Collision avoidance by autonomous braking and steering

Master Thesis
to obtain the academic degree of
Diplom-Ingenieur
in the Master’s Program
Mechatronics
Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, 12. August 2019

Daniel Adelberger, BSc
Abstract

Driver assistance systems have become more and more important in recent years due to the increasing degree of automation in road traffic – especially with regard to increasing safety. These systems do not only provide passive protection for driver and passengers (e.g. through airbags), but – in the near future – will also be able to protect other road users from injury by actively avoiding collisions. Emergency brake assistants are already mandatory for newly registered trucks and buses, and these assistants are becoming increasingly important also in civil road traffic. The next stage of development will be assistance systems that can not only carry out emergency braking independently but – if necessary – also steering manoeuvres if those are required to avoid a collision.

This thesis deals with such an autonomous braking and steering assistant that intervenes in critical situations to avoid accidents. The aim of this thesis is to develop a system that can follow a given route under normal conditions and takes action if necessary, namely if obstacles or other road users would involve the vehicle in an accident. For this purpose, a simple model of a real vehicle is created, including model boundaries and physical limitations. Subsequently, a mathematical formulation of the environment as well as of various other road users is developed – the group of possible dynamic objects is limited to passenger cars, cyclists, and pedestrians. In the next step, an optimisation problem is formulated with the aim to avoid collisions and to follow the original path as well as possible. In each optimisation step, this is done in a two-layer structure. In the first layer, non-linear model predictive control (NMPC) with a prediction- and control-horizon of 5 s is used to check which of the available lanes has the lowest risk for a collision. This lane is then selected as the reference for the next layer if the risk of a collision on the primary lane exceeds a certain limit in this step. In the second layer, the selected trajectory is tracked as well as possible by an NMPC – taking into account the existing collision risk – with the additional involvement of other road users. This happens with a now shorter prediction- and control-horizon of 2 s. In order to be able to estimate the behaviour of other road users, a prediction model is introduced which deduces from the states of the objects to their inputs, and predicts their future behaviour. In order to evaluate this model, data from real road users were recorded by using a test-vehicle, and were analysed later on. In a final step, the developed collision avoidance assistant is applied to a selected set of scenarios. The resulting trajectories are then validated using a realistic vehicle model in IPG CarMaker to obtain information about the modelling quality.

It is shown that the developed assistant fulfils the requirements very well in the analysed situations, and also that the validation of the trajectories leads to a satisfactory result.
Kurzfassung


ausgewählte Menge an Szenarien angewandt. Die resultierenden Trajektorien werden anschließend mittels realitätsnahem Fahrzeugmodell in IPG CarMaker validiert, um Aussagen über die Modellierungsqualität zu erhalten.

Es zeigt sich, dass der entwickelte Assistent in den untersuchten Situationen die an ihn gestellten Anforderungen sehr gut erfüllt, und dass auch die Validierung der Trajektorien zu einem zufriedenstellenden Ergebnis führt.
# Contents

1 Preamble .......................... 1
   1.1 Background ........................................ 1
   1.2 Problem description ............................. 3
   1.3 Structure of the thesis ......................... 4

2 Modelling of the ego-vehicle ....... 7
   2.1 Challenges when modelling a vehicle ............. 7
   2.2 Different modelling approaches ................. 7
   2.3 Single track model ............................... 8
      2.3.1 Physical limitations .......................... 9

3 Representation of obstacles and environment ... 13
   3.1 Environment for test scenarios ............... 13
      3.1.1 Allowed and prohibited regions ............. 15
      3.1.2 Mathematical check if a point exits road boundaries .... 17
      3.1.3 Calculation of the ego-vehicle corners .......... 22
      3.1.4 Minimal distance from the ego-vehicle to a point ......... 23
   3.2 Collision detection for obstacles .......... 24
      3.2.1 Conditions for convexity .................. 25
      3.2.2 Collision detection ......................... 25

4 Formulation of the optimisation problem ... 29
   4.1 Restrictions of the problem .......... 29
   4.2 General design of the optimisation ......... 30
   4.3 First layer of the optimisation .......... 32
      4.3.1 Trajectory generation .................... 32
      4.3.2 Tracking optimisation .................... 34
      4.3.3 Evaluation of the optimised trajectories .... 35
      4.3.4 Selection of a trajectory for the next layer .... 37
   4.4 Second layer of the optimisation .......... 39
      4.4.1 Setting the options for the second optimisation .... 39
      4.4.2 Representation of the obstacles ........... 39
      4.4.3 Performance criterion representing the cost of obstacles .... 40
      4.4.4 Distance-based validation of obstacles ....... 40
      4.4.5 Parallax-angle-based validation of obstacles .... 42
      4.4.6 Combination of both approaches ............ 45
      4.4.7 Optimisation in the second layer ............ 45
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 Short description of non-linear problems in general</td>
<td>46</td>
</tr>
<tr>
<td>4.5.1 General formulation</td>
<td>46</td>
</tr>
<tr>
<td>4.5.2 Condensed and un-condensed formulation of the problem</td>
<td>47</td>
</tr>
<tr>
<td>4.5.3 Limitation of the computation time</td>
<td>47</td>
</tr>
<tr>
<td>4.5.4 Nonlinear programming</td>
<td>48</td>
</tr>
<tr>
<td>4.6 Blocking as a way to reduce computational burden</td>
<td>50</td>
</tr>
<tr>
<td>5 Prediction for dynamic obstacles</td>
<td>53</td>
</tr>
<tr>
<td>5.1 Different prediction methods</td>
<td>53</td>
</tr>
<tr>
<td>5.1.1 Differences in terms of physics-based motion models</td>
<td>54</td>
</tr>
<tr>
<td>5.1.2 Handling of uncertainties</td>
<td>54</td>
</tr>
<tr>
<td>5.2 Selection of a concrete method</td>
<td>55</td>
</tr>
<tr>
<td>5.2.1 Modelling approach</td>
<td>55</td>
</tr>
<tr>
<td>5.2.2 Resulting prediction</td>
<td>61</td>
</tr>
<tr>
<td>5.2.3 Performance criterion for the prediction</td>
<td>62</td>
</tr>
<tr>
<td>5.3 Validation of the model</td>
<td>63</td>
</tr>
<tr>
<td>5.3.1 Validation for cars</td>
<td>64</td>
</tr>
<tr>
<td>5.3.2 Validation for bicycles</td>
<td>64</td>
</tr>
<tr>
<td>5.3.3 Validation for pedestrians</td>
<td>65</td>
</tr>
<tr>
<td>5.3.4 Interpretation of the validation</td>
<td>66</td>
</tr>
<tr>
<td>6 Recording of real measurement data</td>
<td>67</td>
</tr>
<tr>
<td>6.1 Sensor overview</td>
<td>68</td>
</tr>
<tr>
<td>6.1.1 LIDAR</td>
<td>68</td>
</tr>
<tr>
<td>6.2 Acquired Data</td>
<td>69</td>
</tr>
<tr>
<td>6.2.1 Measurement</td>
<td>70</td>
</tr>
<tr>
<td>6.2.2 Data statistics</td>
<td>71</td>
</tr>
<tr>
<td>6.2.3 Usage for scenario generation</td>
<td>75</td>
</tr>
<tr>
<td>7 Selection of the Scenario</td>
<td>77</td>
</tr>
<tr>
<td>7.1 Definition of the Scenario</td>
<td>77</td>
</tr>
<tr>
<td>7.1.1 Definition of ego and dynamic obstacles</td>
<td>78</td>
</tr>
<tr>
<td>7.2 Straight road</td>
<td>78</td>
</tr>
<tr>
<td>7.2.1 Case S1 – Simple collision avoidance</td>
<td>79</td>
</tr>
<tr>
<td>7.2.2 Case S2 – Collision avoidance by braking</td>
<td>79</td>
</tr>
<tr>
<td>7.2.3 Case S3 – Crash-mode</td>
<td>79</td>
</tr>
<tr>
<td>7.3 Junction</td>
<td>80</td>
</tr>
<tr>
<td>7.3.1 Case J1 – Collision avoidance in front of pedestrian</td>
<td>81</td>
</tr>
<tr>
<td>7.3.2 Case J2 – Collision avoidance behind pedestrian</td>
<td>81</td>
</tr>
<tr>
<td>8 Trajectory tracking and scenario evaluation</td>
<td>83</td>
</tr>
<tr>
<td>8.1 Trajectory tracking</td>
<td>83</td>
</tr>
<tr>
<td>8.1.1 Signals in the control structure</td>
<td>84</td>
</tr>
<tr>
<td>8.1.2 Functional blocks of the control structure</td>
<td>84</td>
</tr>
<tr>
<td>8.1.3 Parameters of the PI-controllers</td>
<td>85</td>
</tr>
<tr>
<td>8.2 Performance evaluation</td>
<td>86</td>
</tr>
<tr>
<td>8.2.1 Case S1 – Simple collision avoidance</td>
<td>86</td>
</tr>
<tr>
<td>8.2.2 Case S2 – Collision avoidance by braking</td>
<td>88</td>
</tr>
</tbody>
</table>
Chapter 1

Preamble

In this chapter a short introduction concerning passive and active security systems is provided, as well as the rating scheme used by the European New Car Assessment Program (Euro NCAP). Moreover, the reasons that cause the need for assistants that are not only capable of braking (AEB) but also of steering (AES) in real life driving situations are portrayed. As a last point, the structure of the thesis is shortly described.

1.1 Background

The rising rate of automation in everyday traffic leads to new challenges for car manufacturers and also pushes the need for new and better safety systems for driving. Those safety systems are no longer just the well-known passive ones which protect driver and passengers (e.g. through airbags) and other road users (e.g. through an easier deformability of the bonnet) in case of accidents. More and more systems that are not designed to protect passengers and other road users in the event of a collision but to prevent such collisions by actively intervening are entering the market. While those active safety systems are currently mostly only offered as additional security packages in the luxury segment of passenger cars, the situation is different for commercial vehicles. For example, new trucks and buses in the European Union have had to be equipped with AEB since 2015. The relevant regulations are summarised in UN/ECE R 131 [1]. Taking a look into the future, it can be assumed that this type of emergency assistance system will be standard for newly developed cars in a few years’ time.

To support this kind of development, AEB/AES-systems are getting more and more into focus in the Euro NCAP Rating of the upcoming years, as listed in [2, p. 13].

The Euro NCAP Rating is split into four subgroups:

- Adult Occupant Protection (AOP)
- Child Occupant Protection (COP)

---

1The Euro NCAP (European New Car Assessment Programme) is a Brussels-based company of European transport ministries, automobile clubs and insurance associations. The organisation carries out crash tests on new types of cars and then assesses their safety on the basis of the available safety systems. The tests are not required by law, but are merely intended to inform consumers.
• Vulnerable Road User (VRU) Protection
• Safety Assist (SA)

In the rating, a car can score a certain amount of points in each of these subgroups in different categories. The following table shows the quantity of points for the category in relation to the total number of points achievable in the subgroup, according to [2, p. 13] where also all the other categories are listed.

Table 1.1: Euro NCAP rating overview

<table>
<thead>
<tr>
<th>Test</th>
<th>2018</th>
<th>2019</th>
<th>2020</th>
<th>2021</th>
<th>2022</th>
<th>2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult Occupant Protection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AEB (City)</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>Child Occupant Protection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>Vulnerable Road User Protection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AEB/AES Pedestrian</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>AEB/AES Cyclist</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>AEB/LSS Powered Two Wheeler</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>48</td>
<td>48</td>
<td>54</td>
<td>54</td>
<td>63</td>
<td>63</td>
</tr>
<tr>
<td>Safety Assist</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AEB C2C</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Total</td>
<td>13</td>
<td>13</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

This focus on autonomous braking and steering assistants is not only visible in the distribution of points in the rating but is also expressed in the official Euro NCAP Roadmap [3].

As this Euro NCAP rating is of significant importance for the attractiveness of a car and influences purchasing decisions – as safety usually plays a considerable role when buying a new car, the manufacturers have to focus on the development of such systems.

In this work a proof of concept of such an autonomous braking and steering system is presented. The proposed assistant is then evaluated with certain test scenarios in Chapter 8.

Various papers concerning this topic exist, see e.g. [4] where a non-linear model predictive controller (NMPC) is used in combination with an artificial potential field (APF) to avoid collisions with obstacles. These potential fields are also used in other works, like [5] where the optimisation is implemented separately for longitudinal and lateral motion. They mostly represent solely the distance to the specific object. Other geometrical methods exist as well, like the use of the parallax angle from the base of the ego to the hull points of surrounding obstacles, as shown in [6]. Most of those works use really simple models for trajectory generation and separate – more complex – models to evaluate the result, like in [7] where a simple kinematic model is used in the first place which is later evaluated with an eight
1.2 Problem description

Assistants for autonomous emergency braking and steering are especially useful in situations where the ‘real’ driver does not pay attention to a potential hazard or simply does not have sufficient time to react fast enough in a critical situation.

In such cases the system intervenes to avoid collisions. The first approach to cope with this kind of problems are so-called autonomous emergency braking (AEB) systems that automatically brake if an obstacle occurs to which the driver does not react. These systems are really useful, but still, situations remain where braking alone is not sufficient, e.g. if the braking distance of the vehicle would simply be too long – as in the scenario in Subsection 8.2.3. In such situations autonomous emergency steering (AES) systems come into play. If the vehicle gets the option of autonomous steering in addition to autonomous braking, far more dangerous situations can be well taken care of.

The basic principle of such systems is quite simple, as visualised in the following figure.

![Simplified architecture of a typical AEB/AES assistant](image)

Figure 1.1: Simplified architecture of a typical AEB/AES assistant

For ‘real’ cars and simulation studies the concept varies only slightly.

1. At first, information about the environment, road boundaries, and obstacles is measured by a system of sensors. In simulation studies, and also in this thesis, that kind of data are usually expected as given – if needed, those sensor systems could be modelled by adding noise to the data.
2. Now the future positions of the surrounding obstacles need to be predicted. Various approaches exist here. The procedure is equal for the real world as well as for simulations.

3. In the next step, some kind of risk assessment takes place. The previous prediction is used to define dangerous regions. Then, an optimal trajectory is calculated, usually using a simplified model of the car. The advantage of this method is that inputs for the model are also calculated, which can later be used for the tracking of the path. There also exist other methods for path planning, like building a path with simple geometrical shapes, but those approaches are not further covered here.

4. Now the optimised trajectory is found and needs to be tracked. In the real world this is the point where the vehicle executes the calculated manoeuvre to avoid a collision. In simulation there are two options. Either the tracking is obtained by a more complex vehicle model to validate the result directly (online) just like in the real world, or again the tracking is obtained by the simple model, which means that the calculated trajectory can be used directly. For validation purposes this trajectory has to be tracked with a complex model afterwards. This offline approach is used in the present work due to a better flexibility.

5. In the end, one obtains a manoeuvre consisting of braking and steering actions – in the optimal case without a collision along the trajectory.

An exact description of the single components is given in the corresponding chapters.

1.3 Structure of the thesis

In order to provide a clear overview, the structure of the thesis is similar to the single steps presented before. The content of the chapters is shortly described in the following.

Modelling of the ego-vehicle

At first, in Chapter 2 a simple non-linear mathematical model of a vehicle is developed for further studies. Furthermore, first physical restrictions, that are then used as soft constraints in the optimisation, are defined here. This model is later utilised for the representation of the vehicle in the optimisation as well as for the prediction of obstacle movement.

Representation of obstacles and environment

In the next step, namely in Chapter 3 the representation of environment and obstacles is defined as a base of the upcoming scenarios. The types of dynamic obstacles are restricted to cars, pedestrians, and bicycles, as the main focus of the Euro NCAP rating lies on these groups and also a sufficient amount of measurement data of these kinds of obstacles could be obtained.
Formulation of the optimisation problem

In Chapter 4, the formulation of the trajectory planning optimisation is presented. A two-layer approach is selected to provide optimal results. In the first layer, the safest lane in the scenario is selected in each optimisation step – with some heuristics in the background to keep the focus on the primary planned lane – except when it gets too dangerous. The second layer then tries to track the selected trajectory but additionally minimizes the risk of a collision by avoiding obstacles along the trajectory through braking and steering. A general overview about non-linear model predictive control (NMPC) and methods to reduce the computational burden is presented at the end of the chapter.

Prediction for dynamic obstacles

To be able to provide safe trajectories, the collision avoidance assistant needs to predict future positions of surrounding obstacles as well as possible. Chapter 5 presents a prediction approach which uses the measured obstacle states to estimate the past inputs of the obstacle, which are then used to provide regions where the obstacle may be located in the future. This prediction method is then validated using real world measurement data.

Recording of real measurement data

The process of recording these data is shortly described in Chapter 6. The data are then analysed to provide an overview of the movement behaviour of obstacles in junction situations. The median velocity of several obstacle types is determined to be partly used for the upcoming evaluation scenarios.

Scenario selection

In Chapter 7, parametrized scenarios for a straight road as well as for a junction are defined. The scenarios are selected in such a way that most situations the collision avoidance assistant has to cope with are covered. These scenarios are used to show the behaviour of the assistant in certain situations and are by far not sufficient to provide statements about safety – alternative methods or far more scenarios and testing are needed for that.

Trajectory tracking and scenario evaluation

As a last step, the previously defined scenarios are now evaluated in Chapter 8. Therefore, a simple tracking system using PI-controllers is defined at first. The behaviour of the collision avoidance assistant is compared to its expected behaviour, and the quality of trajectory tracking is analysed.
Chapter 2

Modelling of the ego-vehicle

This chapter first gives a short overview of general modelling of vehicles. In the next step, a simple non-linear model for further usage is proposed. Physical limitations as well as limitations of the model are described and formulated as additional costs (soft constraints) for the upcoming optimisation.

2.1 Challenges when modelling a vehicle

When modelling a system in general or – as in this case – a vehicle, one always has to face trade-offs between sufficient complexity and simplicity. On the one hand, a model should be complex enough to display the most important physical effects, but on the other hand, it also should not be too complex because that might have a negative influence on computation time, following the rule “as simple as possible but as complex as needed” [8, p. 14]. The selection of model complexity should always be chosen, taking into account the application the model is used for. Simple linear models tend to be perfect for fast computation because there are a lot of methods that cope with such kind of systems, but in most cases they are not able to represent all the effects one wants to incorporate for technical use. More complex non-linear models promise a much more realistic representation of actual physical phenomena, but due to their complexity, they are usually not usable for real-time based prediction and control [7, p. 772].

2.2 Different modelling approaches

As a first approach the vehicle could be modelled as a point of mass. One can easily recognize that this model might be useful in some simple cases (when also implementing sufficient constraints) but not for crash avoidance at junctions. In that case the simplicity and direct possibility of intervention through the inputs have negative effects on realistic modelling because a real car simply cannot move in all directions at all times. When looking at a straight road, this model might be sufficient because longitudinal and lateral coordinates of street and vehicle are mostly the same, and constraints for a realistic behaviour are easier to implement. For more complex scenarios, where realistic modelling of a vehicle with a
degree of freedom similar to a real world car is required, this simple model is by far not sufficient.

The next step in modelling would be a model called “Dubins vehicle” which is already a simple non-linear model that gets closer to the behaviour of a ‘real’ car by representing it using just one tire. Here, ‘realistic’ control inputs are assumed, which means, one cannot simply vary the velocity \( v \) but influence it by an integrated acceleration/deceleration \( a \), just like in a ‘normal’ vehicle by using accelerator and brake pedal. This model is getting closer to what is observable in the real world, but still, some problems remain, like for example the orientation of the wheel in terms of rotation is not constrained here. Theoretically, the modelled wheel could just rotate at its current position and then head to an arbitrary point in the room. It is well-known that something like that is not possible for real cars because they are limited by their mechanical construction – in particular the rear axle that is not influenced by steering.

In conclusion, there is to say that the one-wheel-model is still not sufficient. Therefore, in the following subsection, the development of another simple non-linear model – namely the so-called single track model – is shown.

### 2.3 Single track model

The next stage of modelling a vehicle is to take into account the geometry of the car – in particular the rear axle. This leads to a model where the orientation of the vehicle is changed by steering with the front wheels, but these two rotation-angles are not equal. That leads to a quite realistic model that is also common in the literature, see e.g. [7, p. 769].

States of the system:
- \( x(t) \) in m ... position in X-direction
- \( y(t) \) in m ... position in Y-direction
- \( v(t) \) in \( \frac{m}{s} \) ... longitudinal velocity of the vehicle
- \( \theta(t) \) in rad ... orientation of the vehicle

Inputs of the system:
- \( a(t) \) in \( \frac{m}{s^2} \) ... longitudinal acceleration/deceleration of the vehicle
- \( \delta(t) \) in rad ... steering angle (relative angle of the front wheels to the longitudinal axis of the vehicle)

The state equations result as followed.

\[
\begin{align*}
\frac{dx(t)}{dt} &= v \cdot \cos(\theta(t)) \\
\frac{dy(t)}{dt} &= v \cdot \sin(\theta(t)) \\
\frac{dv(t)}{dt} &= a \\
\frac{d\theta(t)}{dt} &= \frac{v(t)}{D_a} \cdot \tan(\delta(t))
\end{align*}
\] (2.1)
2.3. Single track model

Realistic limits for acceleration as well as deceleration can be found in the literature, see [9, pp. 4739] and were set to even values for simplicity here.

- \( a_{\text{braking}}^{\text{max}} = -4 \text{ m/s}^2 \)
- \( a_{\text{accel}}^{\text{max}} = 2 \text{ m/s}^2 \)

![Figure 2.1: Single track model – representation of states and inputs](image_url)

By using the simple forward Euler method, the proposed continuous-in-time-model can be transformed to a discrete-in-time-model where \( T_s = 0.1 \text{ s} \) is selected as the sample time of the system:

\[
\begin{align*}
    x(k+1) &= x(k) + v(k) \cdot \cos(\theta(k)) \cdot T_s \\
    y(k+1) &= y(k) + v(k) \cdot \sin(\theta(k)) \cdot T_s \\
    v(k+1) &= v(k) + a(k) \cdot T_s \\
    \theta(k+1) &= \theta(k) + \frac{v(k)}{D_a} \cdot \tan(\delta(k)) \cdot T_s
\end{align*}
\] (2.2)

As the name suggests, the single track model represents the movement of just a single track – represented by the ‘virtual’ central pair of wheels in Figure 2.1. For simplicity of modelling, one can assume that the single track sufficiently represents the behaviour of a vehicle with four tires, as drawn in the figure.

2.3.1 Physical limitations

According to [8, p. 17], the model is suitable for moderate cornering with a lateral acceleration of up to 0.4 g for dry roads. This should be adequate to represent a vehicle in emergency steering situations in a first attempt. It might be helpful to limit the steering in relation to the velocity to avoid lateral accelerations that would overshoot the limit of 0.4 g. In order to fulfil this condition, a constraint is introduced – the longitudinal acceleration of the vehicle
is limited. This can be expressed by using the current curvature of the movement as well as the velocity of the vehicle.

![Diagram](image)

**Figure 2.2:** Single track model – current rotation centre $C(k)$

In each point in time the single track model rotates around a specific fixed point $C(k)$ (instantaneous centre of rotation) on the two-dimensional plane, in the case $\delta(k) = 0$ rad this point gets shifted to infinity. In all other cases a radius $r(k)$ from the current rotation centre $C(k)$ to the position of the centre of the rear axle can be defined. Using trigonometrical functions, one can easily derive the current curvature – required measures are visualised in Figure 2.2.

$$c(k) = \frac{1}{r(k)} = \frac{1}{D_a} \cdot \tan(\delta(k))$$

(2.3)

This entity is also used for calculating $\theta(k + 1)$ in Equation 2.2 – using the knowledge that the path velocity $v(k)$ in a circular movement is related to the angular velocity and the radius of the circle.

$$\omega(k) = \frac{v(k)}{r(k)} = v(k) \cdot c(k)$$

(2.4)

Knowing the curvature at each time step, one can now formulate a constraint for the lateral acceleration.

$$a_{lat}(k) = v(k)^2 \cdot c(k) = \frac{v(k)^2}{D_a} \cdot \tan(\delta(k)) \leq 0.4 \, g \approx 4 \, m/s^2$$

(2.5)

For all cases where this constraint is not violated, the model should be sufficient according to [8, p. 17]. Therefore, a soft constraint for the optimisation problem can be formulated as

$$J_{\text{model}}(k) = \max \left( 0, \frac{|a_{lat}(k)|}{0.4 \, g} - 1 \right)$$

(2.6)
where invalid lateral accelerations above 0.4 g are penalised. The division by 0.4 g leads to a normalisation of the cost term which is quite intuitive.

Not only the single track model has to be valid in each step. As it is commonly known, a vehicle can only handle a limited amount of applied forces before it loses traction on the street. This is the reason why one should avoid unnecessary braking in curves. That is especially dangerous for two-wheeled vehicles like motorcycles, but also cars are affected by this issue.

Therefore, another criterion is to be implemented to avoid hazardous states where traction might be lost. Again, this is a case where simplification is necessary for online-calculability. When driving in the ‘real’ world, each wheel is loaded differently depending on acceleration, curvature of the road and even the arrangement of passengers in the vehicle.

For really precise simulations with a high degree of freedom vehicle, facts like that are implemented, but for the proposed simplified single track model, a conceptual simplification will be made. In concrete terms this means that the whole vehicle is reduced to one single tire on which all forces act. The amount of forces is also simplified by neglecting e.g. air resistance or the slope of the road.

Forces included in the model are

- \( F_{fric} \) in N ... Friction force that prevents the vehicle from slipping
- \( F_{cent}(k) \) in N ... Centrifugal force acting on the vehicle in a curve
- \( F_{acc}(k) \) in N ... Inertia force acting on the vehicle during acceleration and braking

These forces are calculated as shown in the following:

\[
F_{fric} = m \cdot g \cdot \mu_{sf} \quad (2.7)
\]

\[
F_{cent}(k) = m \cdot \frac{v(k)^2}{D_a} \cdot \tan(\delta(k)) \quad (2.8)
\]

\[
F_{acc}(k) = m \cdot a(k) \quad (2.9)
\]

\( F_{fric} \) defines the radius of Kamm’s circle while the combination of \( F_{cent}(k) \) and \( F_{acc}(k) \) states if the adhesion condition is fulfilled, therefore, they need to be geometrically added.

\[
F_{res}(k) = \sqrt{F_{cent}(k)^2 + F_{acc}(k)^2} \quad (2.10)
\]

The only undefined values at this point are the static friction of the road \( \mu_{sf} \) and the mass \( m \) of the car. The mass might be known but has no importance here because the forces are only set into relation to each other and due to that, the mass can be eliminated. Since this work is only a proof of concept, ideal street conditions are assumed, which means in this case dry asphalt and therefore a static friction coefficient

\[
\mu_{sf} = 0.9 \quad (2.11)
\]

obtained from \[10\] p. 8.
The example in Figure 2.3 shows two cases (superscripts \( v \) and \( i \)) where case \( v \) is a valid situation (\( F_{\text{res}}(k^v) \) inside Kamm’s circle - \( F_{\text{res}}(k^v) < F_{\text{fric}} \)) while case \( i \) is invalid (\( F_{\text{res}}(k^i) \) outside Kamm’s circle - \( F_{\text{res}}(k^i) > F_{\text{fric}} \)).

Once again, a soft constraint for the optimisation problem can be formulated as

\[
J_{\text{kamm}}(k) = \max \left( 0, \frac{\sqrt{F_{\text{cent}}(k)^2 + F_{\text{acc}}(k)^2}}{F_{\text{fric}}} - 1 \right) = \max \left( 0, \frac{F_{\text{res}}(k)}{F_{\text{fric}}} - 1 \right)
\]

As in Equation 2.6, the costs are normalised to obtain a better comparability and scalability.
Chapter 3

Representation of obstacles and environment

This chapter provides a short overview of the required definitions of surrounding obstacles, road boundaries, and the environment in general. In the second part of the chapter some mathematical methods are introduced, which are later used to calculate distances to obstacles and also serve, when it comes to the detections of possible collisions.

3.1 Environment for test scenarios

For the development and simulation of driving assistance systems, the interaction between ego-vehicle and surrounding obstacles as well as environment is of huge importance because the implemented systems are usually created to react to the environment and influence the behaviour of the ego-vehicle to respond in a preordained way. Therefore, one has to create unified scenarios that can be influenced and re-defined in an easy and clear way. Different obstacles are able to carry out dissimilar kinds of movements, and also the differentiation between static and dynamic objects is important. In this thesis, static objects represent the non-moving parts of the environment, like trees, buildings, or parked cars while the group of dynamic obstacles consists of walking pedestrians, bicyclists, and cars – other vehicles like coaches, trucks and so on would also be counted as dynamic obstacles but are not covered in this work to retain as much simplicity as possible. Nevertheless, they could easily be added in later versions of the assistant. Generally, this thesis sets its focus on possible crash situations in urban traffic. That is why a major focus will be set on junction situations.

In the further course of this work, some different scenarios are presented. This is the reason why a standard description of scenarios is proposed here in the beginning.

Due to simplification of computational complexity, all obstacles in this work are modelled by simple two-dimensional bounding boxes shaped as rectangles when dynamic, and polygonal, rectangular, or circular when static. The method of rectangular bounding boxes appears quite obvious because current sensor systems like RADAR\(^1\), LIDAR\(^2\), and cameras usually

\(^1\)Radio Detection And Ranging
\(^2\)Light Detection And Ranging
only supply assistance systems with this kind of information. In [Chapter 6] these sensor types are examined in greater detail.

Typical elements of a test scenario are shown in the following figure.

![Figure 3.1: Elements of a test scenario](image)

There are a few possible object types taken into account. A rough differentiation is possible by splitting them into a group of static and a group of dynamic objects. This is quite important because for static objects the position is known for all time steps – as the description expresses, they are static and will not move over time.

Static objects are:
- T ...trees
- H ...houses and buildings
- S ...other static objects (collective subgroup)

Road boundaries (R) are treated differently than normal obstacles because usually in normal driving they shall not be crossed, but in a crash situation it is, for example, better to cross a road boundary and get to stop in a field than to crash into an approaching car. The condition that tells if a car is allowed to cross road boundaries is also covered in this work, see [Section 4.2](#).

The other group contains all objects that are able to move in the two-dimensional plane, which means their states vary over time.

A task that has to be tackled is that these future obstacle states are not known a-priori, i.e. the collision avoidance assistant has to make predictions concerning the future states of the dynamic objects. That way the whole crash avoidance task turns out to be a lot more complicated because it can be assumed that the prediction accuracy gets worse with
each further step of looking into the future. A prediction method that tackles this task is proposed in Chapter 5.

Dynamic objects are:
- V ...vehicles (other cars)
- P ...pedestrians walking around
- B ...bicyclists

Each of these obstacles possesses different characteristics that will be discussed later. At a fixed point in time they define a set of allowed and prohibited regions for the ego-vehicle in the two-dimensional plane through their geometry, which is shown in the next section.

3.1.1 Allowed and prohibited regions

Figure 3.2, Figure 3.3, and Figure 3.4 show the allowed and prohibited sets caused by road boundaries as well as by static and dynamic objects.

![Diagram](image)

Figure 3.2: Allowed ($A_{road}$) and prohibited ($P_{road}$) regions caused by road boundaries.

In general, the allowed region at a specific point $k$ in time can be mathematically defined as

$$A(k) = A_{road} \cap A_{stat} \cap A_{dyn}(k)$$

(3.1)

whereas the prohibited region follows to

$$P(k) = P_{road} \cup P_{stat} \cup P_{dyn}(k)$$

(3.2)
These regions will be of severe importance because if the ego-vehicle enters a prohibited region (except the region $P_{road}$), it can be counted as a crash which is a non-desirable final state for a collision avoidance assistant.

Figure 3.3: Allowed ($A_{stat}$) and prohibited ($P_{stat}$) regions caused by static obstacles

Figure 3.4: Allowed ($A_{dyn}(k)$) and prohibited ($P_{dyn}(k)$) regions caused by dynamic obstacles – for the current time step where it can be assumed that all states and dimensions are well-known
3.1.2 Mathematical check if a point exits road boundaries

Of course, a defined region or set which a vehicle is allowed or prohibited to enter is defined quite easily, but for practical use one needs a mathematical formulation that is able to return a logical value that tells if a position is valid or prohibited.

A first approach can be formulated by considering the ego-vehicle as a point \( p(k) = [x(k), y(k)]^T \) and not as a rectangle. For road boundaries, this method is almost sufficient because if one uses this check on all four corners of the rectangle, nearly all possible cases are covered (the only exception will be presented later in Subsubsection 3.1.2.1).

Junctions in the real world usually have no sharp corners, to make it easier for vehicles to turn. To keep the environment as simple but also as realistic as possible, this circumstance was also modelled here. For ease of computation, all road boundaries in this work will consist of lines oriented in angles of the set \( \Phi = \{ \phi \mid \pi/4 \cdot n, n \in \mathbb{Z} \} \) – most of them horizontal or vertical, this way the task can be formulated quite easily. For further development, it appears necessary to also model non-linear shapes.

The next step is to look for a mathematical formulation to check if a point \( p(k) \) is inside the prohibited region \( P_{road}^A \).

Therefore, it seems appropriate to split the task into smaller subproblems, namely by taking a look at all boundary lines of the prohibited area. When using this method, it is important to specify which ‘side’ of the line acts as the outside and which acts as the inside of the area. In this work, a counter-clockwise notation is established, which means the lines are defined by points that run around the area counter-clockwise, with the prohibited area on the ‘left’.

The distance \( d(k) \) of the point \( p(k) \) to such a line can be easily calculated as shown in the following Figure 3.6.
\[ [\bar{x}_S, \bar{y}_S]^T \] denotes the start point of the line and \([\bar{x}_E, \bar{y}_E]^T\) the end point.

First, the distance between start and end along the X- and Y-axis are calculated to be able to calculate the orientation \(\alpha\) in a next step.

\[
d_{SE,x} = \bar{x}_E - \bar{x}_S \\
d_{SE,y} = \bar{y}_E - \bar{y}_S
\] (3.3) (3.4)

For the calculation of \(\alpha\), the sign of \(d_{SE,x}\) is important, as it represents the orientation of the line in the plane.

\[
\alpha = \begin{cases} 
\arctan \left( \frac{d_{SE,y}}{d_{SE,x}} \right) & \text{for } d_{SE,x} \geq 0 \\
\arctan \left( \frac{d_{SE,y}}{d_{SE,x}} \right) + \pi & \text{for } d_{SE,x} < 0
\end{cases}
\] (3.5)

The different handling for varying signs of \(d_{SE,x}\) is necessary because if both \(d_{SE,x}\) and \(d_{SE,y}\) are negative, the same \(\alpha\) as if both were positive would be the outcome, which leads to a wrong final result. Now one can easily calculate \(d_1(k)\) and \(d_2(k)\) that give the normal distance \(d(k)\) from the point \(p(k)\) to the line.

\[
d_1(k) = \cos(\alpha) \cdot (y(k) - \bar{y}_S) \\
d_2(k) = -\sin(\alpha) \cdot (x(k) - \bar{x}_S)
\] (3.6) (3.7)

\[
d(p(k), \bar{x}_S, \bar{y}_S, \bar{x}_E, \bar{y}_E) = d_1(k) + d_2(k)
\] (3.8)

If one calculates \(d(k)\) to all lines of a road boundary, the result states how far the road boundary was crossed by the point \(p(k)\). Therefore, a min / max formulation is used and all distances lower than 0 are set to 0. With this formulation all points inside the region yield their correct distance to the closest boundary line.
As an example, the formulation for Figure 3.5 is shown below. One can obtain that the distance from lines that cross only once with other lines is valid until infinity. In case this behaviour is unwanted, the ‘open’ polygon could be simply closed by adding additional lines. The most important criterion here is that the polygon stays convex, convexity is graphically described in Figure 3.11.

\[
d_{\text{road}}^A(p(k)) = \min(\max(0, d(p(k), \bar{x}_{A1}, \bar{y}_{A1}, \bar{x}_{A2}, \bar{y}_{A2})), \\
\max(0, d(p(k), \bar{x}_{A2}, \bar{y}_{A2}, \bar{x}_{A3}, \bar{y}_{A3})), \\
\max(0, d(p(k), \bar{x}_{A3}, \bar{y}_{A3}, \bar{x}_{A4}, \bar{y}_{A4}))
\] (3.9)

This can now be used to define the allowed and prohibited regions shown in Figure 3.5.

\[
\mathcal{P}_{\text{road}}^A : \{p(k) \mid d_{\text{road}}^A(p(k)) > 0\} \\
\mathcal{A}_{\text{road}}^A : \{p(k) \mid d_{\text{road}}^A(p(k)) = 0\}
\] (3.10) (3.11)

For collision avoidance situations, these prohibited regions are not seen as completely forbidden (hard constraints) but as regions with higher costs that should be avoided by the ego-vehicle.

Therefore, the design selection for \(d_{\text{road}}(k)\) makes it an optimal performance criterion for such kind of soft constraints, as the costs rise linearly with further penetration of the prohibited zone. With a formulation as the one stated in Equation 3.9, the value of \(d_{\text{road}}(k)\) starts to rise, however, not until the road boundary has been crossed. This might seem reasonable at a first thought, but in practice a vehicle should always try to keep a certain distance to the road boundary. Luckily, the appearance of Equation 3.9 allows an easy shift of the border by simply adding a constant value \(d_S\) before calculating the single maxima.

\[
d_{\text{road}}^A(p(k)) = \min(\max(0, d(p(k), \bar{x}_{A1}, \bar{y}_{A1}, \bar{x}_{A2}, \bar{y}_{A2} + d_S)), \\
\max(0, d(p(k), \bar{x}_{A2}, \bar{y}_{A2}, \bar{x}_{A3}, \bar{y}_{A3} + d_S)), \\
\max(0, d(p(k), \bar{x}_{A3}, \bar{y}_{A3}, \bar{x}_{A4}, \bar{y}_{A4} + d_S))
\] (3.12)

These calculations can now be combined for all road boundaries in the scenario, as shown in Figure 3.7.

As already realised in Section 2.3, a cost-term can be formulated – this time for crossing the road boundaries (or more accurately the shifted boundaries)

\[
J_{\text{road}}^P(p(k)) = \max(d_{\text{road}}^A(p(k)), d_{\text{road}}^B(p(k)), d_{\text{road}}^C(p(k)), d_{\text{road}}^D(p(k)))
\] (3.13)

where the single distances are calculated the same way (in a counter-clockwise direction, as
already shown in Equation 3.12).

\[
\begin{align*}
    d^B_{\text{road}}(p(k)) &= \min(\max(0, d(p(k), \bar{x}_B1, \bar{y}_B1, \bar{x}_B2, \bar{y}_B2) + d_S), \\
    &\quad \max(0, d(p(k), \bar{x}_B2, \bar{y}_B2, \bar{x}_B3, \bar{y}_B3) + d_S)), \\
    d^C_{\text{road}}(p(k)) &= \min(\max(0, d(p(k), \bar{x}_C1, \bar{y}_C1, \bar{x}_C2, \bar{y}_C2) + d_S), \\
    &\quad \max(0, d(p(k), \bar{x}_C2, \bar{y}_C2, \bar{x}_C3, \bar{y}_C3) + d_S)), \\
    d^D_{\text{road}}(p(k)) &= \min(\max(0, d(p(k), \bar{x}_D1, \bar{y}_D1, \bar{x}_D2, \bar{y}_D2) + d_S), \\
    &\quad \max(0, d(p(k), \bar{x}_D2, \bar{y}_D2, \bar{x}_D3, \bar{y}_D3) + d_S))
\end{align*}
\]  

(3.14)  

\[
\begin{align*}
    (\bar{x}_B4, \bar{y}_B4) &\quad \text{P}^B_{\text{road}} \\
    (\bar{x}_B1, \bar{y}_B1) &\quad (\bar{x}_B2, \bar{y}_B2) \\
    (\bar{x}_B3, \bar{y}_B3) &\quad d_S
\end{align*}
\]  

(3.15)  

\[
\begin{align*}
    (\bar{x}_A4, \bar{y}_A4) &\quad \text{P}^A_{\text{road}} \\
    (\bar{x}_A1, \bar{y}_A1) &\quad (\bar{x}_A2, \bar{y}_A2) \\
    (\bar{x}_A3, \bar{y}_A3) &\quad (\bar{x}_A4, \bar{y}_A4)
\end{align*}
\]  

(3.16)  

The shift \(d_S\) is chosen to 0.5 m for all further implementations.

The cost function in Equation 3.13 only calculates the resulting value for one single point. In practice it seems reasonable to evaluate it for as many points as necessary but as little points as possible of the ego-vehicle. The four corners of the vehicle (as calculated in Equation 3.18) represent a good selection to fulfil this criterion, again a maximum has to be computed.

\[
J_{\text{road}}(k) = \max_{p(k)}(J_{\text{road}}^P(p(k))) \quad \text{with} \quad p(k) \in \{P_1(k), P_2(k), P_3(k), P_4(k)\}
\]  

(3.17)

If all corners of the ego-vehicle are in the set \(A_{\text{road}}\) (expressed by \(J_{\text{road}}(\cdot) = 0\), only one special case is left, where parts of the ego-vehicle could still be overlapping with the prohibited regions, namely if the prohibited area has a corner, as shown in the following Figure 3.8.
3.1.2.1 Edges of the ego-vehicle in prohibited sets

This case visualises the problem that occurs when only the corner points of the ego-vehicle are checked.

All conditions presented in the previous section are fulfilled in this picture (no validation of the road boundary is detected) and still, one can obtain that the vehicle has entered a prohibited region. In this case, a check of all corners is not sufficient. Nevertheless, as a safety-margin $d_S$ is included and an online-optimisation can never cover all situations perfectly, a separate treatment can be dispensed. It has only to be taken into account that this kind of problem exists and might occur, but in general, it induces no hazardous situation.
3.1.3 Calculation of the ego-vehicle corners

As mentioned in Section 3.1, an important part of in silico simulations for crash avoidance is the mathematical clarification if a crash occurred.

The dynamics of the ego-vehicle are sufficiently described in Section 2.3, while the geometry is shown in Figure 3.9. Now one can build up on that knowledge and take a look at the interaction with other objects. To check if the vehicle is in an allowed set, the first step is to calculate the corners of the ego-vehicle from its states and geometry.

\[
\begin{align*}
P_1(k) &= \begin{bmatrix} x_{P_1(k)} \\ y_{P_1(k)} \end{bmatrix} = \begin{bmatrix} x_{P_0(k)} + D_f \cdot \cos(\theta(k)) + \frac{W}{2} \cdot \sin(\theta(k)) \\ y_{P_0(k)} + D_f \cdot \sin(\theta(k)) - \frac{W}{2} \cdot \cos(\theta(k)) \end{bmatrix} \\
P_2(k) &= \begin{bmatrix} x_{P_2(k)} \\ y_{P_2(k)} \end{bmatrix} = \begin{bmatrix} x_{P_0(k)} + D_f \cdot \cos(\theta(k)) - \frac{W}{2} \cdot \sin(\theta(k)) \\ y_{P_0(k)} + D_f \cdot \sin(\theta(k)) + \frac{W}{2} \cdot \cos(\theta(k)) \end{bmatrix} \\
P_3(k) &= \begin{bmatrix} x_{P_3(k)} \\ y_{P_3(k)} \end{bmatrix} = \begin{bmatrix} x_{P_0(k)} - (L - D_f) \cdot \cos(\theta(k)) - \frac{W}{2} \cdot \sin(\theta(k)) \\ y_{P_0(k)} - (L - D_f) \cdot \sin(\theta(k)) + \frac{W}{2} \cdot \cos(\theta(k)) \end{bmatrix} \\
P_4(k) &= \begin{bmatrix} x_{P_4(k)} \\ y_{P_4(k)} \end{bmatrix} = \begin{bmatrix} x_{P_0(k)} - (L - D_f) \cdot \cos(\theta(k)) + \frac{W}{2} \cdot \sin(\theta(k)) \\ y_{P_0(k)} - (L - D_f) \cdot \sin(\theta(k)) - \frac{W}{2} \cdot \cos(\theta(k)) \end{bmatrix}
\end{align*}
\]

(3.18)

As these points define the outer contour of the ego-vehicle, they will be of grave importance when evaluating distance to as well as overlap with obstacles.

Figure 3.9: Corner points of the single track model

The point \(P_0(k)\) is directly defined by the position states of the model \((x_{P_0(k)} = x(k), 
\ y_{P_0(k)} = y(k))\). With knowledge about the geometry of the car and the current orientation \(\theta(k)\), the points \(P_1(k)\) to \(P_4(k)\) can be calculated.
3.1.4 Minimal distance from the ego-vehicle to a point

The first question that occurs is how one calculates the (minimal) distance from the ego-vehicle to another point in the two-dimensional plane.

Important note: all polygons here and in the following are defined counter-clockwise. If defined clockwise, the method of creating normal vectors at each edge would not work sufficiently, because the vectors would point to the inside of the convex polygon and not to the outside as presumed here.

There are three different cases where a point $P_O(k)$ can be located:

a) geometrically oblique to the rectangle, which results in two positive projections on the normal vectors of the rectangle edges

b) directly in front of an edge, which results in one positive projection on the normal vectors of the rectangle edges

c) inside the rectangle, which results in no positive projection on the normal vectors of the rectangle edges

These cases are illustrated in Figure 3.10.

The distance of the point $P_O(k)$ to one of the edges can be simply calculated in one step as shown in d), where $d_c(P_O(k),k)$ denotes the projected distance of a point $P_O(k)$ to the edge spanned by $P_c(k) = [x_c(k), y_c(k)]$ and $P_{c+1}(k) = [x_{c+1}(k), y_{c+1}(k)]$, with $P_0(k) = P_1(k)$.

$$ \vec{n}_c(k) = \begin{bmatrix} y_{c+1}(k) - y_c(k) \\ x_c(k) - x_{c+1}(k) \\ y_{c+1}(k) - y_c(k) \\ x_c(k) - x_{c+1}(k) \end{bmatrix} $$

$$ P^m_c(k) = \frac{P_c(k) + P_{c+1}(k)}{2} $$

$$ d_c(P_O(k),k) = \vec{n}_c(k)^T \cdot \vec{d}_c(P_O(k),k) $$

Now one can obtain the minimum distance from the point $P_O(k)$ to the rectangle using the calculated projected distances from the edges. In case c) this is simply 0 because there is no positive distance. In case b) the only positive distance denotes the minimal distance. In case a) one has to apply Pythagoras’ law to get the resulting minimum distance. Mathematically, this can be expressed as followed.

$$ d_{min}(P_O(k),k) = \left( \sum_{c=1}^{4} d_c(P_O(k),k)^2 \right)^{\frac{1}{2}} $$

This algorithm only works sufficiently for rectangles due to the use of Pythagoras’ law.
3.2 Collision detection for obstacles

The projection method introduced in Subsection 3.2.2 is used to detect collisions with all types of obstacles (except road boundaries) listed in Section 3.1 (including also other static obstacles). It is important to state that the shown methods are only used to validate if a collision occurred on performed simulation steps, and not for collision detection in future predictions where the MATLAB® function inpolygon is used due to faster computation time and its ability to cope with non-convex polygons as used in the predictions. As inpolygon can only check if single points are inside a polygon, it is not as precise as the methods described in the following. Nevertheless, a description of the inpolygon function is provided in Subsection 5.2.1.
3.2.1 Conditions for convexity

The only characteristic obstacles have to fulfil as a condition for the method in Subsection 3.2.2 is the convexity of their descriptive polygon. Since even non-convex polygons can be split into convex ones, the method is quite useful. A polygon is convex if every line segment between two points in the interior, or between two points on the boundary – but not on the same edge – is strictly interior to the polygon, except at its endpoints if they are on the edges. This principle is shown in Figure 3.11. In this figure, the convexity is simply tested by connecting all corners of the polygon with each other. Green lines state connections which are allowed under the previously defined conditions, red ones mark the ones mismatching the condition which therefore identify the polygon as non-convex. If just one such prohibited connection can be found, the whole polygon is denoted non-convex.

![Convexity of polygons](image)

Figure 3.11: Convexity of polygons

If a method explicitly requires convex polygons, and if a splitting of the non-convex polygons causes no further problems, non-convex polygons can be expressed through smaller convex ones, as shown in Figure 3.11.

3.2.2 Collision detection

As the variety of dynamic obstacles was reduced to two geometrical shapes (circles and rectangles), only these two cases have to be taken into account. Methods for collision detection are introduced in the following sections.

3.2.2.1 Circular obstacles

This denotes the easy case for collision detection because the whole systematic is already covered by the method proposed in Subsection 3.1.4 and just needs to be expanded by taking the dimension (in this case the radius) of the obstacle into account.

To guarantee that no collision occurs, the only thing that needs to be checked is if the minimal distance to the centre of the obstacle ($P_O(k)$ in Figure 3.12) – the algorithm for that is listed in Subsection 3.1.4 with the result in Equation 3.23 - is greater than the radius $r_O$ of the obstacle.

\[
\text{collision}(k) = \begin{cases} 
\text{true} & \text{for } d_{\text{min}}(P_O(k), k) < r_O \\
\text{false} & \text{for } d_{\text{min}}(P_O(k), k) \geq r_O 
\end{cases} \quad (3.24)
\]
### Rectangular obstacles

For rectangular objects the base of the algorithm stays the same, only the amount of required iterations rises. One needs to check the projections on all normal vectors of all edges of the ego-vehicle and obstacle to ensure if a collision happens or not. Each edge can yield the result that no collision happened. In this case there is no more need to proceed with the algorithm.

For each edge all points of both rectangles are projected on the normal vector using Equation 3.22 and previous equations. Each of these steps results in minimal and maximal projected (scalar) values for ego (superscript $E$) and obstacle (superscript $O$).

$$
\begin{align*}
    d_{c,\text{min}}^E(k) &= \min (d_c(P_1(k), k), d_c(P_2(k), k), d_c(P_3(k), k), d_c(P_4(k), k)) \\
    d_{c,\text{max}}^E(k) &= \max (d_c(P_1(k), k), d_c(P_2(k), k), d_c(P_3(k), k), d_c(P_4(k), k)) \\
    d_{c,\text{min}}^O(k) &= \min (d_c(P_{O1}(k), k), d_c(P_{O2}(k), k), d_c(P_{O3}(k), k), d_c(P_{O4}(k), k)) \\
    d_{c,\text{max}}^O(k) &= \max (d_c(P_{O1}(k), k), d_c(P_{O2}(k), k), d_c(P_{O3}(k), k), d_c(P_{O4}(k), k))
\end{align*}
$$

(3.25)

c denotes the edge that is currently checked, the algorithm needs to iterate through the set of $c$. A collision occurs if the following condition holds.

$$
\text{collision}(k) = \text{∃} \left( d_{c,\text{max}}^E(k) < d_{c,\text{min}}^O(k) \lor d_{c,\text{max}}^O(k) < d_{c,\text{min}}^E(k) \right) \\
\text{with} \\
c \in \{1,2,3,4,O1,O2,O3,O4\}
$$

(3.26)

For $c = 4$ this check is shown in Figure 3.13.
It can be obtained that in this case a collision can be excluded. The principle should become even clearer by taking a look at the projection A-A. If the projected regions of the two objects do not overlap, the edge to which the current normal vector is assigned acts as a separating plane between the two objects.

In case of a collision the projections overlap for each normal vector and Equation 3.26 yields true.

The method proposed here for rectangles can be easily adapted to general convex polygons. The same mathematical principles hold, and the only difference is the variation in the number of obstacle edges.
Chapter 4

Formulation of the optimisation problem

In the following chapter an overview of the two-layer optimisation approach is provided. Then, the separate layers are described in detail, including the contained optimisation. Finally, a short overview of non-linear optimisation methods as well as of tactics to handle the computational burden is provided.

4.1 Restrictions of the problem

Usually, collision avoidance assistants focus on avoiding collisions on a pre-defined track (e.g. [6, p. 1505]). In general, it is not an easy task to clarify in which state the normal driving of a vehicle is possible without hazard, and for which state intervention by the assistant is necessary. One reason for that is that the planned track is usually not known to the assistant and therefore the trajectory of the car would also have to be predicted to some extent. To be able to tackle this kind of problem, some assumptions have been made to reduce the complexity, as this work should only represent a proof of concept and no fully working collision avoidance assistant that just needs to be implemented into a ‘real’ car.

- The street network is known a-priori, which means all road boundaries and especially the width of the single lanes are known – this way it is possible to define valid trajectories along the lanes.
- The trajectories are also pre-computed, which means that they do not need to be calculated in each step.
- The optimal velocity along these trajectories is known, which means for each point along the trajectory a velocity is given that can be tracked.
- The primary trajectory is known a-priori (reasonable if a navigation system is used).
- The slope of the road is not included, a flat and dry street is assumed (with a friction coefficient, as specified in Equation 2.11).
- Current states and dimensions of all surrounding objects are perfectly known at the time of measurement, this is common in the literature, see e.g. [7, p. 771].
- Only cars, pedestrians, and bicycles occur in the surrounding traffic, later the assistant could be expanded to further categories of traffic participants.
4.2 General design of the optimisation

For the collision avoidance assistant in this work, a two-layer approach is chosen. This means in a first step that the safest trajectory along the available lanes is selected. A performance factor is calculated that specifies how safe the selected trajectory is and according to that the second layer lays its focus more on tracking of the reference or on the avoidance of obstacles. Alternatively to the optimal speed references along the trajectories, a second reference concerning the velocity is also taken into account for each trajectory, in particular one where the deceleration is set to $a(k) = a_{\text{braking}}^{\text{max}} = -4 \frac{m}{s^2}$ to also take into consideration a braking manoeuvre – the velocity reference in these cases is still set to $0 \frac{m}{s}$ to obtain related costs in the optimisation. This principle of trajectories is shown in Figure 4.2.

![Figure 4.2: Concept of a single optimisation step in the two-layer approach](image-url)

Figure 4.1: Concept of a single optimisation step in the two-layer approach
4.2. General design of the optimisation

The advantage of the first layer lies within its parallel computability. All lanes and sub-lanes can be optimised separately, which leads to a far better performance, as parallel computing with multiple CPU-cores is quite common nowadays.

The different steps in Figure 4.1 can be described as followed

1.1) Each lane consists of equally spaced tracking points $x_{\text{ref}}(n)$ and $y_{\text{ref}}(n)$ necessary for position-tracking. Those stay the same for each type of velocity reference on the lane.

1.2) For each lane a typical velocity reference that suits the maximum allowed speed on the road and also speed reduction due to higher curvature are deposited. A second reference concerning the velocity is taken into account, namely one where the deceleration is set to $a(k) = a_{\text{braking}}^{\text{max}} = -4 \text{ m/s}^2$ for each lane, to force a braking reference for the specific lane. The references get passed to the next step.

1.3) The first optimisation process happens in this step, a reference tracking for position and velocity is applied for a (long) prediction horizon of $N_{\text{p}}$ steps in the future. This is possible in a really short computation time because the optimisation problem is quite simple, see 4.3.2.

1.4) Now the results of the optimisations are analysed and a TTC (time to collision) for each trajectory is calculated as a performance criterion to make the references comparable. In this step the prediction model introduced in 3 is used the first time, the previous step 1.3) lays its focus completely on tracking the reference and ignores obstacles. This way one gets a good overview which lane (and which velocity reference) is the least dangerous without obstacle handling.

1.5) As a last step in the first layer, the TTC of all trajectories is compared and the most promising one is selected as a reference for the second layer.

   - the trajectories are not just sorted by their TTC in this step because one can assume that the driver wants to track his primary lane as long as possible
   - over a certain (tunable) TTC threshold the primary lane (with velocity reference) is always selected
   - if the TTC of the primary lane (with velocity reference) is too low, the alternative lane with the best TTC (with velocity reference) is selected
   - if the TTC of all lanes with velocity reference is below a certain threshold, one of the braking references (the one with the best TTC) is selected
   - if all references have a TTC below a certain level, the collision avoidance assistant switches to crashmode, where some elements in the second level cost function get deactivated and obstacles are weighted according to their importance. Also, the deceleration is set to $a(k) = a_{\text{braking}}^{\text{max}} = -4 \text{ m/s}^2$ because a collision is highly probable and the collision energy is generally minimized by lowering the speed of the ego-vehicle. A last important circumstance is to note here: if this mode was entered once, it is kept until the end of the simulation or until a crash happens.

Depending on how low the TTC of the selected reference is, $\lambda_{tr}$ is selected closer to 1 (for situations where the TTC is high) – more weight on tracking in layer 2 – or closer to 0 if the TTC is quite low to achieve the opposite weighting in layer 2.
2.1) The options for the optimisation in layer 2 are set here.

- If the crashmode is not active, nothing is changed in the cost function and the optimisation is executed normally with $\lambda_{tr}$ calculated as in the previous step.

- With active crashmode, $\lambda_{tr}$ is set to 0 which means position-tracking is neglected and only collision avoidance is important now. The cost for crossing the road boundaries is also removed here, since it seems obvious that avoiding a crash with persons involved might be more important than to stay on the road.

2.2) The second optimisation process happens in this step. A weighted reference tracking for position and velocity as well as a weighted obstacle avoidance optimisation is applied for a (short) prediction horizon of $N_p^s$ with options specified as in the step before.

2.3) Now the obtained optimal trajectory is simulated. If a crash is detected (using the methods specified in Section 3.2), the simulation is finished, otherwise the initial position of the ego-vehicle is refreshed and the next optimisation step is computed.

This simple algorithm represents a quite flexible collision avoidance assistant that is also tunable in an easy way – this should happen mostly in step 1.5) where $\lambda_{tr}$ and the favoured trajectory get selected.

4.3 First layer of the optimisation

4.3.1 Trajectory generation

As already mentioned, the references for position and velocity are previously defined using geographical data of the environment and are always accessible for the ego-vehicle. An example is shown in Figure 4.2. The small circles represent the points for position tracking while the bars above represent the velocity reference for the specific point. In addition to the ‘normal’ velocity references, the ones with the fixed deceleration are added to each position reference trajectory. Figure 4.2 contains a primary position trajectory (1) and a secondary position trajectory for the other lane (2). Summing up, 2 lanes (which means 2 different position references) as well as 2 velocity/deceleration references for each of the lanes, which results in a total of 4 different references (each containing of a position- and a velocity/deceleration-reference) can be observed.

![Figure 4.2: Reference-trajectories along a straight road, including velocity reference](image-url)
Each trajectory consists of \( M_{tr} \) points \( (p_{tr}(n), n = 1, \ldots, M_{tr}) \), each containing a reference for position and speed (or deceleration).

The same principle holds for junctions. To provide a better overview, only the velocity reference for the primary trajectory is shown in Figure 4.3. To keep the number of references as low as possible, the junction is first treated as a ‘normal’ road, with a second lane (2) trajectory besides the primary trajectory (1). In a next step, the ‘legal’ lanes for such a junction situation are added, here namely a straight movement (3) and a turn to the right (4) when approaching from a side. As a last step, references for the missing ‘illegal’ lanes straight (5) and to the right of the ego-vehicle (6) are added. After the junction has been crossed, the amount of lanes can again be reduced to two as represented in Figure 4.2 Summing up, 6 lanes (which means 6 different position references) as well as 2 velocity/deceleration-references for each of the lanes, which results in a total of 12 different references (each containing of a position- and a velocity/deceleration-reference) can be observed, which means that a maximum of 12 references has to be handled in junction situation with two crossing streets.

Each of the references consists of points in the X/Y-plane and associated velocity references. For a reference-length of \( M_{ref} \), the following vectors are needed

\[
x_{\text{ref}} = [x_{\text{ref}}(1), \ldots, x_{\text{ref}}(n), \ldots, x_{\text{ref}}(M_{\text{ref}})]
\]

\[
y_{\text{ref}} = [y_{\text{ref}}(1), \ldots, y_{\text{ref}}(n), \ldots, y_{\text{ref}}(M_{\text{ref}})]
\]

\[
v_{\text{ref}} = [v_{\text{ref}}(1), \ldots, v_{\text{ref}}(n), \ldots, v_{\text{ref}}(M_{\text{ref}})]
\]

A distance of 1 m was found to be sufficient between the single reference points in the space domain. This way curves are still tracked in an appropriate way while the amount of reference points is not getting unreasonably high.
4.3.2 Tracking optimisation

In this step, a first optimisation process is executed for each reference trajectory. The optimisation is carried out for the next $N_p$ steps in the future with a sampling time of $T_s = 0.1$ s between the steps and $k^*$ as the present step, as shown in Figure 4.14. $N_p$ is selected to be 50 steps, which means in this case a prediction or optimisation for the upcoming 5 s.

$$u_{opt} = \arg \min_{u(\cdot)} J_{long}(x(k^*), u(\cdot); N_p)$$ (4.4)

with:

$$J_{long}(x(k^*), u(\cdot); N_p) := \sum_{k=k^*}^{k^*+N_p} (J_{tr,pos}(k) + J_{tr,vel}(k) + J_{kamm}(k) + J_{model}(k))$$ (4.5)

subject to:

$$x(k+1) = f(x(k^*), u(k^*)), \quad x(k^*) = x(k^*)$$ (4.6a)

$$u(k) \in U, \quad \forall k \in [k^*, k^* + N_p]$$ (4.6b)

where $x(k)$ represents the states during the optimisation, $u(k)$ represents the inputs during the optimisation, and $f(x(k), u(k))$ represents the non-linear state-equations which were previously defined in Equation 2.2. $J_{kamm}(k)$ and $J_{model}(k)$ were already introduced in Equation 2.12 and Equation 2.6.

For tracking the position, an appropriate function has to be found, in literature, e.g. in [4, p. 1357], an exponential function is used which is quite suitable for this kind of task.

$$J_{tr,pos}(k) = \min \left(1 - \exp \left(-\frac{(x_{ref} - x(k))^2 + (y_{ref} - y(k))^2}{s_{pos}^2}\right)\right)$$ (4.7)

All operations on vectors (except the finding of a minimum in the end) are meant as element-wise operations. For selection of the correct velocity reference, the vector-index of the found minimum is necessary, in MATLAB® this can be easily implemented. It is assumed here that $\text{index}$ returns the index of the minimum of a vector in the same way as $\text{arg}$ yields the position where a function has its minimum value.

$$i_{min}(k) = \text{index} \min \left(1 - \exp \left(-\frac{(x_{ref} - x(k))^2 + (y_{ref} - y(k))^2}{s_{pos}^2}\right)\right)$$ (4.8)

The value of $J_{tr,vel}(k)$ can then be calculated with a simply scaled quadratic function

$$J_{tr,vel}(k) = \frac{(v_{ref}(i_{min}(k)) - v(k))^2}{s_{vel}^2}$$ (4.9)

$s_{pos}$ and $s_{vel}$ are scaling factors and were both selected to the value 5 in this part of the optimisation.
4.3. First layer of the optimisation

Instead of the proposed NMPC tracking approach, the selection of the best trajectory from a bundle of pre-defined trajectories to all other lanes, or a lane switch with trajectories represented by simple geometrical functions as represented in [11, pp. 155] might be promising alternatives – especially in terms of computational burden and time.

4.3.3 Evaluation of the optimised trajectories

The optimisation from Equation 4.4 is evaluated for each reference. In the end, one obtains a single $u_{opt}$ for each trajectory. These can then be used to simulate the future positions of the ego-vehicle using Equation 2.2. In the next step, the position and dimensions of the ego-vehicle are used to represent it as a cloud of points (as shown in Figure 4.5). As this procedure consists only of trivial geometric operations, it is not further described here.

The enveloping polygon of all obstacles in each step is generated using the method from Subsection 5.2.1. This way possible crashes for each step can be evaluated using the MATLAB® function inpolygon.

Figure 4.4: Visualisation of $J_{tr,pos}$ and $J_{tr,vel}$

Figure 4.5: Visualisation of a detected crash for a step
A short step-by-step description of the algorithm can be formulated.

- For each reference
  - For each prediction step
    * calculate the future position of the ego-vehicle using Equation 2.2
    * for the position calculate a cloud of points describing the ego-vehicle and its dimensions
    * predict the enveloping polygon of all obstacles for the specific step using the method from Subsection 5.2.1
    * check if elements of the point-cloud lie within one of the predicted polygons using the MATLAB® function inpolygon
      - if no overlap is detected, proceed with the algorithm
      - if an overlap is detected, save the obtained TTC (resulting from the duration of the collision-free simulation steps times the step-duration $T_s = 0.1$ s) and resume with the next reference
    * check if the last prediction step is reached
      - if yes, proceed with the next step
      - if no, proceed with the next reference

Figure 4.5 shows the visualisation of a detected collision of the ego-vehicle (blue point-cloud enveloped by a blue line) with an obstacle (black point-frame connected by a black line). The red crosses in the ego-car represent points of the point-cloud that are within the region of the predicted obstacle-polygon.

By using this method, not only crashes can be detected quite easily, but also the region of the ego-vehicle involved in the crash. This way the possible severity of a crash for the ego-vehicle can be observed and analysed – which is not yet implemented but promising for future development.

The final result after this algorithm is a calculated TTC for each reference which makes them comparable quite easily. The TTC for references without a crash is set to the prediction time ($5$ s for the used $N_P = 50$ and $T_s = 0.1$ s).
4.3.4 Selection of a trajectory for the next layer

The selection of a reference trajectory is a quite tricky task, as it has a huge influence on the behaviour of the whole collision avoidance assistant and requires a significant amount of tuning.

To give a first impression, Figure 4.6 shows exemplary TTC-values for a situation with two lanes.

![Diagram of TTC values](image)

**Figure 4.6: Visualisation of exemplary TTC-values**

The references are sorted primarily by their type (velocity tracking with reference and as an alternative fixed deceleration – this braking cases are marked with a (b)) and secondarily by their order starting with the primary lane (p). Usually, by braking one will get the best TTC in almost every situation which is quite intuitive, as the same reference is tracked and only the velocity is constantly reduced.

Due to that, the simple comparison of the various references would by far not be sufficient, as the system would mostly brake and make normal regular driving impossible that way.

In standard cases the driver simply wants to follow the desired path to his goal (the primary reference) with the optimal velocity for the situation, which is represented by a tracking of the first reference.

If out of a sudden an obstacle appears and other lanes are free, a manoeuvre to one of those free lanes might be sufficient to avoid a crash without the necessity for sudden braking – the critical situation is avoided and no restriction on comfort is given, the trip can be continued without interruption.

In case that no other lane is adequately safe, the option to simply brake gets more attractive. This way most collisions can be prevented, and if the obstacle disappears, or other lanes get safe again, the trip can be continued as before.
The last and most extreme case appears if not even by braking a sufficiently high TTC can be reached. For these events the previously mentioned crashmode is entered which directly influences the design of the second layer of the optimisation, not only by trajectory selection or tuning of the parameter $\lambda_{tr}(k)$.

Therefore, various thresholds are implemented to convert these verbal requirements to a sufficient technical framework of the collision avoidance assistant.

**Table 4.1: Overview of the TTC-regions and the consequences for the optimisation**

<table>
<thead>
<tr>
<th>Region</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{max} \geq \text{TTC}(k) &gt; t_{prime}$</td>
<td>If the primary reference has a TTC in this region, it is always selected as the reference trajectory. The focus in this region lies on normal driving with the optimal velocity deposited for this reference.</td>
</tr>
<tr>
<td>$t_{prime} \geq \text{TTC}(k) &gt; t_{brake}$</td>
<td>If the primary reference has a TTC in this region, all other non-braking references are also taken into account and the reference with the highest TTC is selected as the reference trajectory for the next layer. This represents the case where a change to a safe lane seems better than staying on the primary reference.</td>
</tr>
<tr>
<td>$t_{brake} \geq \text{TTC}(k) &gt; t_{crash}$</td>
<td>If the non-braking reference with the highest TTC lies within this region, the braking references are also taken into account and out of all available references the one with the highest TTC is selected. This refers to the case where also other references are no longer safe enough for an evasive manoeuvre and therefore the option of braking has to be taken into account.</td>
</tr>
<tr>
<td>$t_{crash} \geq \text{TTC}(k) \geq 0$</td>
<td>If the TTC of all references lies within this region, it can be assumed that a collision is highly probable. Therefore, the second layer is switched to the previously mentioned crashmode. This is the worst case because tracking of a position reference is no longer helpful.</td>
</tr>
</tbody>
</table>

For this work the thresholds were selected as followed

- $t_{\text{prime}} = 2.5\, \text{s}$
- $t_{\text{brake}} = 2.0\, \text{s}$
- $t_{\text{crash}} = 1.5\, \text{s}$

The calculation of $\lambda_{tr}(k)$ is also depending on TTC of the selected reference trajectory ($\text{TTC}_{sel}(k)$). It calculates as

$$\lambda_{tr}(k) = \min(0.7, s_{\lambda} \cdot \text{TTC}_{sel}(k)) \quad (4.10)$$

where $s_{\lambda}$ acts as a tuning factor that was selected to $s_{\lambda} = 1/3$ for the presented collision avoidance assistant.
4.4 Second layer of the optimisation

In the second layer of the optimisation problem, the knowledge obtained from the first layer, concerning the safety of the selected trajectory and importance of collision avoidance versus tracking (via $\lambda_{tr}(k)$), are taken into account.

4.4.1 Setting the options for the second optimisation

The reference trajectory selection and the calculation of $\lambda_{tr}(k)$ as well as the decision of a possible switch to the crashmode are carried out in the first layer. In the second layer, as a first step, this knowledge is used to formulate the optimisation. Before the optimisation problem is formulated, a detailed explanation is given concerning the way obstacles are treated and the way their contribution to the cost function is composed.

4.4.2 Representation of the obstacles

There are many different ways one can handle obstacles when trying to avoid collisions. One option would be to represent the ego-vehicle as well as the obstacles in their natural shape or a simplified version of it, like rectangles for cars, circles for trees, and so on. This leads to the problem that the objects and their exact shapes have to be classified quite well by the sensor system of the car and also the function representing the distance to the object can become quite complex which leads to a high computational burden when evaluating the optimisation in a real-time situation.

Typical sensor systems like RADAR or LIDAR usually recognize the environment as a cloud of points which then get computed to objects with specific states and dimensions in a next step. For this principle, sensor fusion, as described in [12, pp. 142] or [8, pp. 94], often plays a huge role.

The concept of this cloud of points leads to the idea to represent the obstacles in the same way for the optimisation – as a row of primitive points forming the shape of the obstacle. This has the advantage that the distance to a point in space (as well as every other kind of performance criterion) is not complex to compute, and because a lot of simple operations have to be computed in parallel, this would be a task that could be done perfectly by using parallel computing. In this work – as everything is based on pure simulation – the position as well as the dimensions of all obstacles are assumed to be perfectly known at the moment of ‘measurement’. This means that the shapes of the obstacles can be converted to a representation by points, which is especially useful for non-convex shapes, as results from the prediction proposed in Chapter 5.

A further simplification here is that none of the obstacles is shadowed by other ones and therefore, all obstacles in a certain range are detected, even if they would not be visible to the sensor systems of the car in real cases. For a further development of the collision avoidance assistant, this task has to be tackled, but for the current proof of concept this is regarded as an acceptable simplification.
4.4.3 Performance criterion representing the cost of obstacles

For the calculation of costs induced by dynamic objects, many approaches and different types of evaluation exist, mostly involving some kind of function that depends on the distance from the ego-vehicle to the obstacle. In this work, the focus is set on two very promising methods.

- A distance based function similar to Equation 4.7 evaluated from the front and the rear end of the ego-vehicle to all points of all obstacles.
- An approach based on the parallax angle from the rear edge of the ego-vehicle to all points of all obstacles, as similarly proposed in [6, p. 1506].

These two methods are finally combined to receive a criterion that is adapted quite well to the shape of the ego-vehicle and takes into account all possible threats induced by approaching obstacles.

4.4.4 Distance-based validation of obstacles

As already mentioned before, for this method the distance from the rear and the front end of the ego-vehicle to a point on the plane is used to receive a performance criterion of that point.

As the first computation step, the distances from the front \(d_f(k,i)\) and the rear end \(d_r(k,i)\) at step \(k\) of the ego-vehicle to the point \(p_0(k,i) = [x_O(k,i), y_O(k,i)]\) are calculated. This involves geometric measures of the ego-vehicle.

\[
d_f(k,i) = \sqrt{(x_f(k) - x_O(k,i))^2 + (y_f(k) - y_O(k,i))^2} \tag{4.11}
\]
\[
d_r(k,i) = \sqrt{(x_r(k) - x_O(k,i))^2 + (y_r(k) - y_O(k,i))^2} \tag{4.12}
\]
\[
x_f(k) = x(k) + D_f \cdot \cos(\theta(k)) \tag{4.13}
\]
\[
y_f(k) = y(k) + D_f \cdot \sin(\theta(k)) \tag{4.14}
\]
\[
x_r(k) = x(k) - (L - D_f) \cdot \cos(\theta(k)) \tag{4.15}
\]
\[
y_r(k) = y(k) - (L - D_f) \cdot \sin(\theta(k)) \tag{4.16}
\]
This way the cost for a single point \( p_O(k,i) \) can be calculated by applying a similar function, as in Equation 4.7

\[
J_{pot}(k,i) = \max \left( \frac{\text{val}(i) \cdot \exp \left( -\frac{d_f(k,i)^2}{s_{pot}^2} \right)}{\text{val}(i) \cdot \exp \left( -\frac{d_r(k,i)^2}{s_{pot}^2} \right)} \right) \tag{4.17}
\]

The advantage of this type of calculation is that the process is equal for all points. Assuming an amount of points \( M_p \), these points can be arranged as vectors, as previously shown for the tracking approach.

The factor \((i)\) depends on the type of obstacle, the corresponding point \( p(k,i) \) indicates when the assistant is in crashmode and otherwise denotes to 1.

\[
\text{val}(i) = \begin{cases} f_{type}(p_O(\cdot,i)) & \text{for active crashmode} \\ 1 & \text{otherwise} \end{cases} \tag{4.18}
\]

with

\[
f_{type}(p(\cdot,i)) = \begin{cases} 10 & \text{if } p_O(\cdot,i) \text{ represents a pedestrian or a bicycle} \\ 5 & \text{if } p_O(\cdot,i) \text{ represents a car} \\ 1 & \text{otherwise} \end{cases} \tag{4.19}
\]

For each point listed in \( x_O(k) \) and \( y_O(k) \), a corresponding value \( \text{val}(i) \) has to be specified. The values defined in \( f_{type}(p_O(\cdot,i)) \) are selected according to the safety importance for the specific type of obstacle and are just exemplary here. Further studies are required to differentiate between the ‘value’ of obstacles – especially in situations where the life of human beings is at stake.

\[
x_O(k) = [x_O(k,1), \cdots, x_O(k,i), \cdots, x_O(k,M_O)] \\
y_O(k) = [y_O(k,1), \cdots, y_O(k,i), \cdots, y_O(k,M_O)] \tag{4.20}
\]

If it is assumed that all mathematical operations (except matrix multiplications) act elementarily on the single vector elements. The proposed functions for one single element can be calculated for all points \( M_O \) in an easy way.

\[
d_f(k) = \sqrt{(x_f(k) - x_O(k))^2 + (y_f(k) - y_O(k))^2} \tag{4.21}
\]

\[
d_r(k) = \sqrt{(x_r(k) - x_O(k))^2 + (y_r(k) - y_O(k))^2} \tag{4.22}
\]

\footnote{An interesting approach concerning this topic was created by the MIT - see \cite{[13]}.}
J_{pot}(k) = \max \left( \left( \text{diag}(\text{val}) \cdot \exp \left( -\frac{d_f(k)^2}{s_{pot}^2} \right) \right)^T, \left( \text{diag}(\text{val}) \exp \left( -\frac{d_r(k)^2}{s_{pot}^2} \right) \right)^T \right) \\
(4.23)

s_{pot} \text{ is a scaling factor and was selected to the value 2 in this part of the optimisation.}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure4.8}
\caption{Visualisation of $J_{pot}$}
\end{figure}

\subsection{Parallax-angle-based validation of obstacles}

For this method, the parallax angle from the rear end to all available points is calculated and then, the resulting vector is maximized to find the point with the biggest parallax angle. At first, the method is implemented again for one single point and then expanded to all available points.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure4.9}
\caption{Required measures to calculate $J_{par}(k, i)$ for a single point}
\end{figure}

At the first computation step, the distances from the two rear corners of the ego-vehicle to the point $p_O(k,i) = [x_O(k,i), y_O(k,i)]$ ($l_l(k,i)$ for the distance from the rear left corner and $l_r(k,i)$ for the distance from the bottom right corner) are calculated.
4.4. Second layer of the optimisation

\[ l_{l}(k,i) = \sqrt{(x_{l}(k) - x_{p}(k,i))^2 + (y_{l}(k) - y_{p}(k,i))^2} \] \hspace{1cm} (4.24)

\[ l_{r}(k,i) = \sqrt{(x_{r}(k) - x_{p}(k,i))^2 + (y_{r}(k) - y_{p}(k,i))^2} \] \hspace{1cm} (4.25)

\[ x_{l}(k) = x(k) - (L - D_f) \cdot \cos(\theta(k)) - \frac{W}{2} \cdot \sin(\theta(k)) \] \hspace{1cm} (4.26)

\[ y_{l}(k) = y(k) - (L - D_f) \cdot \sin(\theta(k)) + \frac{W}{2} \cdot \cos(\theta(k)) \] \hspace{1cm} (4.27)

\[ x_{r}(k) = x(k) - (L - D_f) \cdot \cos(\theta(k)) + \frac{W}{2} \cdot \sin(\theta(k)) \] \hspace{1cm} (4.28)

\[ y_{r}(k) = y(k) - (L - D_f) \cdot \sin(\theta(k)) - \frac{W}{2} \cdot \cos(\theta(k)) \] \hspace{1cm} (4.29)

As a next step, the parallax angle is to be calculated.

Figure 4.10: Basic triangle to represent the law of cosines

The basis of the calculation is the law of cosines, namely

\[ c^2 = a^2 + b^2 - 2 \cdot a \cdot b \cdot \cos(\gamma) \] \hspace{1cm} (4.30)

Figure 4.11: Detailed view of the geometrical composition when calculating \( J_{\text{par}}(k, i) \) for a single point
Now one can substitute the geometric sizes given here for the corresponding elements in Equation 4.30 and solve the equation for $\gamma_p(i)$. An important fact to mention here is that just the region in front of the car is to be observed in that way. Therefore, an additional criterion has to be introduced. A valid method for this task was already implemented to separate the road boundary region from the street. The proposed function from Equation 3.8 can be used here.

\[ W^2 = l_r(k,i)^2 + l_l(k,i)^2 - 2 \cdot l_r(k,i) \cdot l_l(k,i) \cdot \cos(\gamma_p(k,i)) \] (4.31)

\[ \gamma_p(i) = \begin{cases} \text{val}(i) \cdot \arccos \left( \frac{l_r(k,i)^2 + l_l(k,i)^2 - W^2}{2 \cdot l_r(k,i) \cdot l_l(k,i)} \right) & \text{for } d(p_O(k,i), x_r(k), y_r(k), x_l(k), y_l(k)) \leq 0 \\ 0 & \text{otherwise} \end{cases} \] (4.32)

As $\gamma_p(i)$ returns values in the range $[0, \text{val}(i) \cdot \pi]$, the value of the parallax angle can now simply be normalised on the range $[0, \text{val}(i)]$ by division with $\pi$ to get the performance criterion $J_{\text{par}}(i)$, which is shown in Equation 4.33.

\[ J_{\text{par}}(k,i) = \frac{1}{\pi} \cdot \gamma_p(k,i) \] (4.33)

Once again, the method is to be applied on all $M_p$ points, once more in the form of a vector (see Equation 4.20).

\[ l_l(k) = \sqrt{(x_l(k) - x_O(k))^2 + (y_l(k) - y_O(k))^2} \] (4.34)

\[ l_r(k) = \sqrt{(x_r(k) - x_O(k))^2 + (y_r(k) - y_O(k))^2} \] (4.35)

\[ \gamma_p(k) = \begin{cases} \text{diag(val)} \cdot \arccos \left( \frac{l_r(k)^2 + l_l(k)^2 - W^2}{2 \cdot l_r(k) \cdot l_l(k)} \right)^T & \text{for } d(p_O(k), x_r(k), y_r(k), x_l(k), y_l(k)) \leq 0 \\ 0 & \text{otherwise} \end{cases} \] (4.36)

\[ J_{\text{par}}(k) = \frac{1}{\pi} \cdot \max \left( \gamma_p(k) \right) \] (4.37)

Again, all operations on matrices (except matrix multiplications) are handled as element-wise operations.

![Figure 4.12: Visualisation of $J_{\text{par}}$](image)
4.4. Second layer of the optimisation

4.4.6 Combination of both approaches

Both methods grant their own advantages, therefore, a combination is used to obtain the best performance. The maximum of both criteria is taken to get the best of the two worlds.

\[ J_{par,pot}(k) = \max ([J_{par}(k), J_{pot}(k)]) \]  \hspace{1cm} (4.38)

\( J_{par} \) represents the shape of the ego-vehicle quite well and is still suitable for higher distances. Obstacles on the side are captured as less dangerous as obstacles directly in front of the ego-vehicle.

\( J_{pot} \) comes into action for closer obstacles. Front and rear side of the ego-vehicle were selected as an origin because it seems obvious that most collisions with other traffic participants will happen from the front or the rear side.

![Visualisation of the combination of \( J_{par} \) and \( J_{pot} \) according to Equation 4.38](image)

Figure 4.13: Visualisation of the combination of \( J_{par} \) and \( J_{pot} \) according to Equation 4.38

4.4.7 Optimisation in the second layer

In this step, a second optimisation process is executed for the previously selected reference trajectory. The optimisation is carried out for the next \( N_s^P \) steps in the future with a sampling time of \( T_s = 0.1 \) s between the steps. \( N_s^P \) is selected to be 20 steps, which means in this case a prediction or optimisation for the upcoming 2 s. This is short enough to enable a sufficient optimisation time and quality, but also long enough to give a sufficiently long time interval for reaction.

\[ u_{opt} = \arg \min_{\tilde{u}()} \begin{cases} J_{crash} (x(k^*), \tilde{u}(); N_s^P) & \text{for crashmode}(k^*) = \text{true} \\ J_{short} (x(k^*), \tilde{u}(); N_s^P) & \text{for crashmode}(k^*) = \text{false} \end{cases} \]  \hspace{1cm} (4.39)
with:

\[
J_{\text{short}}(x(k^*), u(\cdot); N_p^s) := \sum_{k=k^*}^{k^*+N_p^s} g(k) \cdot (J_{\text{par,pot}}(k) + J_{\text{road}}(k)) \cdot (1 - \lambda_{\text{tr}}(k)) + \\
\sum_{k=k^*}^{k^*+N_p^s} g(k) \cdot (J_{\text{tr,pos}}(k) + J_{\text{tr,vel}}(k)) \cdot \lambda_{\text{tr}}(k) + \\
\sum_{k=k^*}^{k^*+N_p^s} g(k) \cdot (J_{\text{kamm}}(k) + J_{\text{model}}(k))
\]

(4.40)

\[
J_{\text{crash}}(x(1), u(\cdot); N_p^s) := \sum_{k=k^*}^{k^*+N_p^s} g(k) \cdot (J_{\text{par,pot}}(k) + J_{\text{tr,vel}}(k) + J_{\text{kamm}}(k) + J_{\text{model}}(k))
\]

(4.41)

where \( g(k) \) represents a scaling factor that weights values in the close future higher than values in the far future:

\[
g(k) = \left(1 + 0.5 \cdot \frac{k^* + N_p^s - k}{N_p^s}\right)
\]

(4.42)

subject to:

\[
x(k+1) = f(x(k), u(k)), \quad x(k^*) = x(k^*)
\]

(4.43a)

\[u(k) \in \mathcal{U}, \quad \forall k \in [k^*, k^*+N_p^s]
\]

(4.43b)

where \( x(k) \) represents the states during the optimisation, \( u(k) \) represents the inputs during the optimisation, and \( f(x(k), u(k)) \) represents the non-linear state-equations which were previously defined in Equation 2.2. \( J_{\text{kamm}}(k) \) and \( J_{\text{model}}(k) \) were already stated in Equation 2.12 and Equation 2.6. The expression \( J_{\text{road}}(k) \) that defines if the ego-vehicle stays on the road is defined in Equation 3.17 and is removed from the optimisation in crash-mode where the crossing of the road boundaries is allowed. The tracking of the reference trajectory obtained by \( J_{\text{tr,pos}}(k) \) for the position and \( J_{\text{tr,vel}}(k) \) for the velocity is defined in Equation 4.7 and Equation 4.9. The last – but with regard to collision avoidance most important – expression \( J_{\text{par,pot}}(k) \) is defined in Equation 4.38.

### 4.5 Short description of non-linear problems in general

In this section a rough description of the method used for non-linear optimisation shall be given. As the work lays its focus more on a working approach for a collision avoidance assistant than on the deeper optimisation of the numerical tools used to solve the problem, the concepts are presented in a short and generalised way, mostly relying on [14, p. 39].

#### 4.5.1 General formulation

A NMPC problem can be formulated as a general non-linear programming problem in the form

\[
\min_u V(u)
\]

(4.44)
subject to

\begin{align}
G(u) & \leq 0 && (4.45) \\
H(u) & = 0 && (4.46)
\end{align}

where $u$ is a vector that contains the unknown decision variables.

The choice of the right numerical optimisation solver strategy has a significant impact on the need for computational resources (in other words the CPU time required for the solution of the problem to converge and meet the requirements in terms of tolerance) and the quality of the solution. The initial guess plays a huge role in optimisation, therefore, the solutions of the previous run are used as an initial guess for the more complex optimisation in the second layer of the collision avoidance assistant.

### 4.5.2 Condensed and un-condensed formulation of the problem

When solving an optimisation, one has the options of a condensed and an un-condensed formulation of the problem. This leads to different structures that have to be handled by the solver.

Un-condensed formulations contain states and inputs of the system as optimisation variables. Therefore, the system to solve is less complex, but on the other hand, the amount of optimisation variables is far higher, and also non-linear boundary conditions describing the relations of the non-linear system have to be taken into account.

Condensed formulations handle the problem – as the name suggests – in a condensed way, which means that all the relations of the non-linear system are included into the system to solve which becomes much more complex this way, with the advantage that no additional boundaries are needed and only the inputs act as optimisation variables for this kind of problem.

As one can obtain, both formulations have their advantages, disadvantages, and use-cases which are described in greater detail in [14, pp. 10]. For this work, a condensed formulation was selected after comparing the performance for the use-case.

### 4.5.3 Limitation of the computation time

Since the optimisation is carried out with CasADi [15], it was possible to set a maximum CPU time for the optimisation problems. To stay real-time capable, the whole problem has to be solved in a restricted amount of time. For this concept, a time limitation of 0.05 s for the first level of optimisation and a time limitation of 0.4 s for the second layer of the optimisation were selected. In theory, the optimisation for the first layer could be easily computed in parallel which was not implemented here. Also, neither the time required for the composition of the optimisation problem nor the time required for prediction are

\[2\text{The optimisation works for such small computation times, but in some cases non-optimal trajectories are generated. Therefore, the time restriction was set to 0.5 s for the first layer and 1 s for the second layer when evaluating the scenarios presented in Chapter 7. This way the concept and not the (partly insufficient) optimisation method are set into focus.}\]
taken into account. But as the optimisation finishes in a limited amount of time, it is
without doubt that the whole process can be calculated within 0.5 s after some further code optimisations.

As already mentioned several times, this is just a proof of concept and no finished product ready for a real application.

### 4.5.4 Nonlinear programming

It can be said that Newton’s method for iterative solution of non-linear algebraic equations is the backbone of most numerical optimisation methods, as stated in [14, p. 40]. For a non-linear vector equation \( f(u) = 0 \), Newton’s method starts with an initial guess vector \( u^0 \). It then generates a sequence of guesses \( u(k) \) indexed by \( k = 1, 2, 3, \ldots \). For the generation of the guesses, the following Equation 4.47, which itself results from linearisation using Taylor’s theorem, is used.

\[
f (u(k)) + \nabla_u f (u(k)) (u(k + 1) - u(k)) = 0 \tag{4.47}
\]

Equation 4.47 defines a set of linear algebraic equations. Those can be solved for \( u(k + 1) \) using numerical linear algebra. This calculation as well as the computation of the function \( f \) and its gradient (Jacobian matrix) \( \nabla_u f \) can be seen as the main parts to influence computational complexity. Newton’s method relies on linearisation. Therefore, it has only local convergence with the advance of a quadratic convergence rate, as stated in [16, p. 288].

The method given is used in non-linear programming to solve non-linear algebraic equations which are closely related to the first order optimality conditions of 4.44, which are known as Karush-Kuhn-Tucker (KKT) conditions [16, p. 328].

\[
\begin{align*}
\nabla_u L (u^*, \lambda^*, \mu^*) &= 0 \quad (4.48a) \\
H (u^*) &= 0 \quad (4.48b) \\
G (u^*) &\leq 0 \quad (4.48c) \\
\mu^* &\geq 0 \quad (4.48d) \\
G_i (u^*) \mu_i^* &= 0, i = 1, \ldots, n_G \quad (4.48e)
\end{align*}
\]

\( n_G \) defines the number of inequality constraints. The Lagrangian function is defined as

\[
L(u, \lambda, \mu) = V(u) + \lambda^T H(u) + \mu^T G(u) \quad (4.49)
\]

As clearly visible, inequality conditions are included in the KKT conditions, therefore, a direct application of Newton’s method is no longer allowed.

In order to solve this problem, different non-linear programming methods can be used. These methods differ conceptually in the way the KKT conditions – being mixed equations and inequalities – are used to formulate a sequence of non-linear equations.

The different non-linear programming methods also differ with respect to approximations used for the gradient \( \nabla_u f \) of the resulting set of equations. Equation 4.48a requires the
4.5. Short description of non-linear problems in general

calculation of a gradient – namely for the Jacobian matrix of the Lagrangian \( \nabla_u L \) – to be solved. The computation of \( \nabla_u f \) generally requires the expensive computation or approximation of the matrix \( \nabla^2_u L \), known as the Hessian matrix of the Lagrangian, which is another problem that has to be tackled by the non-linear programming methods.

Various methods exist to handle the proposed KKT-conditions, e.g. Sequential Quadratic Programming (SQP) which is further described in [14, pp. 41], or Interior Point Methods (IP), further described in [14, pp. 43].

In this work an Interior Point Method (namely IPOPT with the linear solver MUMPS which is also the standard method in the MATLAB\textsuperscript{®} toolbox CasADi) is used to solve the optimisations, as it performs quite well for problems with the underlying structure. Therefore, a short introduction to IP-methods, as also presented in [14, pp. 43], is given in the following.

Interior point methods deal with the inequality constraints of the KKT-conditions in a fundamentally different way than SQP methods. The KKT conditions that concern the inequality constraints – in particular Equation 4.48c – are replaced by a smooth approximation, as proposed in [17, p. 83]:

\[
G_i (u^*) \mu^*_i = \tau, i = 1, \ldots, n_G
\]  

Now the process of solving the resulting set of algebraic non-linear equations with Newton’s methods becomes equivalent to solving the following approximate problem. Here, the inequality constraints are handled by a logarithmic barrier function:

\[
\min_u \left( V(u) - \tau \sum_{i=1}^{n_G} \log (-G_i(u)) \right) \quad (4.51)
\]

subject to

\[
H(u) = 0 \quad (4.52)
\]

The parameter \( \tau > 0 \) defines a central path in the interior of the feasible region of the problem towards the optimum as \( \tau \to 0 \) – which is also the reason why IP methods are called as they are called. Once the solution for a given \( \tau > 0 \) is found, the parameter \( \tau \) can be reduced by some factor in the next step of the Newton iteration.

Practical implementations of an IP method typically use Newton’s method to compute a search direction. Previously mentioned challenges related to the computation of the Hessian matrix and a limited validity of the linearisation of the Newton method remain similar to SQP, as described in [14, pp. 41].

This acts just as a small conceptional overview, inspired by [14]. For a deeper insight, refer to [16] or [17].
4.6 Blocking as a way to reduce computational burden

One way to cope with the high computational burden that occurs when optimising a non-linear problem is to simply group the optimisation variables for $N_{\text{block}}$ steps in time. This way the amount of variables to optimise is reduced dramatically and in the same way the computational burden is reduced.

![Solution of the open loop optimisation problem at $k = k^*$](image)

Figure 4.14: Visualisation of a blocking strategy for the first layer of optimisation

The example in Figure 4.14 shows the blocking procedure for the first optimisation layer where $N_p^* = 20$. $N_{\text{block}}$ is selected to the value 5, this way the amount of variables to optimise is reduced from 20 to 4.

Since a car represents a relatively sluggish system compared to other technical applications, the holding of inputs for longer time durations than the sampling time $T_s$ (selected to $T_s = 0.1$ s) seems reasonable. In most applications it is not common to hold the optimised value for longer than a sample as the system input, therefore, the first ‘block’ is usually just one sample long and further blocks are selected bigger to ease computing for regions that are further in the future – this concept is only valid if the computation can be done within one sample. This is at the current state of development not the case here. A minimum of 0.5 s (equivalent to 5 steps) is required for computation. Therefore – for the presented application – a selection of equidistant blocks seems reasonable because of the sluggish dynamics of the system as well as the high computational burden which can be eased by using the values for further blocks for a warm start of the next run.
4.6. Blocking as a way to reduce computational burden

The taken blocking configurations for the first and the second layer of optimisation are listed below in Table 4.2.

Table 4.2: Summary of the blocking preferences

<table>
<thead>
<tr>
<th></th>
<th>First Layer</th>
<th>Second Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimisation steps</td>
<td>$N_p^1 = 50$</td>
<td>$N_p^2 = 20$</td>
</tr>
<tr>
<td>Block size for the input $a_k$</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Block size for the input $\delta_k$</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Optimisation variables without blocking</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>Optimisation variables with blocking</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>
Chapter 5

Prediction for dynamic obstacles

First of all, this chapter provides some theoretical background covering advantages and disadvantages of several prediction methods mostly taken from [8] – with a focus on the physics-based approach used in this work. As a next step, different methods of uncertainty-handling are covered. An overview of the selected prediction method is given, and as a last step, the method is evaluated and described in detail.

5.1 Different prediction methods

Physics-based motion models are based on first principles like constant velocity, acceleration, or yaw rate.

- Advantages: Low complexity and low computational effort.
- Disadvantage: Performance is limited since external influences are omitted.

Manoeuvre based motion models provide a prediction by recognition of possible manoeuvres a driver could perform.

- Advantage: An introduction of prototype trajectories enables in general a higher prediction performance for longer horizons because they represent ‘real’ driving.
- Disadvantage: The interactions between traffic participants are still not considered.

Interaction-aware motion models take into account the actions of other vehicles – mostly by using methods from Game Theory like the Level-k Game Theory approach presented in [18, pp. 91] or [19, pp. 3217].

- Advantage: They consider the interaction between the driver and other traffic participants.
- Disadvantage: The complexity of the problem increases dramatically. This can be challenging for on-line-application in real traffic situations.

Due to the general uncertainties in collision situations as well as the lack of trajectory-data from collisions, a manoeuvre based approach was disqualified for this work. Since Game Theory approaches are also mostly based on standard trajectories (‘action sets’ to reduce the computational burden – see e.g. [19, p. 3216]), the same reasons for disqualification
apply. Therefore, a physics-based motion model was selected in the end, as this kind of model interprets the current movement of the vehicle and is therefore able to give a useful estimate of future movement (for short time horizons).

5.1.1 Differences in terms of physics-based motion models

Physics-based motion models represent vehicles as dynamic entities which are governed by the laws of physics. A linking of control inputs (e.g. steering, acceleration), car properties (e.g. weight), and external conditions (e.g. friction coefficient) is used to evaluate possible future states of the vehicle. Physics-based motion models can be roughly separated into two groups: dynamic and kinematic models.

Dynamic models describe the motion of the vehicle based on Lagrange equations. These models take into account the different forces that affect the vehicles’ motion, like for example longitudinal and lateral tire forces. The complex physics of the car is modelled to a certain depth, therefore, those models can get extremely large and involve many internal parameters of the vehicle. This leads to a high computational burden.

Kinematic models describe the motion of the vehicle based on the mathematical relationship between the movement parameters (position, velocity, acceleration). The forces that affect the motion are usually not considered. Such simple models are more popular for trajectory prediction than dynamic models because they result in a low computational burden, require a lot less parameters of the vehicle, and their accuracy is usually sufficient for this type of applications.

5.1.2 Handling of uncertainties

According to [8], three main concepts of uncertainty handling exist in the literature.

The first concept is to simply apply an evolution model to the current state of a vehicle, assuming that the current state is perfectly known and that the evolution model is a perfect representation of the motion of the vehicle. In other words, perfect conditions are presumed and no uncertainty handling happens. This method is fast but performs quite badly in terms of safety.

A more advanced concept is the Gaussian noise simulation where uncertainties on the current states of the vehicle and the evolution of the states are modelled by a normal distribution. However, modelling uncertainties using a uni-modal normal distribution is insufficient to represent the different possible manoeuvres. This is especially used to cope with noisy sensor measurement, and projects these uncertainties on the predicted trajectory.

The third concept is a Monte Carlo simulation where the main idea is to randomly sample from the input variable of the evolution model in order to generate potential future trajectories. Also, for this case the simple evolution models can be used by sampling on the inputs instead of considering them to be constant. Physical limitations of a vehicle can be taken into account to remove infeasible manoeuvres.
5.2 Selection of a concrete method

In this work, a combination of the mentioned concepts for uncertainty handling is used for prediction. In principle, a single trajectory simulation is combined with the amount of dynamics defined by the previous movement of the obstacle to get a field of future object states that evolves and gets bigger with each prediction step. It can be more or less seen as a concept of reachable area that is not as conservative as approaches like [20, pp. 150] because previous dynamics in movement are taken into account to trim future states to a reduced set.

5.2.1 Modelling approach

The first step when creating a physics-based motion model is to find a model that suits all needs while being neither too simple nor too complex. In Section 2.3 a simple single track model was proposed, which fits these needs (as already discussed) quite well – therefore, it is also selected here for prediction. For recapitulation, the model equations of Equation 2.2 are quoted here once again.

\[
\begin{align*}
x(k+1) &= x(k) + v(k) \cdot \cos(\theta(k)) \cdot T_s \\
y(k+1) &= y(k) + v(k) \cdot \sin(\theta(k)) \cdot T_s \\
v(k+1) &= v(k) + a(k) \cdot T_s \\
\theta(k+1) &= \theta(k) + \frac{v(k)}{D_a} \cdot \tan(\delta(k)) \cdot T_s
\end{align*}
\]

(5.1)

In general, the states of such a system vary over time, which can easily be observed when looking at cars in the real world, while inputs of the system (in this case acceleration or deceleration \(a(k)\) and steering angle \(\delta(k)\)) remain mostly equal during static manoeuvres and only vary when a different manoeuvre is chosen or to readjust the movement during an ongoing manoeuvre.

Based on this hypothesis, the prediction algorithm relies on the inputs of the system and not on the states, as it would do in a direct case. Therefore, a method to calculate inputs based on available states is required, assuming that all states of the vehicle can be measured.

For \(k = k^* - M_s, k^* - M_s + 1, \ldots k^*\) measured states it occurs that from a pure mathematical point of view the estimation of the inputs can be done by simply reshaping and solving the equations for \(a(k)\) and \(\delta(k)\) with \(k = k^* - M_s, k^* - M_s + 1, \ldots k^* - 1\). As the inputs interlink the states, it is obvious that only \(M_i = M_s - 1\) inputs can be calculated from \(M_s\) states.

\[
a(k)^{diff} = \frac{v(k+1) - v(k)}{T_s} \\
\delta(k)^{diff} = \frac{1}{T_s} \cdot \arctan \left( \frac{(\theta(k+1) - \theta(k)) \cdot D_a}{v(k)} \right)
\]

(5.2)

This method holds in a perfect world without disturbances, but really small sensor noise leads to huge errors in these calculations because the term \(v(k+1) - v(k)\) as well as the term \(\theta(k+1) - \theta(k)\) are really sensible to disturbances, as visualised in Figure 5.1 and Figure 5.2.
Therefore, it appears reasonable to filter the state vectors in order to receive a smooth state approximation that performs better in terms of input estimation. This was solved with the function \textit{csaps} in MATLAB\textsuperscript{®}. The parameter \( p = 0.05 \) was selected to provide a sufficiently smooth result.

\[
f^v(t) = \arg \min_f \left( p \cdot \sum_{k = k^* - M_s}^{k^*} \left| v(k) - \hat{f}(k \cdot T_s) \right|^2 + (1 - p) \cdot \int_{(k^* - M_s) T_s}^{k^* T_s} \left| \hat{f}''(t) \right|^2 dt \right) \tag{5.3}
\]

\[
v^{\text{spline}}(k) = f^v(k \cdot T_s) \tag{5.4}
\]

\[
f^\theta(t) = \arg \min_f \left( p \cdot \sum_{k = k^* - M_s}^{k^*} \left| \theta(k) - \hat{f}(k \cdot T_s) \right|^2 + (1 - p) \cdot \int_{(k^* - M_s) T_s}^{k^* T_s} \left| \hat{f}''(t) \right|^2 dt \right) \tag{5.5}
\]

\[
\theta^{\text{spline}}(k) = f^\theta(k \cdot T_s) \tag{5.6}
\]

The optimised function \( \hat{f} \) consists of single cubic spline functions defined between each sample \( k \) with the condition \( \hat{f}''(k \cdot T_s) = 0 \) for \( k^* - M_s, k^* - M_s + 1, \ldots k^* \).

The principle of this smoothing is easily described. The first term containing the sum weights the deviation from the original (noisy) state values while the second term with the integral weights the curvature of the function. \( p \) acts as weighting factor that enables the user to select the trade-off between these two features.

To visualise this principle, an example is shown in figure \textbf{Figure 5.1} and \textbf{Figure 5.2}. \textbf{Figure 5.2} shows representative input trajectories (\( a^{\text{original}} \) and \( \delta^{\text{original}} \)). Using these trajectories, the states in \textbf{Figure 5.1} (\( v^{\text{original}} \) and \( \theta^{\text{original}} \)) can be calculated with the system equations from \textbf{Equation 5.1} where \( D_a \) was set to 2.5 m. Then, normally distributed noise \( \mathcal{N}(\mu, \sigma^2) \) is added to all samples:

\[
v^{\text{noise}} = v^{\text{original}} + \mathcal{N}(0, \sigma_v^2) \quad \text{with} \quad \sigma_v = 0.5 \frac{m}{s} \tag{5.7}
\]

\[
\theta^{\text{noise}} = \theta^{\text{original}} + \mathcal{N}(0, \sigma_\theta^2) \quad \text{with} \quad \sigma_\theta = \pi/20 \text{ rad} \tag{5.8}
\]

Now the inputs are calculated

1. by directly using the method from \textbf{Equation 5.2} resulting in \( a^{\text{diff}} \) and \( \delta^{\text{diff}} \).

2. by smoothing the states using the method from \textbf{Equation 5.3} \( \text{resulting in } v^{\text{spline}} \) and \( \theta^{\text{spline}} \) \( \text{and then the method from } \textbf{Equation 5.2} \)

\[
\cdot \text{resulting in } a^{\text{spline}} \text{ and } \delta^{\text{spline}}.
\]
5.2. Selection of a concrete method

It can easily be obtained that this kind of smoothing is urgent when working with real data. Therefore, it will be used for all further calculations.

Figure 5.1: Influence of noise on input-prediction – states

Figure 5.2: Influence of noise on input-prediction – inputs
Prediction for dynamic obstacles

The estimations of the system inputs are used in the next step to predict further movement of the object.

Five different methods were implemented and compared in terms of prediction performance using the measured data from real traffic participants. \( k^* \) represents the index of the currently measured state, and \( M_{pr} \) the number of states to be predicted in the future.

1. Median value of the training data as constant value for the prediction
   \[
   f^{\text{median}}_{pr,a}(k) = \text{median}([a(k^* - M_s), \ldots, a(k^* - 1)]) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]
   \[
   f^{\text{median}}_{pr,\delta}(k) = \text{median}([\delta(k^* - M_s), \ldots, \delta(k^* - 1)]) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]

2. Zero as constant value for the prediction
   \[
   f^{\text{median}}_{pr,a}(k) = 0 \quad \text{with } k = k^*, \ldots, M_{pr} - 1
   \]
   \[
   f^{\text{median}}_{pr,\delta}(k) = 0 \quad \text{with } k = k^*, \ldots, M_{pr} - 1
   \]

3. Mean value of the training data as constant value for the prediction
   \[
   f^{\text{mean}}_{pr,a}(k) = \text{mean}([a(k^* - M_s), \ldots, a(k^* - 1)]) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]
   \[
   f^{\text{mean}}_{pr,\delta}(k) = \text{mean}([\delta(k^* - M_s), \ldots, \delta(k^* - 1)]) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]

4. Last value of the training data as constant value for the prediction
   \[
   f^{\text{last}}_{pr,a}(k) = a(k^* - 1) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]
   \[
   f^{\text{last}}_{pr,\delta}(k) = \delta(k^* - 1) = \text{const. with } k = k^*, \ldots, M_{pr} - 1
   \]

5. Linear extrapolation of the training data as value for the prediction
   \[
   c^{\text{lin}} = \text{polyfit}([k^* - M_s, \ldots, k^* - 1], [a(k^* - M_s), \ldots, a(k^* - 1)], 1)
   \]
   \[
   f^{\text{lin}}_{pr,a}(k) = \text{polyval}(c^{\text{lin}}_{pr,a}, k) \quad \text{with } k = k^*, \ldots, M_{pr} - 1
   \]
   \[
   c^{\text{lin}}_{pr,\delta} = \text{polyfit}([k^* - M_s, \ldots, k^* - 1], [\delta(k^* - M_s), \ldots, \delta(k^* - 1)], 1)
   \]
   \[
   f^{\text{lin}}_{pr,\delta}(k) = \text{polyval}(c^{\text{lin}}_{pr,\delta}, k) \quad \text{with } k = k^*, \ldots, M_{pr} - 1
   \]

The MATLAB\textsuperscript{®} function \texttt{polyfit}
\[
c = \text{polyfit}(t_{tr}, x_{tr}, \zeta)
\]

yields the best fit (in a least-squares sense) of a \( \zeta \)-degree polygon from a training set with time-vector \( t_{tr} \) and a corresponding data vector \( x_{tr} \) returning the coefficients of the polygon \( c \). With the MATLAB\textsuperscript{®} function \texttt{polyval}
\[
x(t) = \text{polyval}(c, t)
\]

the value of the polygon at any time \( t \) is returned using the previously calculated coefficients \( c \). Using just these predicted input values, one would receive a single trajectory for the object. As previously mentioned, the dynamics of the previous vehicle movement shall also be taken into account for the prediction. A good measure for these dynamics is the
5.2. Selection of a concrete method

The standard deviation of the estimated inputs in the training set. If a lot of movement is present, the standard deviation is bigger. If the inputs are more constant, the standard deviation decreases.

\[
\sigma_{\text{data}}^{\text{tr},a} = \text{std}([a(k^* - M_s), \ldots, a(k^* - 1)])
\]

(5.11)

\[
\sigma_{\text{data}}^{\text{tr},\delta} = \text{std}([\delta(k^* - M_s), \ldots, \delta(k^* - 1)])
\]

(5.12)

The problem with this approach is that for objects in static manoeuvres constant inputs would be predicted to continue this manoeuvre over all prediction steps. Therefore, a lower bound for the measure is added.

- \(\sigma_{\text{tr},a}^{\text{min}} = 0.5 \text{ m/s}^2\) as the minimal uncertainty for the acceleration
- \(\sigma_{\text{tr},\delta}^{\text{min}} = \pi/200 \text{ rad}\) as the minimal uncertainty for the steering angle

Then, the values are combined

\[
\sigma_{\text{tr},a} = \min(\sigma_{\text{tr},a}^{\text{min}}, \sigma_{\text{data}}^{\text{tr},a})
\]

(5.13)

\[
\sigma_{\text{tr},\delta} = \min(\sigma_{\text{tr},\delta}^{\text{min}}, \sigma_{\text{data}}^{\text{tr},\delta})
\]

(5.14)

and minimal as well as maximal values for the predicted inputs can be defined (exemplary for the prediction using the median-method here)

\[
a_{\text{pr}}^{\text{max}}(k) = f_{\text{pr},a}^{\text{median}}(k \cdot T_s) + \sigma_{\text{tr},a}
\]

(5.15)

\[
a_{\text{pr}}^{\text{min}}(k) = f_{\text{pr},a}^{\text{median}}(k \cdot T_s) - \sigma_{\text{tr},a}
\]

(5.16)

\[
\delta_{\text{pr}}^{\text{max}}(k) = f_{\text{pr},\delta}^{\text{median}}(k \cdot T_s) + \sigma_{\text{tr},\delta}
\]

(5.17)

\[
\delta_{\text{pr}}^{\text{min}}(k) = f_{\text{pr},\delta}^{\text{median}}(k \cdot T_s) - \sigma_{\text{tr},\delta}
\]

(5.18)

which can now be used in combination with the system in Equation 5.1 to predict the maximum and minimum future states according to the model.

An example of such a prediction using again the median-method is shown in Figure 5.3.

One can now interpolate between the minimum and maximum inputs with arbitrary fineness to get a hull of an allowed region for the vehicle states \(x_{\text{pr}}(k,j)\) and \(y_{\text{pr}}(k,j)\) defining points \(p_{\text{pr}}(k,j) = [x_{\text{pr}}(k,j), y_{\text{pr}}(k,j)]\) with \(k = k^* + 1, \ldots, k^* + M_{\text{pr}}\) and \(j = 1, \ldots, M_j\), represented as red dots with a red connecting line as its hull – exemplarily shown in Figure 5.4. In Figure 5.5 and Figure 5.6 this state-hull is represented too – this time based on real measurement data.

Now it is possible to overlay the rectangle representing the obstacle above each point of the hull, taking into account the calculated orientation \(\theta_{\text{pr}}(k,j)\) of the obstacle in that particular point. The resulting outer hull of these overlay obstacles (illustrated in blue) represents the possible physical region where the obstacle can be according to the proposed model, and consists of \(i = 1, \ldots, M_i\) points. A further mathematical description is not listed here because the concept is quite intuitive, as shown in Figure 5.4.

The resulting points \(p_{\text{O}}(k,i)\) defining the outer hull are those which are also used for the methods in Figure 4.5, Figure 4.7, and Figure 4.9.
Figure 5.3: States of an example prediction based on real measurement data

Figure 5.4: Graphical representation of hulls, and maximum as well as minimum states
The peaks in the end of the prediction phase appear to be a measurement error. Generally, not too much work was put into the pre-processing of the data to keep the results as close to real situations as possible. With very well pre-processed data the model would most likely work even better than shown here.

### 5.2.2 Resulting prediction

Obviously, this approach has to be evaluated by using data from real traffic participants. Therefore, measurements were performed using a test car with a state of the art LIDAR system, as described in Chapter 6. These data are now used for validation.

Different types of traffic participants were recorded. Afterwards, the data were analysed with a video track recorded in parallel, and only trajectories of cars, bicycles, and pedestrians were selected for further processing.

The data were split into training- (with $N^{tr}$ samples) as well as evaluation-sets (with $N^{ev}$ samples). Therefore, recordings with a length lower than $N^{tr} + N^{ev}$ were discarded.

To enlarge the amount of available samples, multiple predictions were performed for each object detected by simply shifting the training- as well as the evaluation region through the dataset.

Figure 5.5 and Figure 5.6 show a prediction for selected steps with $N^{tr} = 20$ and $N^{ev} = 20$ (in this case a prediction for a car).

![Figure 5.5: Top view of an example prediction based on measurement data](image)
5.2.3 Performance criterion for the prediction

Now a performance criterion for the quality of the prediction is needed. A very intuitive way is to define this prediction error as the maximum distance of the actual measured position ($p_{\text{real}}(k)$) to the predicted region of the position states. The vector $p_0(k)$ defines the points that describe the hull of the possible predicted states in prediction step $k$.

The prediction error calculates to

$$
\varepsilon_{pr}(k) = \begin{cases} 
0 & \text{for } p_{\text{real}}(k) \text{ inside the predicted region} \\
\min \left( \|p_{\text{real}}(k) - p_0(k)\| \right) & \text{otherwise}
\end{cases}
$$

(5.19)

This way the obtained predictions can easily be evaluated and compared.
5.3 Validation of the model

The various methods proposed in Subsection 5.2.1 are compared in the following figures. The maximum available amount of objects and predictions for each type of obstacle were selected – the quantity considered in each case is displayed in the legend of the figures. For each prediction step the median value of all predicted data as well as the 25% quantile ($Q_{0.25}$) and the 75% quantile ($Q_{0.75}$) are displayed. In this case the median tends to be a more reliable performance measure than the mean because some measurement errors in the data might affect the results.

The prediction is obtained with and without a minimum prediction value, as introduced in Equation 5.13. For a better comparability the plots are grouped by object type. The first plot in each subgroup shows the prediction without minimum prediction deviation limits, the second plot in each subgroup shows the prediction with minimum prediction deviation limits.

For the following figures, a prediction for $N_{tr} = 20$ and $N_{ev} = 20$ is shown, more combinations (and also predictions with $N_{ev} = 50$) are delineated in Appendix A.

As covered in Chapter 6, it was not possible to determine the distance between the obstacle axles. Therefore, standard values were used for the predictions. With a correct measurement it can be assumed that the prediction performance would be even better.

### Table 5.1: Common axle spacing values in mm

<table>
<thead>
<tr>
<th>Model</th>
<th>Axle spacing</th>
<th>Model</th>
<th>Axle spacing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars</td>
<td></td>
<td>Bicycles</td>
<td></td>
</tr>
<tr>
<td>Smart Fortwo</td>
<td>1867</td>
<td>Merida eBIG.NINE 500</td>
<td>1160</td>
</tr>
<tr>
<td>Twike</td>
<td>1870</td>
<td>KTM Race</td>
<td>1110</td>
</tr>
<tr>
<td>Fiat Cinquecento</td>
<td>2200</td>
<td>CUBE Cross</td>
<td>1120</td>
</tr>
<tr>
<td>Porsche 911</td>
<td>2211</td>
<td>Hrinkow Team XC</td>
<td>1070</td>
</tr>
<tr>
<td>VW Käfer</td>
<td>2400</td>
<td>Cannondale</td>
<td>1020</td>
</tr>
<tr>
<td>Nissan 370Z Roadster</td>
<td>2550</td>
<td>Puch</td>
<td>1010</td>
</tr>
<tr>
<td>VW Golf VI</td>
<td>2578</td>
<td>Venice</td>
<td>1050</td>
</tr>
<tr>
<td>Ferrari Enzo Ferrari</td>
<td>2650</td>
<td>Canyon Nerve AL</td>
<td>1168</td>
</tr>
<tr>
<td>VW Phaeton</td>
<td>2881</td>
<td>Nakita town &amp; country</td>
<td>1090</td>
</tr>
<tr>
<td>Mercedes-Benz W 126</td>
<td>2850</td>
<td>KTM life adventure</td>
<td>1100</td>
</tr>
<tr>
<td>Mercedes-Benz W 221</td>
<td>3035</td>
<td>SCOTT E-Sub Active</td>
<td>1140</td>
</tr>
<tr>
<td>Hummer H1</td>
<td>3302</td>
<td>Kalkhoff Allround</td>
<td>1100</td>
</tr>
<tr>
<td>Median</td>
<td>2564</td>
<td>Median</td>
<td>1100</td>
</tr>
<tr>
<td>Mean</td>
<td>2533</td>
<td>Mean</td>
<td>1095</td>
</tr>
</tbody>
</table>

The values for $D_a$ are selected by taking into account median and mean of common values listed in Table 5.1. As pedestrians have no actual axle spacing, the value was selected significantly smaller compared to cars and bicycles to model their capability of fast direction changes.

- $D_{a,\text{car}} = 2.5\, \text{m}$
- $D_{a,\text{bicycle}} = 1.1\, \text{m}$
- $D_{a,\text{pedestrian}} = 0.5\, \text{m}$
5.3.1 Validation for cars

![Figure 5.8](image)

Figure 5.8: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars without minimum prediction deviation limits.

![Figure 5.9](image)

Figure 5.9: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits.

5.3.2 Validation for bicycles

![Figure 5.10](image)

Figure 5.10: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles without minimum prediction deviation limits.
5.3. Validation of the model

![Graph showing prediction error for bicycles](image1)

Figure 5.11: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits

5.3.3 Validation for pedestrians

![Graph showing prediction error for pedestrians](image2)

Figure 5.12: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians without minimum prediction deviation limits

![Graph showing prediction error for pedestrians](image3)

Figure 5.13: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits
5.3.4 Interpretation of the validation

When looking at the prediction performance, it becomes clear that some of the methods outperform others by far. Linear extrapolation tends to deviate quite far from the real position, therefore, it can be discarded. The method of taking the last estimated input value from the training set tends to deliver really good results for cars, but is still worse than the other methods when it comes to pedestrian or bicycle prediction. Using the median or mean of the training data provides quite good result for cars, while for bicycles and pedestrians the method of keeping the input at zero (constant velocity and orientation) tends to be slightly superior. This method, however, is performing by far not as good as median or mean when it comes to car predictions. Due to its robustness against outliers that differentiates it from the mean method, the median method was selected for all further predictions, as these two methods perform well for all types of obstacles. All in all, the median method provides quite predictions with a maximum deviation of $0.5\text{ m}$ for all types of obstacles, where the results for bicycles and pedestrians are even better. This holds for short time predictions of 20 steps. For long term predictions (50 steps ahead) the zero-method outperforms the median-method, as displayed in Appendix A. Therefore, it is used for these predictions. It has to be said that while the performance for short time prediction is quite promising, the proposed methods get worse for longer prediction times, as shown in Appendix A. Therefore, a manoeuvre based approach for long-time-prediction might perform better for further development of the assistant.

For further investigations it tends to be promising to tune different prediction styles to different obstacle types. To provide a uniform system that performs sufficiently well in a general use-case, just a single method for all types was selected here for each case (long- and short-time-prediction).
Chapter 6

Recording of real measurement data

In the course of this work, real world measurement data were acquired to validate the prediction model as well as to get sufficient information concerning the behaviour of the surrounding traffic participants to be able to generate realistic scenarios.

For this measurement a test-vehicle (BMW F31) was used. The vehicle is equipped with different kinds of sensor systems, namely

- RADAR (Radio Detection and Ranging)
- LIDAR (Light Detection and Ranging)
- Cameras

Figure 6.1: Picture of the used test-vehicle (BMW F31)
6.1 Sensor overview

According to [8, p. 95], the three types of sensor systems have different strengths and weaknesses, as quickly outlined in the following table.

<table>
<thead>
<tr>
<th>Performance aspect</th>
<th>Human</th>
<th>RADAR</th>
<th>LIDAR</th>
<th>Camera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object detection</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Object classification</td>
<td>Good</td>
<td>Poor</td>
<td>Fair</td>
<td>Good</td>
</tr>
<tr>
<td>Distance estimation</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Edge detection</td>
<td>Good</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Lane tracking</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
<td>Good</td>
</tr>
<tr>
<td>Visibility range</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Poor weather performance</td>
<td>Fair</td>
<td>Good</td>
<td>Fair</td>
<td>Poor</td>
</tr>
<tr>
<td>Dark or low illumination performance</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
</tbody>
</table>

As there was no sensor fusion implemented, only the LIDAR (ibeo LUX 4L) data were used, since no object detection was implemented for the camera stream. The RADAR system lacks performance when it comes to angular resolution as well as accuracy in measurement, therefore, it was not utilised.

6.1.1 LIDAR

Not only the theoretical features of that technology seem promising, but also the obtained measurement data were of high quality. The system provided data in form of detected objects via CAN-Bus (Controller Area Network), containing the attributes of these objects, like dimension, position, orientation, as well as velocity.

According to [21], the LIDAR (ibeo LUX 4L) has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Technical data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>50 m @ 10 % remission</td>
</tr>
<tr>
<td>Horizontal field of view</td>
<td>110° (50° to − 60°)</td>
</tr>
<tr>
<td>Vertical field of view</td>
<td>3.2°</td>
</tr>
<tr>
<td>Multi-layer</td>
<td>4 parallel scanning layers</td>
</tr>
<tr>
<td>Accuracy (distance independent)</td>
<td>10 cm</td>
</tr>
<tr>
<td>Angular resolution (horizontal)</td>
<td>up to 0.25°</td>
</tr>
<tr>
<td>Angular resolution (vertical)</td>
<td>0.8°</td>
</tr>
<tr>
<td>Distance Resolution</td>
<td>4 cm</td>
</tr>
</tbody>
</table>

The pre-processing of the data was simply the elimination of faulty datasets. As the object classification ability of LIDAR systems is fair but not good, as stated in Table 6.1, a manual
identification of the respective object type had to be carried out by comparing the measured data with the synchronised stream of a camera. With the help of a mentioned sensor fusion concept this task could easily be automated.

6.2 Acquired Data

Besides many other data, mainly for classification and detection probability for the obstacles, six primary attributes were detected:

- $W_O$ in m ... box-width
- $L_O$ in m ... box-length
- $x_O(t)$ in m ... horizontal position
- $y_O(t)$ in m ... vertical position
- $v_O(t)$ in m/s ... velocity
- $\theta_O(t)$ in rad ... orientation

As visualised, the position of the obstacles (here the centre of the rectangle is described) does not match the position in the model which is set to the centre of the rear axle, as displayed in Figure 2.1. The measurement of $W_O$ and $L_O$ of the LIDAR system was not consistent over time – therefore, a calculation of the model centre from the measured data was impossible. This is the reason why the position $x_O$ and $y_O$ were directly used for the model too – see validation of the prediction in Section 5.3.

For a real implementation of the system, a reliable measurement of the obstacle’s outer dimensions ($L_O$ and $W_O$) as well as its axial spacing ($D_a$) is required.
6.2.1 Measurement

The measurement took place with a static test-vehicle at the junction “Freistädterstraße - Hauptstraße” (48°18’56.2″N 14°16’49.7″E) on 19 June 2019. The test-vehicle was parked as shown in Figure 6.3 and the traffic was recorded.

![Figure 6.3: Overview of the junction used for measurement, map provided by 22](image)

The circle segment represents the estimated area covered by the LIDAR. Comparing the resulting data with the video sequence during the measurement as well as with the geographical data, the area matches well. The measurement-vehicle is represented in an upscaled size in the figure. The coordinate system of the LIDAR matches with the one from Figure 6.2. The junction where the measurement was taken is uncontrolled, which means that no traffic lights were installed to control the traffic.

Two full crosswalks are placed in the measurement region, therefore, data of pedestrians were collected, too. The one directly in front of the measurement vehicle was only partially covered, still, some data of pedestrians crossing there could be recorded.
6.2. Acquired Data

6.2.2 Data statistics

A total of 195 objects were measured when observing the junction, as listed below. While the majority of the measurements were cars, also a relatively huge amount of pedestrians could be observed. The amount of bicycles was rather small, but still, some data could be extracted.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Car</th>
<th>Bicycle</th>
<th>Pedestrian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount of valid measurements</td>
<td>151</td>
<td>9</td>
<td>35</td>
</tr>
<tr>
<td>Length of the dataset in s</td>
<td>$Q_{0.25}$</td>
<td>4.10</td>
<td>4.65</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.5}$</td>
<td>4.90</td>
<td>5.90</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.75}$</td>
<td>6.68</td>
<td>7.40</td>
</tr>
<tr>
<td>Velocity in $\frac{m}{s}$</td>
<td>$Q_{0.25}$</td>
<td>-0.53</td>
<td>-0.29</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.5}$</td>
<td>0.11</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.75}$</td>
<td>0.92</td>
<td>0.50</td>
</tr>
<tr>
<td>Estimated acceleration in $\frac{m}{s^2}$</td>
<td>$Q_{0.25}$</td>
<td>-0.05</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.5}$</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$Q_{0.75}$</td>
<td>0.09</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Figure 6.4: Spatial frequency of the different objects

Figure 6.4 shows the spatial frequency of the recorded objects. Brighter regions (yellow) are more frequently crossed while darker regions (blue) are not. One can clearly observe the geographical structure of the junction in the figure, and even the lack of valid data for bicycles in plot (B) becomes visible. In general, all plots from Figure 6.4 to Figure 6.11 are labelled with capital letters, where (A) represents the measurement for cars, (B) represents the measurement for bicycles, and (C) represents the measurement for pedestrians.

In the following plots of this section, histograms are shown. The vertical bars represent the frequency in certain regions of different features, specified in the corresponding plot. Above
the individual plots, a box-plot is realised that shows the median \((Q_{0.5})\) of the data as well as the quantiles \(Q_{0.25}\) and \(Q_{0.75}\). Values outside the plotted whiskers are seen as outliers and not plotted. Those outliers are values which are greater than \(Q_{0.75} + 1.5 \cdot (Q_{0.75} - Q_{0.25})\) or lower than \(Q_{0.25} - 1.5 \cdot (Q_{0.75} - Q_{0.25})\).

The value for the whiskers corresponds to approximately \(\pm 2.7 \cdot \sigma\) and 99.3% coverage if the data are normally distributed. The plotted whisker extends to the adjacent value, which is the most extreme data value that is not an outlier.

**Figure 6.5: Length of the measurements**

**Figure 6.5** shows the length of the valid measurements. As one can obtain, some of the measurements are shorter than the 4 s demanded in [Section 5.3](#), which were dismissed for the validation. It can be said that hardly any measurement at the junction exceeds 15 s with a median value of around 5 s, which is quite realistic for a junction without traffic lights.

**Figure 6.6: Velocity distribution**
In the distribution of velocity it is clearly visible that cars in general often have a phase in the beginning where the velocity is \( 0 \text{ m/s} \), which means that they stop and wait at the junction while the other object types tend to move more fluently. The lack of data is observable in Figure 6.7 (B) while a really consistent behaviour can be observed for pedestrians (C).
When looking at the distribution of the orientation of the obstacles, the directions of movement can clearly be determined. The orientation is represented as $\theta_O$ in Figure 6.2. That means that a value of 0 rad describes a movement away from the ego-vehicle while $\pm \pi$ rad describes the opposite. Values around $\pi/2$ rad represent a movement horizontal to the left in front of the ego-vehicle while values around $-\pi/2$ rad stand for a movement in the opposite direction. This can especially be observed for pedestrians (C) due to the horizontal cross-walk in front of the ego-vehicle.

Figure 6.10: Estimated acceleration values

Figure 6.11: Estimated steering angle calculated with the $D_o$-values specified in Section 5.3

Figure 6.10 and Figure 6.11 as well as Figure 6.12 show the estimated inputs of the observed obstacles. It gets clear that even in junction situations the input of the obstacle is often not changed (straight movement without variation of velocity), the variance around zero for bicycles and pedestrians tends to be smaller than for cars. This is most likely the reason why the zero-method in Section 5.3 performed that well for these two object types.
This behaviour is probably caused by the fact that most pedestrians crossed straight crosswalks with a constant speed and also many bicycles crossed the junction straight – see Figure 6.4. Also the fact that cars usually realise higher acceleration rates than the other two obstacle types comes into play here. More measurements from other junctions and situations are needed here to be able to make more qualified statements, especially for bicycles because only 9 objects were recorded.

6.2.3 Usage for scenario generation

The data from Table 6.3 can now be used to specify states for objects in junction scenarios (in the underlying scenario the velocity of the pedestrian was modelled this way). For scenarios with straight roads, typical speed limits in Austria were taken into account, namely the inner-city limitation of $50 \text{ km/h}$ which corresponds to a speed of $13.9 \text{ m/s}$. For the junction, a speed limit of $30 \text{ km/h}$ which corresponds to $8.3 \text{ m/s}$ was selected.

The approach of generating the scenario is described in Chapter 7. Two scenarios selected for the validation of the proposed collision avoidance assistant – one for a straight road and one for a junction situation – are listed in Chapter 7 too.

The parameters in the scenarios are varied to test various conditions while the basic make-up of the scenario remains the same.
Chapter 7

Selection of the Scenario

In the following chapter, two different scenarios are introduced, one for a straight road and one for a junction situation. It is clear that not all safety-critical situations can be covered here, as in real driving a huge amount of situations arise which cannot fully be determined a-priori. The scenarios are chosen in such a way that most of the features of the assistant can be illustrated.

7.1 Definition of the Scenario

At first, the road boundaries and all other known geometrical attributes are defined, the corresponding measures are shown in [Figure 7.1] and [Figure 7.2]. The width of the street ($W_S$ as specified in [Figure 3.1]) is set to 7 m, which results in a lane width of 3.5 m.

As no full coverage of all safety critical situations can be covered with such simple scenarios, some situations were picked where the typical behaviour of the collision avoidance assistant is shown.

Therefore, three different cases are selected for a straight road, and two cases for a junction. The scenarios respectively stayed the same in these cases, only small parameters according to the position of obstacles are varied.

Parameters for the straight road:

- $x_V(k = 0)$ in m ... position of the dynamic obstacle (vehicle) in X-direction
- $x_S(k = 0)$ in m ... position of the static obstacle in X-direction

Parameters for the junction:

- $x_V(k = 0)$ in m ... position of the dynamic obstacle (vehicle) in X-direction
- $x_P(k = 0)$ in m ... position of the dynamic obstacle (pedestrian) in X-direction

These parameters were enough to provide different situations without the need of defining many different scenarios.
7.1.1 Definition of ego and dynamic obstacles

Geometry and initial states were selected to simple measures that still represent actual geometries in the ‘real’ world quite well. The pedestrian (P) was also modelled as a rectangle with equivalent attributes as vehicles – as the same prediction model is applied on this kind of dynamic obstacle.

The initial velocity was selected to the legal limits in Austrian urban traffic (50 $\frac{km}{h} \approx 13.9 \frac{m}{s}$ and 30 $\frac{km}{h} \approx 8.3 \frac{m}{s}$). In case of the junction, this is approximately twice as fast as the median of the measured vehicles, which corresponds to an additional challenge for the ego-vehicle (E).

Table 7.1: Geometry and initial states for selected scenarios

<table>
<thead>
<tr>
<th></th>
<th>Straight road</th>
<th></th>
<th>Junction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E</td>
<td>V</td>
<td>E</td>
</tr>
<tr>
<td>Geometry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$W$</td>
<td>2.0 m</td>
<td>2.0 m</td>
<td>2.0 m</td>
</tr>
<tr>
<td>$L$</td>
<td>5.0 m</td>
<td>5.0 m</td>
<td>5.0 m</td>
</tr>
<tr>
<td>$D_a$</td>
<td>3.0 m</td>
<td>3.0 m</td>
<td>3.0 m</td>
</tr>
<tr>
<td>$D_f$</td>
<td>4.0 m</td>
<td>4.0 m</td>
<td>4.0 m</td>
</tr>
<tr>
<td>Initial states</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x(k = 0)$</td>
<td>5.0 m</td>
<td>$x_V$</td>
<td>5.0 m</td>
</tr>
<tr>
<td>$y(k = 0)$</td>
<td>1.7 m</td>
<td>5.3 m</td>
<td>1.7 m</td>
</tr>
<tr>
<td>$v(k = 0)$</td>
<td>13.9 $\frac{m}{s}$</td>
<td>13.9 $\frac{m}{s}$</td>
<td>8.3 $\frac{m}{s}$</td>
</tr>
<tr>
<td>$\theta(k = 0)$</td>
<td>0 rad</td>
<td>$\pi$ rad</td>
<td>0 rad</td>
</tr>
</tbody>
</table>

7.2 Straight road

Figure 7.1: Overview of the scenario with a straight road, all units in m
7.2. Straight road

Figure 7.1 shows the scenario for the straight road: the initial position of the vehicle (V) and the position of the static obstacle (S) are varied to get different situations the collision avoidance assistant has to cope with. A short introduction of the scenario background is shown in the following sections, as well as a description of the expected outcome. These scenarios are evaluated in Chapter 8.

7.2.1 Case S1 – Simple collision avoidance

In this case the vehicle (V) is far away when the ego-vehicle (E) approaches the static obstacle (S). The second lane is not blocked.

Parameters:

- $x_V(k = 0) = 195 \text{ m}$
- $x_S(k = 0) = 45 \text{ m}$

Expected outcome

The ego should avoid the collision with the static obstacle by simply changing the lane, then it should return to its primary lane and drive past the upcoming vehicle.

7.2.2 Case S2 – Collision avoidance by braking

In this case the vehicle (V) is blocking the other lane when the ego-vehicle (E) approaches the static obstacle (S).

Parameters:

- $x_V(k = 0) = 70 \text{ m}$
- $x_S(k = 0) = 45 \text{ m}$

Expected outcome

The ego should avoid the collision with the static obstacle by simply braking, as a lane change is not possible. It should then either come to a standstill or accelerate again to drive past the static obstacle.

7.2.3 Case S3 – Crash-mode

In this case the vehicle (V) is not yet blocking the other lane in the beginning when the ego-vehicle (E) approaches the static obstacle (S). The distance to the static obstacle (S) is not sufficient for a braking manoeuvre, as it is in Case S2.
Selection of the Scenario

Parameters:
- \( x_V(k = 0) = 65\) m
- \( x_S(k = 0) = 30\) m

Expected outcome

The ego should avoid the collision with the static obstacle by a simple lane change and try to get past the static obstacle before the vehicle (V) blocks the second lane. If a collision cannot be avoided, the ego should switch into crashmode and avoid the collision by exiting the road boundaries.

7.3 Junction

Figure 7.2: Overview of the junction scenario, all units in m

Figure 7.2 shows the scenario for the junction. The initial position of the pedestrian (P) is varied to get different situations the collision avoidance assistant has to cope with. A short introduction of the scenario background is shown in the following sections, as well as a description of the expected outcome. These scenarios are evaluated in Chapter 8.
7.3.1 Case J1 – Collision avoidance in front of pedestrian

In this case the vehicle (V) is approaching the ego-vehicle (E) in a junction situation where the ego’s primary trajectory represents a turn to the left. It is assumed that the ego-vehicle is on a priority road. The vehicle (V) ignores the give way sign and approaches at a constant speed. After the junction a pedestrian (P) crosses the street in an unexpected region.

Parameters:
- \( x_p(k = 0) = 12 \text{ m} \)

**Expected outcome**

To avoid the first collision, the ego should turn left at the junction before the vehicle (V) arrives. Then, it should avoid the collision with the upcoming pedestrian (P) by accelerating and crossing the path of the pedestrian in front of him.

7.3.2 Case J2 – Collision avoidance behind pedestrian

In this case the vehicle (V) is again approaching the ego-vehicle (E) in a junction situation where the ego’s primary trajectory represents a turn to the left. It is assumed that the ego-vehicle is on a priority road. The vehicle (V) ignores the give way sign and approaches at a constant speed. After the junction, a pedestrian (P) crosses the street once again in an unexpected region. The difference to the previous scenario is that the pedestrian is on the lane of the ego when the ego reaches the pedestrian’s position.

Parameters:
- \( x_p(k = 0) = 16 \text{ m} \)

**Expected outcome**

To avoid the first collision, the ego should turn left at the junction before the vehicle (V) arrives. Then, it should avoid the collision with the upcoming pedestrian (P) by switching the lane and crossing the path of the pedestrian behind him.
Chapter 8

Trajectory tracking and scenario evaluation

In this chapter, the proposed collision avoidance assistant is evaluated by tracking the calculated trajectories from Chapter 7 with a complex vehicle model of the ego using IPG CarMaker. The tracking is performed offline on the references previously generated with the simple single-track-model. It has to be evaluated if the collision avoidance assistant performs sufficiently in the scenario situations and also if a good tracking performance is possible with the underlying simple model as a reference.

8.1 Trajectory tracking

The tracking of the calculated path is realised with a feed forward control using the calculated inputs for the model in combination with a feedback control of the desired position, velocity, and orientation at a specific step \((k)\). This approach seems promising, as the vehicle is modelled non-linear in a way that should fit the behaviour of a ‘real’ vehicle quite well. The general structure is shown in the following Figure 8.1.

![Figure 8.1: Structure of the controller used for tracking](image)

---

The figure above shows the control structure used for trajectory tracking. The inputs include the reference yaw angle \(\theta_{ref}(k)\), reference position in the longitudinal direction \(x_{ref}(k)\), and reference velocity in the longitudinal direction \(v_{ref}(k)\). The outputs are the control actions for steering \(u_{steer}(k)\), acceleration \(u_{accel}(k)\), and braking \(u_{brake}(k)\). The feedback control terms include the errors in yaw angle \(e_\theta(k)\), position \(e_x(k)\), and velocity \(e_v(k)\), with the corresponding PI controllers \(PI_\theta\), \(PI_{lat}\), \(PI_{long}\), and \(PI_v\). The system dynamics are represented by the function \(G\).
The PI-blocks represent the PI-controller for the respective variable and G represents the plant, in this case the complex car-model in IPG CarMaker. \( f_{dev} \) and \( f_a \) are functions which are defined in [Subsection 8.1.2]

### 8.1.1 Signals in the control structure

The following signals appear in the structure:

- **References and feed forward values**
  - \( x_{ref}(k) \) ... state reference for position in X-direction
  - \( y_{ref}(k) \) ... state reference for position in Y-direction
  - \( v_{ref}(k) \) ... state reference for velocity
  - \( \theta_{ref}(k) \) ... state reference for orientation
  - \( \delta_{ref}(k) \) ... feed forward value for steering angle
  - \( a_{ref}(k) \) ... feed forward value for acceleration/deceleration

- **(Control) deviations**
  - \( e_x(k) \) ... deviation in X-direction
  - \( e_y(k) \) ... deviation in Y-direction
  - \( e_{lat}(k) \) ... control deviation in lateral direction
  - \( e_{long}(k) \) ... control deviation in longitudinal direction
  - \( e_v(k) \) ... control deviation of velocity
  - \( e_{\theta}(k) \) ... control deviation of orientation

- **Actuating variables**
  - \( u_{steer}(k) \) ... steering angle
  - \( u_a(k) \) ... acceleration/deceleration
  - \( u_{gas}(k) \) ... input for gas pedal position
  - \( u_{brake}(k) \) ... input for brake pedal position

- **Control variables**
  - \( x(k) \) ... position in X-direction
  - \( y(k) \) ... position in Y-direction
  - \( v(k) \) ... velocity
  - \( \theta(k) \) ... orientation

### 8.1.2 Functional blocks of the control structure

The block \( f_a \) realises the following operation:

\[
 u_{gas}(k) = \begin{cases} 
 u_a(k) & \text{for } u_a(k) > 0 \\
 0 & \text{otherwise} 
\end{cases} \tag{8.1}
\]

\[
 u_{brake}(k) = \begin{cases} 
 -u_a(k) & \text{for } u_a(k) < 0 \\
 0 & \text{otherwise} 
\end{cases} \tag{8.2}
\]
8.1. Trajectory tracking

The block $f_{dev}$ realises the following operation:

\[
\begin{align*}
    dx(k) &= x_{ref}(k) - x(k) \\
    dy(k) &= y_{ref}(k) - y(k) \\
    e_{long}(k) &= \cos(\theta(k)) \cdot dx(k) + \sin(\theta(k)) \cdot dy(k) \\
    e_{lat}(k) &= \cos(\theta(k) + \pi/2) \cdot dx(k) + \sin(\theta(k) + \pi/2) \cdot dy(k)
\end{align*}
\] (8.3) (8.4) (8.5) (8.6)

The feedback-control of $\theta(k)$ and $v(k)$ only realises a tracking of the orientation and the velocity which acts more or less as a feed forward-control of the position. To get the deviation from the position inside a feedback loop, the method in $f_{dev}$ was implemented. $e_{lat}(k)$ is well related to the steering angle while $e_{long}(k)$ is well related to the acceleration.

![Figure 8.2: Relationship of inputs and outputs of the block $f_{dev}$](image)

8.1.3 Parameters of the PI-controllers

The parameters of the PI-controllers were selected to fit all tested scenarios listed in Chapter 7. With a parameter switching corresponding to specific situations, the quality of the tracking could still be increased.

<table>
<thead>
<tr>
<th>Controller</th>
<th>P</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI$_v$</td>
<td>10.00</td>
<td>2.00</td>
</tr>
<tr>
<td>PI$_\theta$</td>
<td>20.00</td>
<td>0.01</td>
</tr>
<tr>
<td>PI$_{lat}$</td>
<td>5.00</td>
<td>0.02</td>
</tr>
<tr>
<td>PI$_{long}$</td>
<td>5.00</td>
<td>0.02</td>
</tr>
</tbody>
</table>
8.2 Performance evaluation

8.2.1 Case S1 – Simple collision avoidance

Figure 8.3: Costs, (tracked) states, and inputs for the Case S1
8.2. Performance evaluation

8.2.1.1 Evaluation of the result

The collision avoidance assistant performs quite well in this scenario. As expected, the assistant avoids the collision with the static obstacle by changing the lane through steering and then switches back to its primary lane where it continues normal tracking without further braking and steering.

The moments when the ego drives past the obstacles can be observed clearly in Figure 8.3(B) looking at $J_{\text{par,pot}}$. The physical boundaries of the object are never crossed ($J_{\text{kamm}}$ and $J_{\text{model}}$ are 0 over all time) and also the road-boundary-condition ($J_{\text{road}}$) is never violated.

The costs are displayed for a ‘present’ moment (expressed by a point in the graph) and the upcoming 0.4 s which are predicted and still contain uncertainties according to the positions of surrounding obstacles. This method of representation was selected because it provides a better insight of the optimisation process at each (optimisation) step.

It can be said that the collision avoidance assistant handles the critical situation as expected.

The tracking works quite well in this situation. Just a very low deviation to the reference-states can be observed, and, as one can see in Figure 8.4 and Figure 8.5, the trajectory is nearly identical.

Since the tracking was only performed offline, an online-approach with the complex CarMaker-model in the loop should even increase the performance.
8.2.2 Case S2 – Collision avoidance by braking

Figure 8.6: Costs, (tracked) states, and inputs for the Case S2
8.2. Performance evaluation

8.2.2.1 Evaluation of the result

Again, the collision avoidance assistant performs quite well in this scenario. As expected, the assistant avoids the collision with the static obstacle by braking, as there is an approaching car on the other lane.

The moment when the approaching vehicle drives past the ego can be observed clearly in Figure 8.6(B) looking at $J_{\text{par, pot}}$. The physical boundaries of the object are never crossed ($J_{\text{karm}}$ and $J_{\text{model}}$ are 0 over all time) and also the road-boundary-condition ($J_{\text{road}}$) is never violated.

It can be said that the collision avoidance assistant handles the critical situation as expected. As can be seen in Figure 8.7 and Figure 8.8 the tracked trajectory is again nearly identical with the reference.
8.2.3 Case S3 – Crashmode

Figure 8.9: Costs, (tracked) states, and inputs for the Case S3
8.2. Performance evaluation

8.2.3.1 Evaluation of the result

This is the first scenario where the collision avoidance assistant struggles in terms of performance, even if the outcome is a collision avoidance. First, the system tries to avoid a crash by changing the lane, as braking alone would lead to a collision with the static obstacle which is positioned closer to the ego’s initial position here.

The front of the ego is approximately at 9 m in the beginning, which means an available way for braking of $s_{\text{avail}} = 30 \text{ m} - 9 \text{ m} = 21 \text{ m}$ while the required braking distance calculates to:

$$s_{\text{req}} = \frac{(v(k = 0))^2}{2 \cdot \left| a_{\text{max braking}} \right|} = \frac{(50 \text{ m/s})^2}{2 \cdot 4 \text{ m/s}^2} = 24.11 \text{ m}$$ (8.7)
which means 3.11 m more would be required to avoid this collision only through braking.
Therefore, the steering manoeuvre seems reasonable.

The critical point in this case is that again, an oncoming vehicle drives towards the ego. Since the TTC falls under a critical level here, as no collision avoidance through steering and braking is possible anymore, the crashmode gets activated after 1.5 s, which can be observed in Figure 8.9(A).

Figure 8.9(B) shows that the cost for $J_{\text{model}}$ explodes when the crashmode gets activated, also $J_{\text{kamm}}$ gets active. This means that the optimised inputs here lead to a model mismatch and also that road adhesion cannot be guaranteed anymore. This is mainly caused by the fact that costs for obstacles get weighted according to their type and rise by a huge factor while $J_{\text{model}}$ and $J_{\text{kamm}}$ are not scaled accordingly. For further development this fact should be taken into account. The proposed rise in obstacle costs cannot be observed in Figure 8.9(B) because these costs rocket in further prediction steps which are not displayed here, as already mentioned in Subsection 8.2.1.

The rise of $J_{\text{road}}$ is just displayed for illustration purposes because this component is no longer part of the optimisation if the crashmode is active (crossing of road boundaries is allowed). The moment when the approaching vehicle gets to its closest point to the ego can be observed clearly in Figure 8.6(B) looking at $J_{\text{par,pot}}$.

It can still be said that the collision avoidance assistant handles the critical situation as expected, even if a model mismatch occurs.

As one can observe in Figure 8.7 and Figure 8.8 the tracked trajectory deviates clearly from the reference, due to violated model- and friction-conditions. The struggle can be seen clearly in Figure 8.6(D) and Figure 8.6(F). Still, the crash is avoided, even in the tracking-case (Figure 8.11).

For a possible further development this problem has to be tackled.
8.2.4 Case J1 – Collision avoidance in front of a pedestrian

Figure 8.12: Costs, (tracked) states, and inputs for the Case J1
Figure 8.13: Top view of the reference – Case J1 (1 cm $\equiv 5$ m)

Figure 8.14: Top view of the tracking – Case J1 (1 cm $\equiv 5$ m)
8.2.4.1 Evaluation of the result

Again, the collision avoidance assistant performs quite well in this scenario. As expected, the assistant turns left at the junction, as specified in the primary tracking trajectory, and accelerates to cross the predicted path of the pedestrian faster, to avoid a collision with the pedestrian.

The house as well as the approaching vehicle have no direct influence here but could play a role in further parameterisations of the scenario.

The moment when the ego crosses the pedestrian can be observed clearly in Figure 8.6(B) looking at $J_{\text{par,pot}}$. The physical boundaries of the object are never violated ($J_{\text{kamm}}$ and $J_{\text{model}}$ are 0 over all time) and the cost of the road-boundary-condition ($J_{\text{road}}$) rises just slightly when crossing the junction.

It can be said that the collision avoidance assistant handles the critical situation as expected. As one can observe in Figure 8.7 and Figure 8.8, the tracked trajectory is again nearly identical with the reference with just a small deviation in X-direction observable in Figure 8.6(C).
8.2.5 Case J2 – Collision avoidance behind a pedestrian

Figure 8.15: Costs, (tracked) states, and inputs for the Case J2
8.2. Performance evaluation

Figure 8.16: Top view of the reference – Case J2 (1 cm ≡ 5 m)

Figure 8.17: Top view of the tracking – Case J2 (1 cm ≡ 5 m)
8.2.5.1 Evaluation of the result

Again, the collision avoidance assistant performs quite well in this scenario. As expected, the assistant turns left at the junction, as specified in the primary tracking trajectory, the same way as in the previous scenario. Instead of an intense acceleration to cross the predicted path of the pedestrian in front of the pedestrian (as before) the assistant recognises that not enough space is provided in front of the pedestrian and starts a combined steering manoeuvre to cross the path of the pedestrian behind the pedestrian to avoid a collision.

The moment when the ego crosses the pedestrian can again be observed in Figure 8.6(B) looking at $J_{par,pot}$. The physical boundaries of the object are never violated ($J_{kamm}$ and $J_{model}$ are 0 over all time) and the costs of the road-boundary-condition ($J_{road}$) do not rise this time when crossing the junction (compared to the previous scenario) because the ego executes an extended turn to the left.

It can be said that the collision avoidance assistant handles the critical situation as expected.

As one can observe in Figure 8.7 and Figure 8.8 the tracked trajectory is again nearly identical with the reference with just a small deviation in X-direction observable in Figure 8.6(C) – just like in the previous scenario.

8.3 Conclusions

Under the presented condition, the collision avoidance assistant performs very well and acts as expected. A safe trajectory is planned and the selected model behaves in a way which is so close to the far more complex CarMaker model that the planned trajectories can easily be tracked with simple PI-controllers.

The tracking is still not perfect (as can be observed in Subsection 8.2.3), but with some adaptations in the weighting of the cost function, problems like that could be avoided, as it clearly results from the fact that the model leaves its range of validity.
Chapter 9

Results and outlook

In this chapter, the results are summarised once again, and an outlook for possible further use cases is given.

The proposed assistant is able to avoid collisions in all observed scenarios, and the general behaviour seems promising, even for different situations, as the system always reacts as expected. Except for one situation, sufficiently described in Subsection 8.2.3, the selected model is very well suited for the optimisation, as all collisions are avoided, and also the tracking delivers very good results. The approach with the four proposed PI-controllers including the optimised inputs as feed forward signals leads – without further adoption to specific situations – to a good following of the optimised trajectory.

The optimisation provides quite good results, but still, the online performance of an NMPC is not the best, as stated in Subsection 4.5.3, and even if the computation can be interrupted after a certain amount of optimisation time, it is never guaranteed that the result is the optimal one in the particular situation. Alternatives should be taken into account here. For the short-time NMPC (second layer of the optimisation), this could be difficult due to the rather complex structure of the cost-function, but for the long-time NMPC (first layer of the optimisation), alternatives could be found. Since in this step only the tracking of a reference is necessary, pre-computed trajectories as well as simple geometrical functions could be used, as denoted in Subsection 4.3.2.

While the prediction for short horizons (2 s) performs quite well, there is a lack in performance for long time predictions (5 s). It is hard to say if different prediction approaches (e.g. manoeuvre based ones) would lead to better results, as it is hard to find standard manoeuvres in collision situations. But one next step could be the implementation of such a prediction system.

The next steps would be to put into effect an online tracking instead of the currently used offline-evaluation, as the tracking-concept already works quite well and only the optimisation structure had to be merged with the CarMaker model. More measurements of general and also critical traffic situations would also be a good contribution if a further development of the system is intended, as well as a catalogue of critical testing scenarios or even other methods for the safety evaluation of such a system. An implementation of this method for extra-urban applications would be interesting, as the current system was developed and
tested only for urban scenarios with up to $50 \text{ km} \frac{1}{h}$.

To sum up, there is to say that the two-layer approach leads to quite promising results that should be further investigated.
BIBLIOGRAPHY

Bibliography


List of Acronyms

AEB  Autonomous Emergency Braking
AES  Autonomous Emergency Steering
AOP  Adult Occupant Protection
APF  Artificial Potential Field
C2C  Car-to-Car
CAN  Controller Area Network
COP  Child Occupant Protection
CPU  Central Processing Unit
Euro NCAP European New Car Assessment Program
IP   Interior Point
IPOPT Interior Point OPTimizer
KKT  Karush-Kuhn-Tucker
LIDAR Light Detection And Ranging
LSS  Lane Support System
MIT  Massachusetts Institute of Technology
MUMPS MUltifrontal Massively Parallel Sparse direct Solver
NMPC Non-linear Model Predictive Control
PI-controller Proportional–Integral-controller
RADAR Radio Detection And Ranging
SA   Safety Assist
SQP  Sequential Quadratic Programming
TTC  Time To Collision
UN/ECE United Nations Economic Commission for Europe
VRU  Vulnerable Road User
## List of Figures

1.1 Simplified architecture of a typical AEB/AES assistant .............................................. 3
2.1 Single track model – representation of states and inputs ............................................. 9
2.2 Single track model – current rotation centre $C(k)$ ..................................................... 10
2.3 Graphical representation of the forces in Kamm’s circle, N (North) = braking, S (South) = acceleration, E (East) = steering to the left, W (West) = steering to the right ......................................................................................... 12
3.1 Elements of a test scenario ............................................................................................. 14
3.2 Allowed ($A_{\text{road}}$) and prohibited ($P_{\text{road}}$) regions caused by road boundaries .... 15
3.3 Allowed ($A_{\text{stat}}$) and prohibited ($P_{\text{stat}}$) regions caused by static obstacles .......... 16
3.4 Allowed ($A_{\text{dyn}}(k)$) and prohibited ($P_{\text{dyn}}(k)$) regions caused by dynamic obstacles – for the current time step where it can be assumed that all states and dimensions are well-known.................................................................................. 16
3.5 Prohibited region $P_{\text{road}}^A$ ......................................................................................... 17
3.6 Distance $d(k)$ from a line to a point $p(k)$ ..................................................................... 18
3.7 Prohibited regions and allowed region $A_{\text{road}}$ ................................................................ 20
3.8 Edge of the ego-vehicle inside shifted border of the prohibited set $P_{\text{road}}^A$ ................ 21
3.9 Corner points of the single track model ........................................................................ 22
3.10 Possible cases for a point $P_O(k)$ ................................................................................ 24
3.11 Convexity of polygons ................................................................................................. 25
3.12 Collision detection for circular obstacles ....................................................................... 26
3.13 Collision detection for rectangular obstacles for $c = 4$ ................................................ 27
4.1 Concept of a single optimisation step in the two-layer approach ................................... 30
4.2 Reference-trajectories along a straight road, including velocity reference ................... 32
4.3 Reference-trajectories for a junction, including velocity reference for primary reference ..................................................................................................................... 33
4.4 Visualisation of $J_{tr.pos}$ and $J_{tr.vel}$ ......................................................................... 35
4.5 Visualisation of a detected crash for a step ..................................................................... 35
4.6 Visualisation of exemplary TTC-values .......................................................................... 37
4.7 Required measures to calculate $J_{pot}(k, i)$ for a single point ....................................... 40
4.8 Visualisation of $J_{pot}$ ..................................................................................................... 42
4.9 Required measures to calculate $J_{par}(k, i)$ for a single point ......................................... 42
4.10 Basic triangle to represent the law of cosines ................................................................. 43
4.11 Detailed view of the geometrical composition when calculating $J_{par}(k, i)$ for a single point ................................................................................................................................. 43
4.12 Visualisation of $J_{par}$ ..................................................................................................... 44
LIST OF FIGURES

4.13 Visualisation of the combination of $J_{par}$ and $J_{pot}$ according to Equation 4.38 45
4.14 Visualisation of a blocking strategy for the first layer of optimisation 50

5.1 Influence of noise on input-prediction – states 57
5.2 Influence of noise on input-prediction – inputs 57
5.3 States of an example prediction based on real measurement data 60
5.4 Graphical representation of hulls, and maximum as well as minimum states 60
5.5 Top view of an example prediction based on measurement data 61
5.6 Three-dimensional view of an example prediction based on measurement data 62
5.7 Top view of an example prediction based on measurement data with plotted prediction error 62
5.8 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars without minimum prediction deviation limits 64
5.9 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits 64
5.10 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles without minimum prediction deviation limits 64
5.11 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits 65
5.12 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians without minimum prediction deviation limits 65
5.13 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits 65

6.1 Picture of the used test-vehicle (BMW F31) 67
6.2 Features measured with the installed LIDAR system 69
6.3 Overview of the junction used for measurement, map provided by 22 70
6.4 Spatial frequency of the different objects 71
6.5 Length of the measurements 72
6.6 Velocity distribution 72
6.7 Velocity over time 73
6.8 Distribution of the orientation 73
6.9 Orientation over time 73
6.10 Estimated acceleration values 74
6.11 Estimated steering angle calculated with the $D_a$-values specified in Section 5.3 74
6.12 Estimated acceleration and steering angle 75

7.1 Overview of the scenario with a straight road, all units in m 78
7.2 Overview of the junction scenario, all units in m 80

8.1 Structure of the controller used for tracking 83
8.2 Relationship of inputs and outputs of the block $I_{dev}$ 85
8.3 Costs, (tracked) states, and inputs for the Case S1 86
8.4 Top view of the reference – Case S1 (1 cm $\cong 10$ m) 87
8.5 Top view of the tracking – Case S1 (1 cm $\cong 10$ m) 87
8.6 Costs, (tracked) states, and inputs for the Case S2 88
8.7 Top view of the reference – Case S2 (1 cm $\cong 5$ m) 89
8.8 Top view of the tracking – Case S2 (1 cm $\cong 5$ m) 89
8.9 Costs, (tracked) states, and inputs for the Case S3 90
A.1 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (10 steps training data) A2
A.2 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (20 steps training data) A2
A.3 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (30 steps training data) A2
A.4 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (10 steps training data) A3
A.5 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (20 steps training data) A3
A.6 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (30 steps training data) A3
A.7 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (10 steps training data) A4
A.8 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (20 steps training data) A4
A.9 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (30 steps training data) A4
A.10 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (10 steps training data) A5
A.11 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (20 steps training data) A5
A.12 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (30 steps training data) A5
A.13 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (10 steps training data) A6
A.14 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (20 steps training data) A6
A.15 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (10 steps training data) A7
A.16 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (20 steps training data) A7

A.17 Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (30 steps training data) A7
List of Tables

1.1 Euro NCAP rating overview .................................................. 2
4.1 Overview of the TTC-regions and the consequences for the optimisation ... 38
4.2 Summary of the blocking preferences ........................................ 51
5.1 Common axle spacing values in mm ........................................... 63
6.1 Sensor comparison .............................................................. 68
6.2 Attributes - ibeo LUX 4L ......................................................... 68
6.3 Distribution of relevant characteristics ...................................... 71
7.1 Geometry and initial states for selected scenarios .......................... 78
8.1 Parameters of the used PI-controllers ....................................... 85
Appendix A

Validation of the prediction method
A.1 20 steps ahead prediction

A.1.1 Cars

![Figure A.1: Visualisation of median as well as quantile \(Q_{0.25}\) and \(Q_{0.75}\) of the prediction error for cars with minimum prediction deviation limits (10 steps training data)](image1)

![Figure A.2: Visualisation of median as well as quantile \(Q_{0.25}\) and \(Q_{0.75}\) of the prediction error for cars with minimum prediction deviation limits (20 steps training data)](image2)

![Figure A.3: Visualisation of median as well as quantile \(Q_{0.25}\) and \(Q_{0.75}\) of the prediction error for cars with minimum prediction deviation limits (30 steps training data)](image3)
A.1.2 Bicycles

Figure A.4: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (10 steps training data)

Figure A.5: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (20 steps training data)

Figure A.6: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for bicycles with minimum prediction deviation limits (30 steps training data)
A.1.3 Pedestrians

Figure A.7: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (10 steps training data)

Figure A.8: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (20 steps training data)

Figure A.9: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (30 steps training data)
A.2 50 steps ahead prediction

A.2.1 Cars

Figure A.10: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (10 steps training data)

Figure A.11: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (20 steps training data)

Figure A.12: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for cars with minimum prediction deviation limits (30 steps training data)
A.2.2 Bicycles

Unfortunately, no valid bicycle-dataset long enough for a 50 steps ahead prediction and 30 steps training data was measured.
A.2.3 Pedestrians

Figure A.15: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (10 steps training data)

Figure A.16: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (20 steps training data)

Figure A.17: Visualisation of median as well as quantile $Q_{0.25}$ and $Q_{0.75}$ of the prediction error for pedestrians with minimum prediction deviation limits (30 steps training data)