FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

Human Driver model for high performance closed loop vehicle simulations

Model lidského řidiče pro vysoce výkonné simulace řízení vozidel s uzavřenou smyčkou

Bc.Petr Velkoborský

Diplomová práce

Vedoucí práce: RNDr. Jana Königsmarková

Plzeň, 2024

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Petr VELKOBORSKÝ**
Osobní číslo:	**A21N0126P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Model lidského řidiče pro vysoce výkonné simulace řízení vozidel s uzavřenou smyčkou**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. State of the Art: Vehicle lateral-control algorithms, vehicle longitudinal control algorithms, combined (lat-long) control algorithms. Examine the general issues of trajectory tracking, matching optimal criteria, and control designs.
2. Define the criteria for designing a Driver-Model to follow a specified trajectory so that the resulting behavior is as similar as possible to human control. Formulate mathematically the conditions for the design of the driver-model, and select the optimal criterion meeting the requirements of the client.
3. Design a control method for the Driver-Model to optimally follow the car's route, considering several profile types of drivers. Verify this control against real car data.
4. Implement the Driver-Model in closed loop simulation environment, and verify the proposed solution for different modes. Compare the implemented solution with some corresponding existing solutions.

Rozsah diplomové práce:	**40-50 stránek A4**
Rozsah grafických prací:	
Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

Koga, Ayame, et al. "Realization of different driving characteristics for autonomous vehicle by using model predictive control." *2016 IEEE intelligent vehicles symposium (IV)*. IEEE, 2016.

Vedoucí diplomové práce:	**RNDr. Jana Königsmarková** Výzkumný program 1
Datum zadání diplomové práce:	**2. října 2023**
Termín odevzdání diplomové práce:	**20. května 2024**

Západočeská univerzita v Plzni * Fakulta aplikovaných věd *

L.S.

[illegible]

Doc. Ing. Miloš Železný, Ph.D.
děkan

Radová

Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

V Plzni dne 2. října 2023

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě diplomovou/bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou/diplomovou práci vypracoval(a) samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 20.5.2024

Velkoborský
....
vlastnoruční podpis

Abstrakt

Cílem této práce je návrh a implementace modelu řidiče, který splňuje požadované vlastnosti, specifikované ZF Engineering. Model řidiče by se měl pokud možno chovat co nejpodobněji jako lidský řidič. Navhrovaný model řidiče má sledovat předem definovanou trať s definovanou rychlostí. Samotná trasa neobsahuje provoz ani překážky. Model bude poté použit pro automatické testování na různých platformách, kde je třeba simulovat chování lidského řidiče.

Klíčová slova

Hardware in the loop; Software in the loop; Simulink environment; Autopilot; Linear quadratic regulator

Abstract

The aim of this thesis is designing and implementing a driver model that meets the required properties, specified by ZF Engineering. This driver model should behave in the same way as a human driver and follow a predefined track at a specified speed. The track itself does not contain traffic or obstacles. This model will be used for automatic testing on different platforms to simulate human-like driver behavior.

Keywords

Hardware in the loop; Software in the loop; Simulink environment; Autopilot; Linear quadratic regulator

Acknowledgement

Tímto bych chtěl poděkovat paní RNDr. Janě Königsmarkové za její čas, ochotu a rady, které mi během psaní mé diplomové práce poskytla. Také bych rád poděkoval společnosti ZF Engineering Plzeň, která mi poskytla zázemí pro psaní mé diplomové práce. V neposlední řadě bych rád poděkoval mým rodičům a partnerce, kteří mě velice podporovali během celého studia.

Contents

List of symbols and abbreviations

HIL	–	Hardware in the loop
SIL	–	Software in the loop
LQR	–	Linear quadratic regulator
TCU	–	transmission control unit
ECu	–	electronic control unit
LQR	–	Linear quadratic regulator
ADAS	–	advanced driver assistant system

List of Figures

Chapter 1

Introduction

1.1 Motivation

Testing a new product in the automative industry is very expensive for several reasons. For example, when testing a transmission, for every new development a new prototype has to be made and tested in a real environment where lots of potential problems can lead to the destruction of the product or even of the car. Another issue is that a real person is required to drive the car in a test scenario on a test track, which introduces new problems and increases costs. To mitigate those issues, as much of the testing as possible is done virtually in a simulation. When using this approach, there is no need to create a prototype after every small change, and it is possible to simulate hours of runtime in minutes. If errors occur, the simulation can be restarted and after the fix ran again. This is the main reason for designing this autopilot, eliminating as many errors as possible without the need for frequent test track runs risking damaging the car in the testing process. In general, there are a software in the loop (SIL) simulation and hardware in the loop (HIL) simulation. In SIL testing both the control unit and the car are simulated, whereas in HIL testing the control unit is not simulated, instead the real hardware, such as transmission control (TCU) unit or electronic control unit (ECU), is used and connected with cables to a special computer where the car is simulated. Using the HIL approach, the control unit should act as if it was in a real car. Both HIL and SIL simulations are closed loop because every controller has a feedback loop. When testing the autopilot, closed loop has to be present in the testing environment. The company ZF Engineering requested the development of a new advanced driver assistant system (ADAS). This system aims to help drivers drive more safely, and comfortably it also helps preventing crashes in the case of driver or hardware errors. Examples of established ADAS systems include cruise control or emergency braking. When using this system, there is still a person who drives the car, and this person influences the driving behaviour and therefore the feedback of the controller. For this reason, it is mandatory to have a simulation of a driver in the closed loop. The development of this simulated driver model is the aim of this thesis. ADAS systems are required to work in a wide range of possible human driving scenarios, so all of those scenarios need to be taken into account when testing and developing the driver simulation. For those reasons, the driver simulation (autopilot) should react the same way as a normal driver in several defined criteria.

1.2 Requirements

The main goal of this thesis is to make a driver model that behaves as a human driver as much as possible. Since not every human driver is the same, there should be a way of changing some parameters to represent different types of drivers. Therefore the car needs to be tested for multiple types of drivers. A professional rally driver would behave differently than somebody who has just received their driving license or somebody who has not been driving for a long time. The requirements have been agreed with ZF Engineering and have been defined as follows.

- Cutting corner coefficient - A way to make the model cut corners of a turn, or not.
- Lateral/Longitudinal tolerance - How much can the model deviate from the required speed and track.
- Maximal acceleration and speed of a steering wheel - A limitation how fast is the model able to turn the steering wheel.
- Maximal forces affecting the driver - Acceleration, deceleration, and lateral acceleration in turns.
- Reaction time - For the purpose of testing, the stability of the system when an error occurs by injecting some wrong input, for example, the wheel suddenly starts turning, the model should take some time to start compensating the wrong input.
- Pedal changing speed - How fast can the model go from using the accelerator to using the brakes.
- Acceleration/deceleration time - How long the model should be accelerating for or how soon before the turn the model should start decelerating.
- Look ahead - How far the driver looks in front of him. For example, when does the driver take into account the hill or the turn that is coming up.

The model should also provide several predefined setups, for example, an aggressive driver and a defensive driver, but at the same time, there should be a possibility to change the parameters independently. Some of these parameters can be set to unreasonable values or ignored. This way the simulation of a car trying to go through a turn at 100 km/h while on ice to test the functionality of the ADAS system is possible. Therefore, this thesis focuses on defining and implementing compact driver model. The implementation of the driver will be done in Matlab and Simulink environments as an independent subsystem. The inputs will be provided from the model of the car. The outputs of this subsystem will be:

- Steering wheel angle - Representing the driver's input to turn right or left.
- Acceleration pedal - Regulating how much torque should the motor provide.
- Break pedal - How much force should the brakes apply.

For testing purposes, there will be a track provided with defined points in space that represent the track. Each of these points also includes several parameters like the target speed, the friction and the inclination etc.

In general, a thorough study regarding driver models should be carried out to see if there is already an existing suitable one for our purposes.

Chapter 2

State of the ART

This chapter presents some of the existing implementations of autopilots that are available and explains the way in which the currently implemented autopilot works. The main focus of this thesis is to implement a new better autopilot that will control the speed and steering of the car given the reference velocity and path. The autopilot should also be as human-like as possible. There are lots of factors that affect the behaviour of a driver, for example, age or driving experience. For that reason, there should be some kind of parametrization available. The efficiency of the controller depends on the accuracy of the vehicle model. Planar vehicle models with lateral and longitudinal motion are commonly used in literature because of the simplicity of such models. The x-axis usually corresponds to a longitudinal motion and the y-axis corresponds to a lateral motion of the vehicle.[1]

There are lots of autopilot models and algorithms made for different purposes. Some of the most relevant autopilots, Cognibit, IPG driver, and VI Driver, that satisfy the criteria defined in section 1.2 are shown in the following sections. Different algorithms will be discussed.

2.1 Cognibit Drivebot

This autopilot is an AI-driven model implementing the typical human-like behaviour and problems, for example, limited field of view, reaction time or attention span. The drivebot from Cognibit takes into account all of the mentioned factors. However, this implementation is focused on autonomous driving. For the current specification, it is only necessary to follow some predefined tracks, this implementation is too complicated. There is no need for traffic simulation or lane changes. [2] For detailed information about the specifications see chapter. 1.2

2.2 IPG driver

Carmaker from IPG is very Similar to Cognibit Drivebot, it was designed for testing purposes, and it allows you to automatically control steering, braking, adjusting the gas pedal position, gear shifting, and clutch operation. Again, not everything is useful for the defined problem. However, there is some useful information, for example, the parameters implemented in the carmaker are useful for this thesis, corner cutting coefficient, the time needed to change the brake pedal and acceleration pedal, maximum acceleration and deceleration, and minimal acceleration of the car. There are also some default parameter values depending on the driver's aggressiveness that can be considered when designing a new model. For example, an aggressive driver will be cutting corners more, it will accelerate faster and brake faster than a defensive driver.[3]

2.3 VI driver

Another possibility is a VI driver. It is a very advanced driver model created by a VI-grade company. This model is more advanced than needed for the purposes of the task defined in this thesis, since it deals with problems that are not in the requirements.

To summarize all of the state-of-the-art models meet our requirements but are very expensive and more complex than needed (lane changing, traffic, etc.) for the aim of this thesis. Therefore, a new model designed for the requirements discussed in chapter 1.2 needs to be developed. [4]

2.4 Pure pursuit controller

This technique is often used due to its simplicity and convenience to implement it in any environment. Among other benefits, there are low computational requirements and satisfying performance at lower speeds. The main principle of this type of controller is calculating a point at a given distance ahead of the vehicle. Similar to the behaviour of a real driver. This distance is called look ahead distance. An arc is fitted between the look ahead point and the rear wheel of the car. This is shown in figure 2.1

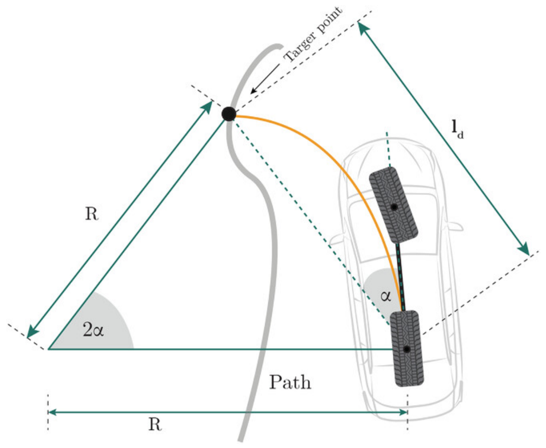

Figure 2.1: Pure pursuit functionality shown on a simple bicycle model.[1]

The main way to tune this controller is by changing the look ahead distance. Setting an excessive look ahead distance may cause the controller to cut corners. On the other hand, diminutive distance can positively influence the accuracy but may also cause some oscillations. Another factor influencing the behaviour and reliability of this controller is velocity. Since it is derived from a kinematic model it lacks the dynamic of the vehicle. Which is more apparent at higher speeds. To ensure stability, a range of speeds and look ahead distances was developed. Adaptive approaches can be used for tuning the look ahead distance. The main drawback of this controller is the limitation of plausible look ahead distances at different speeds, and not taking into account the orientation of the car at the target point. [1], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]

2.5 Stanley Controller

This controller is already implemented in ZF Engineering, therefore we it can be analyzed. It was developed for a 2005 DARPA Grand Challenge by the Stanford Racing team. They were able to win the race with the fastest average speed using this controller. This controller is designed to follow the track in hard terrain and between obstacles. It consists of two controllers, one lateral and one longitudinal, that control the speed and position of the car. The decoupling of those two controllers is possible because of the linear model used to design the controller. The longitudinal velocity is usually assumed constant and the tyre model is only accurate for small slip angles and low lateral accelerations. One of the major differences in using the Stanley controller is that the control action is generated using the front wheel orientation concerning the reference trajectory instead of the orientation of the whole body of the car. There are many iterations and innovations regarding Stanley controller. This autopilot has already been implemented in ZF Engineering. It is only analyzed and explained in this thesis to provide a brief overview. [1], [22], [8], [23]

As for the lateral control, the Stanley controller implements a nonlinear feedback law to follow a given path. The core of the controller is designed by using a kinematic model of a car, and it is improved by considering the dynamic equations of motion. [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

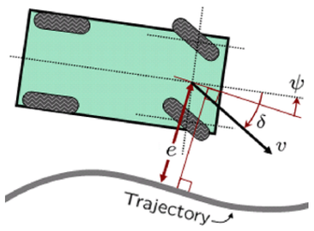

Figure 2.2: Geometry of a the vehicle model.[24]

The following parameters are used to describe the kinematic motion of an automobile, where $e(t)$ is the crosstrack error (distance to the nearest point in trajectory), $v(t)$ is the velocity $\Psi(t)$ is the yaw angle of the vehicle and $\delta(t)$ is the steering angle. As shown in the figure 2.2 Using those we can calculate the angle of the wheels with respect to the trajectory $(\Psi(t) - \delta(t))$ and come up with two kinematic equations. [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

$$\dot{e}(t) = v(t) \sin\left(\Psi(t) - \delta(t)\right) \tag{2.1}$$

$$\dot{\Psi}(t) = -\frac{v(t) \sin \delta(t)}{a+b}, \tag{2.2}$$

Where a and b are distances from the centre of gravity to the front and rear wheels.

Using those equations steering law is derived.

$$\delta(t) = \begin{cases} \Psi(t) + \arctan \frac{ke(t)}{v(t)} & if \quad |\Psi(t) + \arctan \frac{ke(t)}{v(t)}| < \delta_{max} \\ \delta_{max} & if \quad \Psi(t) + \arctan \frac{ke(t)}{v(t)} \geq \delta_{max} \\ -\delta_{max} & if \quad \Psi(t) + \arctan \frac{ke(t)}{v(t)} \leq -\delta_{max} \end{cases} \tag{2.3}$$

After substituting the control law equation 2.3 to the kinematic equation 2.1 we obtain

$$\dot{e}(t) = v(t) \sin \arctan\left(\frac{ke(t)}{v(t)}\right) = \frac{-ke(t)}{\sqrt{1 + \left(\frac{ke(t)}{v(t)}\right)^2}} \tag{2.4}$$

Using this control law when the crosstrack error is high (car is far away from the track), the controller steers the car directly back to the trajectory. As the crosstrack error gets smaller, (2.4) can be approximated to $\dot{e}(t) = -ke(t)$ with solution in time domain $e(t) = exp(-kt)$. That means when the vehicle is near the trajectory it converges to the track exponentially with time constant k. This can be seen in the control law when k is multiplied by $\frac{1}{v(t)}$.[24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

Using this controller, the angle of the wheel is controlled but not the yaw. At lower speeds tyres act as dampers and provide sideways force sufficient enough to stabilize the yaw dynamics. However, at higher speeds this damping force diminishes, and therefore active damping is needed. The author of the paper experimentally showed that the negative feedback on the yaw rate provides the best outcomes. Therefore $k_{dyaw}(r_{meas}(t) - r_{traj}(t))$ is added to the control law. Where k_{dyaw} is tunable gain, r_{traj} is the trajectory yaw and r_{meas} is the measured yaw rate.

The controller uses a steering servo as an actuator, therefore time delay and overshoot in this actuator can cause instability. This issue can be solved by adding $k_{dsteer}(\delta_{meas}(i) - \delta_{meas}(i+1)$ to the control law, where k_{dsteer} is another tunable parameter that is damping the steering wheel response and δ_{meas} is the discrete-time measurement of the steering angle and i is the index of the measurement. [24], [25], [26], [27], [28], [29], [30], [31], [32], [33] Another modification needs to be done since automobiles usually point inward on curves, generating lateral acceleration. Therefore, the controller yaw setpoint should be non-zero and a steady state yaw is introduced as follows

$$\Psi_{ss} = \frac{mv(t)r_{traj}(t)}{C_y(1 + \frac{a}{b})} \tag{2.5}$$

where C_y is tyre stiffness. One last modification needs to be done to prevent the term $\frac{k}{v(t)}$ from being large and becoming oversensitive to the noise of $e(t)$. To solve this problem the term k_{Soft} is added to the denominator. The final control law is.[24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

$$\delta(t) = (\Psi(t) - \Psi_{ss}) + \arctan\frac{ke(t)}{k_{Soft} + v(t)} + k_{dyaw}(r_{meas} - r_{traj}) + k_{dsteer}(\delta_{meas}(i) - \delta_{meas}(i+1)) \tag{2.6}$$

With saturation at $\pm\delta_{max}$. This controller can also be used while reversing using the rear tyres as the guiding wheels.[24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

Longitudinal control of the Stanley controller gets speed requests from the trajectory planner. It uses the throttle level and brake as two opposing actuators that provide a longitudinal force on the car. Experiments carried out in the paper showed that the brake system corresponds almost completely to reality and for the throttle, it is an acceptable simplification. It consists of PI regulator computing a single proportional integral error at discrete iteration.[24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

$$e_v(i+1) = k_{pv}(v(i+1) - v_C(i+1)) + k_{iv}e_{int}(i+1)) \tag{2.7}$$

The integral term is given as

$$e_{int}(i+1) = e_{int}(i+1) + (v(i+1) - v_c(i+1)) \tag{2.8}$$

v_C is the target speed and k_{pv} and k_{iv} are tunable parameters that influence the disturbance and overshoot of the controller. To prevent windup, the integral term is saturated. For a positive PI error, the brake is set proportional to the PI error. If it is negative, the throttle is set to the negative of the PI error.[24], [25], [26], [27], [28], [29], [30], [31], [32], [33]

To analyze the behaviour of the Stanley controller, it has been decided to use a double-lane change scenario as a baseline 2.3 since there was available comparable data from VI driver that it can be compared to.

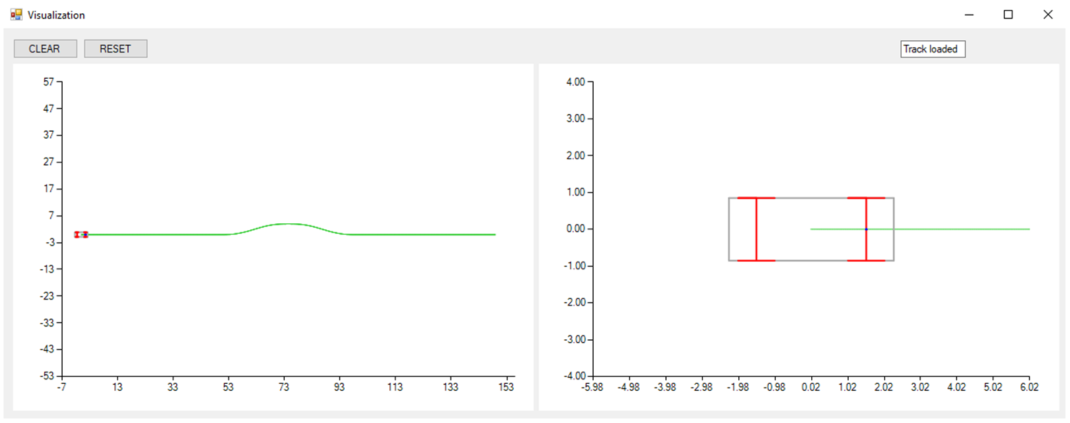

Figure 2.3: Double lane change scenario

As described above, there are three tunable parameters:

1. k_{dyaw} stabilizes yaw dynamics at higher speed.
2. k_{dsteer} damps the steering wheel response to get rid of instability.
3. k changes how fast the car converges to the track

To verify the influence of each parameter, scenarios of the double lane change at 80 km per hour were run using different values for each parameter. For each of the scenarios, the data was gathered and saved, and using Matlab, steer angle and steer rate were plotted to verify plausibility.

Parameter K:

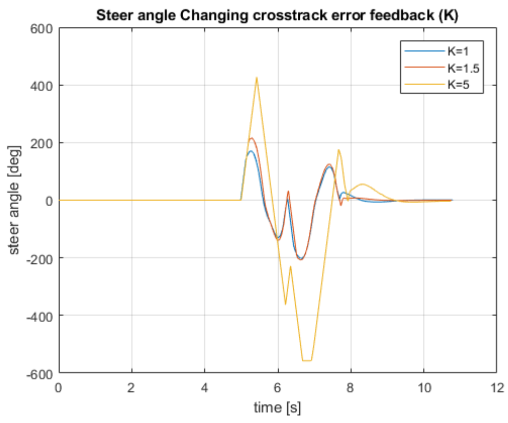

Figure 2.4: Steer angle for different K

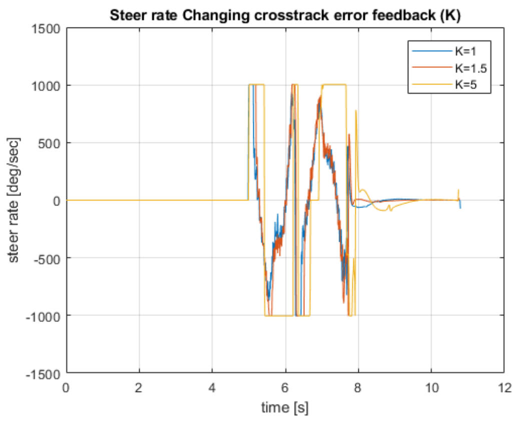

Figure 2.5: Steer rate for different K

As shown in figure 2.5 as the K gets higher the autopilot acts more aggressively which corresponds to the theory described in 2.5 The core of the controller is $\delta(t) = (\Psi(t) - \Psi_{ss}) + \arctan \frac{ke(t)}{k_{Soft}+v(t)}$ where $\arctan \frac{ke(t)}{k_{Soft}+v(t)}$ corrects the cross track error and $\delta(t) = (\Psi(t) - \Psi_{ss})$ corrects the heading error. Those two parts provide different outputs. The bigger the cross track error is, the more dominant (to the point of saturation) is the correction for cross track error (the car is going straight to the track). As the cross track error gets smaller, the second part starts to influence the outcome and the heading error begins to be corrected. That means that for high values of K even when the cross track error $e(t)$ is relatively small, the cross track error correction is still dominant, meaning the controller starts to correct for the heading error closer to the track, so the car converges faster to the track. When the K is too high, there is a risk of overshooting since there is not enough time to correct the heading error.

Parameter k_{dyaw}:

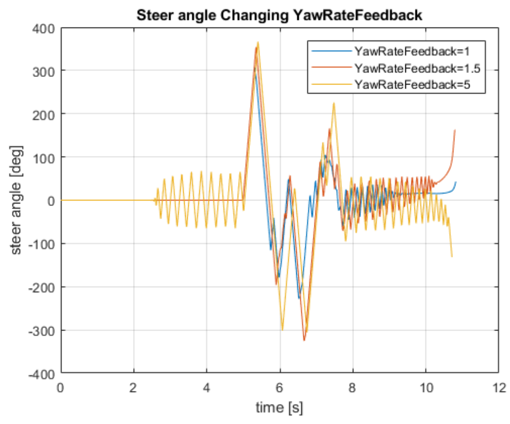

Figure 2.6: Steer angle for different k_{dyaw}

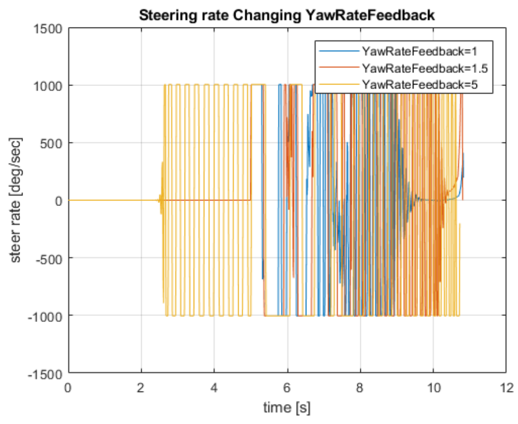

Figure 2.7: Steer rate for different k_{dyaw}

Changing parameter k_{dyaw} makes the system unstable. Therefore, using this parameter to change the behaviour of the system is inconvenient.

Parameter k_{dsteer}:

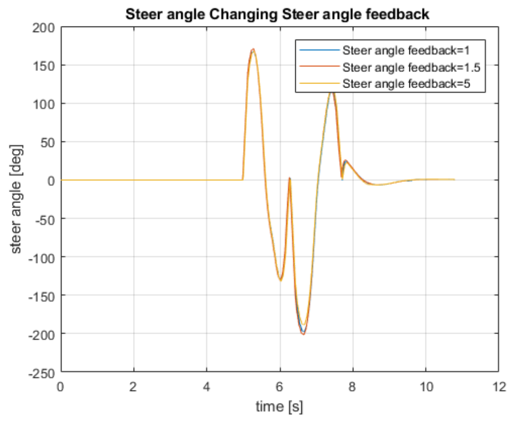

Figure 2.8: Steer angle for differentk_{dsteer}

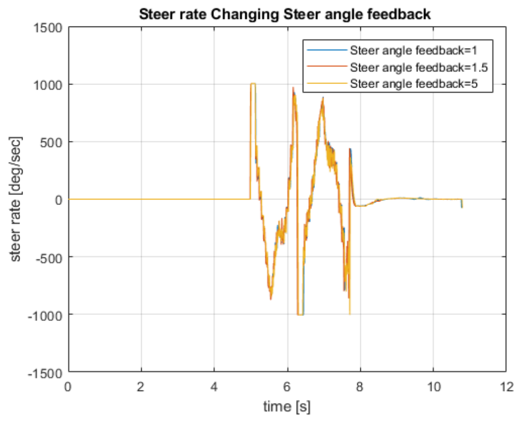

Figure 2.9: Steer rate for different k_{dsteer}

Changing parameter k_{dsteer} has almost no influence so using this parameter is inconvenient.

Using those tests, it has been shown that the only parameter that can be practically used to change the behaviour of the system is the gain (K). To investigate this further, the double lane change scenario with changing K was used. However, this time with changing velocities going from 20 km per hour to 80 km per hour. For reference, the cross track error was plotted and the sum of this error was calculated.

1. Velocity of 20 km per hour. For lower speeds, it was expected to get a more satisfying outcome.

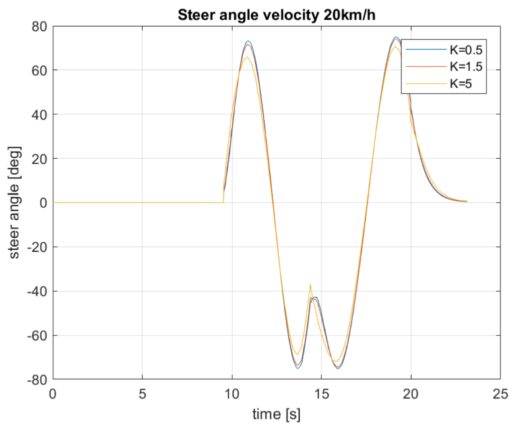

Figure 2.10: Steer angle for different K at 20 km per hour

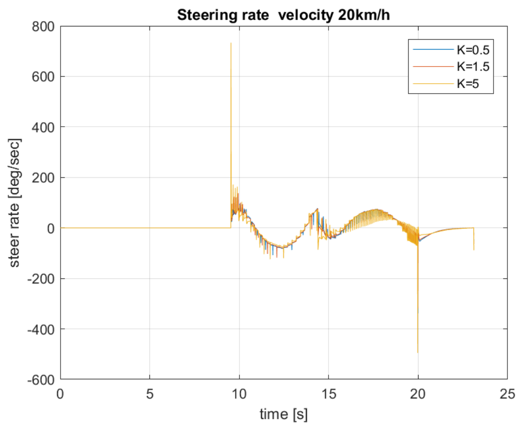

Figure 2.11: Steer rate for different K at 20 km per hour

As expected, the steering angle is significantly smaller since the action does not have to be that substantial at lower speeds. The steering rate gets saturated for a short time, for excessive K but that can be caused by the used model or a different steering ratio.

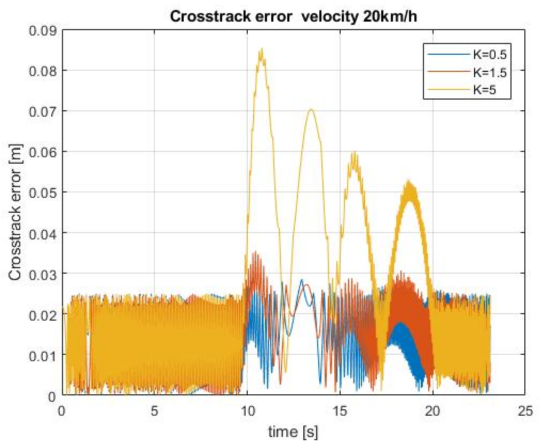

Figure 2.12: Cross track error for different K at 20 km per hour

We can see that the error is overall minor but increasing K leads to an increasing error. The sum of cross track error in meters was:

(a) K = 0.5 - 33.4 m

(b) K = 1.5 - 35.4 m

(c) K = 5 - 62.2 m

At 20 km per hour, the best results were for K = 0.5 so the smaller the K the better result.

2. Velocity of 40 km per hour.

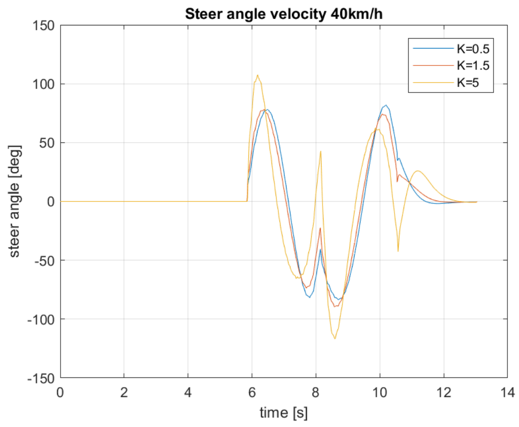

Figure 2.13: Steer angle for different K at 40 km per hour

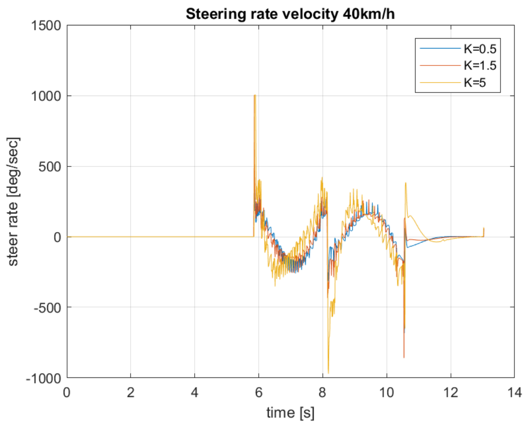

Figure 2.14: Steer rate for different K at 40 km per hour

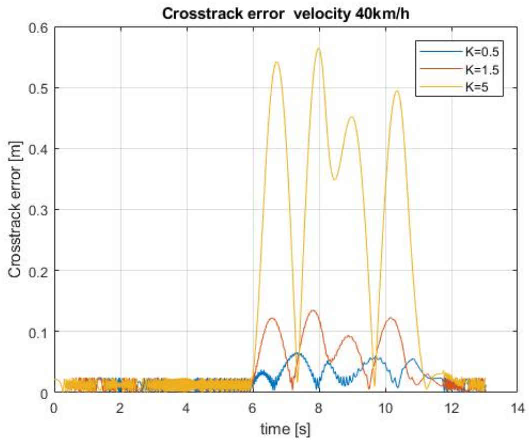

Figure 2.15: Cross track error for different K at 40 km per hour

(a) K = 0.5 - 29.8 m

(b) K = 1.5 - 49 m

(c) K = 5 - 183 m

At 40 km per hour, similar trend may be observed. The best results were again for K = 0.5 with an even lower cross track error sum than at 20 km per hour, which indicates that optimal K for 20 km per hour is probably even smaller.

3. Velocity of 60 km an hour.

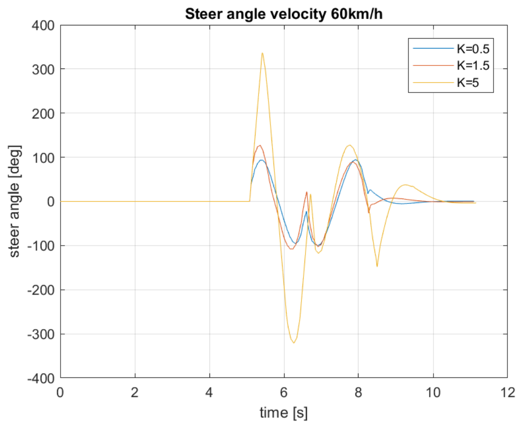

Figure 2.16: Steer angle for different K at 60 km per hour

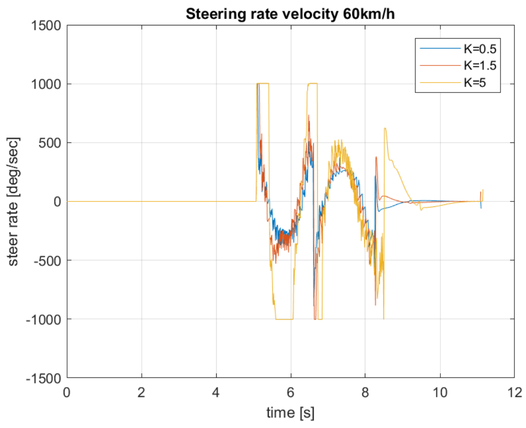

Figure 2.17: Steer angle for different K at 60 km per hour

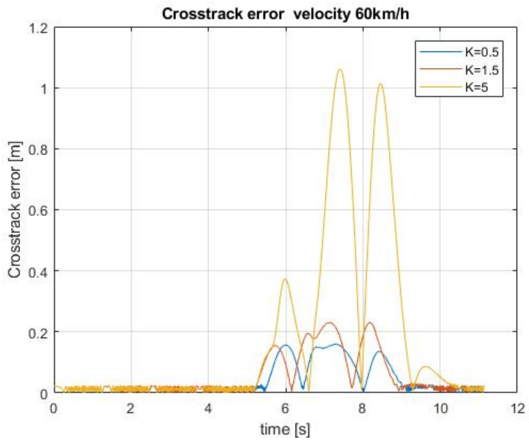

Figure 2.18: Cross track error for different K at 60 km per hour

A comparable tendency can be observed as with the lower speeds. The overall errors are increasing as expected.

(a) K = 0.5 - 47.2 m

(b) K = 1.5 - 60.1 m

(c) K = 5 - 198 m

At 60 km per hour, we can see a similar trend happening.
The best results were again for K = 0.5.

4. Velocity of 80 km an hour.

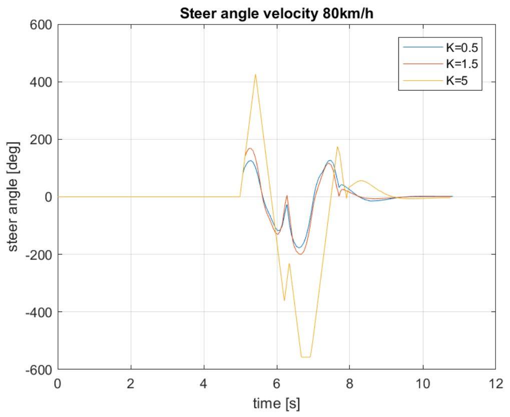

Figure 2.19: Steer angle for different K at 80 km per hour

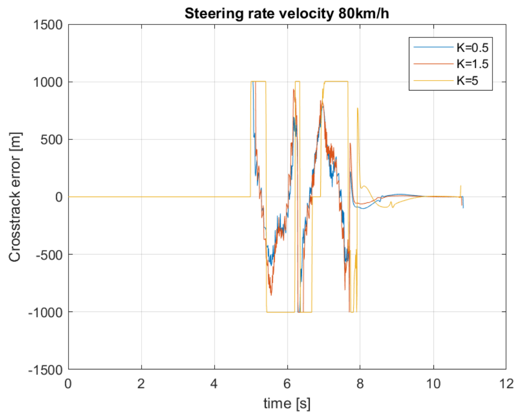

Figure 2.20: Steer rate for different K at 80 km per hour

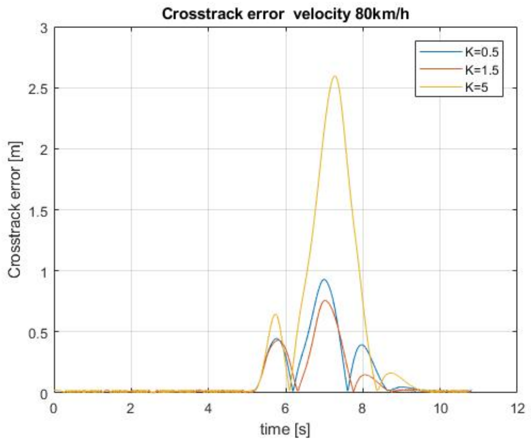

Figure 2.21: Cross track error for different K at 80 km per hour

A comparable tendency can be observed as with the lower speeds. In comparison with the values at lower speeds, the overall errors are increasing.

(a) K = 0.5 - 138 m

(b) K = 1.5 - 109.2 m

(c) K = 5 - 349.5 m

At 80 km per hour the behaviour changes. At this speed it becomes problematic to go through the double lane change and the best results are with K = 1.5.

Using those measurements is clear that the higher the velocity of the car, the bigger the error gets. As was shown to minimize the cross track error at a higher speed, K needs to be increased. However, if K is excessive an overshoot happens, and increases the error greatly, as shown in two following figures 2.22 and 2.23.

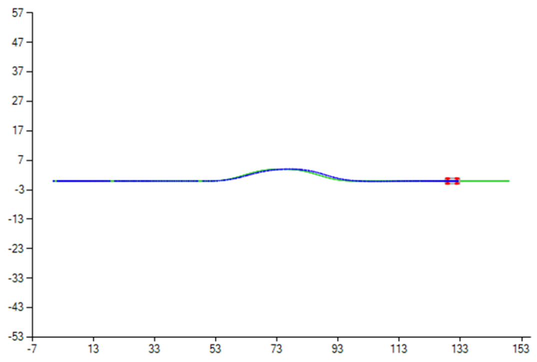

Figure 2.22: Trajectory of a car in double lane change scenario

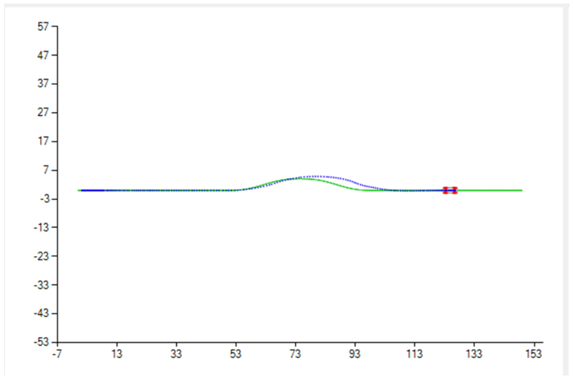

Figure 2.23: Trajectory of a car in double lane change scenario

To conclude, the aggressiveness of the driver may be influenced. However, several requirements can not be achieved, for example the reaction time. As shown in all the figures above, the autopilot reacts immediately regardless of the chosen parameter.

2.6 Linear quadratic regulator (LQR)

LQR uses a linear model to calculate optimal feedback control gain. This linearized model is expressed using state space representation in a form

$$\dot{x} = Ax + Bu \tag{2.9}$$

The goal of LQR is to find gain K that using feedback control law

$$u(k) = -Kx(k), \tag{2.10}$$

minimizes the following quadratic cost function

$$J = \sum_{k=0}^{\infty} (x^T(k) \cdot Q \cdot x(k) + u^T(k) \cdot R \cdot u(k)), \tag{2.11}$$

where Q and R are weighing matrices. Those matrices are diagonal positive and definite and are used to tune the controller. The optimal gain K is obtained using next formula

$$K = R^{-1} \cdot B^T \cdot P, \tag{2.12}$$

where P is the solution of the discrete Riccati matrix equation

$$P = A^T \cdot P \cdot A - (A^T \cdot P \cdot B) \cdot (R + B^T \cdot P \cdot B)^{-1} \cdot (B^T \cdot P \cdot A) + Q \tag{2.13}$$

The benefit of the LQR approach is that the conroller may be influenced by changing the diagonal of matrices Q and R, where each component on the diagonal of Q corresponds to a specific state. By increasing the value the controller focuses more on the error in that state. Similarly, in the R matrix each component on the diagonal corresponds to an input, and by increasing the value, the controller tries to use this input less.[1]

Chapter 3

Design of the driver model

In this chapter, the process of developing the LQR controller will be described. First, the model used for linearization will be described, and then the process of designing the controller itself.

3.1 Model of the car used for linearization

The model used in the simulation has six degrees of freedom. This model is unnecessarily complicated for the linearization that is required in the new controller. The simplified model was provided by ZF Engineering. It is a simple single-track model with some modifications. It is a dynamic model with seven states and two inputs. The states are

- X_c which is the position of the car in x axis.
- Y_c which is the position of the car in y axis.
- V_x which is the velocity of the car in x axis.
- V_y which is the velocity of the car in y axis.
- ψ which is the heading angle of the car.
- ω which is the change of the heading angle.
- δ which is the steering rate.

It can be written as a vector of states

$$X = [x_1, x_2, x_3, x_4, x_5, x_6, x_7] = [X_c, Y_c, V_x, V_y, \psi, \omega, \delta] \tag{3.1}$$

and the inputs are

- τ of the motor.
- $\dot{\delta}$ steering rate of the car.

That corresponds to the vector of inputs

$$U = [u_1, u_2]. \tag{3.2}$$

Those two outputs can be directly translated to the position of the acceleration/brake pedal and the steering wheel. The fundamental parameters of the car model were again provided by ZF Engineering.

- m = 2736 (vehicle mass [kg]).
- l_r = 1.491 (distance between the centre of gravity and the rear axle [m]).
- l_f = 1.528 (distance between the centre of gravity and the front axle [m]).
- r = 0.39 (tyre radius [m]).
- g = 9.81 (gravitational accelaration [m/s^2]).
- I_z = 4411.9 (moment of inertia [kgm^2]).
- C_r = 1.6
- C_f= 1.6
- B_r = 13
- B_f= 13

Where C and B are the tyre parameters.

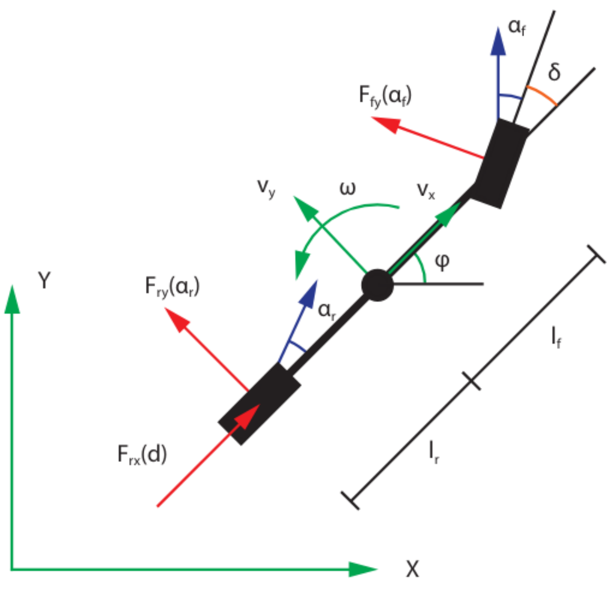

Figure 3.1: Schematic of the car model
[34]

The differential equations are described in 30 as follows:[34]

$$\dot{x_1} = x_3 \cdot cos(x_5) - x_4 \cdot sin(x_5) \tag{3.3}$$

$$\dot{x_2} = x_3 \cdot sin(x_5) - x_4 \cdot cos(x_5) \tag{3.4}$$

$$\dot{x_3} = a_x \tag{3.5}$$

$$\dot{x_4} = a_y \tag{3.6}$$

$$\dot{x_5} = \omega \tag{3.7}$$

$$\dot{x_6} = \dot{\omega} \tag{3.8}$$

$$\dot{x_7} = u_2 \tag{3.9}$$

Those equations were the slip angle and tyre forces were modelled using a simplified tyre model from Pacejka's article. Those equations are as follows. [35]
Slip angle:

$$\alpha_r = atan(\frac{x_4 - l_r \cdot x_6}{x_3}) \tag{3.10}$$

$$\alpha_f = atan(\frac{x_4 - l_r \cdot x_6}{x_3}) - x_7 \tag{3.11}$$

Tyre forces:

$$F_x = \frac{u_1}{r} \tag{3.12}$$

$$F_z = \frac{m \cdot g}{2} \tag{3.13}$$

$$FY_f = -sin(C_f \cdot atan(B_f \cdot \alpha_f)) \cdot F_z \tag{3.14}$$

$$FY_r = -sin(C_r \cdot atan(B_r \cdot \alpha_r)) \cdot F_z \tag{3.15}$$

Accelerations:

$$\dot{x_3} = \frac{F_x - FY_f \cdot sin(x_7) + m \cdot x_4 \cdot x_6}{m} \tag{3.16}$$

$$\dot{x_4} = \frac{FY_r + FY_f \cdot cos(x_7) - m \cdot x_3 \cdot x_6}{m} \tag{3.17}$$

$$\dot{x_6} = \frac{FY_f \cdot l_f \cdot cos(x_7) - FY_r \cdot l_r}{I_z} \tag{3.18}$$

To get the system in state space form:

$$\dot{x} = A \cdot x + B \cdot u \tag{3.19}$$

It is needed to calculate the matrix A and B. To do that derivatives of the slip angle in front and in the rear, and the derivatives of forces with respect to all the states need to be calculated. Slip angle in the front:

$$\alpha_f = atan(\frac{x_4 + l_f \cdot x_6}{x_3}) - x_7 \tag{3.20}$$

$$\frac{\partial \alpha_f}{\partial x_3} = \frac{1}{\frac{(x_4 + l_f \cdot x_6)^2}{x_3^2} + 1} \cdot \frac{-\frac{x_4 + l_f \cdot x_6}{x_3}}{x_3} \tag{3.21}$$

$$\frac{\partial \alpha_f}{\partial x_4} = \frac{1}{(\frac{(x_4 + l_f \cdot x_6)^2}{x_3^2} + 1) \cdot x_3} \tag{3.22}$$

$$\frac{\partial \alpha_f}{\partial x_6} = \frac{l_f}{(\frac{(x_4 + l_f \cdot x_6)^2}{x_3^2} + 1) \cdot x_3} \tag{3.23}$$

$$\frac{\partial \alpha_f}{\partial x_7} = -1 \tag{3.24}$$

Slip angle in the rear:

$$\alpha_r = atan(\frac{x_4 - l_r \cdot x_6}{x_3}) \tag{3.25}$$

$$\frac{\partial \alpha_r}{\partial x_3} = \frac{1}{\frac{(x_4 - l_r \cdot x_6)^2}{x_3^2} + 1} \cdot \frac{-\frac{x_4 - l_r \cdot x_6}{x_3}}{x_3} \tag{3.26}$$

$$\frac{\partial \alpha_r}{\partial x_4} = \frac{1}{(\frac{(x_4 - l_r \cdot x_6)^2}{x_3^2} + 1) \cdot x_3} \tag{3.27}$$

$$\frac{\partial \alpha_r}{\partial x_6} = \frac{-l_r}{(\frac{(x_4 - l_r \cdot x_6)^2}{x_3^2} + 1) \cdot x_3} \tag{3.28}$$

Forces:

Force in z axis:

$$F_z = \frac{m \cdot g}{2} \tag{3.29}$$

Lateral force front

$$FY_f = -sin(C_f \cdot atan(B_f \cdot \alpha_f)) \cdot F_z \tag{3.30}$$

$$\frac{\partial FY_f}{\partial \alpha_f} = \frac{-F_z \cdot B_f \cdot C_f \cdot cos(C_f \cdot atan(B_f \cdot \alpha_f))}{{B_f}^2 \cdot {\alpha_f}^2 + 1} \tag{3.31}$$

Lateral force rear

$$FY_r = -sin(C_r \cdot atan(B_r \cdot \alpha_r)) \cdot F_z \tag{3.32}$$

Longitudinal force

$$\frac{\partial F_x}{\partial u1} = \frac{1}{r} \tag{3.33}$$

Matrices A and B may now be constructed to obtain the state space representation of the system. Both matrices are constructed using partial derivatives of each state.

$$A = \left[\begin{array}{ccccc}
0 & 0 & cos(x_5) & -sin(x_5) & ... \\
0 & 0 & sin(x_5) & cos(x_5) & ... \\
0 & 0 & \frac{-sin(x_7)\cdot\frac{\partial FY_f}{\partial x_3}}{m} & \frac{-sin(x_7)\cdot\frac{\partial FY_f}{\partial x_4}}{m} + x_6 & ... \\
0 & 0 & \frac{\frac{\partial FY_r}{\partial x_3}+\frac{\partial FY_f}{\partial x_3}\cdot cos(x_7)}{m} - x_6 & \frac{\frac{\partial FY_r}{\partial x_4}+\frac{\partial FY_f}{\partial x_4}\cdot cos(x_7)}{m} & ... \\
0 & 0 & 0 & 0 & ... \\
0 & 0 & \frac{\frac{\partial FY_f}{\partial x_3}\cdot l_f\cdot cos(x_7)-\frac{\partial FY_r}{\partial x_3}\cdot l_R}{I_z} & \frac{\frac{\partial FY_f}{\partial x_4}\cdot l_f\cdot cos(x_7)-\frac{\partial FY_r}{\partial x_4}\cdot l_R}{I_z} & ... \\
0 & 0 & 0 & 0 & ...
\end{array}\right.$$

$$\left.\begin{array}{ccc}
-x_3\cdot sin(x_5) - x_4\cdot cos(x_5) & 0 & 0 \\
x_3\cdot cos(x_5) - x_4\cdot sin(x_3) & 0 & 0 \\
0 & \frac{-sin(x_7)\cdot\frac{\partial FY_f}{\partial x_6}}{m} + x_4 & \frac{-sin(x_7)\cdot\frac{\partial FY_f}{\partial x_7 - FY_f\cdot cos(x_7)}}{m} \\
0 & \frac{\frac{\partial FY_r}{\partial x_6}+\frac{\partial FY_f}{\partial x_6}\cdot cos(x_7)}{m} - x_3 & \frac{\frac{\partial FY_f}{\partial x_7}\cdot cos(x_7) - FY_f\cdot sin(x_7)}{m} \\
0 & 1 & 0 \\
0 & \frac{\frac{\partial FY_f}{\partial x_6}\cdot l_f\cdot cos(x_7)-\frac{\partial FY_r}{\partial x_6}\cdot l_R}{I_z} & \frac{\frac{\partial FY_f}{\partial x_7}\cdot l_f\cdot cos(x_7) - FY_f\cdot l_f\cdot sin(x_7)}{I_z} \\
0 & 0 & 0
\end{array}\right] \tag{3.34}$$

The partial derivatives were defined by using 3.21 to 3.32 as:

$$\frac{\partial FY_r}{\partial x_3} = \frac{\partial FY_r}{\partial \alpha_r}\cdot\frac{\partial \alpha_r}{\partial x_3} \tag{3.35}$$

$$\frac{\partial FY_r}{\partial x_4} = \frac{\partial FY_r}{\partial \alpha_r}\cdot\frac{\partial \alpha_r}{\partial x_4} \tag{3.36}$$

$$\frac{\partial FY_r}{\partial x_6} = \frac{\partial FY_r}{\partial \alpha_r}\cdot\frac{\partial \alpha_r}{\partial x_6} \tag{3.37}$$

$$\frac{\partial FY_f}{\partial x_3} = \frac{\partial FY_f}{\partial \alpha_f}\cdot\frac{\partial \alpha_f}{\partial x_3} \tag{3.38}$$

$$\frac{\partial FY_f}{\partial x_4} = \frac{\partial FY_f}{\partial \alpha_f}\cdot\frac{\partial \alpha_f}{\partial x_4} \tag{3.39}$$

$$\frac{\partial FY_f}{\partial x_6} = \frac{\partial FY_f}{\partial \alpha_f}\cdot\frac{\partial \alpha_f}{\partial x_6} \tag{3.40}$$

$$\frac{\partial FY_f}{\partial x_7} = \frac{\partial FY_f}{\partial \alpha_f}\cdot\frac{\partial \alpha_f}{\partial x_7} \tag{3.41}$$

The same can be done to get matrix B.

$$J_B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m \cdot r} & 0 \\ \frac{x_7 \cdot l_r}{r \cdot m \cdot (lr+lf)} & \frac{-x_3 \cdot l_R}{lr+lf} \\ 0 & 0 \\ \frac{x_7}{r \cdot m \cdot (lr+lf)} & \frac{x_3}{lr+lf} \\ 1 & 0 \end{bmatrix}$$

Using the matrices A and B the linearization can be calculated in each step of the simulation. Then with the linearized model, the LQR controller is used to control the original system, by updating the feedback gain in every time step. The first implementation of the LQR controller could be done in Matlab to test plausibility before implementing it in Simulink. In this test, the lqr function was used to calculate the feedback gain from the linearized matrices A and B. Matrix Q and R were used to change the behaviour of the LQR controller. For this first test, the matrices Q and R were chosen as follows:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

In Matlab, a simple circular track with a defined centre and diameter was used as the reference. All the references were calculated in each step of the simulation. Reference for the x and y position is just the closest point on the circle. Heading angle:

$$\psi = atan2(y - y_{circle}, x - x_{circle}) \tag{3.42}$$

Where y_{circle} and x_{circle} are the coordinates of the centre of the circle. To get the ψ in a range from 0 to $2 \cdot \pi$ it is required to modulate it:

$$\psi = mod(\psi + \frac{\pi}{2}, 2 \cdot \pi) \tag{3.43}$$

The reference speed of the car V_X is constant. And in the y axis is set to 0. The yaw rate is calculated as follows:

$$\omega = \frac{V}{r} \tag{3.44}$$

Where V is the reference speed and r is the radius of the circle. And lastly, the steering rate is calculated using the next formula:

$$\delta = \frac{\omega \cdot l_f}{V_x} \tag{3.45}$$

Using this test, the functionality of the controller in a simplified environment with a simplified track can be validated.

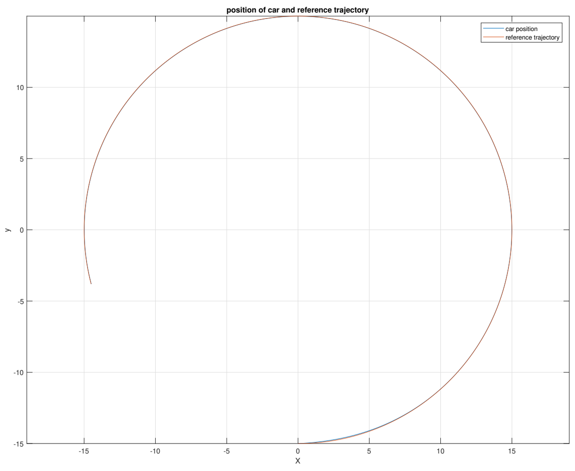

Figure 3.2: Car following a circular trajectory with constant radius

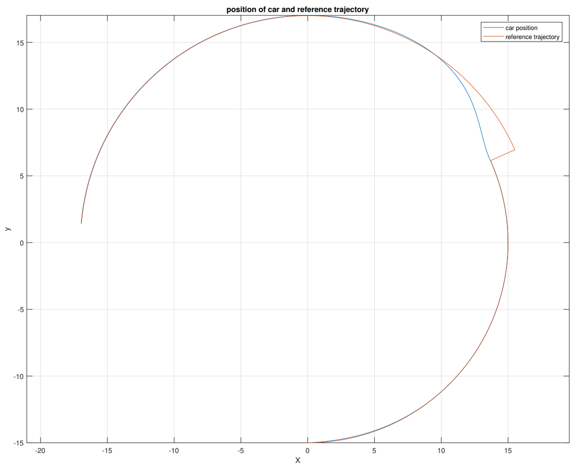

Figure 3.3: Car following a circular trajectory with change in radius

The functionality of the controller can be seen in graphs 3.2 and 3.3. As shown there the car follows the track quite neatly, and when there is a jump in radius it corrects for that. Other tests were ran to verify the functionality of the controller before implementing it in the Simulink environment. For example, driving around the circle with changing speeds or changing the initial position. Furthermore, the behaviour of the eigenvalues when changing parameters of the Q matrix was analyzed at this point to help with the tuning in the future. The next step was to implement this controller in the Simulink environment and integrate it into the existing model provided by ZF Engineering. This model is very complex and this autopilot is just a small part of it. All the necessary signals such as the position of the car or the reference are provided by this model. After the controller is incorporated in Simulink it can be further compiled and then run in softcar, which is a software design for SIL testing developed by ZF Engineering.

Chapter 4

Implementation of the Driver-Model in a closed-loop simulation environment

A diagram of the Controller and the model with all corresponding signals is shown in the figure 4.1, as it will be integrated in simulink. This controller can be used later as a subsystem and inserted into the whole model.

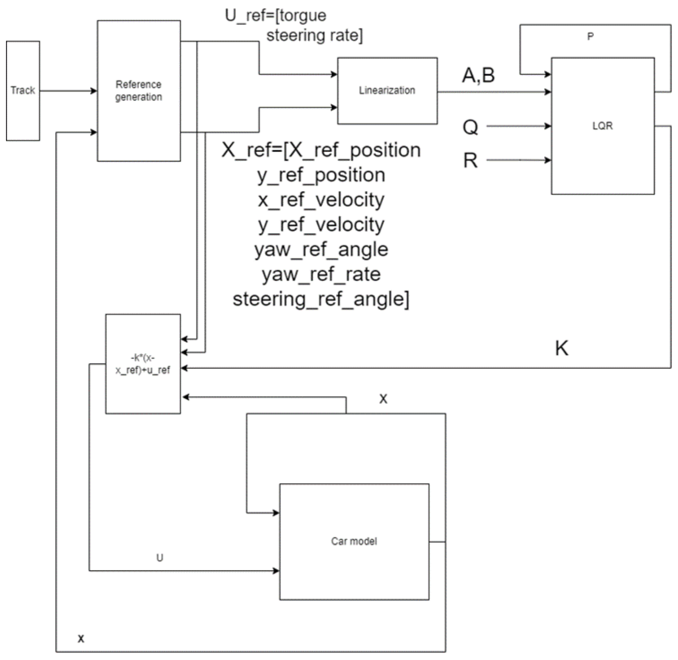

Figure 4.1: Block diagram of the planned controller

4.1 Integration to Simulink

First, the controller needed to be integrated in Simulink. There were several issues that had to be considered. For example, LQR function from Matlab can not be used in Simulink and references have to be calculated from the track that has been generated. To use LQR online in Simulink, it was necessary to solve the Riccaty equation. For that purpose, Newton method was used that which returns the local minimum. Since a global minimum is needed, the method was initialized at the beginning with a precalculated solution to get the global minimum. In each step, the solution of the Riccati equation from the previous step is used as the initial value. This approach was soon discovered to be unpractical and not robust enough. Using this iterative method was fast but any time controlability was lost. There was a risk that the algorithm could not recover. For that reason a new method was introduced providing an exact solution for the Riccati equation, using the Hamiltonian matrix and subspace method.[36] This method provides an exact solution but it is more complex. Therefore, this method is used only at the beginning of the simulation, when controllability is lost or when the Newton method does not converge to a solution after a certain number of iterations. Some computational time may be saved this way while still providing a robust controller. This setup was again tested in a Simulink environment on a circular track. Another issue came in the form of a jump in reference angle. The reference heading angle ψ was generated in a range between $-180\,\text{deg}$ to $180\,\text{deg}$ with respect to the x and y axis. This caused an issue when the car drove perpendicularly to the X axis in a negative direction. At this point the reference heading angle jumped from $180\,\text{deg}$ to $-180\,\text{deg}$. This was calculated as a deviation of 360 degrees which led to controller steering rapidly. This was solved by changing the reference frame of the heading angle to the car. The error has been calculated by using the next formula

$$diff = -atan2(\sin(heading - heading_{ref}), \cos(heading - heading_{ref})) \tag{4.1}$$

The reference heading is the error and the current heading is always zero. The available references from the track were only the position and velocity of the car and the heading angle and curvature. The missing references were needed to calculate, velocity in each axis, yaw rate and steering rate. Kinematic equations were used for simplicity. The velocity in the y axis was set to 0 and the reference velocity in the x axis was set to the reference velocity of the car. The ω was calculated as follows:

$$\omega = V_{car} \cdot curvature \tag{4.2}$$

end δ:

$$\delta = (\omega * lf)/v_{car} \tag{4.3}$$

Where lf is the distance from the centre of gravity to the front axle. When the issues above were solved it was possible to compile the Simulink model and run the simulation in a SIL environment.

Using the built-in visualization, the behaviour of the car could be seen and compared to the outcome of the Stanley controller.

4.2 Tuning of the LQR parameters

To tune the controller, initial values for Q and R were calculated using the Bryson rule. After that manual tuning needed to be done. It was advantageouos that each parameter on the diagonal of the matrices corresponds to one state or input. The tuning matrices Q and R are as follows. Also, the influence on the eigenvalues of each parameter was plotted to help with the tuning.

$$Q = \begin{bmatrix} Q_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Q_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & Q_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Q_7 \end{bmatrix}$$

$$R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

Where Q_1 corresponds to state x_1 etc. And R_1 corresponds to input u_1. For example, by increasing R_2 the steering rate can be penalized, which can help to get rid of oscillations, or by increasing R_1 and R_2 which corresponds to the position states, the controller can be forced to follow the track more precisely. The goal is to balance those parameters to satisfy the requirements defined in 1.2 because, for example, increasing the precision in track following leads to more oscillation in steering rate. We can use this knowledge to fulfill the requirements defined in 1.2

- Lateral/Longitudinal tolerance Cutting corner coefficient - This can be controlled by increasing the parameters Q_1, Q_2 and R_2. Making the Q parameters higher makes the controller follow the track more precisely and decreasing the R parameter penalizes the steering rate less. Other parameters also influence this behaviour. However, these three are the most relevant ones.

- Maximal acceleration and speed of a steering wheel - Parameter saturating the steering rate was added.

- Maximal forces affecting the driver - Acceleration, deceleration, and lateral acceleration in turns.

- Reaction time - A delay of the current data for the controller can be introduced.

- Pedal changing speed - Since the braking pedal and acceleration are both represented as 1 input (torque). Where positive torque means acceleration negative torque means breaking. A rate limiter with parameter controlling the rate of change of torque can be used.

- Acceleration/deceleration time and Look ahead - Look ahead parameter is used. This parameter is in seconds and based on current speed it is calculated the future position. Taking into account the curvature on the track at that position the reference speed is set so the car can go safely through the turn.

Using the following parameters

$$Q = \begin{bmatrix} 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 6 \end{bmatrix}$$

The car was going around the track quite neatly as can be seen in the figures 4.2 and 4.3

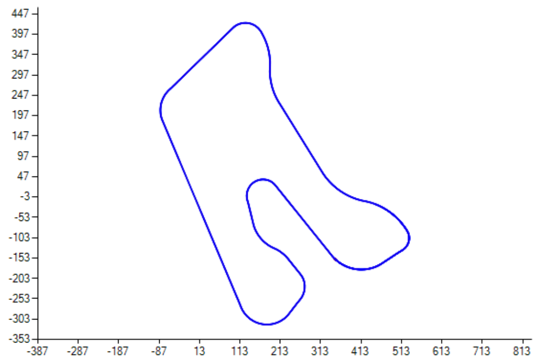

Figure 4.2: Car following a trajectory (Hockenheim race track

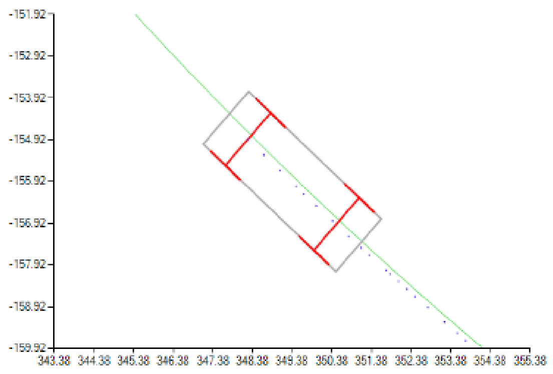

Figure 4.3: Car on track zoomed

Chapter 5

Discussion

To verify the new autopilot it was tested on the same scenario as the Stanley controller. It was a double-lane change scenario with constant velocity. 40 km per hour was used as a baseline with different parameters of Q and R matrixes. For the first test, a set of parameters representing an aggressive driver was used meaning that the controller tried to follow the track as nicely as possible while sacrificing high steering rates. The parameters were.

$$Q = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 3 \end{bmatrix}$$

In this case, the sum of cross track errors was 20.4 meters. When compared to the Stanley controller 2.12 the cross track error is smaller and the car follows the trajectory better. The sum of cross track error for Stanley controller was 33.4 metres which again shows better performance of LQR. This outcome was balanced by small oscillations in steering rate, that are still much better then in the original Stanley controller.

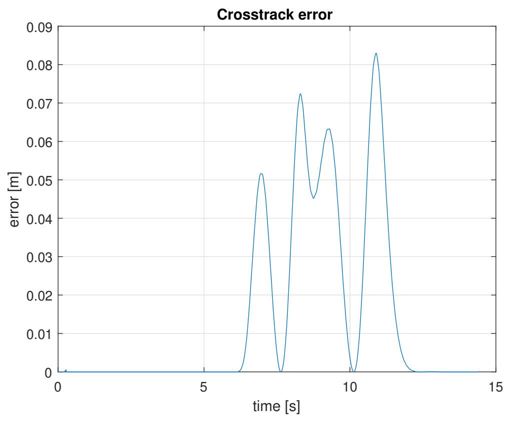

Figure 5.1: Cross track error using LQR aggressive driver

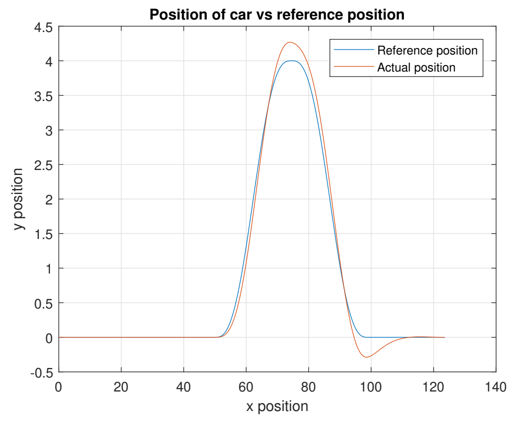

Figure 5.2: Position of the car vs the reference position aggressive driver

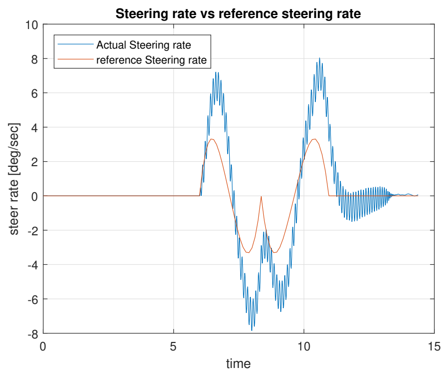

Figure 5.3: Steering rate of the aggressive driver

The main benefit in using LQR instead of Stanley controller is in the posibility to change the behaviour. For the next test, a casual driver with following parameters was used for the next test:

$$Q = \begin{bmatrix} 7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 6 \end{bmatrix}$$

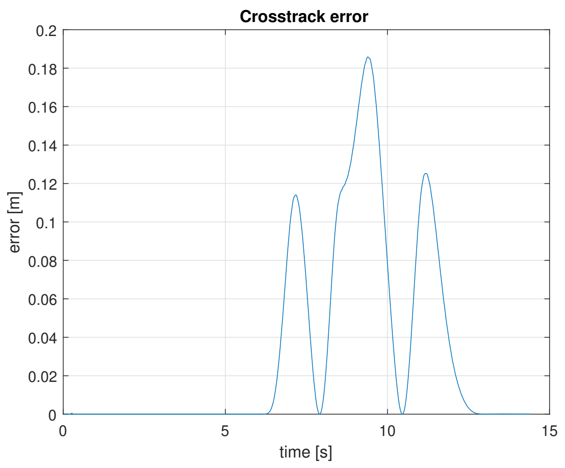

Figure 5.4: Cross track error using LQR with casual driver

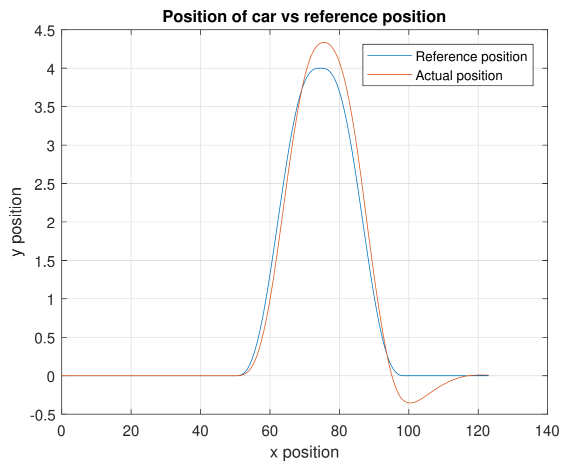

Figure 5.5: Position of the car vs the reference position

In this case, the sum of cross track errors was 46.6 meters. Slightly lower steering rate and oscillations are shown in figure 5.6.

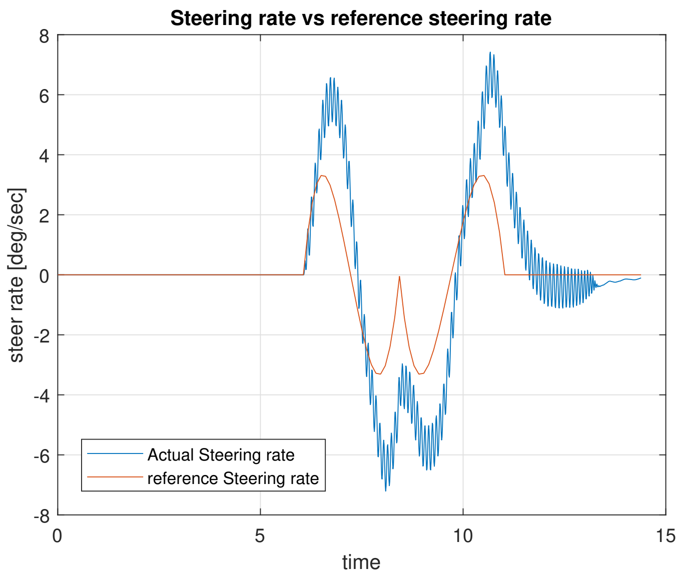

Figure 5.6: Steering rate of the casual driver

Lastly, for more passive driver the steering rate has to be as small as possible. However, by implementing that, the cross track error increased.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.01 & 0 \\ 0 & 20 \end{bmatrix}$$

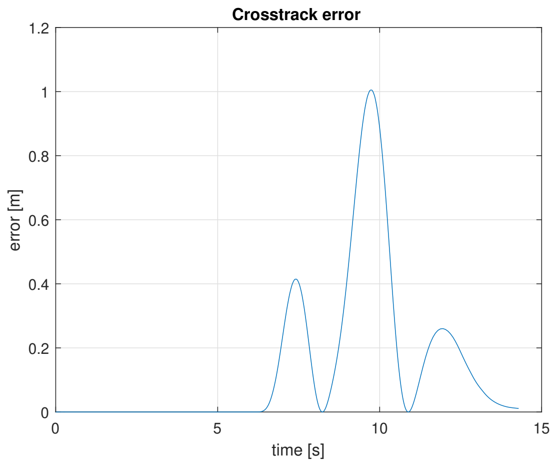

Figure 5.7: Crosstrack error using LQR with passive driver

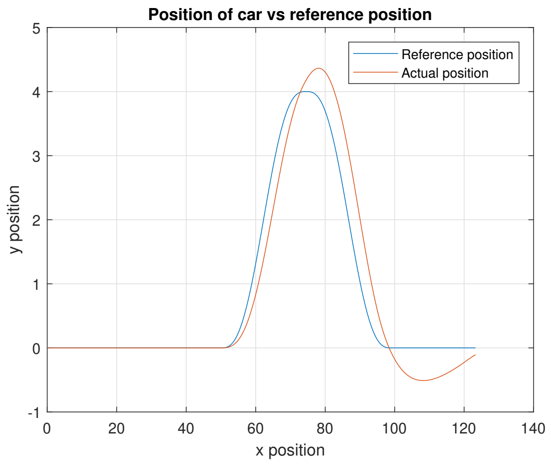

Figure 5.8: Position of the car vs the reference position

In this case, the sum of cross track errors was 202.6 meters. Which is quite a lot, at one point the error is over 1 meter, but as you can see the steering rate is not that aggressive with lower oscillations. By changing the parameters the oscillations may be eliminated whatsoever, however doing so increases the cross track error to 2.5 meters at some points. Which is unrealistic while doing double lane change at 20 km/h.

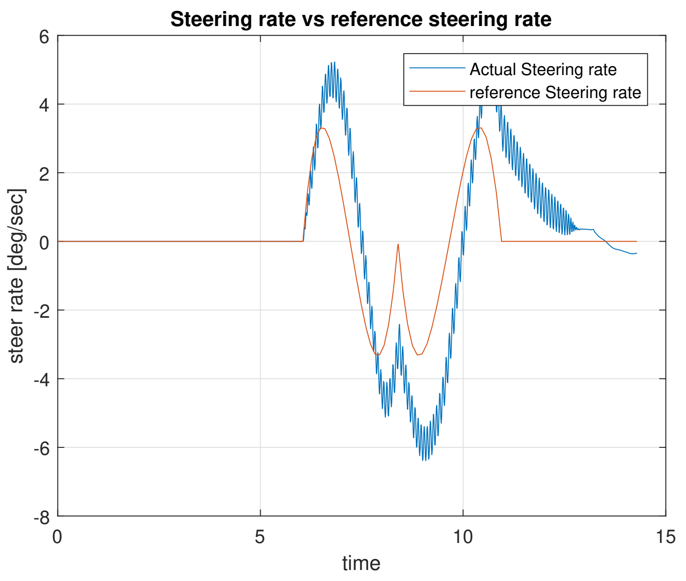

Figure 5.9: Steering rate of the passive driver

As you can see, by changing the parameters we can change the behaviour of the controller quite drastically. Those parametrizations might change in the future depending on the feedback from ZF Engineering.

Chapter 6

Conclusion

The aim of this thesis was to formulate criteria for the design of the driver model. Subsequently, to design a method to control the car to follow a specific route while considering several types of driver profiles. Finally, the driver model was implemented in a closed-loop simulation environment. The influence of the parameters was analyzed and several parameterizations were set.

This thesis was created with the cooperation of ZF Engineering and the autopilot is implemented in a bigger model that will be used for testing. Since the whole model and specifically the track generation and the physical model of the car is still in progress, the controller will probably require some adjustments or upgrades. For this reason, my thesis reflects only the current state. That may change change in the future.

Bibliography

1. ROKONUZZAMAN, Mohammad; MOHAJER, Navid; NAHAVANDI, Saeid; MOHAMED, Shady. Review and performance evaluation of path tracking controllers of autonomous vehicles. *IET Intelligent Transport Systems* [online]. 2021-05, vol. 15, no. 5, pp. 646–670 [visited on 2024-04-08]. ISSN 1751-956X, ISSN 1751-9578. Available from DOI: `10.1049/itr2.12051`.
2. *driveBOT – cogniBIT* [online]. [visited on 2024-04-13]. Available from: `https://cognibit.de/drivebot/`.
3. *Solutions for Virtual Test Driving | IPG Automotive* [online]. [visited on 2024-04-13]. Available from: `https://www.ipg-automotive.com/en/`.
4. *VI-Driver | VI-grade* [online]. [visited on 2024-04-13]. Available from: `https://www.vi-grade.com/en/products/vi-driver//`.
5. SCHARF, Louis; HARTHILL, William; MOOSE, Paul. A comparison of expected flight times for intercept and pure pursuit missiles. *IEEE Transactions on Aerospace and Electronic Systems* [online]. 1969-07, vol. AES-5, no. 4, pp. 672–673 [visited on 2024-05-12]. ISSN 0018-9251. Available from DOI: `10.1109/TAES.1969.309951`.
6. WALLACE, Richard; STENTZ, Anthony; THORPE, Charles; MARAVEC, Hans; WHITTAKER, William; KANADE, Takeo. First Results in Robot Road-Following. In: 1985-01, pp. 1089–1095.
7. CAMPBELL, Stefan. Steering control of an autonomous ground vehicle with application to the DARPA Urban Challenge. 2007-01.
8. SNIDER, Jarrod. Automatic Steering Methods for Autonomous Automobile Path Tracking. 2011-04.
9. MORALES, J.; MARTÍNEZ, Jorge; MARTÍNEZ, María; MANDOW, Anthony. Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile Robots with a 2D Laser Scanner. *EURASIP Journal on Advances in Signal Processing*. 2009-12, vol. 2009, pp. 1–10. Available from DOI: `10.1155/2009/935237`.

10. BUEHLER, Martin; IAGNEMMA, Karl; SINGH, Sanjiv. *The 2005 DARPA Grand Challenge: The Great Robot Race*. 2007-01. ISBN 978-3-540-73428-4. Available from DOI: `10.1007/978-3-540-73429-1`.

11. AMIDI, Omead; THORPE, Chuck E. Integrated mobile robot control. In: CHUN, Wendell H.; WOLFE, William J. (eds.) [online]. Boston, MA, 1991-03-01, pp. 504–523 [visited on 2024-05-12]. Available from DOI: `10.1117/12.25494`.

12. RANKIN, Arturo L.; CRANE, Carl D.; ARMSTRONG, David G. Evaluating a PID, pure pursuit, and weighted steering controller for an autonomous land vehicle. In: *Other Conferences*. 1998. Available also from: `https://api.semanticscholar.org/CorpusID:1528715`.

13. OLLERO, Aníbal; HEREDIA, Guillermo. Stability analysis of mobile robot path tracking. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*. 1995, vol. 3, 461–466 vol.3. Available also from: `https://api.semanticscholar.org/CorpusID:10628549`.

14. AMER, Noor Hafizah; ZAMZURI, Hairi; HUDHA, Khisbullah; KADIR, Zulkiffli Abdul. Modelling and Control Strategies in Path Tracking Control for Autonomous Ground Vehicles: A Review of State of the Art and Challenges. *Journal of Intelligent & Robotic Systems* [online]. 2017-05, vol. 86, no. 2, pp. 225–254 [visited on 2024-05-12]. ISSN 0921-0296, ISSN 1573-0409. Available from DOI: `10.1007/s10846-016-0442-0`.

15. COULTER, Craig. Implementation of the Pure Pursuit Path Tracking Algorithm. In: 1992. Available also from: `https://api.semanticscholar.org/CorpusID:62550799`.

16. HEREDIA, Guillermo; OLLERO, Anibal. Stability of autonomous vehicle path tracking with pure delays in the control loop. *Advanced Robotics* [online]. 2007-01, vol. 21, no. 1, pp. 23–50 [visited on 2024-05-12]. ISSN 0169-1864, ISSN 1568-5535. Available from DOI: `10.1163/156855307779293715`.

17. OLLERO, Anibal; GARCIA, Alfonso; MARTÍNEZ, Jorge; MANDOW, Anthony. Fuzzy tracking methods for mobile robots. 1997-02.

18. PARK, Myungwook; LEE, Sangwoo; HAN, Wooyong. Development of Steeing Control System for Autonomous Vehicle using Geometry based Path Tracking Algorithm. *ETRI Journal* [online]. 2015-05-04 [visited on 2024-05-12]. ISSN 1225-6463. Available from DOI: `10.4218/etrij.15.2314.0123`.

19. SHAN, Yunxiao; YANG, Wei; CHEN, Cheng; ZHOU, Jian; ZHENG, Ling; LI, Bijun. CF-Pursuit: A Pursuit Method with a Clothoid Fitting and a Fuzzy Controller for Autonomous Vehicles. *International Journal of Advanced Robotic Systems* [online]. 2015-09-01, vol. 12, no. 9, p. 134 [visited on 2024-05-12]. ISSN 1729-8814, ISSN 1729-8814. Available from DOI: `10.5772/61391`.

20. RODRIGUEZ-CASTAÑO, A.; HEREDIA, G.; OLLERO, A. Analysis of a GPS-Based Fuzzy Supervised Path Tracking System for Large Unmanned Vehicles. *IFAC Proceedings Volumes* [online]. 2000-09, vol. 33, no. 25, pp. 125–130 [visited on 2024-05-12]. ISSN 14746670. Available from DOI: 10.1016/S1474-6670(17)39327-8.

21. WIT, Jeffrey. Vector Pursuit Path Tracking For Autonomous Ground Vehicles. 2000-09.

22. THRUN, Sebastian; MONTEMERLO, Michael; DAHLKAMP, Hendrik; STAVENS, David; ARON, Andrei; DIEBEL, James; FONG, Philip; GALE, John; HALPENNY, Morgan; HOFFMANN, Gabriel; LAU, Kenny; OAKLEY, Celia; PALATUCCI, Mark; PRATT, Vaughan; STANG, Pascal; STROHBAND, Sven; DUPONT, Cedric; JENDROSSEK, Lars-Erik; KOELEN, Christian; MAHONEY, Pamela. Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robotics*. 2006-01, vol. 23, pp. 661–692.

23. AMER, Noor Hafizah; HUDHA, Khisbullah; ZAMZURI, Hairi; APAROW, Vimal Rau; ABIDIN, Amar Faiz Zainal; KADIR, Zulkiffli Abd; MURRAD, Muhamad. Adaptive modified Stanley controller with fuzzy supervisory system for trajectory tracking of an autonomous armoured vehicle. *Robotics and Autonomous Systems* [online]. 2018-07, vol. 105, pp. 94–111 [visited on 2024-05-12]. ISSN 09218890. Available from DOI: 10.1016/j.robot.2018.03.006.

24. HOFFMANN, Gabriel M.; TOMLIN, Claire J.; MONTEMERLO, Michael; THRUN, Sebastian. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In: *2007 American Control Conference* [online]. New York, NY, USA: IEEE, 2007-06, pp. 2296–2301 [visited on 2024-01-13]. ISBN 978-1-4244-0988-4 978-1-4244-0989-1. Available from DOI: 10.1109/ACC.2007.4282788.

25. GILLESPIE, T. D. *Fundamentals of vehicle dynamics*. Warrendale, PA: Society of Automotive Engineers, 1992. ISBN 978-1-56091-199-9.

26. BYRNE, R.H.; ABDALLAH, C.T.; DORATO, P. Experimental results in robust lateral control of highway vehicles. In: *Proceedings of 1995 34th IEEE Conference on Decision and Control* [online]. New Orleans, LA, USA: IEEE, 1995, vol. 4, pp. 3572–3575 [visited on 2024-05-11]. ISBN 978-0-7803-2685-9. Available from DOI: 10.1109/CDC.1995.479140.

27. GULDNER, J.; SIENEL, W.; HAN-SHUE TAN; ACKERMANN, J.; PATWARDHAN, S.; BUNTE, T. Robust automatic steering control for look-down reference systems with front and rear sensors. *IEEE Transactions on Control Systems Technology* [online]. 1999-01, vol. 7, no. 1, pp. 2–11 [visited on 2024-05-11]. ISSN 10636536. Available from DOI: 10.1109/87.736743.

28. RAJAMANI, R.; HAN-SHUE TAN; BOON KAIT LAW; WEI-BIN ZHANG. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Transactions on Control Systems Technology* [online]. 2000-07, vol. 8, no. 4, pp. 695–708 [visited on 2024-05-11]. ISSN 10636536. Available from DOI: 10.1109/87.852914.

29. ROSSETTER, E; SWITKES, J; GERDES, J. Experimental validation of the potential field lanekeeping system. *International Journal of Automotive Technology*. 2004-01, vol. 5.

30. GERDES, J.; ROSSETTER, Eric. A Unified Approach to Driver Assistance Systems Based On Artificial Potential Fields. *Journal of Dynamic Systems Measurement and Control*. 2001-09, vol. 123. Available from DOI: `10.1115/1.1386788`.

31. COOMBS, D.; MURPHY, K.; LACAZE, A.; LEGOWIK, S. Driving autonomously off-road up to 35 km/h. In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)* [online]. Dearborn, MI, USA: IEEE, 2000, pp. 186–191 [visited on 2024-05-11]. ISBN 978-0-7803-6363-2. Available from DOI: `10.1109/IVS.2000.898339`.

32. KELLY, Alonzo; STENTZ, Anthony. Rough Terrain Autonomous Mobility—Part 2: An Active Vision, Predictive Control Approach. *Autonomous Robots*. 1998-05-01, vol. 5, no. 2, pp. 163–198. ISSN 1573-7527. Available from DOI: `10.1023/A:1008822205706`.

33. BROGGI, Alberto; BERTOZZI, Massimo; FASCIOLI, Alessandra; GUARINO, Corrado; GUARINO LO BIANCO, Corrado; PIAZZI, Aurelio. The Argo Autonomous Vehicle's Vision And Control Systems. *Int. J. Intelligent. Control. Syst.* 2000-12, vol. 3.

34. LINIGER, Alexander; DOMAHIDI, Alexander; MORARI, Manfred. Optimization-Based Autonomous Racing of 1:43 Scale RC Cars. *Optimal Control Applications and Methods* [online]. 2015-09, vol. 36, no. 5, pp. 628–647 [visited on 2024-05-17]. ISSN 0143-2087, ISSN 1099-1514. Available from DOI: `10.1002/oca.2123`.

35. BAKKER, Egbert; NYBORG, Lars; PACEJKA, Hans B. Tyre Modelling for Use in Vehicle Dynamics Studies. *SAE Transactions* [online]. 1987, vol. 96, pp. 190–204 [visited on 2024-05-17]. ISSN 0096736X. Available from: `http://www.jstor.org/stable/44470677`.

36. BENNER, Peter. Computational methods for linear-quadratic optimization. *Rendiconti del Circolo Matematico di Palermo, Supplemento*. 1999, pp. 21–56.