expected computation time can be reduced. If one point does not fulfill the test the trajectory is not used and can be directly dumped.

6.1.3 Problems

The path planner explained until now has some problems, in a real implementation.
First the width of the track has to be shrunk, because the equation of motion only describes the CG and thus it is possible that CG is still inside the borders, but the real car is already outside. Additionally to the pure width, some safety factor has to be added to to get some robustness and to deal with the fact that the car is not always parallel to the borders. Thus the track is shrunk by 6 cm, where the car is only 5 cm wide.

This leads to the problem that the car may still leave the allowed borders. If the car is outside the border and is not able to come back into the border constraints within the first time step, the path planner finds no solution, see Figure 6.7. If the car once leaves the constraints and is parallel to the physical border on the track, most of the times it is not possible to find a trajectory which allows it to come back within one time step. Thus it is important to find an efficient recovery strategy in such a case. If there is no possible trajectory which brings the car back within one time step, the recovery strategy allows to bring the car back inside the constraints within a few time steps. For the used sampling time 4 or 5 time steps is a good choice.

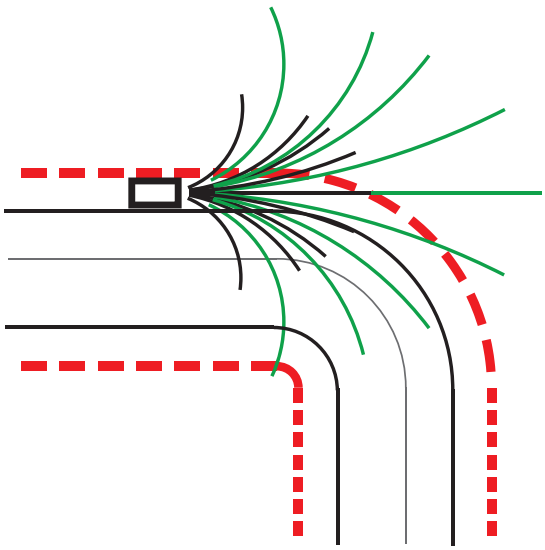

Figure 6.7: Graphical example of a situation where a recovery strategy is necessary. Where the red dashed line is the physical border and the black line is the constraint of the algorithm

This leads to an additional step in the algorithm.

1. Generate trajectories

2. Calculate progress on track for each trajectory

3. Sort the trajectories by their progress on the track

4. Repeat until optimal trajectory is found: start with the best trajectory

4.1 Check if stationary velocities are close to measured
→ if No: go back to 4.

4.2 Check if trajectory stays inside borders
→ if No: go back to 4.
→ if Yes: optimal trajectory is found

5 If there is no possible solution
→ use recovery trajectory which fulfills the border constraints within 4 time steps and has the biggest progress on the track

The search for this trajectory requires no additional computation time because it can be found during step 4., by saving the first trajectory which fulfills the border constraints at least until the fourth to last time step. This still does not guarantee that the path planner has a solution, however, the path planner needs a solution otherwise the reference tracking MPC cannot run.
If the path planner including the recovery strategy still does not find a solution, the optimization problem is solved without constraints. However, just forward velocities of 0.75 and 1 m/s are allowed. Not using any border constraints often leads to trajectories which cut corners. This can solved by adding an heuristic approach. The heuristic approach uses the knowledge of how fast the car is planed to drive, this allows to give an approximate upper bound to the progress. If the car has a bigger progress than this upper bound, the trajectory has to cut a corner to achieve such a progress, thus it can be rejected.
This leads to the final path planning algorithm.

1. Generate trajectories

2. Calculate progress on track for each trajectory

3. Sort the trajectories by their progress on the track

4. Repeat until optimal trajectory is found: start with the best trajectory

4.1 Check if stationary velocities are close to measured
→ if No: go back to 4.

4.2 Check if trajectory stays inside borders
→ if No: go back to 4.
→ if Yes: optimal trajectory is found

5 If there is no possible solution
→ use recovery trajectory which fulfills the border constraints within 4 time steps and has the biggest progress on the track

6 If there is no recovery trajectory
→ chose trajectory with a forward velocity of 0.75 or 1 m/s and the biggest progress on the track without violate heuristic

To further increase the robustness against crashes, an additional shrinking of the track around corners can be used, see Figure 6.8. These helps to decrease the number of crashes in this corners while allowing the path planner in the rest of the track to use its full potential.

6.1.4 Multi Radii Algorithm

The path planner formulation until now has one big disadvantage: it limits the path planner to have a stationary velocity and so just one radius within the whole horizon. This leads to problems in narrow curve combination and makes the path planner less flexible. To deal with this fact the horizon length has to be quite short.

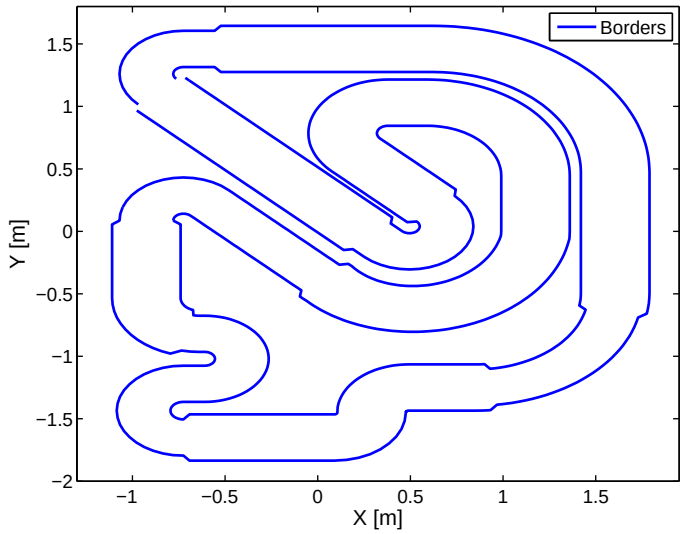

Figure 6.8: Locally shrinking in critical curves

To allow more flexibility in the path planner, it would be of advantage if the path planner could choose between different stationary velocities in different parts of the horizon.
The best results can be obtained by allowing all possible combinations of of stationary velocities. This, however, leads to an exponentially growing number of trajectories. In other words, just allowing to split the horizon into two intervals leads to 2704 possible trajectories, which is a problem if the whole controller should be implemented with a sampling time below 50 ms.
The goal is so to come up with an algorithm which has a linear complexity and still increases the performance of the path planner. The simplest version of such an algorithm would be to separate the problem into individual optimization problems, thus choose for the first short interval the optimal solution and based on this trajectory choose the next and so on. Such an algorithm leads to problematic trajectory, because the intervals of each optimization problem are very short. Therefore the optimal trajectory is often very straight. For example it is optimal to drive straight with full throttle until the center line starts turning. However, often it is already too late to take the curve exactly at at the beginning of the curve and the path planner does not find a solution for the next interval. The horizon of such a path planner is too short to generate useful trajectories.
Thus beside the linear complexity a further point is important, the algorithm should guarantee that the additional intervals have a possible solution. To fulfill this, a combination of the original and the previously explained algorithm is used, which first generates the optimal trajectory for the whole horizon. This trajectory is only used until the end of the first interval. For the second interval, the trajectory until the end of the horizon from the new starting point is optimized. Because the stationary velocity used in the first interval is also part of the optimization in the second interval, the second interval can have the same solution as the first optimization. Thus the algorithm has a guaranteed

solution and the progress from first/previous optimization is a lower bound. For a visualization see Figure 6.9.

Figure 6.9: Graphical example of the multi radii path planner. Left all the possible trajectories for the second section and right the resulting trajectory in green compared to the old trajectory in purple

This algorithm has some nice properties: Firstly if the original one radius path planner has a solution, the new path planner also has a solution. Additionally, the original solution is a lower bound on the performance, and lastly it has a linear complexity. However, the algorithm calculates a lot of 'optimal' trajectories which are at the end only partly used. And the first interval of the path planner which is the most important, is the same as in the original path planner.

6.2 MPC Formulation

From the path planner an optimal trajectory containing stationary velocities as well as the position and orientation is obtained. The idea is to use an MPC reference tracking controller to track the output of the path planner.

The MPC controller also deals with the assumption in the path planner, that the stationary velocities can be reached within one time step. To have the lowest cost the MPC controller has to reach the stationary velocity within one time step and stay there. This is generally not possible, so the controller has two tasks: to reach the stationary velocity as quickly as possible and at the same time track the position and orientation of the path planner. The priority on the velocities or the position and orientation tracking can be adjusted using the weighting matrices. The general non linear reference tracking MPC is given as

$$
\begin{aligned}
&\min_{x,u}(\sum_{k=1}^{N-1}(x_k - x_{k,ref})^T Q(x_k - x_{k,ref}) \\
&+ \sum_{k=0}^{N-1}(u_k - u_{k,ref})^T R(u_k - u_{k,ref}) + (x_N - x_{N,ref})^T Q_N(x_N - x_{N,ref})) \\
&s.t. \\
&x_0 = x \\
&x_{k+1} = f(x_k, u_k) \\
&g(x_k, u_k) \leq 0 \\
&x_k \in \mathcal{X} \quad u_k \in \mathcal{U}
\end{aligned}
\tag{6.17}
$$

A quadratic cost function is used, penalizing the deviation of the states from the reference value as well as the deviation from the stationary control inputs. Lastly a terminal cost on the deviation of the last state is used. The optimization is subject to the non linear model, inequality constraints which limits the car to stay inside the track and state and control bounds. The problem as posted here can only be solved using a NLP, which is not suited for a real time implementation.

6.2.1 Problem Formulation

To have an optimization problem which can be solved in real time, a convex quadratic program needs to be derived. Such problems can be very efficiently solved using optimized MPC QP solvers. The QP solver needs linear equality and inequality constraints and a convex quadratic cost function, which is given if $Q = Q^T \geq 0$, $Q_N = Q_N^T \geq 0$, $R = R^T > 0$.
This leads to the following optimal control problem

$$
\begin{aligned}
&\min_{x,u}(\sum_{k=1}^{N-1}(x_k - x_{k,ref})^T Q(x_k - x_{k,ref}) \\
&+ \sum_{k=0}^{N-1}(u_k - u_{k,ref})^T R(u_k - u_{k,ref}) + (x_N - x_{N,ref})^T Q_N(x_N - x_{N,ref})) \\
&s.t. \\
&x_0 = x \\
&x_{k+1} = A_k x_k + B_k u_k + g_k \\
&A_{k,ineq} x_k \leq b_{k,ineq} \\
&x_k \in \mathcal{X} \quad u_k \in \mathcal{U}
\end{aligned}
\tag{6.18}
$$

The non linear continuous dynamic model, needs to be linearized and discretized. First the affine time varying model is obtained, by linearizing around the trajectory of the path planner, which gives the necessary linearization points.

The linearization is done for each time step of the path planner. This is necessary because the position and orientation varies strongly during the horizon. Mainly the orientation which can change up to nearly 180° influences the linear position model strongly, see the linearization of the position and orientation model.

$$\begin{aligned}
&\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi) \\
&\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi) \\
&\dot{\varphi} = \dot{\varphi} \\
&\Rightarrow \\
&\dot{X} \approx \cos(\varphi_k) v_{x,lin} - \sin(\varphi_k) v_{y,lin} - (v_{x,k} \sin(\varphi_k) + v_{y,k} \cos(\varphi_k)) \varphi_{lin} \\
&\dot{Y} \approx \sin(\varphi_k) v_{x,lin} + \cos(\varphi_k) v_{y,lin} - (v_{y,k} \sin(\varphi_k) - v_{x,k} \cos(\varphi_k)) \varphi_{lin} \\
&\dot{\varphi} \approx 1 \dot{\varphi}
\end{aligned} \tag{6.19}$$

Where the values with subscript k are linearization points and the one without are the states of the linear model.
The linearization of the velocities leads to the same affine model over the whole horizon, because the stationary velocity stays the same. If the path planner with multiple radii in the horizon is used, there can also be different affine models for each section. But again for each segment with the same stationary velocities the same affine model is used.
The affine continuous time varying model is then discretized using a zero order hold method and a matrix exponential function. The discretization is necessary for the MPC.

$$\begin{bmatrix} A_d & B_d & g_d \\ 0 & 0 & 0 \end{bmatrix} = expm\left(\begin{bmatrix} A_c & B_c & g_c \\ 0 & 0 & 0 \end{bmatrix} T_s \right) \tag{6.20}$$

Subscript d indicates the discrete and c the continues model matrices.
The linear inequality constraints are derived as in previous ORCA projects where each time step of the MPC problem is constrained to be inside two half spaces, which are tangential to the track. The position of the two half constraints is based on the trajectory of the path planner. The same points as in the border constraints of the path planner are used to generate the two tangential half spaces, see Figure 6.10.
Every time step of the MPC problem is constrained to lie inside the two half spaces, corresponding to the point in the path planner, see Figure 6.10. For these constraints it is crucial that the planned trajectory is close to real achievable trajectory of the MPC, otherwise the constrains lead to problems.

6.2.2 Problems

The derived MPC controller has some problems, the first problem comes from the previously explained half space constraints. If the difference between the measured and the planned velocities is too big the trajectory from the path planner is not achievable for the MPC. However, the linearization and the half space constraints are based on the path planner. This leads to problems, as the

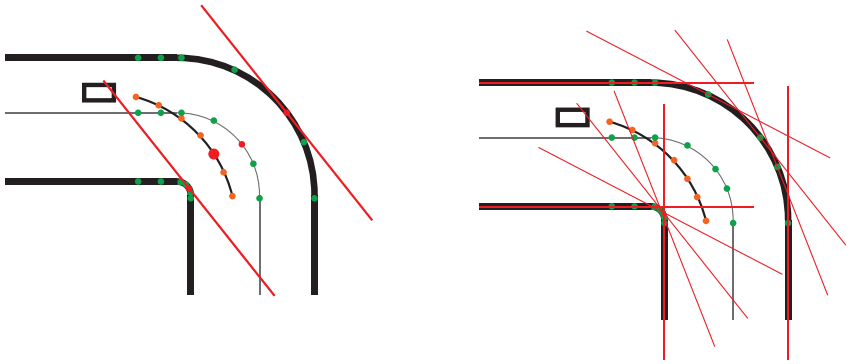

Figure 6.10: Graphical example of the hale space constraints. Left for one time step in the horizon and right for all time steps

main concern are the half space constraints, which can cause feasibility problems and wrong behavior of the controller.
The feasibility problem occurs, if the QP solver cannot reach the region allowed by the constraints, see Figure 6.11. The problem mainly occurs in corners if the velocities of the path planner are not chosen well. The constraint in the path planner that the planed and measured velocities have to be close diminishes the problem. However, for security a back up plan is used if the QP is infeasible, the same problem is solved without the half space constraints.
The second problem caused by the half space constraints is that they can bring the QP to choose a trajectory where the car is driving straight, such that the planed MPC trajectory reaches the constraints, see Figure 6.11. This effect is unwanted because car should already start turning to optimally drive through the curve. The problem is difficult to detect, which makes it hard to react to it. However, the constraint forcing the planed and measured velocities to be close in the path planner solve the problem efficiently.
Another feasibility problem of the MPC controller is introduced by the back up plans of the path planner. If the path planner does not find a solution which stays inside the borders also trajectories which do not stay inside are allowed. Such trajectories lead to an infeasible optimization problem in the MPC, as long as the same track width is used. Thus the track for the half space constraints is shrunk less than the one for the path planner. The border constraints which are only slightly shrunken are enough to prevent the car from hitting borders, and at the same time solving a lot of the infeasibility problems. In case the problem is still infeasible the back up QP without half space constraints is used.
On real laps the back up QP is used in around 1 to 5% of the time steps. It is important that the main part of the infeasible problems are caused by recovery and back up trajectories, and in less than 2% (most of the time below 1%) of the time steps the infeasibility is caused by not reaching the allowed region. What is also interesting, but logical, is that the standard path planner has less infeasible QPs due to reaching problems than the path planner using multiple radii, because the standard path planner is less aggressive. However, the more advanced path planner has less overall infeasible optimization problems because it gives better trajectories, which leads to fewer problems in terms of leaving

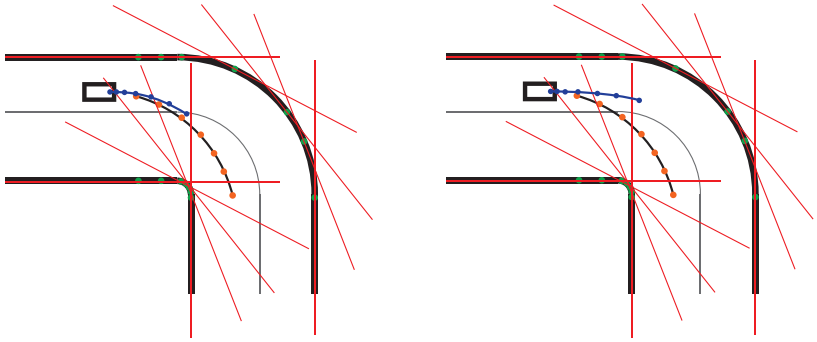

Figure 6.11: Graphical example of the half space constraints problems, under the assumption that the measured forward velocity is much smaller than the planed. Left the first problem where the MPC cannot reach the allowed set for the last time step and right the second problem where the MPC drives straight to reach the be the last half space constraints allowed set

the constraints.
The last problem comes from the longitudinal velocity model, which is not identified in this work but a model from an old project is used. The problem is that the model has a maximal stationary velocity of about 2.4 m/s, which is clearly slower than the car can drive. The second problem is that during a stationary drift the duty cycle necessary to maintain the forward velocity would be around 3. Even driving a normal curve with a speed of 2.4 m/s would need duty cycle of 1.9, where the maximum is 1.
Thus first the original model from the former thesis is used, which has a much simpler breaking term, instead of the more complicated bicycle model.

$$\dot{v}_x = \frac{1}{m}(m((C_{m1} - C_{m2}v_x)D - C_d v_x^2 - C_r) - F_{f,y}\sin\delta + m v_y \dot{\varphi}) \tag{6.21}$$

$$\dot{v}_x = (C_{m1} - C_{m2}v_x)D - C_d v_x^2 - C_r - C_\delta (v_x \delta)^2 \tag{6.22}$$

However, the old model still has the same maximal stationary forward velocity, to allow realistic velocities the duty cycle control bounds of the MPC are set to be between [-1, 2].
To rigorously solve the problem the model has to be identified once again, however this was not possible in the limited time after the problem was identified. Furthermore, the used approach was successful, mostly because the car is driven most the time by full throttle and full breaking inputs.

6.2.3 Tuning

The tuning of the MPC cost matrices is important for the whole RHC. As cost matrices, diagonal matrices are used which fulfill the necessary conditions, if all elements are bigger than zero.
The problem in the tuning of the matrices is that if the controller is too aggressive, the inputs gets too close to bang bang inputs, which amplify unmodeled dynamics. Mainly load change dynamics are a problem, which leads to a slalom

driving behavior. The behavior is initialized by too aggressive duty cycle inputs, which leads to a load change and the slalom driving is even further amplified by the controller. The behavior cannot be tuned in simulation with the existing model, thus the tuning needs to be done mainly experimentally. The problem can be solved using quite high costs on the control deviation, primarily on the duty cycle input to not amplify the load change dynamics.
The weights on the velocities and the positions are similar with a slightly higher weight on the velocities, whereas the weight on the orientation is smaller. Simulations also show that higher weights on position make the optimization problem harder to solve, which is logical, because the controls can only affect the velocities directly which then influence the position and orientation. High weights on the orientation has an even stronger influence on the optimization and leads for too high values to an unsolvable problem.
Lastly the terminal cost is tuned such that the prediction does not deviate more and more from the reference. This is achieved by introducing generally higher weights and relatively higher weights on the position and orientation than during the rest of the horizon. Unfortunately it is not possible to use a standard approach to generate the terminal cost, such as LQR or Lyapunov methods, because the position and orientation model cannot be brought into a linear time invariant version. Furthermore the terminal cost would be different for each stationary velocity. The used cost functions are given by

$$R = \begin{bmatrix} 10^2 & 0 \\ 0 & 10^5 \end{bmatrix} \tag{6.23}$$

$$Q = 10^2 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.24}$$

$$Q_N = 10^3 \begin{bmatrix} 10^3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.25}$$

6.2.4 QP Solver - Forces

To solve the given optimal control problem as a QP, the problem has to be brought into the corresponding form, which is generally.

$$\min_z \frac{1}{2} z^T H z + f z \tag{6.26}$$

$$s.t. \tag{6.27}$$

$$A_{ineq} z \leq b_{ineq} \tag{6.28}$$

$$A_{eq} z = b_{eq} \tag{6.29}$$

$$lb \leq z \leq ub \tag{6.30}$$

To solve this problem efficiently and fast, the QP solver FORCES is used, which is a fast interior point solver for MPC problems [2]. The solver is specially generated for each problem and uses the given sparsity pattern of an MPC problem to solve the problem fast.
FORCES just works with a block diagonal Hessian matrix (H), which is obtained if the optimization variable z is chosen as

$$z = \begin{bmatrix} x_0 & u_0 & x_1 & u_1 & \cdots & x_{N-1} & u_{N-1} & x_N \end{bmatrix}^T \tag{6.31}$$

Using z as defined above the cost function can be rewritten as

$$\begin{aligned} J_k &= (x_k - x_{k,ref})^T Q (x_k - x_{k,ref}) + (u_k - u_{k,ref})^T R (u_k - u_{k,ref}) \\ &= x_k^T Q x_k - 2x_{k,ref}^T Q x_k + x_{k,ref}^T Q x_{k,ref} + u_k^T R u_k - 2u_{k,ref}^T R u_k + u_{k,ref}^T Q u_{k,ref} \\ &= z_k^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} z_k + \begin{bmatrix} -2x_{k,ref}^T Q \\ -2u_{k,ref}^T R \end{bmatrix} z_k + x_{k,ref}^T Q x_{k,ref} + u_{k,ref}^T Q u_{k,ref} \\ &\Rightarrow \\ H_k &= 2 \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \quad \text{and} \quad f_k = \begin{bmatrix} -2x_{k,ref}^T Q \\ -2u_{k,ref}^T R \end{bmatrix} \end{aligned} \tag{6.32}$$

The equality constraints, which are given as

$$x_0 = x \tag{6.33}$$
$$x_1 = A_0 x_0 + B_0 u_0 + g_0 \tag{6.34}$$
$$x_2 = A_1 x_1 + B_1 u_1 + g_1 \tag{6.35}$$
$$\vdots \tag{6.36}$$
$$x_N = A_{N-1} x_{N-1} + B_{N-1} u_{N-1} + g_{N-1} \tag{6.37}$$
$$\tag{6.38}$$

can be reformulated to

$$\mathbb{I} z_0 = x \tag{6.39}$$
$$[A_0 \;\; B_0] z_0 + [-\mathbb{I} \;\; 0] z_1 = -g_0 \tag{6.40}$$
$$[A_1 \;\; B_1] z_1 + [-\mathbb{I} \;\; 0] z_2 = -g_1 \tag{6.41}$$
$$\vdots \tag{6.42}$$
$$[A_{N-1} \;\; B_{N-1}] z_{N-1} + [-\mathbb{I}] z_N = -g_{N-1} \tag{6.43}$$

This leads to an optimization problem which is block diagonal and where the different stages, z_k, are only connected in the equality constraints by the $[-\mathbb{I}\ 0]$ matrix.
The explained controller, including path planning and MPC is only possible, due to the very fast QP solver. In average the computation time for the different tasks can be seen in Table 6.1.
It is important that these are average computational times and not worst case, but it is clear that the controller could not run at 25 ms, if the QP solver would already take about 18 ms in worst case, which is the time CPLEX needs (average

Table 6.1: Computation Times RHC

Path Planner	4.13 ms
MPC Problem Generation	1.19 ms
QP with FORCES	0.81 ms

6ms). FORCES has a worst case computational time of below 10 ms. The fast QP solver also allows to use the back up QP.
What is also interesting is that the linearization and discretization of the problem takes in average longer than the actual computational time of the QP. This is also caused by the fact that the discretization needs a full matrix exponential function. It is possible that a lower order approximation would lead to sufficiently good results and could reduce the computation time, if needed.

6.3 Results

The receding horizon drift controller is implemented using a sampling time of 25 ms, which is not the fastest possible sampling time, but gives good results, while not having any run time problems on the real time system. An horizon length of 14 is used, which is equivalent to look 0.35 seconds ahead. The horizon length is limited by the performance of the path planner, which gets too conservative with longer horizons. This is due to the discussed problem that only one radius can be chosen. Even with the more complex path planning which allows multiple radii in the horizon the problem is not solved, because the first segment is still the same.
Three different implementations of the path planner are tested, first the standard path planner, with just one radius in the horizon. Furthermore the track is locally shrunk in the critical curves to reduce crashes, for the path planner as well as the MPC half space constraints. Secondly the path planner with multiple radii in its horizon is implemented using two different sections or three sections. The implementation using three sections is implemented with a horizon of 15, such that each section has the same length. With the more advanced path planner it is not necessary to additionally shrink the track, which helps the performance of the controller.
All three versions work with the real set up, however the different path planners have an influence on the lap time. The more complicated the path planner the lower the average lap time gets.

Table 6.2: Lap Times RHC

One radius in Horizon	∼9.7 s
Two radii in Horizon	∼9.4 s
Three radii in Horizon	∼9.2 s

The lap time is only faster than the previously discussed PID path tracking controller if a more advanced path planner is used. The lower lap times, with the more complex path planner, are also consistent with the theory that the

planed trajectory has a bigger progress on the track, and show that one of the limiting factors of this controller is the performance of the path planner.
Beside the lap time also the trajectories are changing, see Figure 6.12. It is clearly visible the more complex path planning leads to better overall trajectories. However, also the path planner using three radii in the horizon cannot solve the main problem of the presented controller which is narrow curve combinations. It can be seen that both presented path planner implementations have the same problem in the narrow chicane of the track. For such narrow combination a long horizon is necessary such that the controller is already taking the next curve in consideration. This is however not possible with the current path planner due to its formulation.

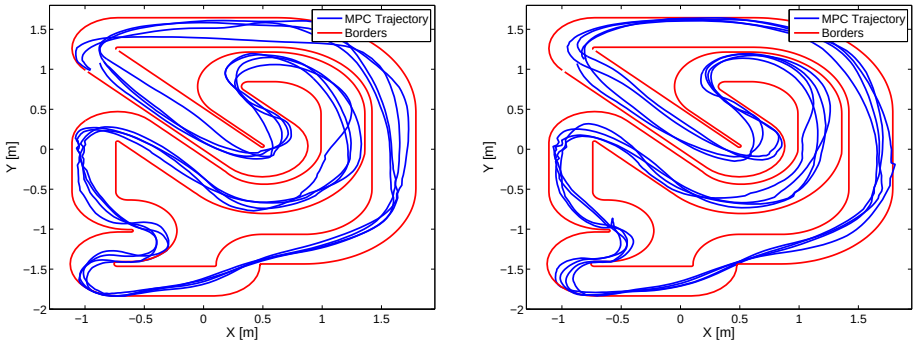

Figure 6.12: Four different trajectories of the controller, left using the standard path planner and right using the path planner with three radii in the horizon

One of the big advantages of the RHC, is that due to the path planner the car is now drifting if it is advantageous, and does not just react correctly if the car starts drifting. This leads to more and better visible drifts than when the PID path tracking controller is used. This can also be seen in Figure 6.13, where the side slip angle of the implementation with three radii in the horizon is shown.
The other big advantage is the online optimization of the velocity profile, which makes the tuning of the velocity profile redundant and leads to better velocity trajectories than the PID path tracking controller, see Figure 6.14.

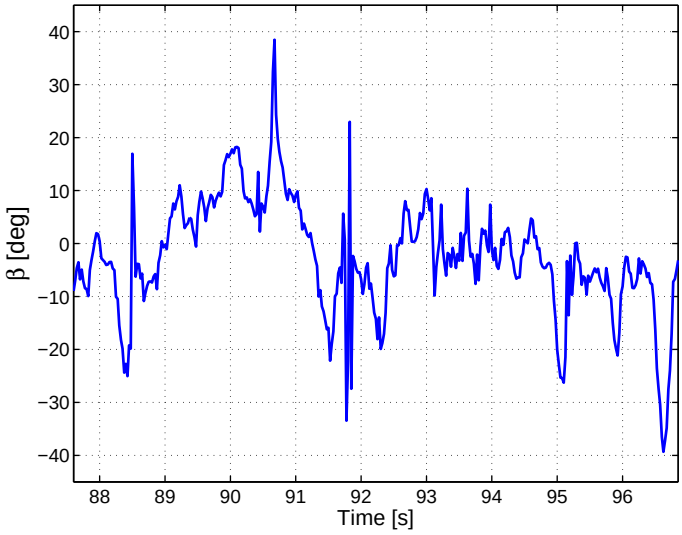

Figure 6.13: Side slip angle over one lap, using the path planner with three radii in the horizon

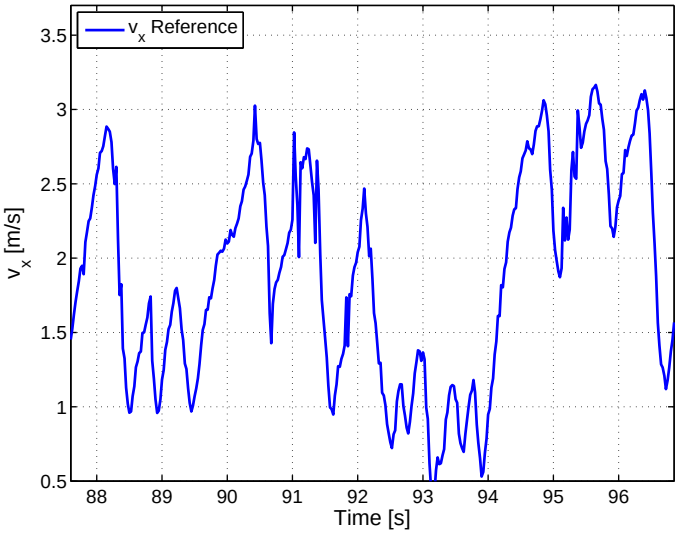

Figure 6.14: Forward velocity over one lap, using the path planner with three radii in the horizon

Chapter 7

Conclusion

In this thesis it is shown that it is possible to drift with the given dNano RC cars. To achieve this goal a lateral slip model, with non linear tire friction laws is used. The model is successfully identified with a focus to be able to represent drift. The identified non linear model is analyzed and the stationary velocity points of the model (stationary cornering conditions) are investigated. These points are the basis for all implemented control structures. Two different control approaches to control the car in normal as well as in the drifting region are derived. First a PID path tracking controller, and secondly a receding horizon controller augmented with a real time path planning. The first controller is a PID tracking controller which can follow a given path. The controller is derived so that it can be augmented with a drift detection algorithm. This makes it possible to use different steering controllers in order to control the car in the drifting and the normal driving region. The second controller is a receding horizon drift controller, consisting of a path planner which chooses the optimal trajectory out of a set of different stationary cornering trajectories. Thus the path planner can decide when it is time optimal to drift. The optimal trajectory is then tracked using a fast MPC controller. Beside the standard path planner where just one stationary velocity is allowed in the horizon, a more complex path planner is derived which allows multiple stationary velocities in the horizon, and so multiple radii. Both derived controllers are faster than all controllers previously presented in the ORCA project. It is shown that a drifting controller can be used to drive faster than when controllers using simpler models and control structures are used. The lap time of the PID path tracking controller is around 9.4 s and the lap time of the RHC controller is about 9.2 s, which is 23% faster compared to the existing controllers which have a minimal lap time of around 12 s. Experiments show that the predictive controller is superior to the non-predictive controller in most terms. The RHC has lower lap times and better drifting abilities, by drifting only when it is the optimal thing to do. Furthermore, the predictive controller can be used in a competitive racing environment due to its online path planning ability. The performance of the RHC is limited by the complexity of the online path planner, which cannot be increased due to computational limitation. However, the increase from one to three radii already reduces the lap time by 0.5 s from 9.7 to 9.2 s.

Chapter 8

Outlook

During the thesis several ideas came up which could not be investigated due to time limitation or hardware limitations. In the report already several ideas are mentioned, which together with possible future work are listed here.
An interesting future project would be to investigate combined slip models. This is not possible until now, but the necessary embedded hardware should very soon be available, which together with the wheel speed sensor developed in this project allows to use combined slip models. The same ideas as in the two derived drift controllers could be used to control the combined slip model. This should mainly increase the performance of the drift controller and help the steering controller, because the throttle input is no longer a disturbance for the steering controller.
There are several ideas to increase the robustness of the RHC and the overall performance. First, the performance and the robustness of the path planner could probably be increased by using more stationary model points for the path planner. This allows to use model based or generally harder constraints on which possible models are allowed based on the current measurement. If a model based approach is used to determine the constraints it is important that the drive train model needs to be identified accurately.
To increase the robustness of the MPC new border constraints could be derived, which have less feasibility problems than the currently used half space constraints.
The performance of the controller could be improved if the path planner were to be formulated as a convex optimization problem or maybe even by combining the path planner and the MPC to one optimization problem.
To race several cars against each other the RHC could be combined with the existing dynamic programming high level path planner, which allows to overtake competing cars.
As a last point, a high speed controller which does explicitly not try to drift could be derived, as a benchmark for the drift controller, to check if it is faster to drift or to keep the car inside the normal driving region. Such a controller could be a MPCC controller using soft constraints to prevent side slip angles which are higher than the saturation point of the force laws. However, this is only possible if a bicycle model using force laws is used and not using a no slip model.

Appendix A

Control of a Combined Slip Model

During this project, the control of a combined slip model is also partly investigated, but only in simulation. The most important points are summarized here.

A.1 Wheel speed sensor

A wheel speed sensor is derived, which uses a magnetic encoder principle to measure the wheel speed. The sensor uses a bipolar hall switch and a magnetic band with alternating plus and minus poles, see Figure A.1. The bipolar hall switch, switches to 1 if the magnetic field is higher than a certain threshold and stays on 1 until the magnetic field is below the same negative threshold. This hysteresis prevents the random switching around zero.

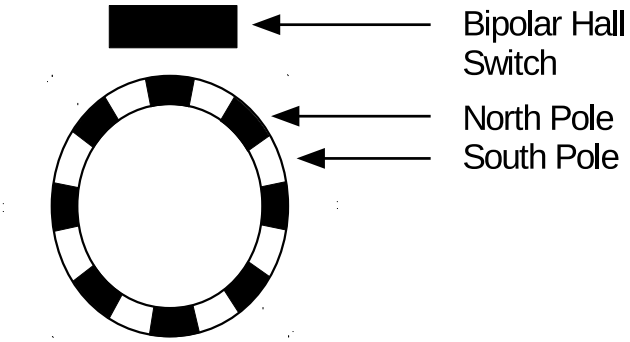

Figure A.1: Schematic image of the wheel speed sensor

The output of the sensor is a rectangular digital signal, which gives the position of the wheel, and by the width between the pulses also the speed of the wheel.

The prototype of the sensor has 6 periods, which allows a high update rate of the sensor.

A.2 Control Structure

In the used lateral slip model the duty cycle is the control input to control the forward velocity. In a combined slip model the wheel speed can be used as a control input, if additionally a cascaded wheel speed controller is used, see Figure A.2. This control structure has the advantage that a more complicated controller which determines the steering angle and the wheel speed can run at a slower computation time and a simple wheel speed controller can run on the embedded hardware at a high frequency, as already done for the steering angle.

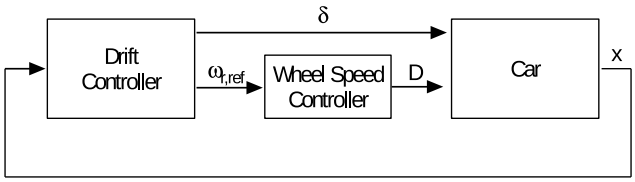

Figure A.2: Cascade control structure for combined slip control, using inner loop wheel speed controller

A.3 Stationary Cornering Conditions

The stationary cornering conditions are more complicated if a combined slip model is used. In the lateral model case there is only one control input of interest, in the combined slip case also the slip angle is of interest, because the relative velocity between the wheel and the ground influences the model.
Completely the same analysis as with the lateral model can be used by assuming the wheel speed to be the free rolling speed.

$$\omega_r = \frac{v_x}{r_r} \tag{A.1}$$

This is reasonable in the normal driving region, but in the drifting region this assumption is not valid, because drifting is characterized by significant wheel spin. To deal with this problem the wheel speed can be added to the calculation of the stationary points, which leads to maps for the stationary points instead of curves. The problem is still to find good drifting points on this map, which is not solved in a systematic fashion until now, and is an interesting open question.

Bibliography

[1] Advanced topic in control. *http://control.ee.ethz.ch/ ifaatic*, January 2012.

[2] M.N. Zeilinger M. Morari A. Domahidi, A. Zgraggen and C.N. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. 2012.

[3] Mujahid Abdulrahim. On the dynamics of automobile drifting. *Proceedings of the SAE World Congress*, 2006.

[4] R.Y. Hindiyeh C. Voser and J.C. Gerdes. Analysis and control of high sideslip maneuvers. *Proc of the Intl Association for Vehicle System Dynamics Stockholm*, 2009.

[5] C. Manzie D. Lam and M. Good. Model predictive contouring control. *Decision and Control CDC 2010 49th IEEE Conference*, 2010.

[6] L. Nyborg E. Bakker and H.B. Pacejka. Tyre modelling for use in vehicle dynamics studies. *Society of Automotive Engineers*, 1987.

[7] H.D. Tuan E. Ono, S. Hosoe and S. Doi. Bifurcation in vehicle dynamics and robust front wheel steering control. *IEEE Transactions on Control Systems Technology*, 1998.

[8] E. Frazzoli E. Velenis and P. Tsiotras. On steady-state cornering equilibria for wheeled vehicles with drift. *Proceedings of the 48h IEEE Conference on Decision and Control CDC held jointly with 2009 28th Chinese Control Conference*, 2009.

[9] E. Frazzoli P. Tsiotras E. Velenis, D. Katzourakis and R. Happee. Steady-state drifting stabilization of rwd vehicles. *Control Engineering Practice*, 2011.

[10] P. Tsiotras E. Velenis and J. Lu. Aggressive maneuvers on loose surfaces: Data analysis and input parametrization. *2007 Mediterranean Conference on Control Automation*, 2007.

[11] P. Tsiotras E. Velenis and J. Lu. Modeling aggressive maneuvers on loose surfaces : The cases of trail-braking and pendulum-turn. *2007 Mediterranean Conference on Control Automation*, 2007.

[12] J. Edelmann and M. Ploechl. Handling characteristics and stability of the steady-state powerslide motion of an automobile. *Regular and Chaotic Dynamics*, 2009.

[13] S. Karaman J.H. Jeon and E. Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the rrt*. *IEEE Conference on Decision and Control and European Control Conference*, 2011.

[14] T. Keviczky and G.J. Balas. Flight test of a receding horizon controller for autonomous uav guidance. *Proceedings of the 2005 American Control Conference*, 2005.

[15] J. Asgari H.E. Tseng P. Falcone, F. Borrelliand and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 2007.

[16] H.B. Pacejka. *Tire and Vehicle Dynamics*. Elsevier, 2006.

[17] P. Spengler and C. Gammeter. Modeling of 1:43 scale race cars. *ORCA Thesis*, 2010.

[18] C. Trabert. Implementation of model predictive contouring control for rc race cars. *ORCA Thesis*, 2012.

[19] E. Velenis. Fwd vehicle drifting control: The handbrake-cornering technique. *IEEE Conference on Decision and Control and European Control Conference*, 2011.

[20] E. Velenis and P. Tsiotras. Minimum time vs maximum exit velocity path optimization during cornering. *Proceedings of the IEEE International Symposium on Industrial Electronics 2005*, 2005.

[21] E. Velenis, P. Tsiotras, and J. Lu. Trail-braking driver input parameterizationfor general corner geometry. *SAE Technical Paper Series*, 2008.

[22] L. Wunderli. Mpc based trajectory tracking for 1:43 scale race cars. *ORCA Thesis*, 2011.

[23] S. You and S. Jeong. Controller design and analysis for automaticsteering of passenger cars. *Mechatronics*, 2012.