Technische Universiteit Delft

Automated Driving Functionality for Intersection Crossing

Master of Science Thesis

A. van der Heide

TU Delft

Automated Driving Functionality for Intersection Crossing

Master of Science Thesis

by

A. van der Heide

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday March 22, 2018 at 10:00 AM.

Student number:	1523341	
Project duration:	June 6, 2017 – March 22, 2018	
Supervisors:	Dr. ir. S. Moten,	Siemens PLM Software
	Ir. F. Celiberti.	Siemens PLM Software
Thesis committee:	Dr. ir. M. Mazo Jr.,	TU Delft
	Dr. A.V. Proskurnikov,	TU Delft
	Dr. ir. R. Happee,	TU Delft
	Dr. ir. Y. Lemmens.	Siemens PLM Software

This thesis is confidential and cannot be made public until March 22, 2023.

TU Delft

Abstract

When vehicles are approaching an intersection, their trajectories might be overlapping. If two or more vehicles are arriving at the same time at this overlapping region, they will end up in a collision. To prevent this from happening, a driver can adjust the speed of the vehicle and enter the intersection before the other vehicle enters or after the other leaves the intersection.

The goal of the project is to develop an autonomous driving feature for a driving simulator, specifically for the intersection scenario. This controller has to replace the inputs, that are normally done by a human and has to navigate the vehicle safely through the intersection by avoiding possible collisions, while going straight or by taking left or right turns. The controller is verified on a high fidelity model, that is placed in an urban-like environment model, and validated on a driving simulator with human-in-the-loop capabilities. In the simulator the system is also used as an Advanced driver-assistance systems, which runs in the background when the human driver is in control of the vehicle and takes over controller when a collision needs to be avoided.

To solve the problem of a vehicle approaching an intersection, several challenges have to be faced. A first step is to split the problem into two parts. The first part is the longitudinal control and the second the lateral. In the longitudinal control, collision avoidance is achieved when one vehicle leaves the intersection before the other enters. For the collision avoidance, the problem is reduced to a one dimensional problem, by assuming that the trajectories are predetermined and do not change. Using this assumption, the control inputs on the vehicle are reduced to throttle and brake. The second assumption is that there are only two vehicles approaching the intersection, from which the first vehicle with the controller is considered as the *Ego* vehicle and the second vehicle is considered as the *other* vehicle. To avoid collision, it is possible to define a ”bad set” of states in which the vehicles collide. To prevent the flow of vehicles from entering this bad set, a second set, called the ”capture set”, is defined and computed. This capture set consists of all the states that lead to a collision, irrespective of the applied inputs. The generation of this capture set is done by taking the two extreme inputs, full throttle and full brakes, and by using the back propagation of the bad set

For the collision avoidance system, two scenarios are considered. In the first scenario, both cars are driving autonomous and decide together about who takes priority by making use of direct communication like Vehicle to Vehicle (V2V) or indirect by making use of Vehicle to Infrastructure (V2I). The decision about who goes first is based on the distance to the intersection and the velocities of the vehicles. The result is that one of the vehicles applies full throttle, while the other applies full brakes. The control is applied at the border of the capture set and to compute this set, two separate sets are generated: the first set consists of all the states that lead to the bad set with applying full throttle on the other vehicle and full brake on the Ego vehicle, the second consists of all the states that lead to the bad set with applying full throttle on the Ego vehicle and full brakes on the other vehicle. The intersection of these sets results in the capture set. When the capture set is found, a feedback map is generated that prevents the flow of vehicles from entering the capture set and, consequently, leads to avoiding a collision.

In the following scenario, the control of the other vehicle is unknown to the Ego vehicle. This other vehicle can be seen as a human driven vehicle or an autonomous vehicle without communication. The exact dynamics of the second vehicle are unknown and have to be estimated. Vehicles are assumed to be accelerating and decelerating within certain boundaries, when they are approaching an intersection. This range of accelerations is used as the kinematics to generate the capture set. The following step is to use a feedback map, as was used in the case with two autonomous vehicles. However, the resulting capture set is much larger in this case and to decrease the size of the capture set, the intention of the other vehicle has to be estimated. By assuming that a driver does not change the type of input at a

certain range, from a few meters before the intersection till the end of the intersection, the size of the capture set is reduced and the collision avoidance system becomes less conservative.

For the lateral control, a path tracking algorithm is used. It is assumed that the coordinates of the trajectory are known and by using a look-ahead distance, a coordinate is found to track. The next step is to use the pure pursuit algorithm, which finds a circle with a certain radius that intersects with the tracked coordinate and with the coordinates of the vehicle. From the radius of the circle and the Ackermann steering principle, a steering wheel angle is determined. Because the speed is limited in the corners, the collision avoidance system is adjusted. If the velocity of the Ego vehicle is higher than the maximum velocity in the corner for preventing an overshoot at the trajectory, the capture set takes the shape of the set of states that lead to the bad set with the brakes as input. This adjusting of the algorithm makes sure that the collision avoidance is still achieved and the vehicle does not overshoot the corner.

The testing of the two controllers is done in two ways. A part of the testing is done by making use of simulations of a fully autonomous vehicle with predefined speed profiles. The second method is by testing with a human-in-the-loop on a simulator. In this part, a human driver uses pedals for throttle and brake to set the speed of the vehicle. If the collision avoidance notices that a collision might happen, the system takes over control and guides the vehicle safely through the intersection.

In the end, a system is obtained that can increase the safety of vehicles at intersections. The system is able to safely navigate the vehicle through an intersection, while taking turns, and by avoiding a collision with one other road user. This road user can be an autonomous or a human driven vehicle.

Preface

This report is the master thesis as final part of the Vehicle Engineering master program at Delft University of Technology in the Mechanical Engineering department. The work was performed in collaboration with Siemens PLM Software in Leuven, Belgium, as part of the ENABLE-S3 project.

Special thanks to my supervisors Sikandar Moten and Francesco Celiberti for their support during the weekly progress meeting, which kept me on track during the time I spent there. I am also very grateful for their help with getting used to all the different software. I would also like to thank Manuel Mazo for pointing me in the right direction during the research.

Furthermore, I want to thank all my colleagues at Siemens PLM Software for making my time at the company very enjoyable.

Finally, I want to thank my family and friends for supporting me during my entire studies in Delft. Without their support, I would have not been able to successfully finish my engineering degree.

A. van der Heide
Delft, March 2018

Contents

List of Figures

List of Tables

List of Abbreviations

ABS	Anti-Lock Braking System.
ADAS	Advanced driver-assistance systems.
DWA	Dynamic Window Approach.
EPS	Electronic Power Steering.
ESP	Electronic Stability Program.
HITL	Human-in-the-Loop.
ICE	Internal Combustion Engine.
PDF	Probability density function.
RTDB	Real-Time Database.
UDP	User Datagram Protocol.
V2I	Vehicle to Infrastructure.
V2V	Vehicle to Vehicle.

List of Symbols

Sign	Description	Unit
I_s	Coefficient between steering wheel angle and steering angle of the wheels	
L	Lower boundary of the bad set	m
T_f	Simulation time in the state estimator	s
T	Simulation time in the capture set generator	s
U	Upper boundary of the bad set	m
α	Angle between the longitudinal axis of the vehicle and the vector directing from the vehicle to the tracked coordinate	rad
β	Typical acceleration or deceleration of the other vehicle at intersections	$\mathrm{m\,s^{-2}}$
δ_s	Steering wheel angle	rad
δ_w	Steering angle of the wheels	rad
γ	Standard deviation of the acceleration or deceleration of the other vehicle at intersections	$\mathrm{m\,s^{-2}}$
κ	Curvature	$\mathrm{m^{-1}}$
$\mathcal{B}$	Bad set: the unsafe set of states	
$\mathcal{C}_{u_B}$	Braking capture slice: set of states lead the trajectory to the bad set when the Ego vehicle applies full brakes	
$\mathcal{C}_{u_C}$	Throttle capture slice: set of states that lead the trajectory to the bad set when the Ego vehicle applies full throttle	
$\mathcal{C}$	Capture set: the set of states that lead the trajectory to the bad set	
ω_{RPM}	Engine speed	RPM
ϕ	Flow or trajectory in longitudinal motion of both vehicles	
ψ	Heading of the vehicle in global coordinates	rad
c_a	Look-ahead distance tuning parameter	
d_p	Distance between the center of the pure pursuit circle and the y coordinate of the waypoint	m
g	Gear number	
i	Vehicle: 1 is the Ego vehicle and 2 is the other vehicle	
j	Step in the future state estimator	
k	Step in the generation of the capture slices	
l_a	Look-ahead distance	m
l_g	Distance between the tracked coordinate and the vehicle coordinates	m
l_w	Wheelbase, distance between front and rear axle	m
p	Longitudinal position	m
q	Mode of the other vehicle, {A}: braking, {B}: accelerating {A,B}: unknown	
r	Radius of the pure pursuit circle	m
t	Simulation time	s
u_B	Control input torque with the Ego vehicle applying full brakes	N m
u_C	Control input torque with the Ego vehicle applying full throttle	N m
u	Control input torque	N m
v_a	Maximum longitudinal velocity in a corner to prevent overshooting	$\mathrm{m\,s^{-1}}$

Sign	Description	Unit
v	Longitudinal velocity	$\mathrm{m\,s^{-1}}$

1 Introduction

To reduce the number of traffic accidents on public roads, nowadays computers are aiding humans or taking over control from humans in vehicles. The necessity of Advanced driver-assistance systems (ADAS) in vehicles is growing. New vehicles are being equipped with ADAS, like Adaptive Cruise Control (ACC), to avoid head-to-tail collisions. The final goal is to achieve a fully autonomous vehicle by combining these systems. In the past, a lot of progress in the field of automated driving was made during the DARPA Grand Challenges in 2004, 2005 and 2007. Especially in the 2007 edition [26], when the challenge was placed in an urban like environment with simulating real-life situations, the potential of autonomous vehicles was demonstrated. Nowadays, a lot of companies, like Google [16], are doing research in the field and some companies are already testing on public roads. Other companies, like Tesla [25], are including all the necessary sensors for automated driving on their models to meet the full automation standards.

Because there are many combinations of how much of the control is automated and how much is still controlled by a human driver, the Society of Automotive Engineers (SAE) categorized the levels of automation in a table. The table consists of six levels reaching from no automation to fully automated vehicles [11]:

0. No Automation
1. Driver Assistance
2. Partial Automation
3. Conditional Automation
4. High Automation
5. Full Automation

From level zero to two, the human is monitoring the environment and the automation is only helping the driver moderately. In these levels, the driver has the main responsibility. In level three to five, the system is monitoring the environment and has full responsibility in some situations or in all situations.

To reach a high level of automation, several engineering fields need to be combined. First, it is important to use the right sensors to obtain data of the environment. Data from several types of sensors are combined to make a map of the environment and to categorize the upcoming situation. The following step is the motion planning, which plans a route from initial location to a destination and makes sure that this plan is being followed. The final part in autonomous driving is controlling the actuators. For this project, the focus is laid on the motion planning part.

Motion planing for autonomous vehicles is a high complexity problem and it is not solvable in one step. Therefore, the problem is split in several layers, that are solved separately. A typical way is to split the problem in four components [20]:

- Global planning

- Behavioral layer
- Local planner
- Local feedback control

These four components are shown as an overview in figure 1.1. In this figure, the global planning is called the route planning and the local planner is called motion planning.

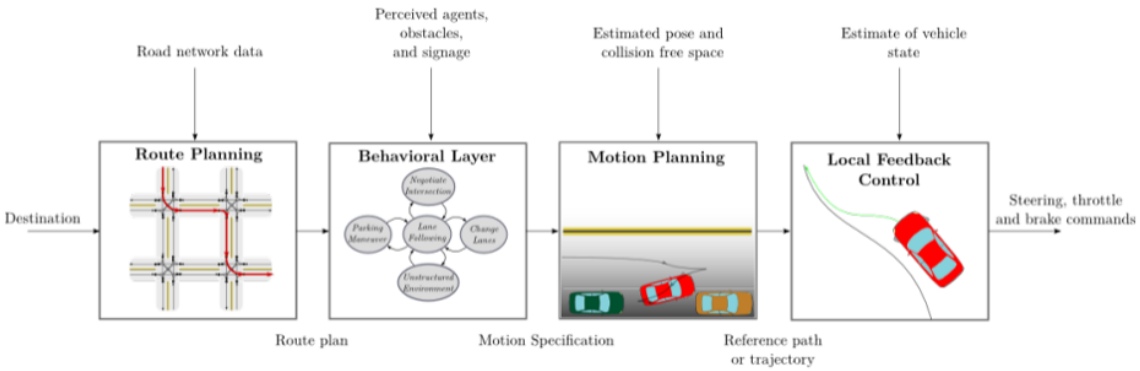

Figure 1.1: Overview of the different layers in motion planning for autonomous vehicles [20]

In the first part, the global planning delivers a route from an initial location to a destination. The second block is the behavioral layer, which determines the type of situation the autonomous vehicle is encountering. For example, this could be the type of intersection, the amount of other road users, high way driving, etc. In the third part, the local planner, collisions with other road users or obstacles is being avoided. Finally, the local feedback control keeps the autonomous vehicle on the previously determined trajectory.

1.1. Problem statement

One of the challenges in driving is the safe crossing of the vehicle at an intersection. During this maneuver, trajectories of various vehicles may overlap at a certain moment. When two vehicles reach this overlapping part at the same time, they are colliding and this can be dangerous. It is possible that only damage is being done on the vehicles, but when the impact is too high, the occupants can get injured. Therefore, these situations have to be avoided. The problem that is being solved in this project is the safe passage of a vehicle at intersections, while having an overlapping trajectory with one other road user.

1.2. Objective of the project

The goal of the project is to develop autonomous driving features for a driving simulator with X-in-the-loop capabilities [17], where X is model, software, hardware, driver/human, etc. This driving simulator is developed at Siemens PLM Software for testing and validating ADAS. The system is equiped with a full multibody vehicle model with subsystems, Human-in-the-Loop (HITL) interface, a motion platform like is shown in figure 1.2 and a virtual environment in rFactor Pro for visualization. In this project, autonomous driving features are added to the simulator and the virtual environment is replaced by a realistic urban-like environment model.

The controller for this project is an automated driving functionality for intersection crossings. It has to be able to make a vehicle safely pass an intersection from initial location to its destination, while avoiding a collision with one other road user. This controller is used in a fully autonomous vehicle and is used as an ADAS. In the first case, when it is used in a fully autonomous vehicle with no driver giving any input, the controller guides the vehicle safely through the intersection, while going left, right or straight, and a cruise control is determining the velocity. It is also possible to use the controller as

an ADAS. In this situation, there is a human driver in control of the vehicle and if a possible collision scenario occurs, the system takes over control to be sure that the vehicle is safe.

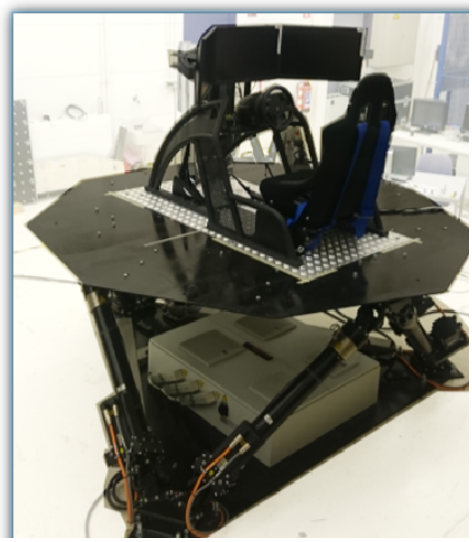

Figure 1.2: Driving simulator with a human-in-the-loop interface

To limit the scope of the problem, it is assumed that there are only two vehicles approaching the intersection. One of these vehicles is considered as the *Ego* vehicle, which is the vehicle that has the controller on board. The second vehicle is considered as the *other* vehicle and can be autonomous or human driven. If the other vehicle is autonomous, information about the input is being shared with the Ego vehicle to avoid the collision. If the vehicle is human driven, there is no information shared and the other vehicle is not responding to any actions of the Ego vehicle. In this scenario, the other vehicle is not making any intentions to avoid a collision. In both scenarios, the goal is to have an Ego vehicle that guarantees that there is no collision. In the visualization of the simulations, the Ego vehicle is represented by a red Mazda RX-8 and the other vehicle by a black Audi A8, as is shown in figure 1.3.

Figure 1.3: The vehicles in the simulations are visualized by a red car (left) for the Ego vehicle, and a black car (right) for the other vehicle

1.3. Methodology and state-of-the-art

To solve the problem of intersection crossing, the method of occupation grids was often used in the past [30],[14],[12],[15]. In this method, the space surrounding a vehicle is discretized in nodes. The next step is to check whether the resulting nodes are occupied by obstacles or other road users and nodes are given a value between zero and one. Because the location of the other road users is dynamic, an estimation is made on their future locations. These estimations are added to the value of the nodes. After giving a value to each node, the nodes with a value less than the threshold are used in a grid searching algorithm, like D* [23], to find a safe trajectory. Because the nodes with a low value are included, there is a chance of a collision. To make sure that no collision occurs, the system is often

combined with an extra collision avoidance system, like the Dynamic Window Approach (DWA) [7]. This is a low-level controller that checks the velocities for being safe and, when this is not the case, changes the velocity of the vehicle.

Another method for collision avoidance is reachability analysis [1]. With reachability analysis, a set of initial states and a set of unsafe states is determined. The unsafe set is the set of states to be avoided and in the intersection case, this is the set of states in which the vehicles are colliding. The goal is to check whether future trajectories end up in the unsafe set. If they do not end up in the unsafe set, the trajectory is assumed as safe, which is visualized in figure 1.4.

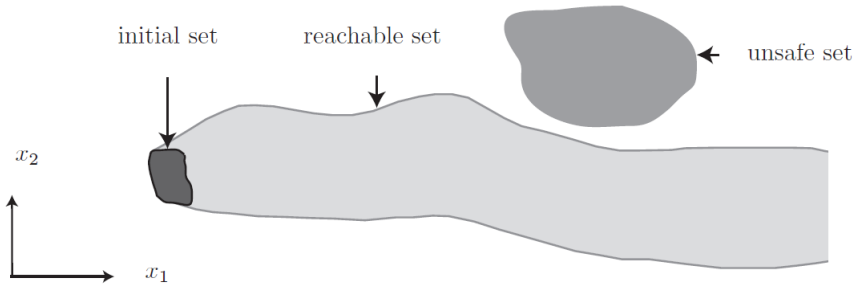

Figure 1.4: A safe system verified by using reachability analysis [1]

The general type of reachability analysis starts from an initial set and works towards future sets, it is also called forwards reachability analysis. On the other side, it is possible to start from the unsafe set and propagate towards the initial states, which is called backwards reachability analysis. With backwards reachability analysis, the states that lead the system to the unsafe set, irrespective of the input, have to be found [29]. After this method, a control map is designed to make sure that the system never ends up in this set of states [3]. This method has been applied on scaled vehicles, that had an overlapping trajectory forming a type of intersection [27]. In these tests, one of the scaled models was driving autonomous and the other was driven by a human. During the tests, the human was trying to force a collision with the autonomous vehicle. The method has also been applied on full scale vehicles [8]. In these tests, both cars were driving autonomous and were using Vehicle to Vehicle (V2V) communication to avoid a collision.

Next to the collision avoidance, the vehicle has to be able to track a trajectory for making a turn at the intersection. For path tracking, many different solutions are available in the field of robotics. For a car-like vehicle, the nonholonomic constraints have to be taken into considerations, because a vehicle can not suddenly move sideways. Path following can be obtained by detailed kinematic or dynamic path tracking [22]. These methods require a detailed model of the vehicle and are hard to tune. A different method is geometrical path tracking [2], which has as advantage that it is easy to implement and can be shared between various vehicles without having to change many parameters.

1.4. Applying the state-of-the-art

Reachability analysis for collision avoidance was in the research, from the previous subsection, applied in only a few type of scenarios. Furthermore, the velocity of the Ego vehicle in the tests was always lower than the maximum velocity and there was no extra speed limit at a part of the trajectory, which is similar as crossing the intersection with a straight trajectory and without turns. When a vehicle is approaching an intersection and has to turn left or right, the velocity of the vehicle is currently higher than the velocity at a future location of the trajectory. The vehicle has to slow down to make sure that it can make the turn without overshooting the planned trajectory. This additional requirement introduces the problem that the throttle input is not available for avoiding a collision. Furthermore, the controllers were only tested on real vehicles or scaled vehicles. By testing the controllers also in a virtual environment with a driving simulator, more scenarios can be tested and the interaction with the subsystems or other ADAS can be investigated. In the virtual environment, it is also easier to test the controller on different vehicles.

In the project, the controllers are implemented in an existing full mutibody model with subsystems, like the internal combustion engine, anti-lock braking system, etc. The other road user is modelled by

simple kinematics and dynamics. The tests are split into two groups: offline and online tests. In the offline tests results are investigated after simulation and there is no time limit for the models to solve. The vehicles are placed in a virtual environment on two straight roads that form an intersection at a certain moment.

Before implementation is possible on a real vehicle, the system has to be tested for real-time solving. If the controllers are not able to give an output at the right time, a collision might still occur. Therefore, the controllers are tested on a Concurrent platform, which guarantees a deterministic solution and guarantees having one second in simulation as one second in real life. Therefore, the algorithms of the controllers have to be solvable in the time that they are allowed to spent. With this real-time requirement, it is possible to test the system as an ADAS with an human-in-the-loop. This human driver can use the pedals for throttle and brakes to determine the velocity of the vehicle, when the situation is safe. When an accident might occur, the collision avoidance system takes over control to keep the Ego vehicle from colliding with the other road user.

To deliver a realistic experience for the driver, a detailed virtual environment is made with two intersections in a row, each with a different other road user arriving at a similar time as the Ego vehicle. Next to the virtual environment, the driver experiences the accelerations in a vehicle by reproducing them on a motion platform. The complete testing environment is split into two parts: hard and soft real-time. The hard real-time has to solve the components that require a deterministic solution and the soft real-time consists of the parts that have to run real-time, but when there is a small delay or a missed integration step, this is not a problem.

1.5. Report overview

This report is divided in several chapters. The second chapter is about the general collision avoidance theory and algorithms. In this chapter, only longitudinal motion of the vehicles is considered. The following chapter introduces the ability of making turns at the intersection. The first part is about trajectory tracking and the second part is about combining the lateral motion with the collision avoidance system from the previous chapter. In the fourth chapter, the testing environment is discussed by describing the software and hardware used for the simulations and tests. This chapter also includes the vehicle models used for verifying the controllers. The fifth chapter shows the results from the simulations and tests in multiple scenarios. The report ends with conclusions about the research and gives a discussion about future research on the subject.

2

Collision Avoidance at Intersections

When two vehicles are approaching an intersection, there is a chance of a collision. The vehicles are colliding, when they reach a range of locations at the same time. To avoid this, one of the vehicles has to take priority over the other. Who needs to take priority is based on many things. In general, this is dictated by the traffic rules. To prevent a vehicle from waiting till all other vehicles, that have priority, left the intersection, one can pass the intersection if the situation is safe. In a scenario, where one car arrives much earlier than the other vehicle and it can leave the intersection before the other arrives, the first arriving vehicle has to take priority for having a high traffic flow through the intersection. The requirement for having no collision between the vehicles, that have an overlapping trajectory, is solved by having one of the vehicles to leave the intersection before the other one enters. The collision avoidance system has to work in all situations with one other road user. The system also has to consider the worst case scenario, which includes having the other vehicle trying to force a collision. For the Ego vehicle, this results in a system that only takes priority when it is safe and there is no chance of a collision.

In the collision avoidance system, two cases are considered. In the first case, two autonomous vehicles are approaching the intersection and information is shared about the input. This information sharing can be done by making use of direct communication using Vehicle to Vehicle (V2V) or indirect by using Vehicle to Infrastructure (V2I) for both vehicles. The communication is necessary to determine which of the vehicles goes first at the intersection. In the second case, only the Ego vehicle is autonomous and has the collision avoidance system on board. The other vehicle is assumed as a human driven vehicle and might try to hit the Ego vehicle. The Ego vehicle does not have any information about the intentions of the other vehicle.

To reduce the scope of the problem, it is assumed that both vehicles are crossing the intersection in a straight trajectory. Using this assumption, only the longitudinal motion of each vehicle is considered. The turning at the intersection will be discussed in the following chapter. Figure 2.1 shows a visualization of the problem that is discussed in this chapter and shows an intersection from the top view, with two vehicles having a trajectory that is overlapping at a certain moment. In this figure, the Ego vehicle is visualized by a red vehicle and is crossing the intersection from West to East direction. The other vehicle is shown by a black vehicle and is approaching the intersection from South direction and will move towards the North. If both vehicles are reaching the overlapping region at the same time, they end up in a collision.

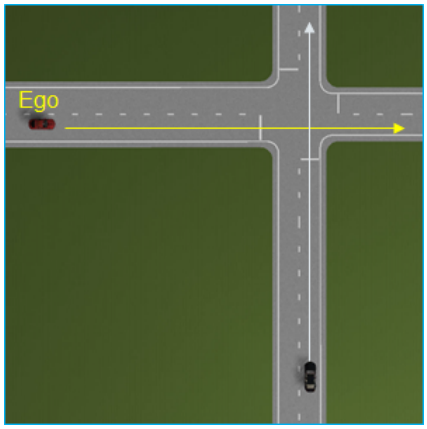

Figure 2.1: Two vehicles approaching an intersection with trajectories that are overlapping

2.1. Autonomous - Autonomous case

The first scenario that is considered is having two autonomous vehicles approaching the intersection at the same time. If no control is applied, they will collide. To solve this problem, one vehicle has to apply the brakes, while the other accelerates to leave the intersection as fast as possible. Because the vehicles are able to communicate with each other, they each know what the other vehicle will do. Many combinations of inputs are available to avoid the collision and, therefore, a strategy is required. If the goal is to have the collision avoidance only acting at the last moment, when a collision can still be avoided, it is possible to only look at the most extreme cases of input. These most extreme cases are having one vehicle apply full brakes with the other full throttle and the other way around. Using this strategy, the collision avoidance is only being used in emergency and when it is absolutely necessary.

2.1.1. Bad and capture set definition

To make sure that there is no collision between the vehicles, the method of backwards reachability analysis is used [1]. In reachability analysis, a set of initial states and a set of unsafe sates is determined. The set of initial states are the longitudinal positions and velocities of the vehicles. The unsafe states are all the combinations of longitudinal positions of the vehicles at which the vehicles are colliding. When using forward reachability analysis, the possible trajectories are formed from the initial set by integrating over time and using a certain input on the dynamics. A trajectory is assumed to be safe, when it does not end up in the unsafe set.

In the case of backwards reachability analysis, the integration is done by using the unsafe set as starting point of integration and the states leading to the unsafe set are found. With the intersection problem from figure 2.1, the unsafe set of states consist of four boundaries in the longitudinal positions. For both vehicles i, there is a lower boundary L_i and an upper boundary U_i in longitudinal positions at the trajectory. Figure 2.2 shows how these boundaries are defined in this problem.

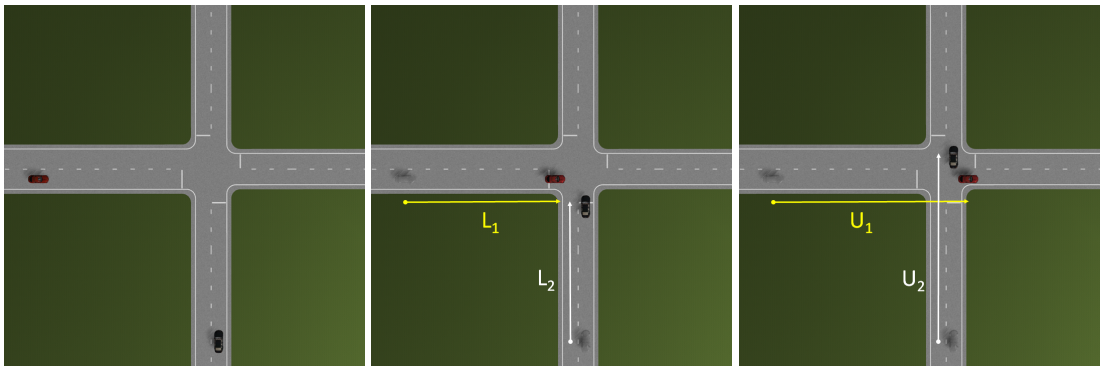

Figure 2.2: Boundaries of the collision area between the two vehicles

The next step is to define the continuous state space of the system. This is noted by X. The

state vector $x \in X$ is such that $x = (p_1, v_1, p_2, v_2)$. In which p_i is the longitudinal position and v_i the longitudinal velocity of vehicle i. Combining this with the boundaries for a collision, it is possible to define the unsafe set of states. This unsafe set is called the "**bad set**" $\mathcal{B}$[27].

$$\mathcal{B} = \{(p_1, v_1, p_2, v_2) \in \mathbb{R}^4 | (p_1, p_2) \in [L_1, U_1] \times [L_2, U_2]\} \tag{2.1}$$

This definition says that the bad set is the set of states at which the longitudinal positions of both vehicles are within the boundary limits of each vehicle. If at a certain moment $x \in \mathcal{B}$, the vehicles are colliding. Because only the longitudinal motion is considered, this bad set can be visualized in a 2D plot with the longitudinal positions of the vehicles on each of the axes. Figure 2.3 shows the longitudinal position of the Ego vehicle on the horizontal axis and the position of the other vehicle on the vertical axis. In this graph, it is now possible to define a black square as the bad set.

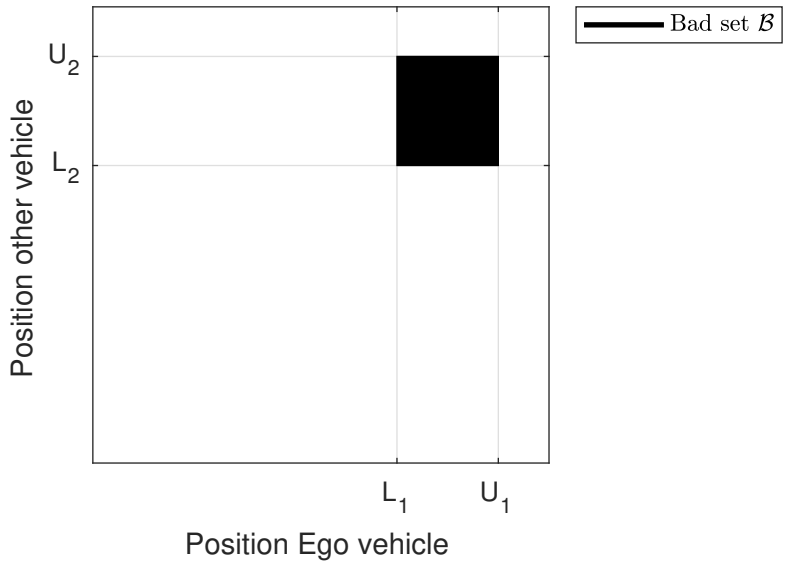

Figure 2.3: Location of the bad set in a 2D graph of the longitudinal positions of the vehicles

After defining the unsafe set, a strategy will be developed to make the system avoid it. Before the strategy can be designed, the system has to be defined. The system of two vehicles approaching an intersection is modelled as a hybrid automaton [5]. It can be defined as a tuple $H = (Q, X, U, f, R)$. Where Q is the set of modes of the system, which are accelerating and decelerating or keeping a constant velocity. The continuous state space X was introduced in the previous section together with the state $x \in X = \mathbb{R}^4$, such that (p_1, v_1, p_2, v_2). The following component is the continuous set of control inputs $U = [u_{\text{min}}, u_{\text{max}}] \subset \mathbb{R}$, which are the control input torques. The maximum braking torque is given by u_{min} and the maximum accelerating torque by u_{max}. The mode reset map is given by $R(x, u) := q$ if $(x, u) \in \text{Dom}(q)$, in which $\text{Dom}(q) : Q \to 2^{X \times U}$ is a map that is attached to a mode with its set of continuous states and inputs. The last component is the vector field $f : X \times Q \times U \to X$. This vector field is piece-wise continuous and depends on the dynamics of the vehicle. The discontinuity comes from the assumptions that the vehicles have a minimum velocity to prevent them from going backwards and a maximum velocity for the speed limitation on the road. The vector field is given by $f(x, q, u) = (f_1(p_1, v_1, u_1), f_2(p_2, v_2, u_2))$. Where:

$$f_i(p_i, v_i, u_i) = \begin{pmatrix} v_i \\ \begin{cases} 0 & \text{if } (v_i = v_{i,\text{min}} \text{ and } \alpha_i < 0) \\ & \text{or } (v_i = v_{i,\text{max}} \text{ and } \alpha_i > 0) \\ \alpha_i & \text{otherwise} \end{cases} \end{pmatrix} \tag{2.2}$$

In which $\alpha_i = a_i u + b_i - c_i {v_i}^2$ and $\dot{p}_i = v_i$. The derivation of the coefficients is coming from the longitudinal dynamics of the vehicle. A more thorough elaboration on the coefficients is given in appendix A.1. It can be shown that $H = H_1 || H_2$, in which H_i are the order preserving hybrid automata of each vehicle [3]. The individual hybrid automaton for each vehicle is shown in figure 2.4.

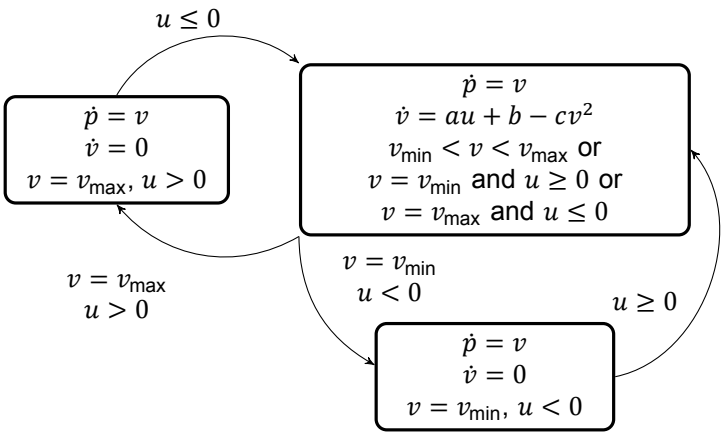

Figure 2.4: Hybrid automaton modelling the longitudinal dynamics of each vehicle [3]

From the defined model of the system and the unsafe set, the set of states that lead the flow to the the unsafe set, irrespective of the input, has to be found. This set of states is called the "**capture set**" $\mathcal{C}$. The definition of the capture set is given by [5]:

$$\mathcal{C} = \{x \in X | \exists t \geq 0 \text{ s. t. } \phi(t, x, u) \in \mathcal{B}\} \tag{2.3}$$

In which t is the simulation time and ϕ is the trajectory or flow of the vehicles. The generation of the capture set is in general a complicated process. However, it is not hard to find the set of states that lead to the bad set when a fixed input is used. For this fixed input, the extreme inputs of the vehicles can be used. The extreme inputs are defined as the situation in which one vehicle applies maximum throttle and the other maximum brakes. As the inputs can also be turned around, there are two extreme inputs. The first extreme input is defined as the brake input $u_B := (u_{1,\text{min}}, u_{2,\text{max}})$, in which the Ego vehicle applies brakes and the other vehicle applies throttle. The second extreme input is defined as the throttle input $u_C := (u_{1,\text{max}}, u_{2,\text{min}})$, with the Ego vehicle applying throttle and the other vehicle brakes. For each of these inputs, the set of states that lead the system to the bad set can be generated. For the brake input, this set of states is called the braking capture slice $\mathcal{C}_{u_B}$, which is defined as: $\mathcal{C}_{u_B} = \{x \in X | \exists t \geq 0 \text{ s.t. } \phi(t, x, u_B) \in \mathcal{B}\}$. A visualization of this capture slice in the longitudinal position diagram is shown in figure 2.5.

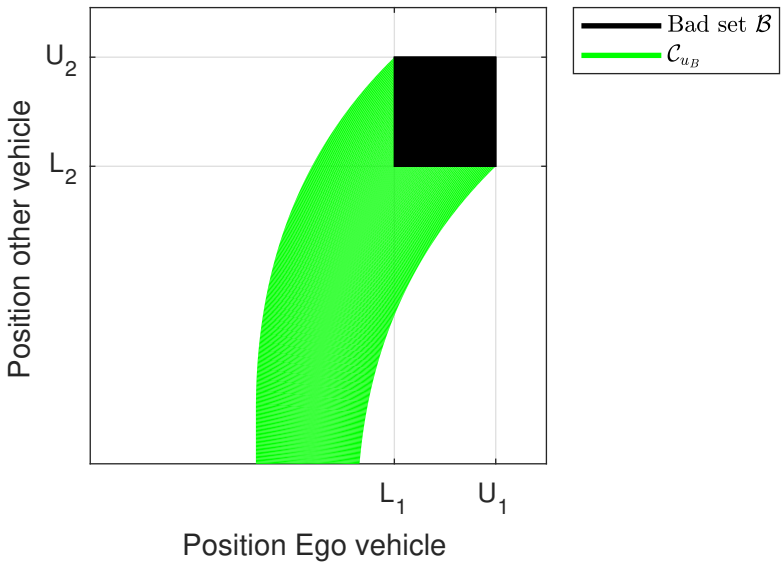

Figure 2.5: The braking capture slice and the bad set in the longitudinal position diagram

The same thing is done for the throttle input, which results in the throttle capture slice. This throttle

capture slice is defined as: $\mathcal{C}_{u_C} = \{x \in X | \exists t \geq 0 \text{ s.t. } \phi(t, x, u_C) \in \mathcal{B}\}$. Figure 2.6 shows the resulting capture slice in the longitudinal position diagram.

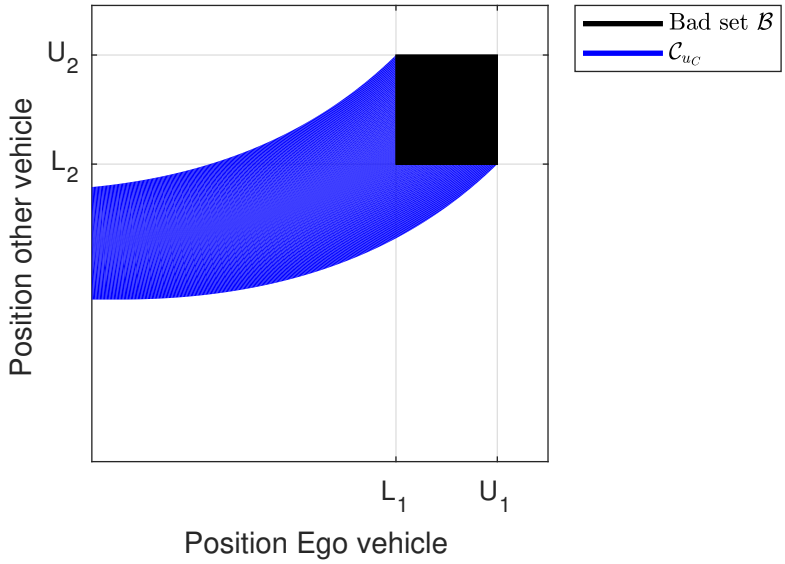

Figure 2.6: The throttle capture slice and the bad set in the longitudinal position diagram

Using the two capture slices, the capture set is generated. By the definitions of the capture slices and by looking at the figures, it is easy to see that the set of unavoidable states is the intersection of both slices [3]. Because both extreme inputs lead the states in this set to a collision, there is no other input available to make the trajectory avoid the bad set. This is equal to the the definition of the capture set and therefore the capture set is found by taking the intersection of both capture slices.

$$\mathcal{C} = \mathcal{C}_{u_B} \cap \mathcal{C}_{u_C} \tag{2.4}$$

This intersection of the slices can be visualized in the longitudinal position diagrams by combining figure 2.5 and 2.6. The resulting capture set is shown in figure 2.7.

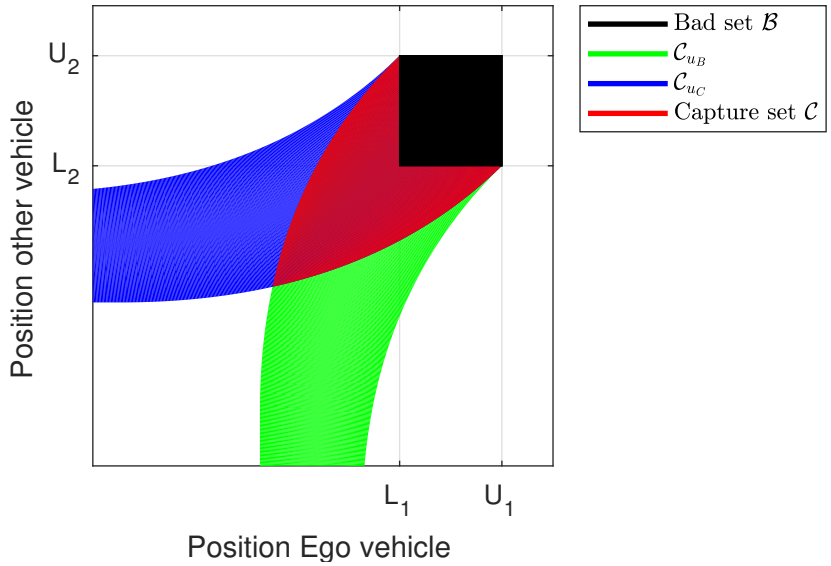

Figure 2.7: The capture set, capture slices and bad set in the longitudinal position diagram

2.1.2. Capture set generation algorithm

The algorithm of the generation of the capture set consists of two parts. One part is the generation of capture slices $\mathcal{C}_{u_B}$ and $\mathcal{C}_{u_C}$ and the other part is about the actual capture set $\mathcal{C}$. The generation of the

capture slices does not depend on the current positions p_i of the vehicles, but only on the velocities v_i and inputs u_i. Therefore, a new position state, or distance travelled, $\bar{p}$ is introduced. This position state has as initial value zero and is used for the back propagation of the bad set. In the first time step k, using a discretization time ΔT and Euler integration, the distance travelled is equal to the velocity times the time step. Using the dynamics and inputs of the vehicle, the new velocity of the vehicle is calculated in the same way.

$$\begin{aligned} \bar{p}_i[k+1] &= \bar{p}_i[k] + v_i[k]\Delta T \\ v_i[k+1] &= v_i[k] + (a_i u_i + b_i - c_i {v_i}^2[k])\Delta T \end{aligned} \tag{2.5}$$

If the velocity in the step $[k+1]$ is reaching the boundary $v_{\max}$ or $v_{\min}$, the velocity takes the boundary value. For the generation of the capture slice, the locations of the bad set are back propagated by using the travelled distance.

$$\begin{aligned} L_i[k+1] &= L_i - \bar{p}_i[k+1] \\ U_i[k+1] &= U_i - \bar{p}_i[k+1] \end{aligned} \tag{2.6}$$

The combination of L_i and U_i for all time steps k results in the capture slice $\mathcal{C}_u$. It is also possible to define the algorithm in symbols [4]. First, a new state vector has to be defined with the new position state as $\bar{x}_i := (v_i)$. The hybrid automaton is used with $H = H_1 || H_2$ and each vehicle is given by $i \in \{1, 2\}$, with $i = 1$ for the Ego and $i = 2$ for the other vehicle. It can be shown that $\bar{p}' = \bar{p} + F_{i,1}(\bar{x}_i, u_i)$ and $\bar{x}_i' = \bar{F}_i(\bar{x}_i, u_i)$. In which the primed variables are the states in the following time step. $F_{i,1}(\bar{x}_i, u_i) = f_{i,1}(\bar{x}_i, R_i(\bar{x}_i, u_i))\Delta T$, $\bar{F}_i(\bar{x}_i, u_i)$ and $F_{i,1}(\bar{x}_i, u_i)$ are order preserving in the state and input. It can be set: $\bar{F}_i^0(\bar{x}_i, u_i) := \bar{x}_i$ and $\bar{F}_i^{k+1}(\bar{x}_i, u_i) := \bar{F}_i(\bar{F}_i^k(\bar{x}_i, u_i))$ for each time step $k = 0, 1, \ldots$. The generation of the capture slices is done in a discrete way using Euler integration. In symbols the total algorithm is given by [4]:

Algorithm 1 Capture slice generation in symbols

For each $i \in \{1, 2\}$ and $k \in \mathbb{N}$ let $L_i^k(\bar{x}_i, u_i) = L_i - \sum_{j=0}^{k-1} F_{i,1}(\bar{F}_i^j(\bar{x}_i, u_i), u_i)$, $U_i^k(\bar{x}_i, u_i) = U_i - \sum_{j=0}^{k-1} F_{i,1}(\bar{F}_i^j(\bar{x}_i, u_i))$. Then $\mathcal{C}_u = \{x \in X | \exists k \geq 0$ with $L_i^k(\bar{x}_i, u_i) < x_{i,1} < U_i^k(\bar{x}_i, u_i) \forall i\}$.

During the generation of the capture slices, it is checked whether the flow of the vehicles is inside the capture slice: $x \cap \mathcal{C}_u$. The following statement is checked on true or false for each vehicle:

$$L_i[k+1] < p_i < U_i[k+1] \tag{2.7}$$

If all of the statements are true, the algorithm outputs a boolean with the value true. The algorithm terminates in two ways. First, there is the maximum number of steps in the capture slice generation to guarantee that the algorithm terminates in a fixed time for the real-time requirement. The second way the algorithm is terminating, is when the upper boundary becomes less than the longitudinal position of the Ego vehicle. Because there is a minimum velocity defined that is zero or positive, there is no possibility that the flow is in that capture slice at a future integration step.

The generation of the capture slices is done for both inputs u_B and u_C. If the booleans of both capture slices have as value true, the flow is inside the capture set and there is no need to generate the actual capture set.

2.1.3. Control strategy

After the generation of the capture slices, it is possible to define a control map. This control map has to make sure that the flow of the vehicles does not enter the capture set. As the two capture slices are build using a certain control type, it is easy to see that control has to be applied at the border of the capture set $\mathcal{C}$.

The type of control depends on the location of the trajectory at the border of the capture set. The strategy is to implement a control map that decides about the type of control that is going to be used. When the trajectory is at the border of the capture set and inside the capture slice $\mathcal{C}_{u_C}$, control u_B has to be used to avoid the bad set. This is the control in which the Ego vehicle applies the brakes and the other vehicle throttle. If the trajectory is again at the border, but now inside capture slice $\mathcal{C}_{u_B}$, the opposite control u_C, in which Ego applies throttle and the other applies brakes, has to be used. When

the trajectory is not in any of the slices, but still at the border, u_B or u_C has to be applied to avoid ending up in the bad set. It does not matter which one it uses, as both make the trajectory avoid the bad set. When the vehicles are not at the border of the capture set, it does not matter what control is used, because the system is safe in a future time step. The control map in symbols becomes [3]:

$$\begin{cases} u_B & \text{if} \quad x \in \mathcal{C}_{u_C} \cap \partial\mathcal{C}_{u_B} \\ u_C & \text{if} \quad x \in \mathcal{C}_{u_B} \cap \partial\mathcal{C}_{u_C} \\ \{u_C, u_B\} & \text{if} \quad x \in \partial\mathcal{C}_{u_B} \cap \partial\mathcal{C}_{u_C} \\ - & \text{otherwise} \end{cases} \tag{2.8}$$

In the final controller, it was decided to use u_B in the third case of the control map. Visually in the graph it is easy to see that when the trajectory hits the border, one of the two types of control has to be used to avoid entering the capture set. Using the algorithm of section 2.1.2, it is not that straightforward. This due to the fact that the real capture set is not computed, but it is only checked if the trajectory is inside one or both of the capture slices. To determine if the trajectory is at the border, a future state estimator is used. This future state estimator estimates the two extreme cases of a following state. The future state is noted by $\hat{x}_i = (\hat{p}_i, \hat{v}_i)$. The following step is to find the lower estimate $\wedge\bar{x}_i$, which is found by applying the brake on both vehicles and the higher estimate $\wedge\bar{x}_i$ is found by applying the throttle. In the future state estimator, Euler integration is used with a step size ΔT_f and each step is given by j. The lower estimate becomes:

$$\begin{aligned} \wedge\hat{p}_i[j+1] &= \wedge\hat{p}_i[j] + \wedge\hat{v}_i[j]\Delta T_f \\ \wedge\hat{v}_i[j+1] &= \wedge\hat{v}_i[j] + ((a_i u_{i,\text{min}} + b_i - c_i \wedge {v_i}^2[j])\Delta T_f \end{aligned} \tag{2.9}$$

With the same procedure, the higher estimate is generated by:

$$\begin{aligned} \vee\hat{p}_i[j+1] &= \vee\hat{p}_i[j] + \vee\hat{v}_i[j]\Delta T_f \\ \vee\hat{v}_i[j+1] &= \vee\hat{v}_i[j] + ((a_i u_{i,\text{max}} + b_i - c_i \vee {v_i}^2[j])\Delta T_f \end{aligned} \tag{2.10}$$

With the lower and higher estimates, new capture slices are generated and these capture slices are checked for intersection with the estimated trajectory. The lower estimate is used for the propagation of the upper boundary and the higher estimate for the lower boundary [9]. In the following equations $\bar{p}_{i,1}$ is used for the position of vehicle i with the higher estimate of the velocity and $\bar{p}_{i,2}$ with the lower estimate. Both have zero as initial value for using of the back propagation of the bad set. The estimates of the velocity are noted as $v_{i,1}[0] = \vee\hat{v}_i[j_{\text{max}}]$ and $v_{i,2}[0] = \wedge\hat{v}_i[j_{\text{max}}]$. The algorithm for each vehicle in the generation of the capture slices becomes:

$$\begin{aligned} \bar{p}_{i,1}[k+1] &= \bar{p}_{i,1}[k] + v_{i,1}[k]\Delta T \\ v_{i,1}[k+1] &= v_{i,1}[k] + (a_i u_i + b_i - c_i {v_{i,1}}^2[k])\Delta T \end{aligned} \tag{2.11}$$

$$\begin{aligned} \bar{p}_{i,2}[k+1] &= \bar{p}_{i,2}[k] + v_{i,2}[k]\Delta T \\ v_{i,2}[k+1] &= v_{i,2}[k] + (a_i u_i + b_i - c_i {v_{i,2}}^2[k])\Delta T \end{aligned} \tag{2.12}$$

$$\begin{aligned} L_i[k+1] &= L_i - \bar{p}_{i,1}[k+1] \\ U_i[k+1] &= U_i - \bar{p}_{i,2}[k+1] \end{aligned} \tag{2.13}$$

The combination of these lower and upper boundaries for all time steps results in the capture slice $\hat{\mathcal{C}}_u$ for the estimated states. Finally, it is checked whether the future state is in the capture slice. The range of the lower and higher estimates has to be inside the propagation of the lower and upper boundary of the bad set and, therefore, the following statements is checked for truth.

$$\begin{aligned} \vee\hat{p}_i[j_{\text{max}}] &> L_i[k+1] \\ \wedge\hat{p}_i[j_{\text{max}}] &< U_i[k+1] \end{aligned} \tag{2.14}$$

If both statements are true for each vehicle, the future state is inside the capture slice and the boolean has as output true. An overview of all components in the collision avoidance is given in figure 2.8.

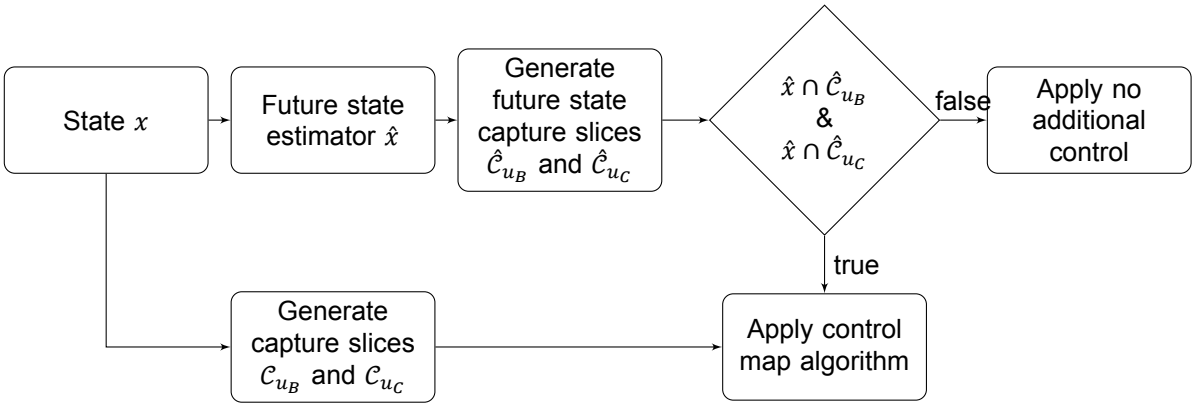

Figure 2.8: Overview of the collision avoidance system

This figure shows that, from the current state, the capture slices are generated and at the same time the future states are calculated. From the future states, the velocities are used to generate the two capture slices $\hat{\mathcal{C}}_{u_B}$ and $\hat{\mathcal{C}}_{u_C}$ with inputs u_B and u_C. If the future states are not inside these capture slices, no additional control is applied. If they are inside, the collision avoidance gives an output based on the current states and capture slices $\mathcal{C}_{u_B}$ and $\mathcal{C}_{u_C}$. The algorithm, to determine the output, is based on the control map of equation 2.8 and is in pseudo-code given by [8]:

Algorithm 2 Control map algorithm

if $x \cap \mathcal{C}_{u_B} = 0$ and $x \cap \mathcal{C}_{u_C} = 1$ **then**
  $u = u_B$
else if $x \cap \mathcal{C}_{u_C} = 0$ and $x \cap \mathcal{C}_{u_B} = 1$ **then**
  $u = u_C$
else
  $u = u_B$
end if

This algorithm shows that, when the trajectory is not yet in the braking capture slice and it is in the throttle capture slice, the braking input is applied. When it is in the braking capture slice and not yet in the throttle capture slice, the throttle input is applied. In all other cases, the system also chooses for the braking input.

2.2. Autonomous - Human driven case

In this section, the same scenario with two vehicles approaching the intersection is used. The difference is that this time there is no information about the input shared between the vehicles. Therefore, it is assumed that only the input of the Ego vehicle can be used for collision avoidance. The other road user can be human driven, but can also be another autonomous vehicle with no option for communciation. To make the system work also work in the worst case scenario, it is assumed that the other road user acts completely independent of the Ego vehicle and makes no effort to avoid a collision. Again, it is assumed that ground truth information about the position and velocity of the vehicles is known.

2.2.1. Bad and capture set re-definition

Since the input and dynamics of the other vehicle are not known, the dynamics in the algorithm is being removed and replaced by simple kinematics. It is assumed, that vehicles are accelerating and decelerating within certain boundaries, when they are approaching an intersection. Therefore, the kinematic system is given by: $\dot{p}_2 = v_2$ and $\dot{v}_2 = \beta_q + \gamma_q d$ [27]. In these equations, $q \in \{A, B\}$ is the mode of the vehicle. Where, $\{A\}$ is braking, $\{B\}$ is accelerating and $\{A, B\}$ is unknown. The parameter β_q corresponds to an average value of the acceleration during mode q, γ_q is the standard deviation from this average value and together with $d \in [-\bar{d}, \bar{d}]$ it forms the disturbance. This disturbance covers the differences between drivers and different types of cars. The values for the typical accelerations and standard deviations have to be obtained from testing with human drivers approaching an intersection.

The results from these tests give a Gaussian distribution and look like the functions in figure 2.9 [28]. With these new kinematics, the maximum and minimum velocity are limited again for preventing driving backwards and respecting the maximum velocity dictated by the traffic rules.

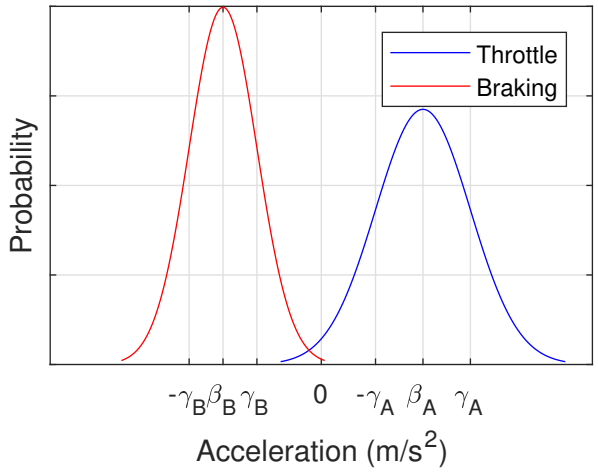

Figure 2.9: Probability density functions for the braking and accelerating at intersections

In the capture set generation algorithm, the new kinematics are replacing the dynamics of the other vehicle. In the part of the Ego vehicle nothing is changed. On the total system, the input is adjusted to only the Ego vehicle with the full braking $u_B = u_{1,\min}$ and the full throttle $u_C = u_{1,\max}$.

The initial conditions of the longitudinal position in the back propagation of the bad set $p_{2,1}[0]$ and $p_{2,2}[0]$ are defined as zero. The initial conditions of the velocity $v_{2,1}[0]$ and $v_{2,2}[0]$ have the value of the current velocity of the other vehicle. Two velocities are being used, because one of them is computed by using the boundary of the lowest acceleration and the other is using the highest. The change in position is done in a similar way as in equation 2.5 with Euler integration of the two velocities.

$$\begin{aligned} \bar{p}_{2,1}[k+1] &= \bar{p}_{2,1}[k] + v_{2,1}[k]\Delta T \\ \bar{p}_{2,2}[k+1] &= \bar{p}_{2,2}[k] + v_{2,2}[k]\Delta T \end{aligned} \tag{2.15}$$

The propagation of the boundaries each uses one of the double integrated accelerations. The lower boundary propagates with the highest acceleration and the higher boundary with the lowest.

$$\begin{aligned} L_2[k+1] &= L_2 - \bar{p}_{2,1}[k+1] \\ U_2[k+1] &= U_2 - \bar{p}_{2,2}[k+1] \end{aligned} \tag{2.16}$$

The velocity of the other vehicle over time changes due to the newly defined kinematics. These kinematics depend on the mode of the other vehicle. When the mode is known, the velocities in the capture slice generation change as followed:

$$\begin{aligned} v_{2,1}[k+1] &= v_{2,1}[k] + (\beta_q + \gamma_q \bar{d})\Delta T \\ v_{2,2}[k+1] &= v_{2,2}[k] + (\beta_q - \gamma_q \bar{d})\Delta T \end{aligned} \tag{2.17}$$

If the mode of the other vehicle is unknown, the capture slices are generated in the worst case scenario. The propagation of the lower boundary is done by the accelerating kinematics and the propagation of the higher boundary with the braking kinematics.

$$\begin{aligned} v_{2,1}[k+1] &= v_{2,1}[k] + (\beta_B + \gamma_B \bar{d})\Delta T \\ v_{2,2}[k+1] &= v_{2,2}[k] + (\beta_A - \gamma_A \bar{d})\Delta T \end{aligned} \tag{2.18}$$

If $v_{2,1}$ or $v_{2,2}$ become lower than the minimum or higher than the maximum, they take the value of the limit. Typical capture sets using these new kinematics look like the one shown in figure 2.10. By comparing this figure to figure 2.7, it can be concluded that the size of the capture set has increased. This makes sense, because the Ego vehicle does not know what the other vehicle is going to do and has to consider all options.

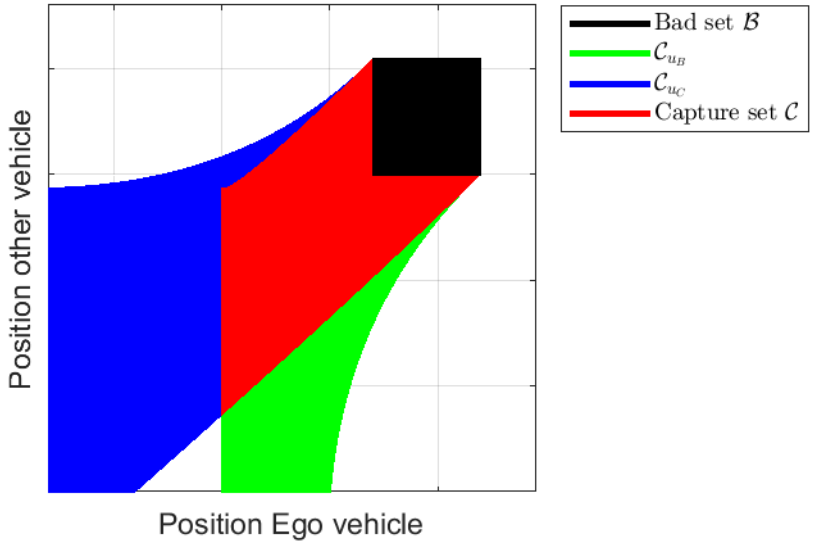

Figure 2.10: Capture set and capture slices with the human driver kinematics

2.2.2. Control strategy

The control strategy is making use of the same approach as the autonomous-autonomous case and the control map of equation 2.8 is used [27]. The only difference, in the case with the human other vehicle, is that there is only one control output, the one on the Ego vehicle. It also makes use of a future state estimator to determine if the flow is at the boundary of the capture set. This future state estimator has to be adjusted for the other vehicle. The lower estimate is done by taking the braking mode for deceleration and the higher estimate is taking the acceleration mode.

$$\begin{aligned} \wedge\hat{p}_2[j+1] &= \wedge\hat{p}_2[j] + \wedge\hat{v}_2[j]\Delta T_f \\ \wedge\hat{v}_2[j+1] &= \wedge\hat{v}_2[j] + (\beta_A - \gamma_A\bar{d})\Delta T_f \end{aligned} \tag{2.19}$$

With the same procedure the higher estimate is generated:

$$\begin{aligned} \vee\hat{p}_2[j+1] &= \vee\hat{p}_2[j] + \vee\hat{v}_2[j]\Delta T_s \\ \vee\hat{v}_2[j+1] &= \vee\hat{v}_2[j] + (\beta_B + \gamma_B\bar{d})\Delta T_s \end{aligned} \tag{2.20}$$

The final values of the velocities in the future state estimator are used as initial conditions in the capture slice generator. The initial conditions are defined as $v_{2,1}[0] = \vee\hat{v}_2[j_{\max}]$ and $v_{2,2}[0] = \wedge\hat{v}_2[j_{\max}]$.

The next step is to decide which equations will be used for the capture set generation, equation 2.17 or 2.18. The decision depends on the mode of the other vehicle. The reason for the two equations is that the size of the capture set is preferably as small as possible, otherwise the Ego vehicle becomes too conservative. Therefore, a mode estimator is being used [29]. It is assumed that in the last meters before the intersection and at the intersection the human driver decides to apply brakes or throttle and does not switch between input type. Therefore, the mode estimator measures the acceleration of the other vehicle from this decision point till the end of the upper boundary of the bad set. The measured acceleration is given by by $\hat{\beta}$ and if the measured acceleration is out of the range of the braking decelerations, it is assumed that the mode is $\{B\}$ accelerating. This is shown in figure 2.11 with using $\bar{d} = 3$ for the limit.

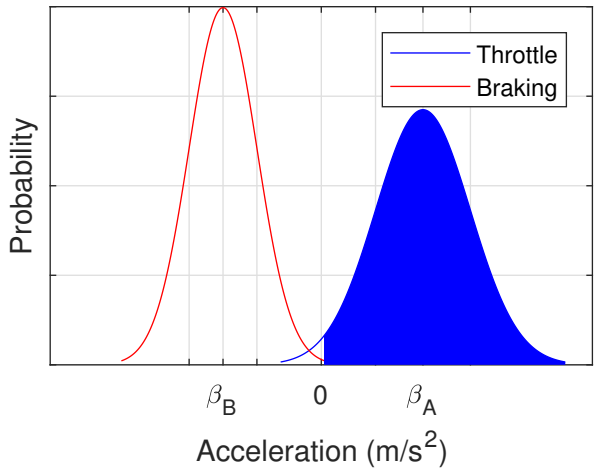

Figure 2.11: The range of accelerations at which the mode estimator assumes accelerating of the other vehicle

If it is out of the range of the throttle accelerations, the mode is assumed $\{A\}$ braking, which is shown in figure 2.12.

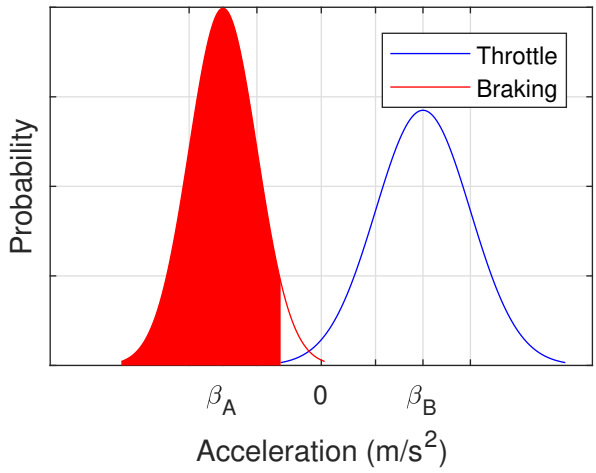

Figure 2.12: The range of accelerations at which the mode estimator assumes braking of the other vehicle

In the final case, where it is still in both the ranges, the mode is unknown $\{A, B\}$. The estimated mode is given by $\hat{q}$ and is determined according to the following map [29]:

$$\begin{aligned} \hat{q} &= A \quad \text{if} \quad |\hat{\beta} - \beta_B| > \gamma_B \bar{d} \\ \hat{q} &= B \quad \text{if} \quad |\hat{\beta} - \beta_A| > \gamma_A \bar{d} \\ \hat{q} &= \{A, B\} \quad \text{otherwise} \end{aligned} \tag{2.21}$$

With the mode of the other vehicle, the capture slices of the future states $\hat{\mathcal{C}}_{u_B}$ and $\hat{\mathcal{C}}_{u_C}$ are generated and checked for intersection. The overview of the collision avoidance system in the human driven case is shown in figure 2.13.

This figure shows that the system is almost the same as in the autonomous-autonomous case. The only difference is that there is one block extra for the mode estimator. The algorithm for the control map is the same as the one in algorithm 2.

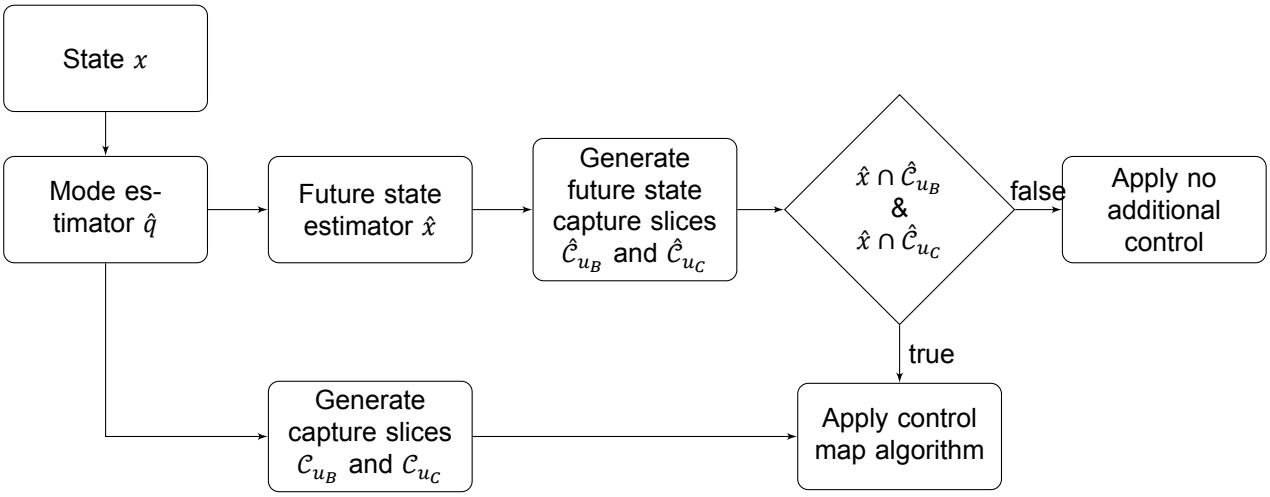

Figure 2.13: Overview of the collision avoidance system in the human driven case

2.2.3. System uncertainties

In the collision avoidance system, a simplified model of the dynamics is used. In the entire system there are several ways to make sure that the system is safe, even when the vehicle model is not exactly representing the real dynamics. By taking a larger step size or more steps in the future state estimator, the collision avoidance system interferes at an earlier stage. More steps or larger step size, make the future states closer to the bad set and the intersection of these future states with the estimated capture slices happen at an earlier stage. This adjustment makes the system more conservative, but also uncertainties are covered.

Another option is to make the boundaries of the human driven kinematics larger by increasing the value of γ_q. This increases the size of the capture set, because the Ego vehicle is expecting larger accelerations and decelerations than the other vehicle is actual able to do. Figure 2.14 shows an increase in size of the capture set by changing the value of γ_q. The original capture set is shown by the dashed line.

Figure 2.14: The capture set with increasing the value for the standard deviation of the acceleration of the other vehicle

The same approach can be used for the uncertainties in the dynamics of the Ego vehicle. The inputs are reduced by taking a lower value than the maximum brake and throttle torques, which results in an increase of the size of the capture set. Figure 2.15 shows the increased capture set due to lower maximum inputs and it also shows the boundaries of the original capture set with the black dashed line.

The larger capture set area makes the collision avoidance system interfere earlier than actually necessary. The only problem that now arises, is that the control map causes the trajectory to bounce on and off the capture set. The control map outputs are only full throttle and full brakes. To prevent this from happening, the output of the collision avoidance system is changed to a desired velocity. This desired velocity is the velocity at a certain time step in the capture set generation. Using the current velocity and this new desired velocity, the error between them is found. The following step is to use a PID controller to reduce this error to zero. Using this method the uncertainties in the system are covered, while the trajectory will not be bouncing on and off the capture set.

Figure 2.15: The capture set with lowering the values for the maximum brake and throttle torque of the Ego vehicle

3

Turning at the Intersection

In the previous chapter, collision avoidance was obtained for a scenario in which two vehicles are approaching an intersection and both have a straight trajectory without making a turn. In this chapter, the Ego vehicle is going to make left or right turns. First, a path tracking method for lateral motion is discussed. Followed by combining the collision avoidance with the ability to make turns. When a vehicle is making a turn at the intersection, the speed is limited at a part of the trajectory. If the velocity of the vehicle is higher than this limit, it overshoots the corner. Usually, the velocity is higher on the straight in front of the corner and the vehicle has to brake to reduce it. Therefore, an adjustment to the collision avoidance system is suggested to safely make turns at intersections.

3.1. Lateral motion controller

For making a turn at the intersection, the Ego vehicle has to track a trajectory. It is assumed that the trajectories are predefined by a set of (x, y) coordinates in the global reference frame. There are many options available for path tracking [22]. Most of the methods require a precise model of the kinematics or dynamics of the vehicle and make use of the error between the vehicle and the trajectory coordinates. Path tracking can also be done by making use of geometry and an example for this is the Pure Pursuit controller [2]. An advantage of geometric path tracking is that it can be applied on different vehicles without having to change many parameters and without having to know much details of the vehicle dynamics or kinematics.

3.1.1. Coordinate transformation

The first step, for path tracking, is to transform the coordinates of the planned trajectory from the global coordinate system to the vehicle axis system. It is assumed that the vehicle is only moving in a 2D plane and does not change in altitude, or in other words the z-axis. Therefore, the equations for the coordinate transformations become:

$$x_{w,v} = (x_{w,g} - x_{r,g})\cos(\psi) + (y_{w,g} - y_{r,g})\sin(\psi) \tag{3.1}$$

$$y_{w,v} = -(x_{w,g} - x_{r,g})\sin(\psi) + (y_{w,g} - y_{r,g})\cos(\psi) \tag{3.2}$$

In which the subscripts w and r are for the way point and vehicle coordinates respectively. The subscripts g and v are for the global and vehicle axis system. Finally, ψ is the heading angle of the vehicle. The heading angle is defined as the angle between the x-axis of the vehicle system and the x-axis of the global coordinate system.

3.1.2. Look-ahead distance

The following step is to find the right coordinate on the trajectory. In most cases, the closest trajectory coordinate is not the preferred coordinate as it might result in a jerky motion. It is often better to take a coordinate at a certain distance from the vehicle. To find the right coordinate, a look-ahead distance l_a is used. This look-ahead distance can be seen as a circle, with a radius equal to the distance,

surrounding the vehicle as is shown in figure 3.1. In the original pure pursuit algorithm the distance was considered as a constant [2], in more recent studies, it was found that for getting better results, it is often beneficial to make the look-ahead distance variable with the longitudinal velocity [18]:

$$l_a = vc_a \tag{3.3}$$

In which c_a is a constant and has to be tuned, the longitudinal velocity is given by v. The equations show that the look-ahead distance is going to zero, when the longitudinal velocity goes to zero. This results in a problem at standstill and, at low speed, it results in a very jerky motion on the steering wheel. When the vehicle is reaching high velocities, the distance becomes very large and is also not desired, because a large look-ahead distance causes the vehicle to cut corners. Therefore, the look-ahead distance is saturated at a minimum and maximum: $l_a \in [l_{a,\text{min}}, l_{a,\text{max}}]$.

3.1.3. Finding the coordinate to track

Now, it is needed to find the right coordinate on the trajectory to track. In the perfect situation, the coordinate to track is the point where the circle of the look-ahead distance intersects with the trajectory.

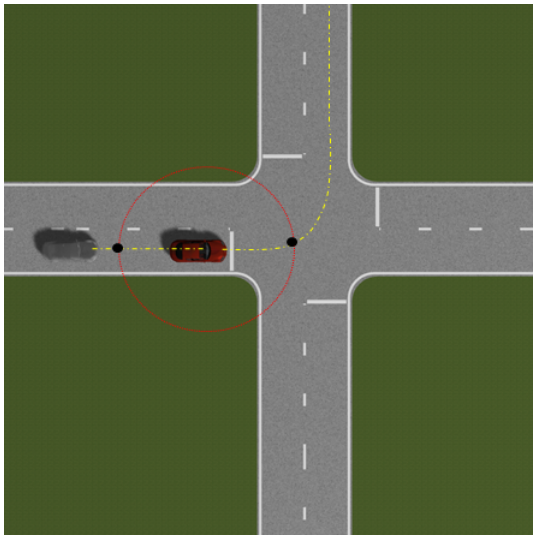

Figure 3.1: The look-ahead distance circle intersects at two locations with the planned trajectory

Figure 3.1 shows that there are two points for this. For the tracking, it is only interesting to use the point laying in front of the vehicle. As the trajectory consists of discrete points, the goal is to find the closest coordinate outside of the look-ahead distance circle. To only look for the coordinates that are laying in front of the vehicle, all coordinates with $x_{w,v} < 0$ are omitted. The next step, is to remove all the coordinates that are inside of the look-ahead distance circle. Therefore, the distance from the vehicle to the coordinate points is calculated by:

$$l_g = \sqrt{x_{w,v}^2 + y_{w,v}^2} \tag{3.4}$$

The coordinates with $l_g < l_a$ are left out and the final step is to find the coordinate with the lowest distance towards the vehicle. This is the coordinate to use in the pure pursuit algorithm.

3.1.4. Pure pursuit algorithm

The goal of the pure pursuit algorithm is to find a circle that goes through the vehicle coordinates and the coordinate from the previous subsection [2]. Therefore, the geometry of figure 3.2 is used.

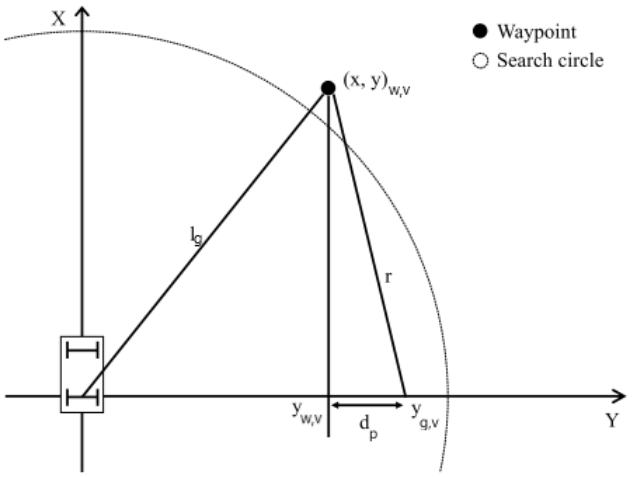

Figure 3.2: Illustration of the pure pursuit algorithm [18]

In this figure, a new point is introduced. This point will be defined as the center of the circle, that goes through the vehicle and the trajectory. The center of this circle is located on the y-axis at a certain distance d_p from the projection of the trajectory coordinate on the y-axis:

$$y_{g,v} = y_{w,v} + d_p \tag{3.5}$$

In which $y_{g,v}$ is the coordinate of the center point. Because it is located on the y-axis, it is equal to the radius r of the circle.

$$r = y_{g,v} = y_{w,v} + d_p \tag{3.6}$$

For the following steps, the equation is rewritten to d_p:

$$d_p = r - y_{w,v} \tag{3.7}$$

From Pythagoras and figure 3.2, the right triangle gives:

$$d_p^2 + x_{w,v}^2 = r^2 \tag{3.8}$$

Combining this equation with equation 3.7, results in:

$$(r - y_{w,v})^2 + x_{w,v}^2 = r^2 \tag{3.9}$$

This can again be rewritten as:

$$r^2 - 2ry_{w,v} + x_{w,v}^2 + y_{w,v}^2 = r^2 \tag{3.10}$$

Now, the radius squared terms are removed and inserting equation 3.4 results in the equation for the radius of the circle:

$$r = \frac{l_g^2}{2y_{w,v}} \tag{3.11}$$

The curvature κ is the inverse of the radius:

$$\kappa = \frac{2y_{w,v}}{l_g^2} \tag{3.12}$$

It is useful to define an angle α, which is the angle between the longitudinal axis and the vector from the vehicle to the goal coordinate. The sinus of this angle is equal to the y distance from the vehicle to the goal point.

$$\sin(\alpha) = y_{w,v} \tag{3.13}$$

Using this definition in equation 3.14, results in:

$$r = \frac{l_g}{2\sin(\alpha)} \tag{3.14}$$

3.1.5. Ackermann steering

After finding the curvature of the circle through the vehicle and the goal coordinate, the steering angle has to be set. For setting the steering angle, the Ackermann steering simplification is used, which is visualized in figure 3.3.

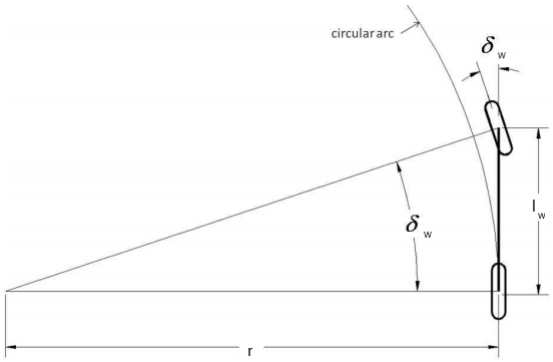

Figure 3.3: Geometry of the Ackermann steering simplification on a bicycle model [22]

The Ackermann steering simplification relates the steering wheel angle δ_w to the radius of the circle [22]:

$$\tan(\delta_w) = \frac{l_w}{r} \tag{3.15}$$

Using the previously found expression for the radius of the circle in equation 3.14, this can be rewritten as:

$$\delta_w = \frac{2l_w \sin(\alpha)}{l_g} \tag{3.16}$$

The steering angle is applied on the wheels of the bicycle model. For a vehicle, which consists of four wheels, a correction has to be applied. However, it is also possible to apply the steering on the wheel that the driver normally holds. Now, it is only needed to find a relationship between the steering wheel angle and the steering angle of the wheels. In general, this is often assumed to be linear.

$$\delta_s = I_s \delta_w \tag{3.17}$$

3.1.6. Velocity planning

Next to the adjustment of the steering wheel angle, the velocity of the vehicle has to be be adjusted. The maximum velocity in a corner is the allowable velocity v_a, which depends on the curvature of the corner. It is assumed that the vehicle keeps a constant velocity during cornering and brakes on the straight before the corner. The location at which the velocity starts to decrease depends on the difference between v_a and the current velocity v.

3.2. Effect on collision avoidance

The following step is to combine the lateral motion at the intersection with the collision avoidance system of the previous chapter. This introduces a new problem. Because the speed is limited in the corner, it

might not be possible to accelerate to avoid a collision. If the collision avoidance system is not altered, the Ego vehicle might end up in a situation where it needs to slow down to not overshoot the corner and, at the same time, it needs to accelerate to avoid collision with the other road user.

To make sure that the vehicle never ends up in this situation, the collision avoidance assumes to be in the $\mathcal{C}_{u_C}$ capture slice, if the current velocity is higher than the maximum velocity v_{a} in the corner. This results in the capture set being equal to the capture slice from the u_B input, $\mathcal{C} = \mathcal{C}_{u_B}$ as is shown in figure 3.4.

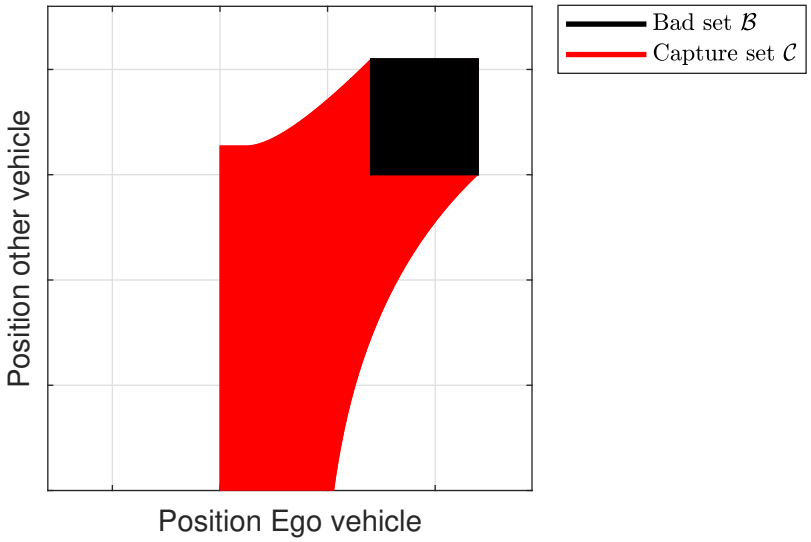

Figure 3.4: The capture set is equal to the braking capture slice, if the current velocity is higher than the maximum velocity in a turn

The second problem, that is introduced with the turning, is that the hybrid automaton of figure 2.4 is not completely valid anymore, because the current velocity might be higher than the maximum velocity at a later stage on the trajectory. To solve this problem, the maximum velocity in the collision avoidance system is made variable and set on the current velocity. The maximum velocity in the capture slice generation of section 2.2.1 depends on the current velocity and is the maximum velocity in the corner or the current velocity, if this is higher than the maximum cornering velocity. The adjustment to the collision avoidance becomes:

Algorithm 3 Adjustment to the collision avoidance system for safe turning at intersections

if $v > v_a$ **then**
  $v_{\text{max}} = v$
  $\mathcal{C} = \mathcal{C}_{u_B}$
else
  $v_{\text{max}} = v_a$
  $\mathcal{C} = \mathcal{C}_{u_B} \cap \mathcal{C}_{u_C}$
end if

Using the current velocity as a maximum velocity makes sure that when the trajectory hits the capture set, the Ego vehicle is still able to use the full braking distance to avoid the bad set. If the maximum cornering velocity was used, the capture set would have been smaller and full braking would lead the system to the bad set.

The result of these adjustments is that when the Ego vehicle approaches the intersection together with another vehicle, it will automatically slow down when the flow touches the capture set, because the capture set reaches over the total vertical plane in the position diagrams. As it is already assuming to be in the $\mathcal{C}_{u_C}$ set, the braking input u_B is applied at the border. When the velocity reaches below the maximum cornering velocity, the intersection between $\mathcal{C}_{u_C}$ and $\mathcal{C}_{u_B}$ becomes the capture set again. If the situation is that the flow is on the lower part of a graph, like the graph in figure 2.10, and moving towards the capture set, the Ego vehicle accelerates till the maximum speed in corner and the Ego

vehicle takes priority. If the flow comes from the left part of the graph, the Ego vehicle brakes and the trajectory follows the boundary of the $\mathcal{C}_{u_B}$ slice. The results is that the Ego vehicle gives the priority to the other vehicle.

4

Hardware & Software Implementation

This chapter gives an overview on the implementation of the controllers and the testing environment. The first part is about the software side and the models used for both vehicles. In the second part, the implementation of the controllers is discussed and an overview of the inputs and outputs is given. The third section gives an overview of the whole system in offline simulations and the chapter ends with the software and hardware used in the real-time implementation on the simulator.

4.1. Vehicle modelling

In this section, the modelling of the vehicles is discussed. The vehicles in the simulations are both modelled using a different approach. The Ego vehicle is being modelled with a high fidelity and consists of a full multibody model and several subsystems. The other road user, on the other hand, is being modeled by simple dynamics and kinematics. To verify the collision avoidance system, only the position and velocity of this vehicle is interesting. Therefore, it is not necessary to increase the fidelity of the model. The model has to act like a vehicle, but it does not have to be exact vehicle dynamics as this only introduces new problems. The collision avoidance system has to work with many types of vehicles and not one particular vehicle.

4.1.1. Simple car model

The simple car model of the other vehicle is split into two parts. The first part includes the longitudinal motion and second part the lateral motion. For the longitudinal motion, a simplified quarter-car model from section A.1 is used. It is assumed that there is no slip at the contact point between the tire and the road. The other assumption is that there is no drag working on the vehicle. These assumptions result in the following equation for the longitudinal acceleration of the vehicle.

$$\dot{v}_2 = a_2 u_2 \tag{4.1}$$

The high fidelity model was used as a reference for values of the parameters to make sure that dynamics of the simple car model were semi-realistic. The control applied was therefore limited $u_2 \in [u_{2,\min}, u_{2,\max}]$. The chosen values for the parameters in the simulations are shown in table 4.1.

Next to the longitudinal motion, the simple car model has to be able to move in the (x, y) plane. Therefore, the kinematic model of a bicycle or car-like robot is used. In this kinematic model, the nonholonomic constraints are taken into consideration, which prevent the vehicle from suddenly going sideways. The heading of the vehicle is connected to the steering wheel angle $\delta_{w,2}$ by:.

$$\dot{\psi}_2 = \frac{v_2}{l_{w,2}} \sin(\delta_{w,2}) \tag{4.2}$$

In which $l_{w,2}$ is the wheelbase. The change of location in the global coordinate system is calculated by making use of the heading angle and the longitudinal velocity.

$$\dot{y}_{2,g} = v_2 \sin(\psi_2 + \delta_{w,2}) \tag{4.3}$$

$$\dot{x}_{2,g} = v_2 \cos(\psi_2 + \delta_{w,2}) \tag{4.4}$$

Group	Symbol	Value
Simulation	a_2	$0.0017\ \mathrm{N^{-1}\,s^{-2}}$
	$u_{2,\min}$	$-1500\ \mathrm{N\,m}$
	$u_{2,\max}$	$1100\ \mathrm{N\,m}$
	$l_{w,2}$	$2.5\ \mathrm{m}$
Collision avoidance	β_A	$-1.45\ \mathrm{m\,s^{-2}}$
	β_B	$0.5\ \mathrm{m\,s^{-2}}$
	γ_A	$0.5\ \mathrm{m\,s^{-2}}$
	γ_B	$0.3\ \mathrm{m\,s^{-2}}$

Table 4.1: Parameters of the other vehicle in simulations. The first ones are used in the simulation of the dynamics, the second part is used in the collision avoidance system

The modelling and control of the simple car is done in Simulink. The control of the simple vehicle model is split into two parts. The longitudinal control is done by a predefined velocity plan and the lateral control uses a predefined trajectory in a pure pursuit controller from section 3.1.

The last part, that is needed for the collision avoidance system, is the typical accelerations and decelerations for the human driven case β_q combined with the disturbance $\bar{d} = 3$ times the standard deviation γ_q. These values are usually found from tests with a human driver and producing a probability density function from the results. For time limitations, however, it was chosen to make the boundaries of both Probability density function (PDF) such that they are outside of the maximum possible accelerations and decelerations and there is a small overlap below zero speed. It was reasoned that the other vehicle could be slowing down before the intersection to check if it was safe and still apply throttle when it is at the intersection. The PDF of the braking had as upper limit the zero velocity. As it makes no sense to first accelerate in the last meters before the intersection and than choose to apply the brakes. The resulting values are shown in table 4.1 and a plot of the PDF in figure 4.1.

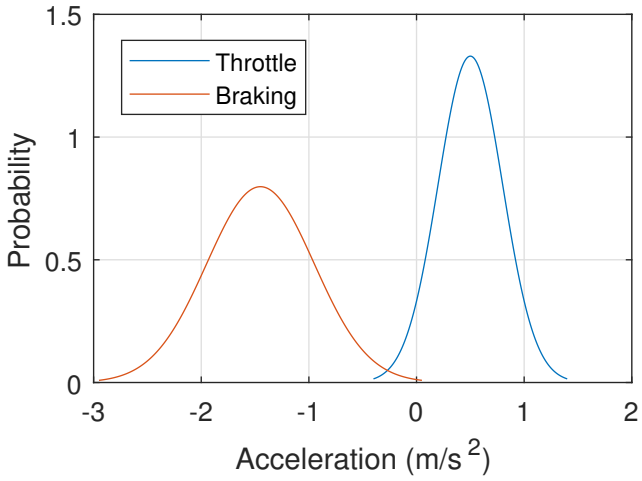

Figure 4.1: Probability density functions for the throttle and braking of the other human driven vehicle at intersections

4.1.2. High fidelity model

For the Ego vehicle, a high fidelity model is used. This high fidelity model simulates a real car more close than the simple car model. The high fidelity model consists of multiple parts and solvers. The main scheme, connecting all the separate parts, is a Simulink model. The vehicle dynamics are modelled by a 3D multibody model in LMS Virtual.Lab Motion. This multibody model is obtained from the LMS Driving Dynamics tool. An image of the multibody model is shown in figure 4.2.

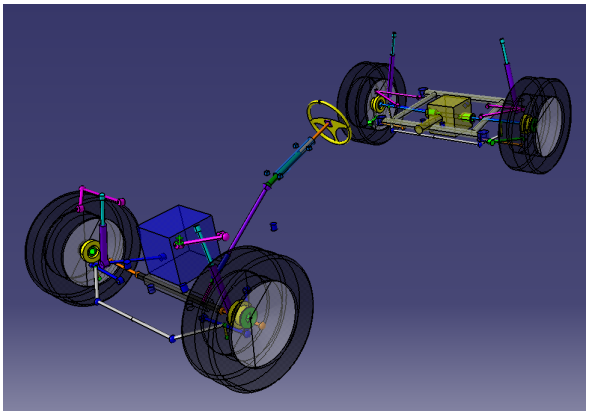

Figure 4.2: Multibody model of a rear wheel driven vehicle in LMS Virtual.Lab Motion

The model represents a typical rear wheel driven vehicle. In the multibody model and software, it is possible to use different models of the tires. For the simulations, a simple tire model with no rolling resistance was used to reduce the amount of tuning in the controllers. It is possible to increase the complexity of the model by introducing complex tire models or Pacejka magic tire models [19]. The aerodynamic forces were also discarded for the simulations. The solver used for the vehicle model is the C-code solver of Virtual.Lab. It uses Euler integration and a step size of $0.001\,\mathrm{s}$. This solver is also used in the offline testing to have similar testing conditions as when testing with the simulator. As the multibody model consists of many degrees of freedom, the model was split in a front part and rear part to solve each on a separate core. The parts are connected by a stiff connection.

Next to the multibody model, there are the internal subsystems of the vehicle in the high fidelity model. These systems are modelled in Simcenter Amesim and consist of three separate systems:

- Internal Combustion Engine (ICE) with powertrain
- Anti-Lock Braking System (ABS) and Electronic Stability Program (ESP)
- Electronic Power Steering (EPS)

The Amesim models are shown in figure 4.3 and are also exported to C-code for the simulations. For the Amesim models, there are two types of interfaces between Simulink and Amesim available. One type uses the the Simulink solver and the other makes use of a co-simulation between the Simulink solver for the total scheme and an Amesim solver for the Amesim models. For the simulations, the co-simulation interface was used, because it results in better and faster solutions. The models of the subsystems are solved by making use of Euler integration with a step size of $0.0001\,\mathrm{s}$.

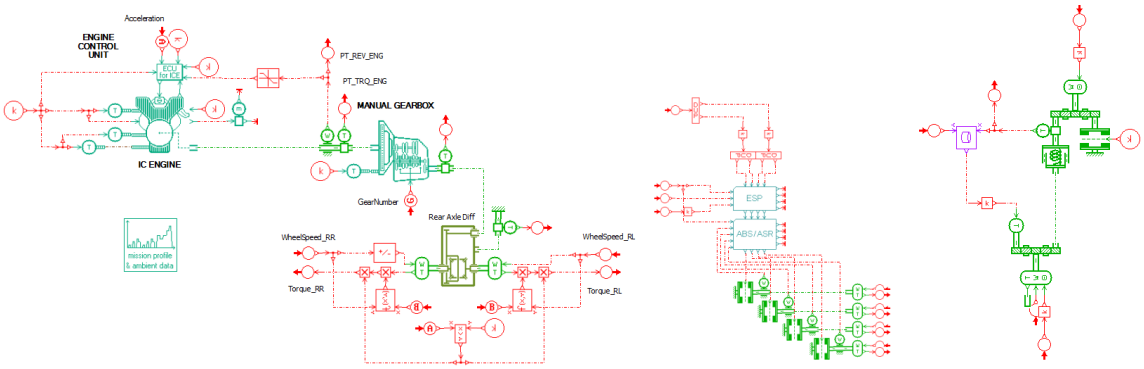

Figure 4.3: Simcenter Amesim models of the subsystems, from left to right: ICE with powertrain, ABS and ESP, and EPS

Because the powertrain of the vehicle includes a manual gearbox, an automatic gear shifter was implemented. This automatic gear shifter increases the gear, when the engine revolutions reach a

Algorithm 4 Automatic gear shifting algorithm

if $\dot{\omega}_{\text{RPM}} > 0$ and $\omega_{\text{RPM}} > \omega_{\text{th,high}}$ and $g < g_{\text{max}}$ **then**
  $g \uparrow$
else if $\dot{\omega}_{\text{RPM}} < 0$ and $\omega_{\text{RPM}} < \omega_{\text{th,low}}$ and $g > g_{\text{min}}$ **then**
  $g \downarrow$
end if

certain value. On the contrary, it reduces gears when the engine revolutions are reducing and reaching below a certain threshold. The algorithm used for the automatic gear shifter is given by algorithm 4.

In which ω_{RPM} is the engine speed and g is the gear number. The algorithm increases the gear when the engine speed is increasing, passing a maximum threshold $\omega_{\text{th,high}}$ and the gear is less than the final gear g_{max}. On the other side, if the engine speed is decreasing, becoming less than the lower threshold $\omega_{\text{th,low}}$ and the gear is higher than the neutral gear g_{min}, it decreases the gear number.

4.1.3. Parameter identification

The parameters of the collision avoidance system were found by perfroming straight line tests with the vehicle model. The tests consisted of full braking and full throttle maneuvers. From these tests, the coefficients a_1, b_1 and c_1 were found to use in the hybrid state model of figure 2.4 for colission avoidance. There are no aerodynamic forces acting on the vehicle, which resulted in $c_1 = 0$ and the rolling resistance was also set to zero $b_1 = 0$. To find the value for a_1, the straight line tests with known control torques were used.

From the braking test, it was possible to find the maximum braking torque $u_{1,\text{min}}$. From the throttle test, the maximum torque $u_{1,\text{max}}$ by the ICE and powertrain was found. Because the torques depend on the velocity of the vehicle, the values were chosen in a conservative way. This means that the maximum torques used in the collision avoidance system were lower than the actual maximum torques. Using these values the capture set becomes larger, but a collision easier to avoided.

For the pure pursuit controller, it was needed to find the relationship between the angle on the steering wheel δ_s and the steering angle δ_w of the wheels. The relationship was found by applying a constant steering wheel angle and constant longitudinal velocity. Using this input, the vehicle makes a turn with a constant radius. If this is not be the case, the vehicle is understeering or oversteering and an extra correction has to be made in the lateral controller. The look-ahead distance coefficient was tuned by tracking trajectories that included turns on intersections. The aimed velocities were $25\,\text{km}\,\text{h}^{-1}$ for the left turn and $15\,\text{km}\,\text{h}^{-1}$ for the right turn. To prevent the look-ahead distance to go from zero to infinity, it was saturated with a minimum set on $l_{a,\text{min}} = 4\,\text{m}$ and the maximum $l_{a,\text{max}} = 12\,\text{m}$.

An overview of all the parameters is given in table 4.2.

Group	Symbol	Value
Collision avoidance	a_1	$0.0017\,\text{N}^{-1}\,\text{s}^{-2}$
	$u_{1,\text{min}}$	$-1500\,\text{N m}$
	$u_{1,\text{max}}$	$1100\,\text{N m}$
Lateral controller	I_s	-6
	c_a	1.1
	$l_{a,\text{max}}$	12 m
	$l_{a,\text{min}}$	4 m

Table 4.2: Tuned parameters of the Ego vehicle in the longitudinal and lateral controller

4.2. Implementation of the controllers

The two controllers, longitudinal and lateral, are developed in a combination of MATLAB and Simulink. The algorithms are written in MATLAB. The cruise control and hierarchy in longitudinal control as well as the connections to the other components of the vehicle are done in Simulink.

4.2.1. Longitudinal controller

The longitudinal control of the Ego vehicle consists of two parts. One part is the collision avoidance system from chapter 2 and the other part is a cruise control. The cruise control is trying to keep the vehicle at a desired velocity $v_{1,\text{desired}}$, when there is no input from the collision avoidance system. This cruise control works by using a PID controller on the error between the desired velocity and the current velocity.

The second part is the collision avoidance system. The dynamic inputs of this system are the longitudinal positions and velocities of both vehicles. The longitudinal acceleration of the other road user is used for the mode estimator. These inputs are obtained, for the Ego vehicle, from the multibody model and, for the other vehicle, from the simple dynamics model. Next to the dynamic inputs, the system requires information about the upcoming intersection. This information consists of the lower and upper boundaries of the bad set and the maximum velocity allowed for the vehicles. The trajectory information inputs are obtained from an external global planner. The final inputs, necessary to make the system work, depend on the use-case. In case the cruise control is used, a desired velocity is required. In case there is a human driver, the throttle and brake from the pedals are used. An overview of all the inputs is given in table 4.3.

The output of the system is the throttle and brake to the subsystems. For the throttle and brake, there are three systems that deliver an output. The final output is determined by the hierarchy of the systems:

1. Collision avoidance system
2. Driver
3. Cruise control

Group	Parameter	Symbol	Vehicle
Dynamic	Longitudinal position	p_i	Both
	Longitudinal velocity	v_i	Both
	Longitudinal acceleration	$\dot{v}_2$	Other
Intersection information	Lower boundary	L_i	Both
	Upper boundary	U_i	Both
	Maximum velocities	$v_{i,\text{max}}$	Both
Cruise control	Desired velocity	$v_{1,\text{desired}}$	Ego
Human-in-the-Loop	Throttle driver		Ego
	Brake driver		Ego

Table 4.3: Inputs of the collision avoidance controller

The last parameter for the longitudinal controller is the decision point of the mode estimator. The location was chosen to be at 10 meters before the lower boundary of the bad set. It was assumed that if the human driver applies throttle or brake between this decision point and the upper boundary of the bad set, it will not completely switch the input again. It is possible for the driver to release or change the amount of input, but not switch from brakes to throttle or throttle to brakes.

4.2.2. Lateral controller

The lateral control makes use of a trajectory, consisting of a list with (x, y) coordinates. This trajectory can come from a global planner, which is a planner that finds the best route from a starting location to a destination, or be predefined. The other inputs of the lateral controller are the states of the Ego vehicle, the location in global coordinates, heading and the longitudinal velocity. An overview of all the inputs is given in table 4.4 and the only output is the steering wheel angle δ_s.

Group	Parameter	Symbol	Vehicle
Trajectory	Coordinates	$(x_{w,g}, y_{w,g})$	Ego
Vehicle	Coordinates	$(x_{r,g}, y_{r,g})$	Ego
	Heading	ψ_1	Ego
	Velocity	v_1	Ego

Table 4.4: Inputs of the lateral motion controller

4.3. Offline implementation

In the offline implementation, the components of the previous subsection are all combined and solved together. In these simulations, the inputs are predetermined and there is no deadline for the simulation to solve. To make the offline simulations as similar as the real-time simulations, the same type of solvers are used for each model as are used in the real-time application. Therefore, every component is using a fixed-step solver with Euler integration. The Amesim and Virtual.Lab models are also used in combination with their C-code solvers to make the differences between real-time and offline as small as possible.

An overview of the connection between all the components is shown in figure 4.4. This figure shows that the other vehicle is acting completely independent from the Ego vehicle. There is no information shared from the Ego vehicle to the other and, therefore, the other vehicle does not respond to any of the Ego vehicle's actions.

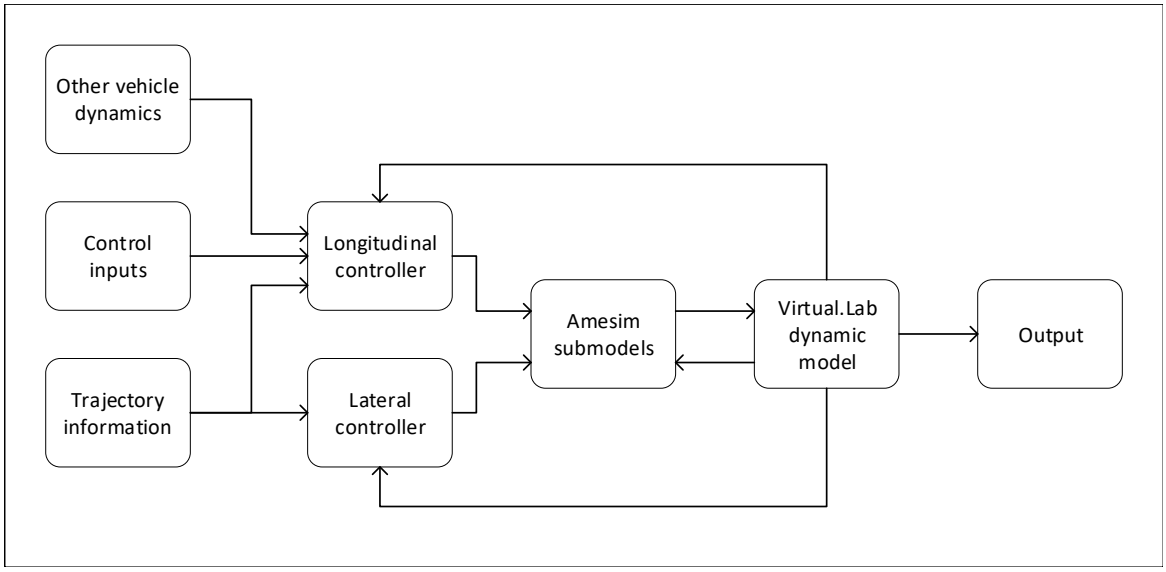

Figure 4.4: Overview of the system for offline simulations

Both controllers are sending their outputs, the control settings throttle, brake and steering angle, to the Amesim submodels. These submodels give the torques as output to the Virtual.Lab model. This Virtual.Lab model returns the location and velocity of the vehicle to the controllers and the angular velocities of the wheels and steering torques to the Amesim models .

For the offline testing, a desired velocity in combination with the cruise control was used as input of the longitudinal controller. The trajectory information to the longitudinal controller included the location of the bad set and the maximum allowed velocity. The trajectory coordinates were sent to the lateral controller.

The last block shows the output, which contains the locations and orientations of both vehicles. This output is used to place the vehicles in a virtual environment. The virtual environment is generated in Simcenter PreScan. For the offline testing, an environment was generated with two straight roads forming an intersection and the trajectories were exported for tracking in the lateral controller. The Simulink scheme, with the simulation of the vehicles, was making use of a local User Datagram Protocol

(UDP) signal to send the location data to a separate Simulink scheme with the PreScan environment. A direct link between the dynamics and the environment was not possible, because the solvers for the dynamics required a different version of MATLAB than the version that was required by PreScan.

4.4. Real-time implementation

Before implementing the controllers in real vehicles, the controllers have to be tested in real-time. In offline testing, it does not matter how long simulations take to solve, because the results are investigated afterwards. In real-time testing, the system has to be guaranteed solvable in the used step size for integration. To reduce the stress on the system, the total system is split into two parts. The first part has to be solved hard real-time, which means that it is not allowed to miss one time step. The components in this part are the controllers, the vehicle dynamics and the subsystems. The second part is solved in soft real-time, which means that missing some of the deadlines for integration are not crucial. This includes the visualization and the supplying of the trajectory information.

4.4.1. Hard real-time

For the hard real-time computing, a system from Concurrent is used. This computer guarantees that one second in simulation is one second in real life and guarantees a deterministic solution of the system. This system is running a special operating system, RedHawk Linux, for this purpose. For the real-time application, the entire system has to be coded in C. As the Virtual.Lab and Amesim models are already exported to C, only the MATLAB and Simulink have to be converted. For this conversion, SIMulation WorkBench was used in combination with the Simulink Coder. This combination converts the models automatically, when they are uploaded to the Concurrent machine. The systems running hard real-time are:

- Multibody vehicle model
- Internal subsystems (ICE & powertrain, ABS & ESP, EPS)
- Longitudinal controller
- Lateral controller
- Dynamics and kinematics of the other vehicle

Because the Concurrent system has multiple cores, it was possible to run each component on a separate core. By splitting the model, the controllers were running parallel to the simulation of the vehicles. Communication between the components was done by making use of a Real-Time Database (RTDB). This RTDB is also used for communicating with the soft real-time system. The other system changes values in the RTDB by sending UDP signals to the Concurrent machine.

4.4.2. Soft real-time

The soft real-time part runs on a separate computer and consists of several components,that also need to run real-time, but if sometimes an integration step would be missed, the system does not have to fail. The main components in the soft real-time part are the inputs, like trajectory information, and outputs, like visualization, of the dynamic system. The soft real-time part consists of the following parts:

- Virtual environment with visualization
- Trajectory information
- Driver information display
- Driver input from the pedals
- Cruise control settings

To give the driver a more realistic experience with the intersection controller, rFactor Pro in the simulator was replaced by an environment model generated in Simcenter PreScan. In PreScan, an

urban-like environment model is created by connecting several roads and by placing actors, that represent the road users. Furthermore, decoration like buildings, houses and road signs are added to give the driver a realistic view in the virtual environment. This results in an experience close to driving an actual vehicle on real roads. The created environment model consists of two intersections in a row, which each had another road user approaching. The environment model is shown in figure 4.5.

The PreScan software is running on a different computer than the vehicle models, because it is not necessary to run the visualization hard real-time. This computer communicates with the Concurrent by means of UDP signals. The UDP signals, from the Concurrent to the soft real-time, consist of the coordinates and orientations of both vehicles.

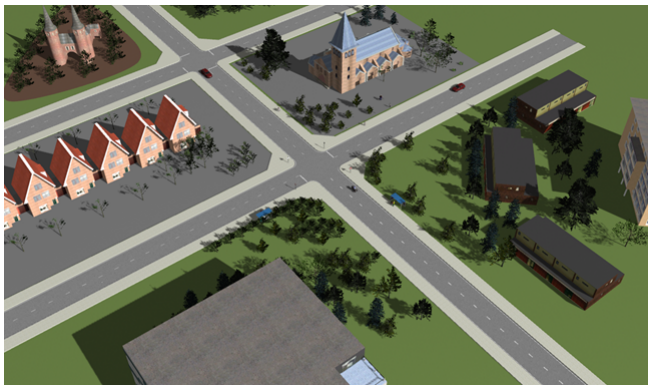

Figure 4.5: Environment model in Simcenter PreScan for online simulations

To reach from an initial starting point to a destination, usually a global planner is used. This planner determines the shortest or best route for the vehicle. There are many methods available for finding a trajectory, methods like Dijkstra's [6], A* [10] or one of the D* algorithms [23] [13]. For the simulations and tests, the global planning was not implemented. The environment model was not detailed enough to make this necessary. When there are multiple routes to a certain destination, this extra planner has to be used. In this case, however, there are only a few routes in this environment model. Therefore, all routes were given as an option to the driver before the test started. The trajectories were found by exporting multiple $(x_{w,g}, y_{w,g})$ coordinates of the road data from the environment model. Using a UDP signal, it was determined if the other vehicles go straight, left or right at the intersections. Based on the location of the Ego vehicle, it was determined if it was approaching the first of the second intersection. If the Ego vehicle was approaching the first intersection, the locations of the bad set of this intersection and the kinematics of the actor, that was approaching this intersection, were communicated to the longitudinal controller. When the Ego vehicle passes this intersection, the details of the second intersection and second actor were send to the longitudinal controller.

4.4.3. Human machine interface

The first part of the human machine interface are the controls. In the tests with the human machine interface, the driver is able change the velocity of the vehicle. When the vehicle is safe and the collision avoidance system does not need to take over, the driver uses the pedals to apply throttle and brakes. Because the system is making use of predefined trajectories the steering wheel is not activated. The throttle and brake pedals are connected to the soft real-time computer and for both inputs signals between 0-100% are sent to the longitudinal controller.

As the visualization only shows what is going on outside of the vehicle, a separate display was developed to inform the human driver about the velocity of the Ego vehicle. The information display is shown in figure 4.6 and also includes the engine RPM for the human driver. When the automatic gear shifter is not used, the driver can use the RPM for deciding to switch gears manually.

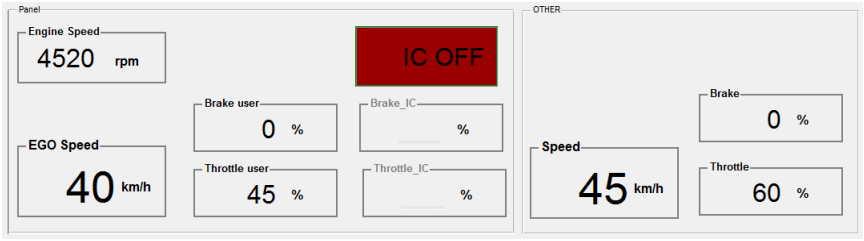

Figure 4.6: Information display for the human driver with the intersection controller not interfering

Next to the engine RPM and the Ego velocity, the control inputs by the driver and the inputs by the intersection controller are shown as a percentage reaching from $0 - 100\%$. Above the inputs of the controller, there is a block that indicates if the intersection controller is taking over control. In case the block is red and shows ”IC OFF”, the driver is in full control. On the other hand if the block gets green and shows ”IC ON”, as is shown in figure 4.7, the intersection controller takes over and the driver can not apply any inputs. The right part of the display shows the velocity and control inputs on the other vehicle.

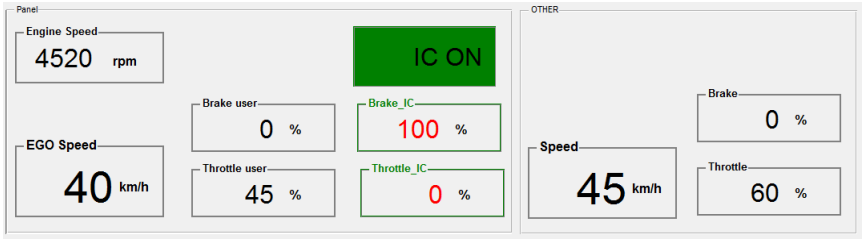

Figure 4.7: Information display for the human driver with the intersection controller taking over control

The last part of the human machine interface is the motion platform. The motion platform, that is being used, is a 6 degree of freedom Stewart platform [24], like is shown in figure 4.8. This platform reproduces the accelerations, that the driver would feel in a real car. For the tests, a MOOG controller is used with its internal motion cuing algorithms. The accelerations, that were reproduced, are obtained from the multibody model of the vehicle.

Figure 4.8: A 6 DOF Stewart platform for reproducing the accelerations

5

Results from the Simulations

This chapter demonstrates the results obtained in simulations. In the simulations, two vehicles are approaching the intersection and the result is a collision, if no additional control is being applied. The chapter is split in three main parts. The first one is considering the case with two autonomous vehicles. The second part is about the Ego as autonomous vehicle and the other vehicle as human driven. This human driven vehicle is assumed to apply control, throttle or brake without taking the position and speed of the Ego vehicle into consideration. All of the simulations are done offline and without the real-time requirement. In the last part, the system is tested on a real-time application. In these tests, a Human-in-the-Loop (HITL) interface is included.

In all the simulations, the lower boundaries of the bad set L_i for both vehicles i are set to longitudinal position $p_i = 0\,\mathrm{m}$. In the position diagrams, the bad set $\mathcal{B}$ is represented by a black area and the capture set $\mathcal{C}$ by a red area. The two capture slices $\mathcal{C}_{u_B}$ and $\mathcal{C}_{u_C}$ are given by a green and blue area respectively. The Ego vehicle is modelled by the high fidelity vehicle model and visualized by the red car in the screenshots of the simulation. The other vehicle is modelled by the simple car dynamics and visualized by the black car.

5.1. Autonomous - Autonomous case

In this section, both vehicles are autonomous. Together, the vehicles decide which vehicle is going to apply throttle and which one the brakes to avoid the collision. In the simulations, both vehicles are only moving in a straight line at the intersection without steering and the collision avoidance system of section 2.1 is used. The time step in the future state estimator $\Delta T_f = 0.1\,\mathrm{s}$ and a maximum of two steps ($j_{\max} = 2$) is used. The reachability analysis in the capture slice generation is done with a stepsize $\Delta T = 0.01\,\mathrm{s}$. The resulting trajectories of the simulations are shown in figure 5.1.

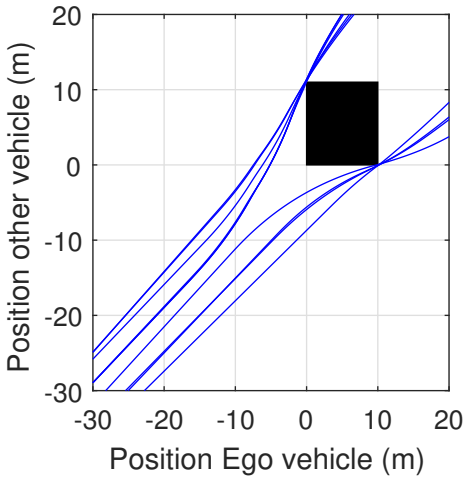

Figure 5.1: Resulting trajectories of the autonomous-autonomous case in the longitudinal positions diagram

In this figure, only the trajectories, in which the collision avoidance was taking over control, are included and it is shown that the trajectories are approaching the bad set, but do not enter it. Part of the trajectories are bend up vertical, which means that the other vehicle is accelerating, while the Ego vehicle is braking. The other part of the trajectories are bend more horizontal, which means the opposite.

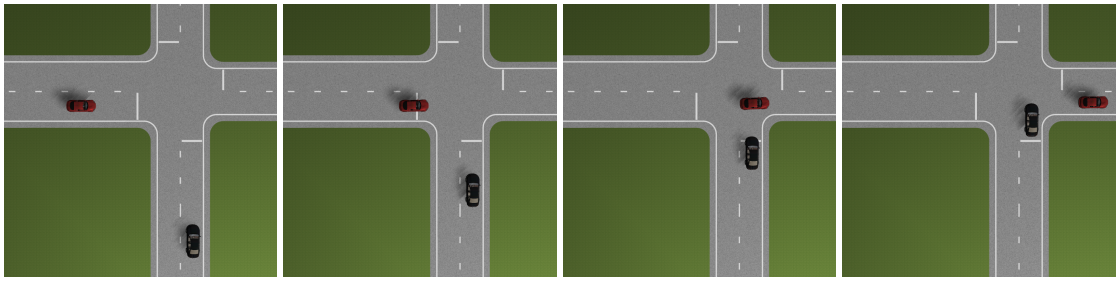

Figure 5.2: Visualization of the autonomous-autonomous case with the Ego vehicle taking priority $t = 3 - 6\,\mathrm{s}$

Figure 5.2 shows the results from the first simulation with the Ego vehicle taking priority. The velocity profile and controls of the first simulation are shown in figure 5.3. The figure shows that both vehicles are approaching the intersection at the same velocity. After approximately $t = 3.7\,\mathrm{s}$, control is applied and one vehicle accelerates, while the other decelerates. Which vehicle accelerates and which decelerates is determined in the position diagrams given in figure 5.4. This figure shows that the trajectory of the vehicles moves through the green capture slice towards the red capture set. Therefore, the control of the blue capture slice, throttle for the Ego and braking for the other, is applied. Using this control the bad set is avoided and there is no collision between the vehicles. When the trajectory passes the bad set, the velocities are returning to the initial desired velocities.

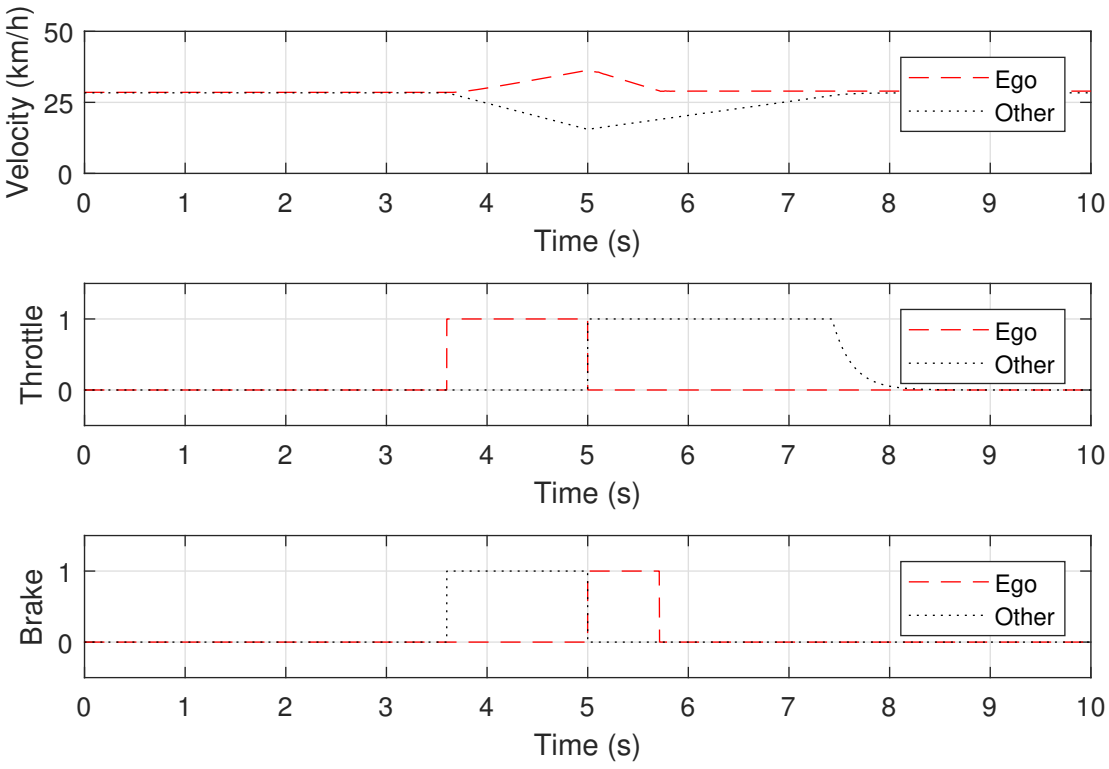

Figure 5.3: Velocity profiles and control inputs of the scenario with the Ego vehicle taking priority

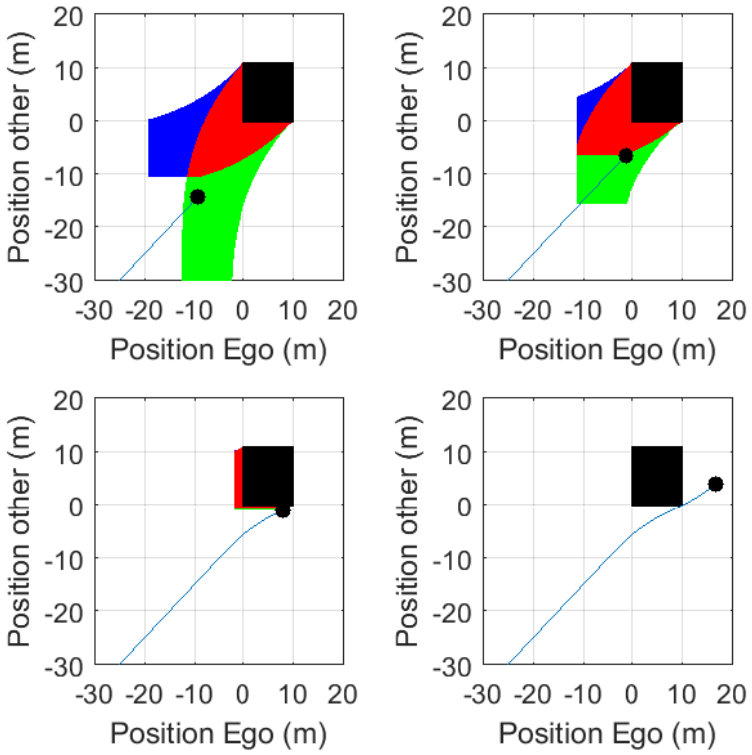

Figure 5.4: Position diagrams and capture set progression between $t = 2 - 6\,\mathrm{s}$ of the autonomous-autonomous case with the Ego vehicle taking priority

In the second simulation, the initial conditions of the vehicles are changed. This results in having the other vehicle taking priority over the Ego vehicle, which is shown in figure 5.5. In this simulation, the Ego vehicle brakes to avoid the collision, while the other vehicle accelerates.

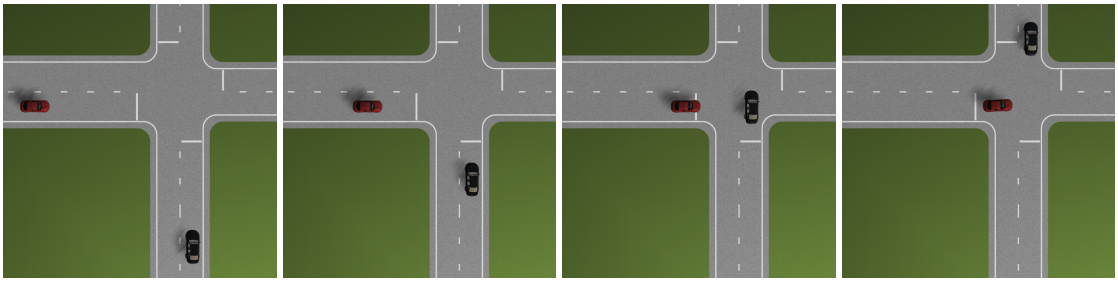

Figure 5.5: Visualization of the autonomous-autonomous case with the other vehicle taking priority $t = 3 - 6\,\mathrm{s}$

Figure 5.6 shows the velocities and control inputs from this simulation. In this figure, it is shown that both vehicles are approaching the intersection at a similar velocity. Around $t = 3\,\mathrm{s}$, the Ego vehicle starts to slow down and the other vehicle starts accelerating. Figure 5.7 shows the position diagrams and the capture set over time of this simulation. The figure shows that in this scenario the trajectory is approaching the capture set from the blue throttle slice. Therefore, the control of the green slice is applied, which results in having the Ego vehicle braking and the other vehicle accelerating in order to avoid the bad set. In this simulation, the velocity profiles are less continuous than in the previous simulation, because the flow is bouncing on and off the capture set. When the flow is starting to move away from the capture set, the collision avoidance system stops working and the vehicles are trying to return to their desired velocity. When they return to their desired velocity, the trajectory bends back towards the capture set and the same thing happens again.

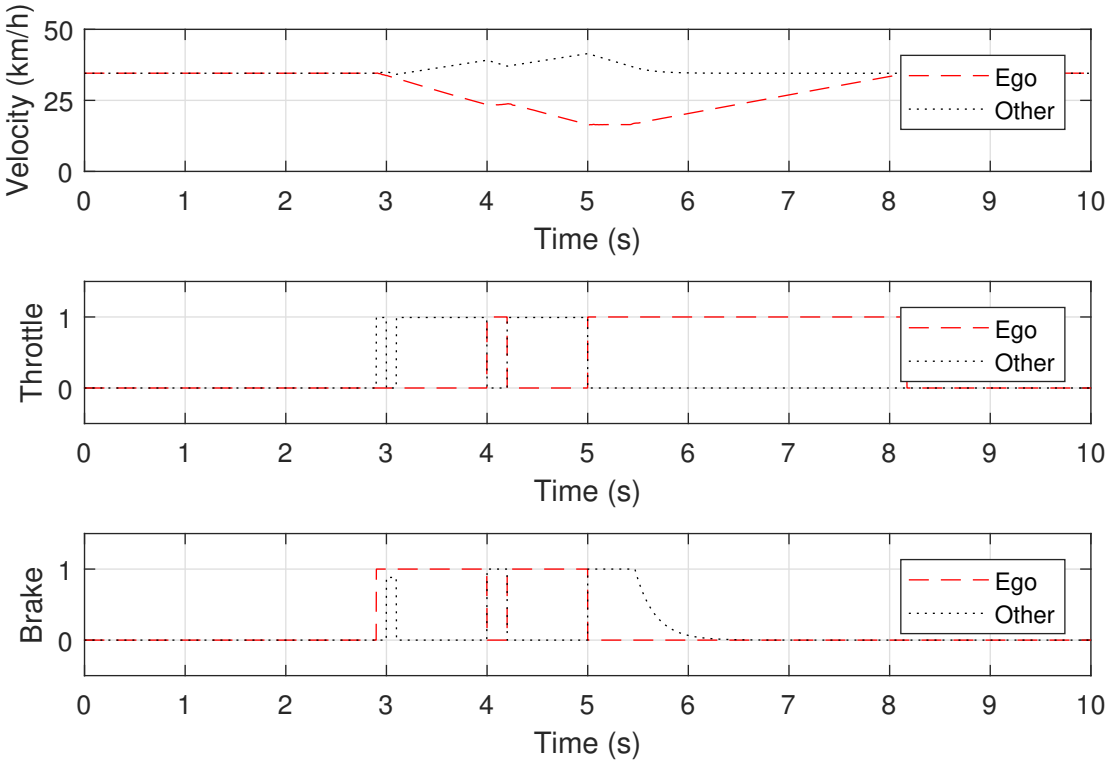

Figure 5.6: Velocity profiles and control inputs of the scenario with the other vehicle taking priority

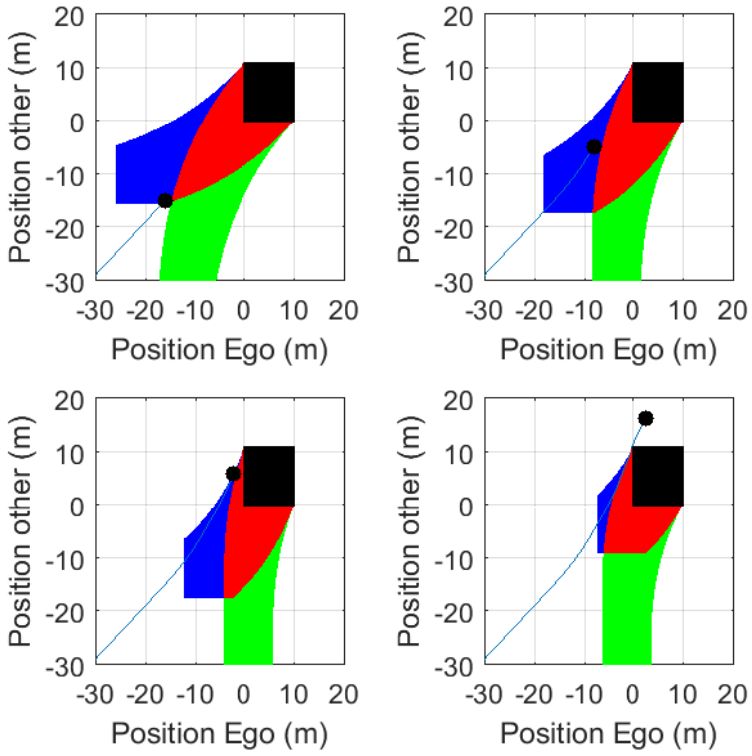

Figure 5.7: Position diagrams and capture set progression from $t = 2 - 6\,\mathrm{s}$ of the autonomous-autonomous case with the other vehicle taking priority

5.2. Autonomous - Human driven case

The next step is to remove the knowledge about the input of the other vehicle. Therefore, the collision avoidance system of section 2.2 is used. The amount of steps in the future state estimator was increased to $j_{max} = 4$. From simulations, it was found that if the amount of steps was lower, the trajectory sometimes ended up inside the borders of the bad set. There are a few explanations for this necessary increase of steps. The control on the trajectory of the system has decreased, because in the previous case control was applied on both vehicles, it was easier to control the direction of the trajectory. Another reason is that the capture slices are discrete and there is a chance that the trajectory ends up just between two blocks in an integration step. The final reason is that the collision avoidance system runs on $10\,\mathrm{Hz}$, which is the same step size as the one of the future state estimator. These three reasons combined made the trajectory sometimes end up in the bad set and with the increase of the amount of steps in the future state estimator, this problem was solved.

The simulations were split in a few groups. The first part only considers both vehicles going in a straight line. The second part also includes the turning at the intersection and the adjustment to the collision avoidance system discussed in section 3.2 was implemented. Finally, there are two other scenarios discussed in which the collision avoidance system can be used.

5.2.1. Going straight

In the first part, both vehicles are going straight at the intersection. The maximum velocity of both cars is assumed to be $50\,\mathrm{km\,h^{-1}}$. Figure 5.8 shows the resulting trajectories obtained in the simulations.

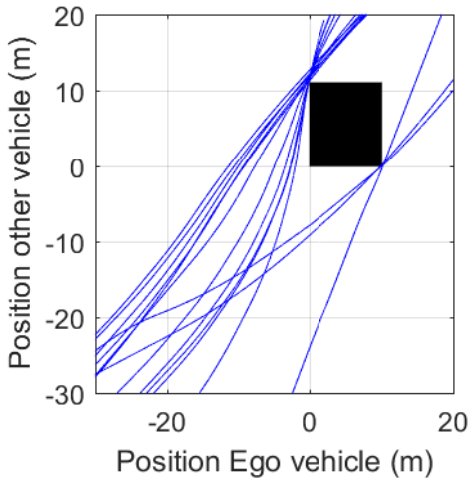

Figure 5.8: Resulting trajectories of the autonomous-human driven case with both cars going straight at the intersection

This figure shows that the trajectories do not end up in the bad set and it also shows that many of these trajectories are causing the Ego vehicle to give priority. This is shown by the fact that the trajectories are touching the bad set in the upper left corner. The first reason is that the human vehicle is not taking the actions of the Ego vehicle into consideration. The second reason is that only the trajectories that touch the capture set are shown and the chance of hitting the capture set from the throttle capture slice is much higher than the chance of touching it from the braking capture slice. For the Ego vehicle to take priority combined with the touching of the capture set, the flow has to move vertically in the position diagram. This can happen by having a high velocity of the other road user compared to the velocity of the Ego vehicle. Another option is having the other vehicle suddenly accelerating when the flow is below the capture set. The Ego vehicle can also force it by slowing down at the intersection. Because the Ego vehicle is crossing the intersection in a straight line with the cruise control at a fixed speed, the vehicle is not slowing down at the intersection unless it has to for collision avoidance.

Two trajectories of figure 5.8 are discussed in more detail. In the first trajectory, the other road user keeps its velocity constant and takes priority. The Ego vehicle recognizes that there is a chance of collision and brakes to avoid it. Figure 5.9 shows the two vehicles approaching the intersection.

Both vehicles are reaching the intersection at the same time. To avoid the collision, the Ego vehicle is slowing down, while the other vehicle is entering the intersection. After the other vehicle has left the intersection, the Ego vehicle enters.

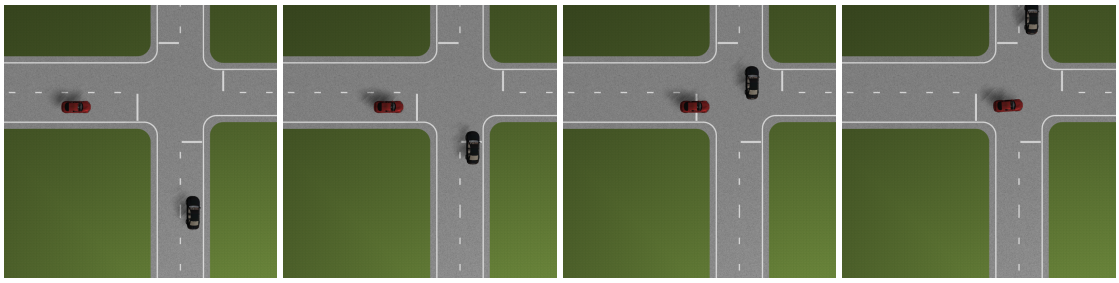

Figure 5.9: Visualization of the autonomous-human driven case with the other vehicle taking priority between $t = 4 - 7\,\mathrm{s}$

The velocities and inputs of both vehicles are shown in figure 5.10. The figures show that the human driven vehicle is not applying any control and is keeping its velocity constant, while the Ego vehicle is braking and slowing down. After 6 seconds, the Ego vehicle starts accelerating, because the road is clear and there is no collision possible anymore.

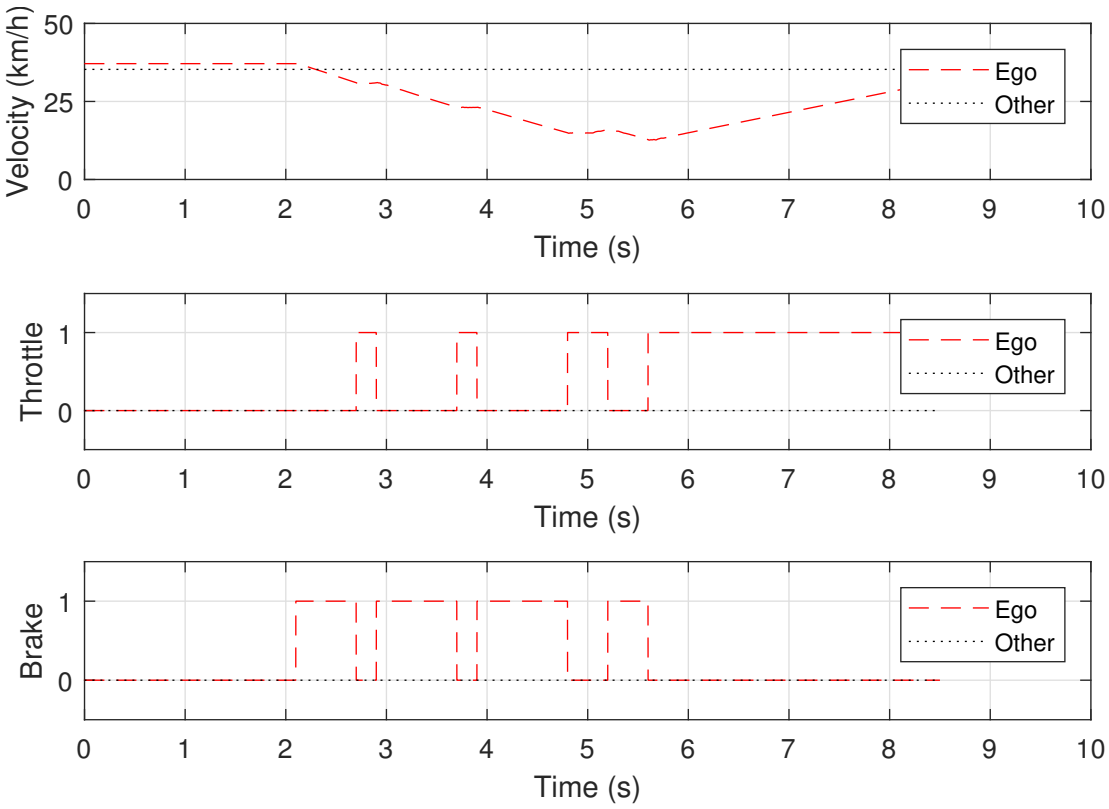

Figure 5.10: Velocity profiles and control inputs of the scenario with the human driven vehicle taking priority

The capture sets with trajectories over time of this simulation are shown in figure 5.11 and show that both vehicles are approaching the bad set. At $t = 3\,\mathrm{s}$, the Ego vehicle is $1\,\mathrm{m}$ closer to the bad set than the other and the velocity diagram shows that the Ego vehicle was driving $2\,\mathrm{km\,h^{-1}}$ faster. However, the Ego vehicle does not know what the other vehicle will do and still considers that it might start accelerating. Therefore, it slows down to make sure that there will be no collision. In the capture set diagrams, this is shown by having the flow coming from the throttle capture slice and moving towards the capture set. Therefore, the Ego vehicle applies the brakes to stay out of the capture set.

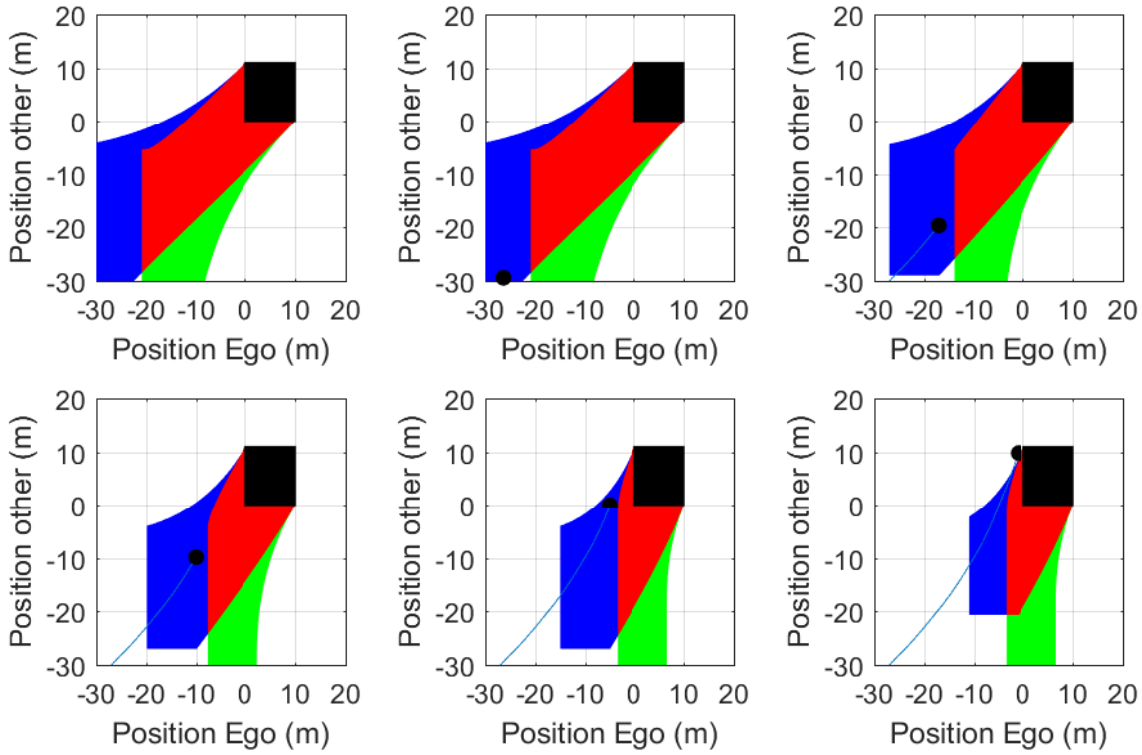

Figure 5.11: Position diagrams and capture sets over time of the scenario with the human driven vehicle keeping a constant velocity and taking priority $t = 2 - 7\,\mathrm{s}$

In the second scenario from figure 5.8, the other road user slows down in front of the intersection to give priority to the Ego vehicle. However, when the Ego vehicle decides to enter, the other vehicle tries to force a collision. Figure 5.12 shows both vehicles at the intersection and it shows that the Ego vehicle enters the intersection first.

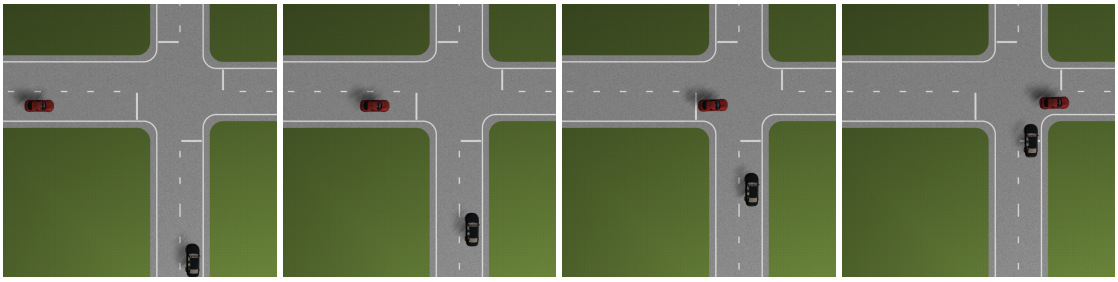

Figure 5.12: Visualization of the autonomous-human driven case with the Ego vehicle taking priority between $t = 3 - 6\,\mathrm{s}$

The velocities and control inputs of this scenario are shown in figure 5.13. This figure shows that at $t = 3\,\mathrm{s}$, the Ego vehicle applies the throttle to take priority and make sure that there is no chance of a collision. As the other vehicle is accelerating, it has to apply throttle to stay safe. At $t = 3.7\,\mathrm{s}$, the Ego vehicle releases the throttle and applies brakes, due to the desired velocity in combination with the cruise control, which is set lower than the current velocity. With the velocities of both vehicles at this stage, there is no chance of collision. Because the velocity is now decreasing, The trajectory is approaching the capture set again. At $t = 3.9\,\mathrm{s}$, the trajectory is touching the capture set and the Ego vehicle is applying throttle another time to avoid the collision.

The development of the capture sets over time is shown in figure 5.14. In these graphs, it is shown that the trajectories are first entering the braking capture slice and than move towards the capture set. This makes the Ego vehicle to apply throttle and, in the end, avoid the bad set. The figure also shows that the trajectory is moving towards the capture set and is running perpendicular between $t = 3 - 4\,\mathrm{s}$. This is the period in which no control was needed to avoid the collision.

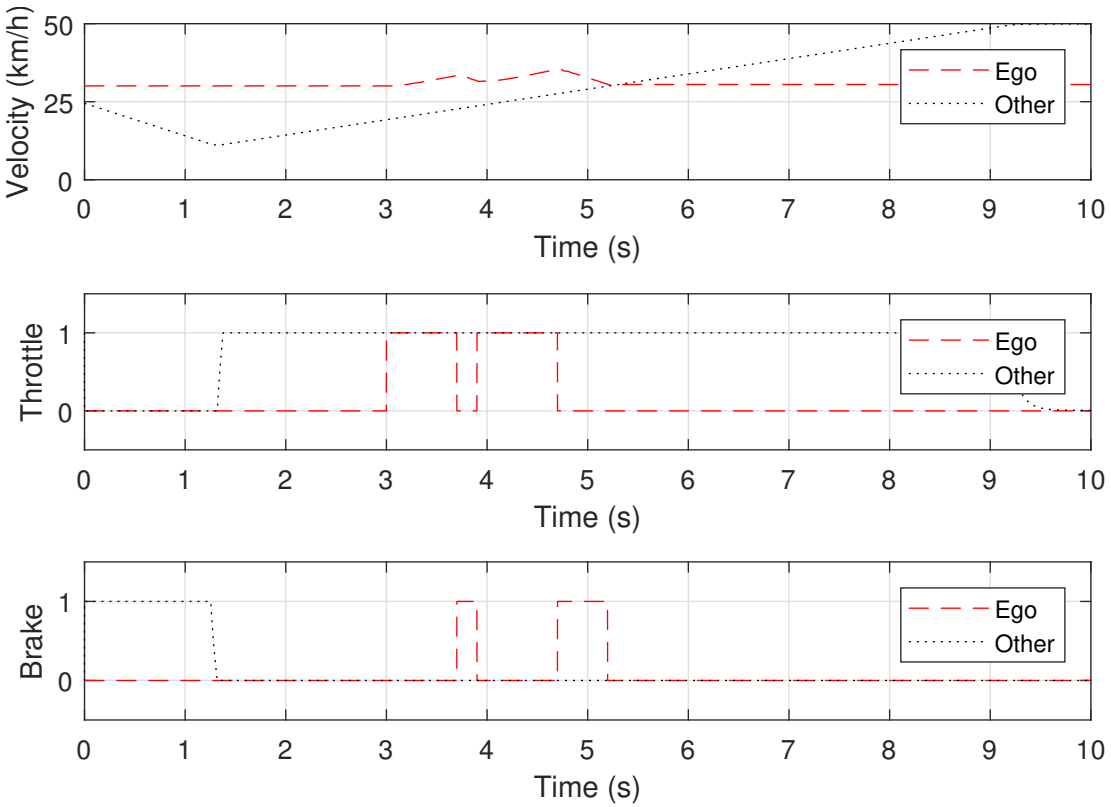

Figure 5.13: Velocity profiles and control inputs of the scenario with the Ego vehicle taking priority and the human driven trying to force a collision

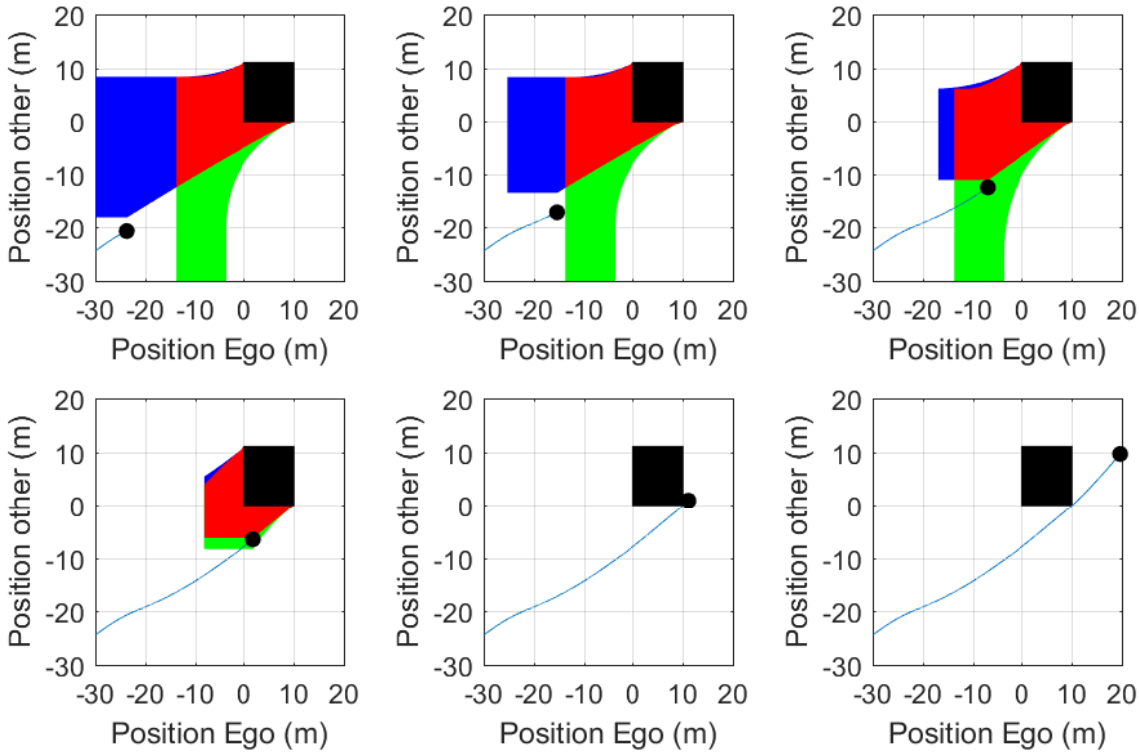

Figure 5.14: Position diagrams and capture sets over time of the scenario with the Ego vehicle taking priority and the human vehicle trying to force a collision, $t = 2 - 7\,\mathrm{s}$

5.2.2. Turning at the intersection

In this part, the Ego vehicle starts making turns at the intersection. This extra action introduces a speed limit for the vehicle around the boundaries L_1 and U_1. The adjustment to the collision avoidance from section 3.2 is implemented to make sure that the system avoids accidents. In the following simulations, the Ego vehicle is turning left and the other vehicle is going straight. This means that the maximum longitudinal velocity of the Ego vehicle is $v_{1,\max} = 25\,\mathrm{km\,h^{-1}}$ and of the other $v_{2,\max} = 50\,\mathrm{km\,h^{-1}}$. Resulting longitudinal displacement trajectories of the simulations are shown in figure 5.15. Simulations with the Ego going right are showing similar results.

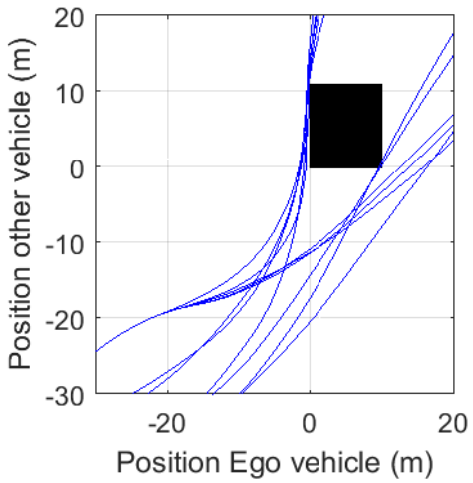

Figure 5.15: Resulting trajectories of the autonomous-human driven case with the autonomous car turning left at the intersection

The figure shows that in these simulations the trajectory passes the lower right corner, which means that the collision avoidance applied throttle, more often than in the scenario with both cars going straight. As the Ego vehicle is slowing down before it reaches the intersection, the flow moves vertical in the position diagrams. Therefore, the flow reaches the capture set from the braking capture slice, resulting in a throttle control output. The trajectories, in which the Ego has to brake, are almost aligned perpendicular to the vertical border of the bad set. This means that the Ego vehicle has braked to almost zero velocity. This makes sense, because the braking capture slice is used as capture set when the velocity of the Ego vehicle is above the maximum velocity in the corner. The distance between the left border of this capture slice and the left border of the bad set is the braking distance of the Ego vehicle. If the Ego vehicle is not slowed down before it touches the capture set and it has to give priority to the other, it needs all the braking distance and ends up at zero velocity to avoid a collision.

Both scenarios, one with the Ego vehicle taking priority and the other with giving priority, are discussed in more detail. In the first simulation, the Ego vehicle approaches the intersection at a higher velocity than the allowable velocity in the corner and it gives the priority to the other vehicle. Figure 5.16 shows the visualization of this simulation and shows that the Ego vehicle reaches the intersection first, but brakes to an almost standstill to make the other vehicle safely cross the intersection. When the other vehicle has passed the intersection, the Ego vehicle starts accelerating and makes a turn to the left.

The velocity profiles and control inputs of both vehicles are shown in figure 5.17. The longitudinal position diagrams with the capture sets are given in figure 5.18. In these figures, it is shown that, in the first seconds, the velocity of the Ego vehicle is higher than the maximum velocity at the intersection. Which is concluded from having the capture set equal to the braking capture slice. Therefore, the Ego vehicle slows down at the moment the flow touches the capture set. After approximately $t = 3.5\,\mathrm{s}$, the current velocity becomes less than the maximum velocity and the brakes are released. However, at $t = 4\,\mathrm{s}$, the flow is approaching the capture set from the throttle capture slice and makes the Ego vehicle applying the brakes again. The flow stays on this side of the capture set and the vehicle gives priority to the other car. When the other car has passed the upper limit U_2, the Ego vehicle accelerates as the intersection is now free.

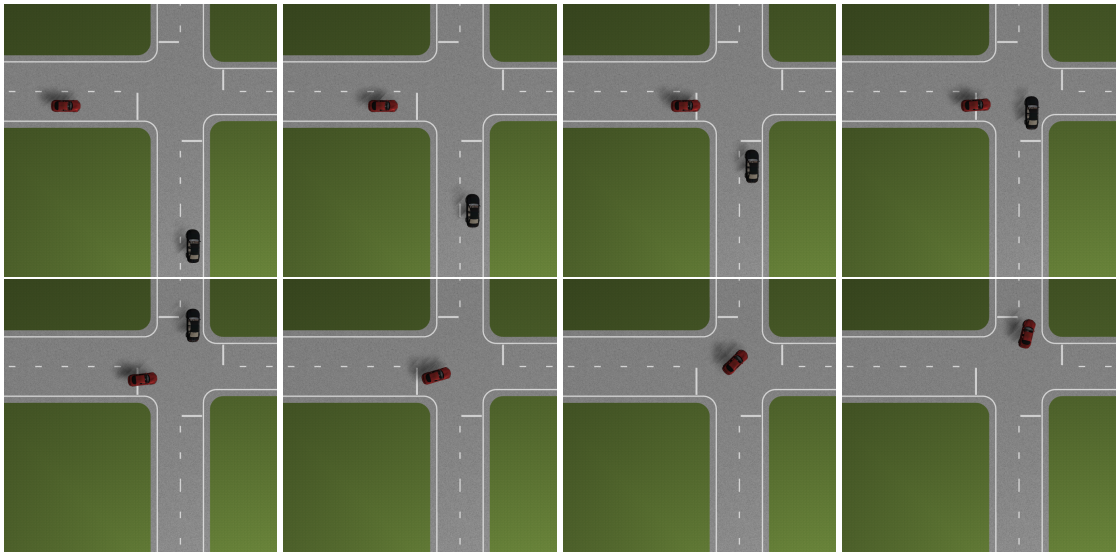

Figure 5.16: Visualization of the scenario with the Ego vehicle making a left turn, the other vehicle taking priority and going straight $t = 2 - 9\,\mathrm{s}$

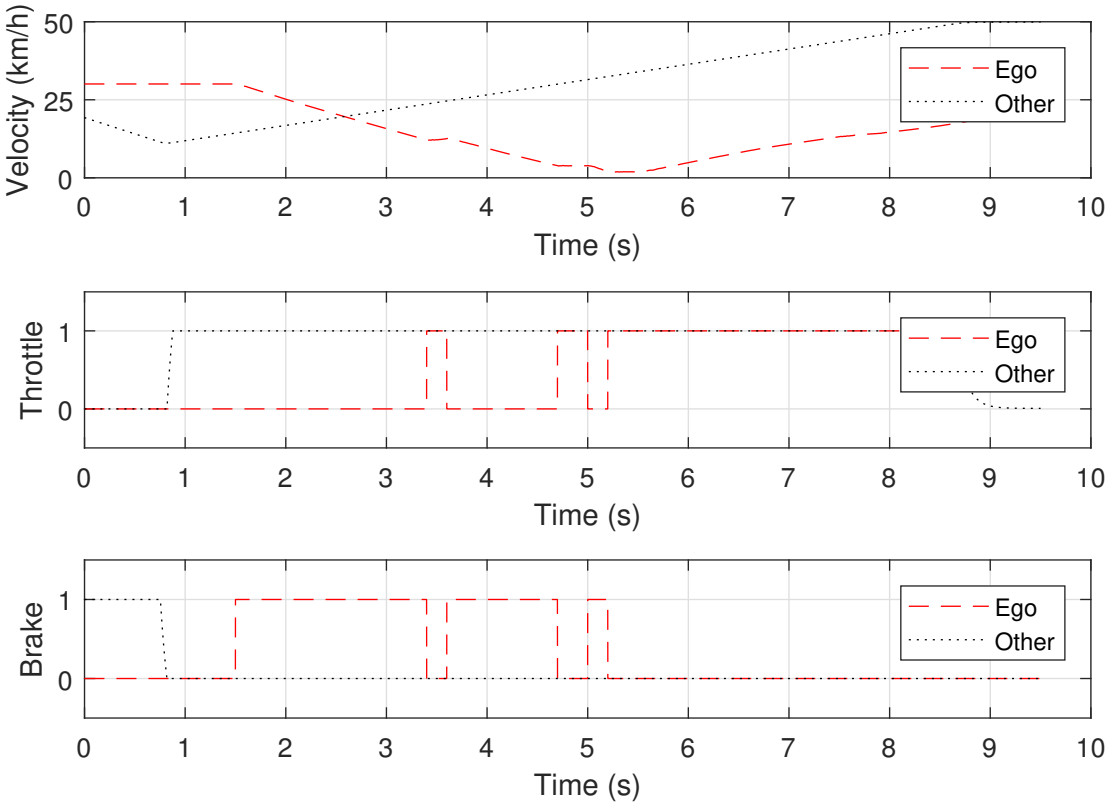

Figure 5.17: Velocity profiles and control inputs of the scenario with the Ego vehicle making a left turn, the other vehicle going straight and taking priority

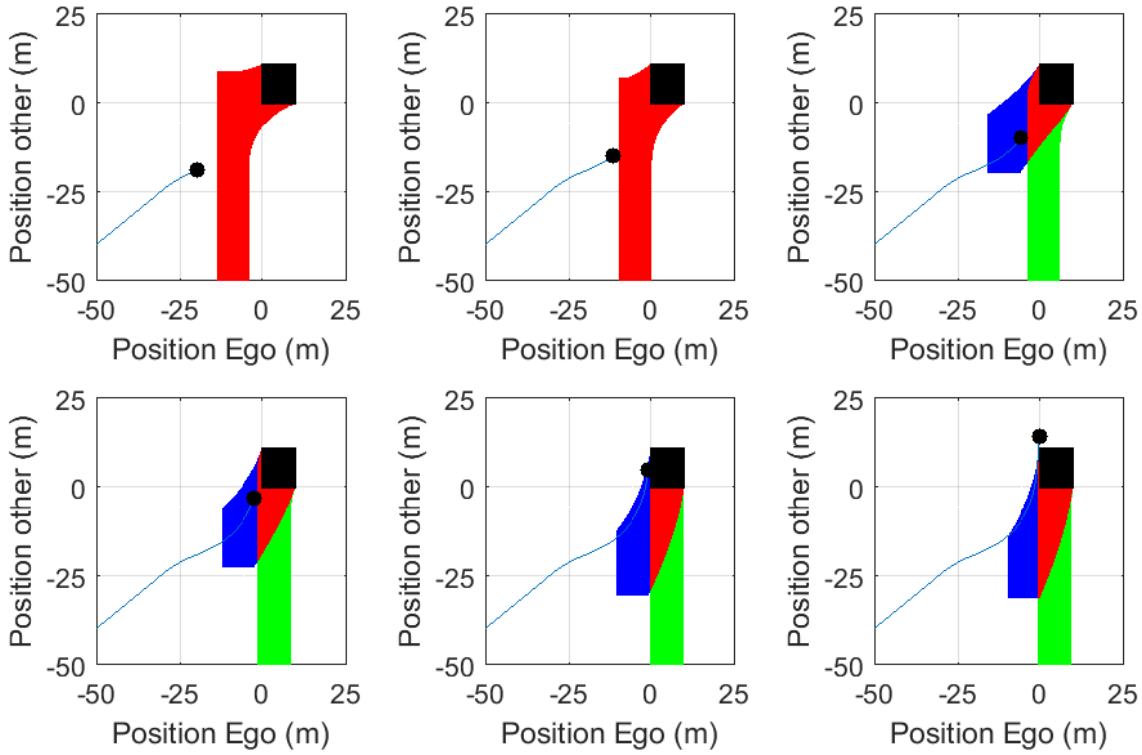

Figure 5.18: Progression of the capture set during the simulation of a left turn for the Ego vehicle and a straight trajectory of the other vehicle. From left to right and up to down: $t = 1 - 6\,\mathrm{s}$

In the second simulation, a similar scenario is used with the Ego vehicle turning left, but is now arriving much earlier. However, the other vehicle is approaching the intersection and increasing its velocity. The collision avoidance system of the Ego vehicle recognizes that it is safe to enter the intersection and takes priority as is shown in figure 5.19.

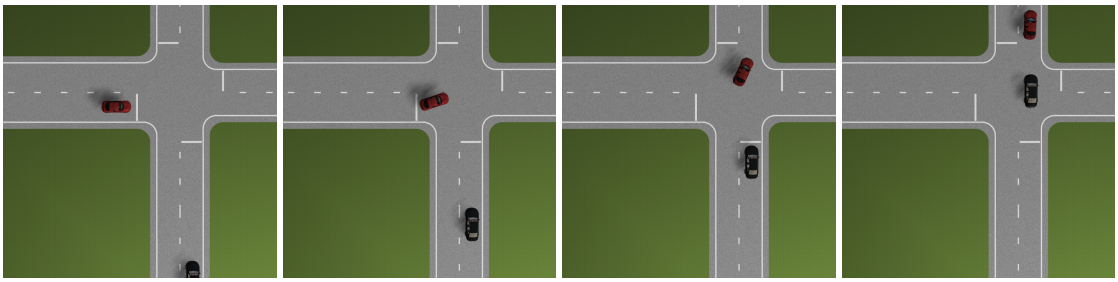

Figure 5.19: Visualization of the scenario with the Ego vehicle taking priority in a left turn and with the other vehicle going straight $t = 4 - 7\,\mathrm{s}$

The normal speed of taking a left turn is set at $v_{1,\mathrm{desired}} = 20\,\mathrm{km\,h^{-1}}$ and the maximum speed for a left turn is $v_a = 25\,\mathrm{km\,s^{-1}}$. In this scenario, the other vehicle keeps accelerating, while the Ego vehicle is already entering the intersection. To avoid a collision, the Ego also starts accelerating. The velocity profiles and applied control inputs are shown in figure 5.20. The longitudinal position diagrams and the capture sets are given in figure 5.21. These figures show that at $t = 0\,\mathrm{s}$, the trajectory is approaching the capture set and the vehicle starts braking. At $t = 2\,\mathrm{s}$, the velocity is less than the maximum velocity in the corner and the Ego vehicle is approaching the intersection. The cruise control tries to slow down the Ego vehicle to the desired velocity. This points the trajectory towards the capture set, as the other vehicle is increasing its velocity. Therefore, at $t = 3\,\mathrm{s}$ the flow is touching the capture set, but in this simulation it is touching the capture set from the braking capture slice. This makes the Ego vehicle to start accelerating again up till its maximum velocity of $25\,\mathrm{km\,h^{-1}}$. When the turn is done, the Ego vehicle accelerates to the normal desired velocity.

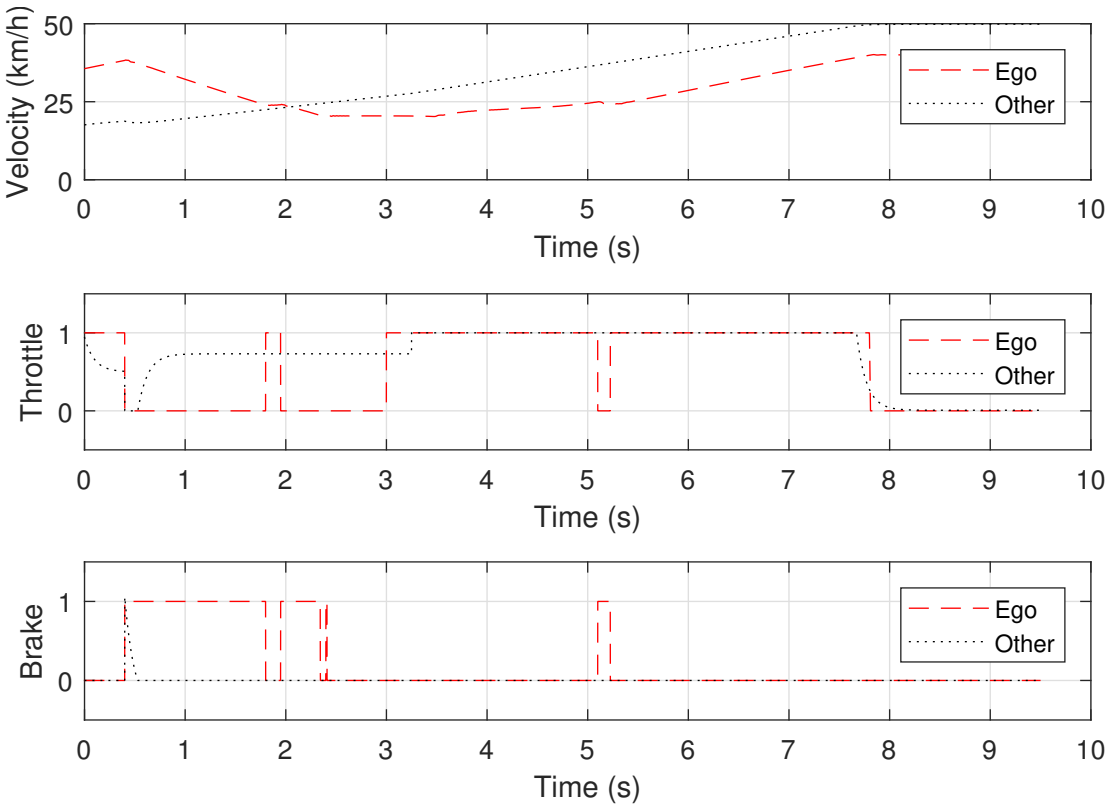

Figure 5.20: Velocity profiles and control inputs of the scenario with the Ego vehicle taking priority in a left turn and with the other vehicle going straight

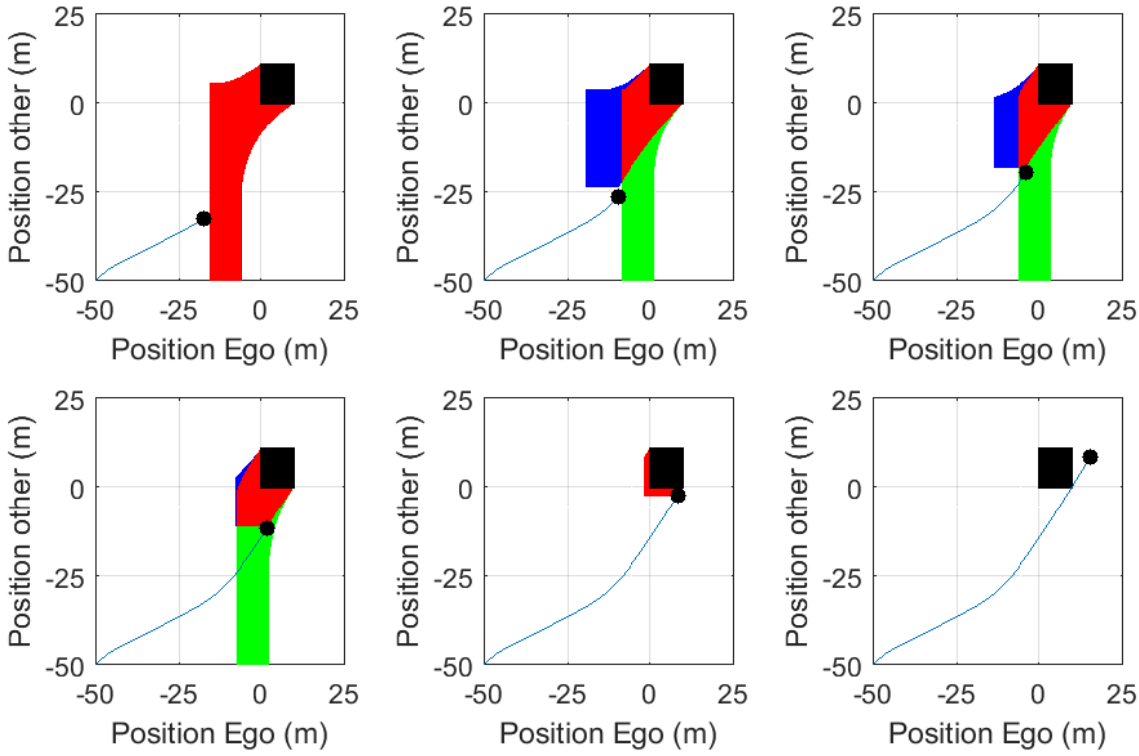

Figure 5.21: Progression of the capture set during the simulation of a left turn for the Ego vehicle and a straight trajectory for the other vehicle. From left to right and up to down: $t = 1 - 6\,\mathrm{s}$

5.2.3. Other scenarios

In case the autonomous vehicle is approaching an intersection with a stop sign or a red traffic light, the collision avoidance system can also be used to make the vehicle slow down. Figure 5.22 shows the two scenarios discussed in this section.

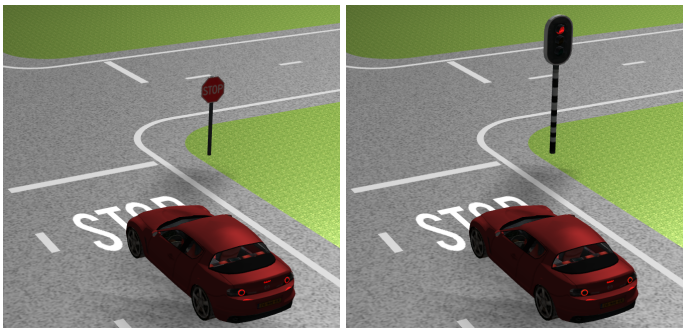

Figure 5.22: The two alternative scenarios for the collision avoidance system, left: the stop sign, right: the traffic light

The first scenario, that is discussed, is the one with the stop sign. An imaginary car is placed in between a lower and upper boundary of the bad set, when the Ego vehicle is approaching the sign. This imaginary care forces the autonomous vehicle to a complete standstill, which is mandatory according to the traffic rules. When there is no other road user, the imaginary car gets placed outside these boundaries at the moment the autonomous vehicle reaches a standstill. If there is another road user, the imaginary car gets replaced by the ground truth information of this vehicle.

The velocity profiles from this scenario are shown in figure 5.23. The stopping sign is located at the point where the longitudinal position of the Ego vehicle is at $0\,\mathrm{m}$. The velocity of the vehicle decreases to $0\,\mathrm{m\,s^{-1}}$ at this location, which means that the vehicle stopped in front of the stop sign. The reason for not reducing in a smooth line, is that at some certain moments the vehicle is not exactly following the capture set trajectory. This is due to the simplified dynamics in the generation of the capture slices.

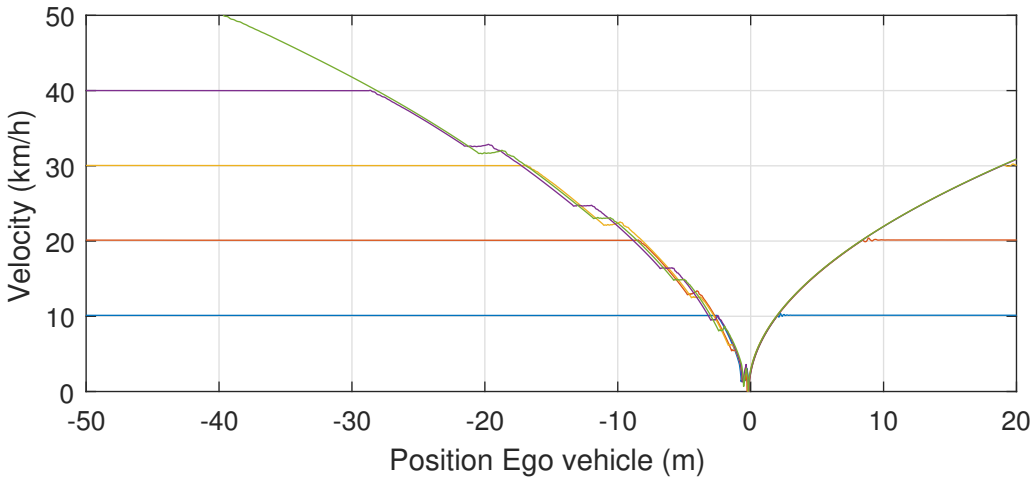

Figure 5.23: The autonomous vehicle approaching an intersection with a stop sign at various starting velocities

The next scenario is the one with a traffic light. In this scenario the same method, with the imaginary car, is used. The autonomous vehicle is approaching the traffic light with a certain velocity. While the traffic light is showing the red light, it is slowing down. When the traffic light color changes to green, the imaginary car is moved outside of the boundaries and the autonomous vehicle accelerates. The results of the simulations are shown in figure 5.24. The traffic light is located at longitudinal position equal to $0\,\mathrm{m}$ and the vehicle is approaching it at a velocity of $50\,\mathrm{km\,h^{-1}}$. The switching of the traffic light, from red to green, has been done at time steps varying with $1\,\mathrm{s}$, starting at $t = 3$. In the figure, the same, not smooth, behavior in the braking part is shown as with the stop sign.

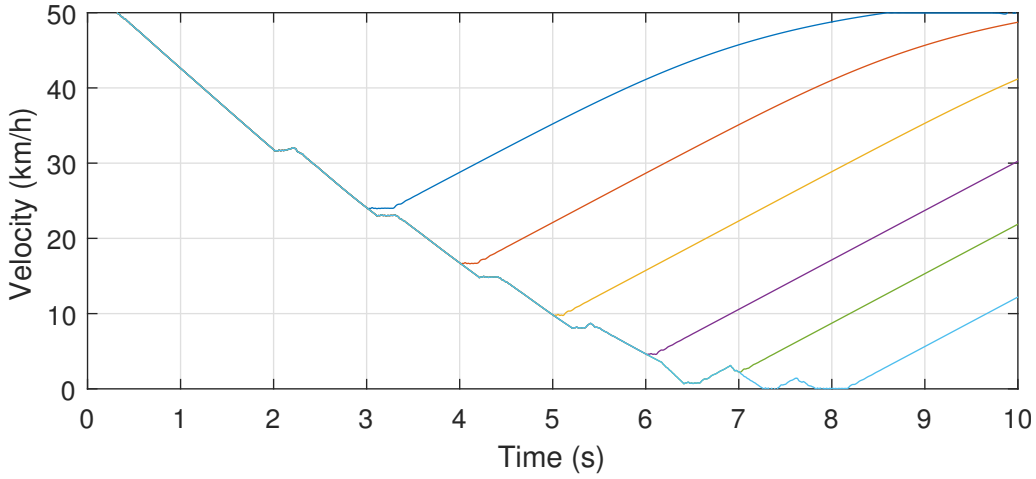

Figure 5.24: The autonomous vehicle approaching a red traffic light and at various times the light turns green

5.3. Real-Time testing

The following tests were performed on the real-time machine and simulator. For real-time testing, it was necessary to increase the step time in the capture slice generation to $\Delta T = 0.1\,\mathrm{s}$ to make sure that the collision avoidance system was working. A smaller step-size makes the system too large to solve in the required time. The main difference, between the previous simulations and this testing, is that there is a driver included that influences the velocity of the Ego vehicle. The resulting trajectories of the tests are shown in figure 5.25 and shows that the bad set of states was avoided. The trajectories, that are touching the upper left corner of the bad set, are trajectories in which both vehicles are going straight at the intersection. The ones that touch the lower right corner, are the situation in which both vehicles are turning left at the intersection. From both scenarios, one result is investigated more in detail.

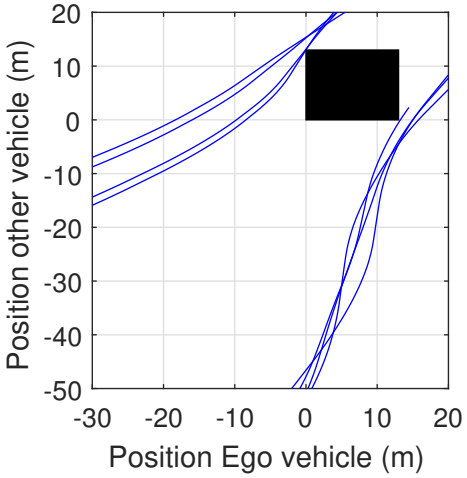

Figure 5.25: The trajectories of the vehicles of the real-time testing with the simulator

In the first scenario, both vehicles are going straight at the intersection. Figure 5.26 shows the velocities of both vehicles and the control inputs of the Ego vehicle. This figure shows that the driver is slowing down the vehicle by applying the brake at $t = 4\,\mathrm{s}$, but releases it after half a second. Approximately one second later, the controller starts applying brakes to avoid the collision. At $t = 8.4\,\mathrm{s}$, the driver applies throttle, but the output of the system is still the brakes, due to the collision avoidance system. When the system notices that the intersection is safe, the driver gets back control and the vehicle accelerates. Figure 5.27 shows the resulting trajectory and the change of the capture slices during the test. The capture set graphs show that ΔT was increased, as the capture set looks less continuous.

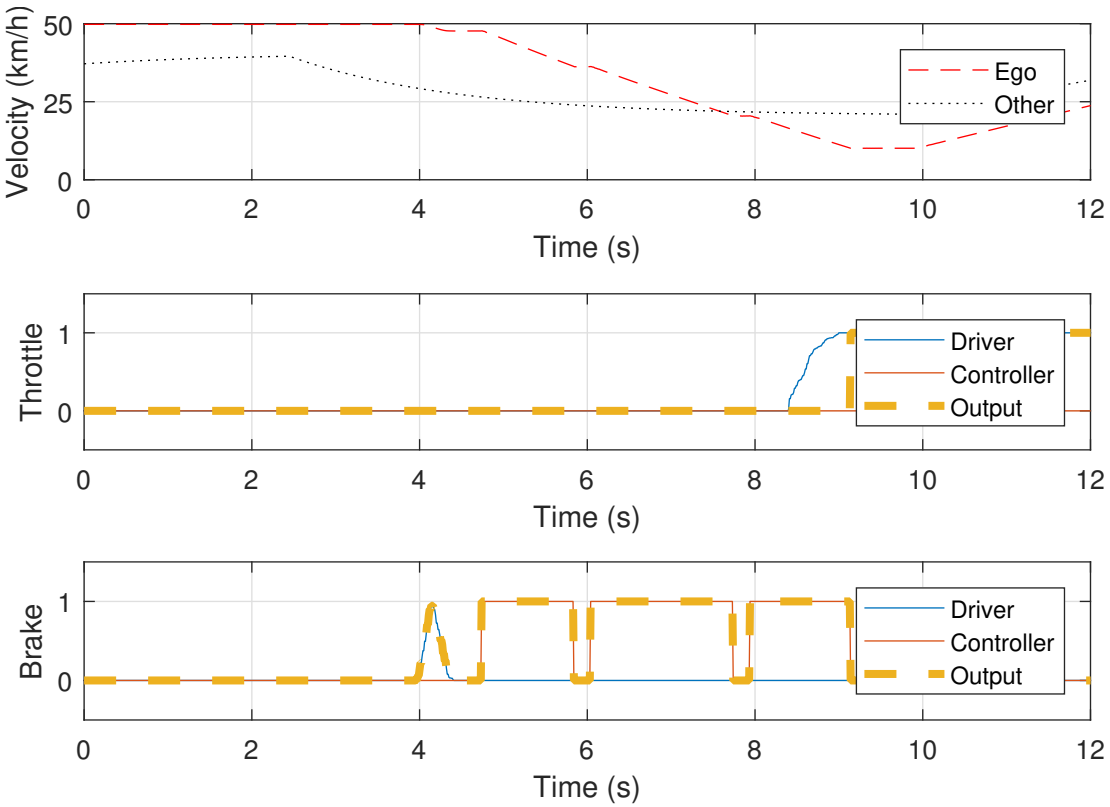

Figure 5.26: The velocity profiles of both vehicles and the inputs of the Ego vehicle in the scenario with the other vehicle taking priority during the real-time test

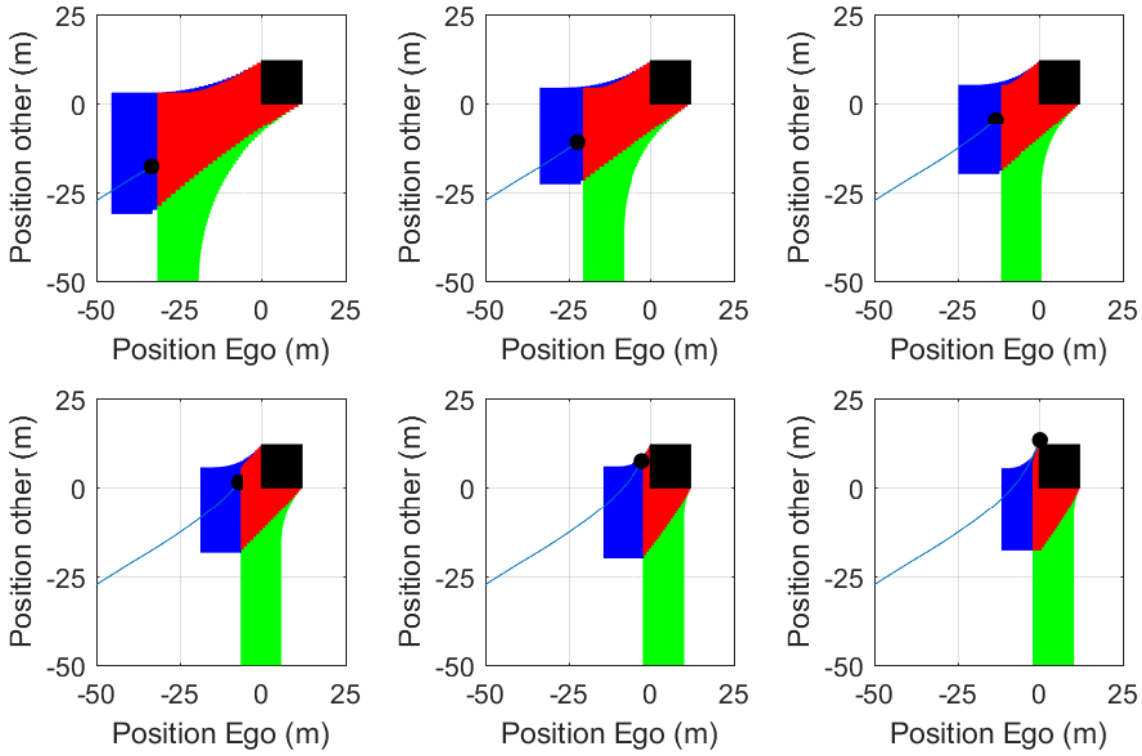

Figure 5.27: The position diagrams and capture sets of the scenario with the other vehicle taking priority during the real-time test $t = 6 - 11\,\mathrm{s}$

Figures 5.26 and 5.27 show that both vehicles are slowing down before the intersection. Because the flow is coming from the blue acceleration capture slice to the red capture set, the Ego keeps on braking. If the flow passes by the lower left part of the capture set, the Ego takes priority. However, the flow stays at the left part and makes the Ego vehicle decelerate till the other vehicle passed the intersection. Now, control is being returned to the driver.

In the second scenario, both vehicles are turning left at the intersection. Therefore, they are both slowing down to less than $25\,\mathrm{km\,h^{-1}}$ to be able to make the turn safely. Figure 5.28 shows the velocity of both vehicles and the inputs on the Ego vehicle. Figure 5.29 shows the trajectory and the capture set over time. In these figures, it is shown that the driver slows down the vehicle before the controller finds it necessary for collision avoidance. However, the driver slows the vehicle down too much in between the boundaries of the bad set, which makes the system vulnerable for a collision. In the position diagrams, this is shown by having the flow move vertical, because the other vehicle is driving faster than the Ego vehicle. As the flow is moving in this direction, it is approaching the capture set. When it comes to close to the capture set, the collision avoidance system takes over control. Because it moves through the braking capture slice towards the capture set, throttle is applied on the Ego vehicle. This accelerates the vehicle and bends the trajectory more horizontal in the diagram. By bending the trajectory more horizontal the bad set is avoided and no collision occurs.

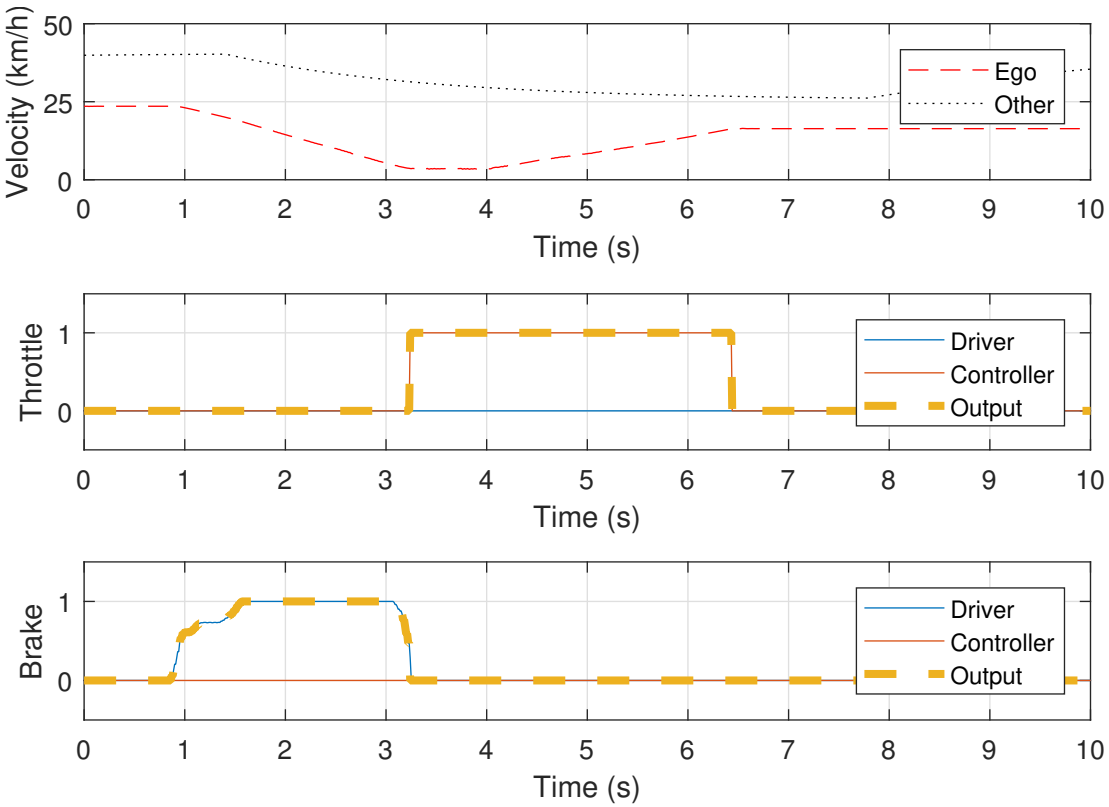

Figure 5.28: The velocity profiles of both vehicles and the inputs of the Ego vehicle in the scenario with the Ego vehicle taking priority during the real-time test

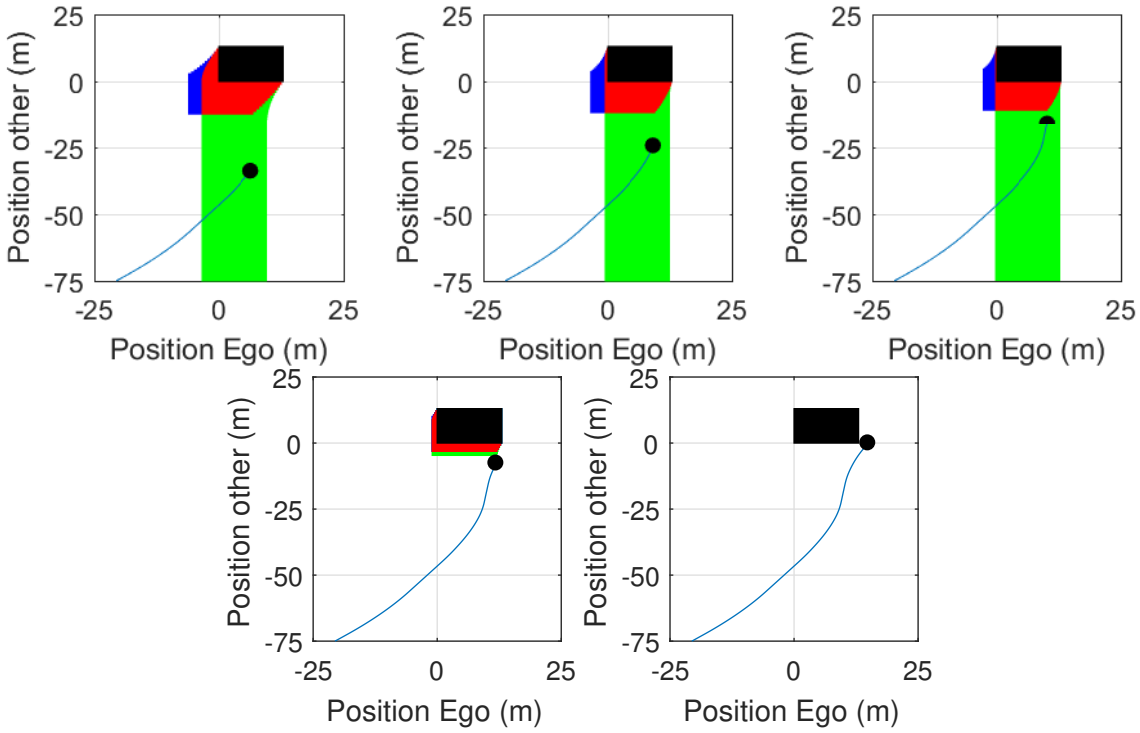

Figure 5.29: The position diagrams and capture sets of the scenario with the Ego vehicle taking priority during the real-time test $t = 3 - 7\,\mathrm{s}$

6

Conclusions and Recommendations

6.1. Conclusions

In this report, a controller for collision avoidance and trajectory tracking was demonstrated on a full autonomous vehicle and as an Advanced driver-assistance systems (ADAS) supporting a human driver. The controller was tested in scenarios with one other road user that had an overlapping trajectory with the controlled vehicle. To prevent a collision, one of the two vehicles had to leave the intersection before the other entered. To achieve this result, the method of backwards reachability analysis was used. A set of unsafe states was defined, which consisted of the locations in which the vehicles are colliding. Using reachability analysis, the states that lead the system to the unsafe set, irrespective of the input on both vehicle, was found. With these two sets, a strategy was developed to avoid entering the unsafe set.

For the collision avoidance system, two scenarios were considered. In the first scenario, both vehicles were driving autonomous and were communicating about the input. The result was that one vehicle applied full brakes, while the other applied full throttle to avoid collision. In the second scenario, the Ego vehicle was avoiding the collision without having any knowledge on what the other vehicle was going to do. By assuming limits on the acceleration and deceleration of the other vehicle, it was possible to avoid a collision.

For making turns at the intersection, a trajectory tracking method was implemented. A pure pursuit controller was developed for making it possible to track predefined trajectories. With the ability to make turns, a new problem was introduced. The maximum velocity at a certain moment in the trajectory was lower than the current velocity of the Ego vehicle. To solve this problem, an adjustment to the collision avoidance system was proposed and tested. The adjustment was that the set of states to be avoided was equal to the set of states that lead to the unsafe set with the braking input. When the flow of vehicles approaches this set, the Ego vehicle automatically applies the brakes, until the velocity was lower than the maximum allowed velocity. Depending on the position of the flow in the longitudinal position diagrams, it was then decided whether the Ego vehicle started accelerating towards the maximum velocity or applied the brakes and gave priority to the other vehicle. To avoid the bouncing on and off the set of states that lead to a collision, the output of the control strategy was replaced by a velocity instead of full brake or throttle. A PID controller was used on the difference between the current velocity and the velocity from the collision avoidance system to set the throttle or brake level.

The controllers were tested on a high fidelity vehicle model consisting of several components from different software. Several scenarios were simulated offline without the real-time requirement. In these simulations, it has been shown that the flow did not end up in the unsafe set. This has been done for the case of two autonomous vehicles and for the case where there was no communication between the vehicles, which can be seen as a case with a human driven other vehicle. First, the scenarios were tested with both vehicles going straight at the intersection. From these simulations, it was found that in case of two autonomous vehicles, there were no issues with the collision avoidance system. When the control of the other vehicle was removed, for the case of a human driven other vehicle, the Ego vehicle became slightly conservative and was often giving priority to the other vehicle. The final offline simulations included the turning of the Ego vehicle. From these simulations, it was found that

the adjustments to the collision avoidance systems were giving the desired results. The vehicles were not ending up in a collision and the Ego vehicle was not exceeding the speed limit in the corner.

In the last part, the collision avoidance system was used as an Advanced driver-assistance systems (ADAS). The collision avoidance system was running on the background and was helping a driver in avoiding a collision. Therefore, the system was tested on a real-time application with a human in the loop. For these tests, a detailed environment model was produced to have a realistic scenario for the driver. Furthermore, a human driver was setting the velocity of the vehicle by making use of the throttle and brake pedals and when the collision avoidance system noticed that there was a chance of a collision, it interfered and took over control. When the risk of having a collision was gone, the control was returned to the human driver.

6.2. Recommendations for future work

In the developing of the intersection controller, many assumptions were made and this makes room for a lot of future work. The recommendations given in this section are be split in a few groups. The first part is on improving the testing environment. This is followed by suggestions on improving the controllers. Finally, some other suggestions for future work are guveb.

6.2.1. Improving the testing environment

First of all, the fidelity of the Ego vehicle can be increased to make it represent a real vehicle better. The used multibody model did not include a complex tire model or any aerodynamic forces, which both have a large influence on the vehicle dynamics characteristics. The goal of the thesis was to demonstrate the method for collision avoidance in combination with turning at the intersection on the driving simulator. The simplifications on the vehicle model reduced the number of uncertainties and reduced the time for tuning of the controllers. By adding the more complex tire model, the number of coefficients for generating the capture slices increases and more tuning has to be done. To demonstrate that this method is also feasible to use on real vehicles, the controller has to be tested on a vehicle model, that has these extra difficulties included.

For the simulations, it was assumed that ground truth information was available for the controllers and in reality this is never the case. If this assumption is not valid, the Ego vehicle has to obtain the information from sensors. This introduces several new problems and difficulties. It will introduce extra uncertainties, but also the delays, which have an influence on the system and control strategy. To make sure that the system is still safe, the size of the capture set can be increased or the time for the future state estimator can be increased. However, this will also make the Ego vehicle more conservative and a balance between how conservative the vehicle is and safety has to be found.

6.2.2. Improving the controllers

The way the control map was designed and how the capture slices were generated, result in most of the time with full throttle or full brakes as output. This is desired for last minute collision avoidance, but to make the system more comfortable the inputs on the vehicle have to be applied in a smooth transition. To keep the guarantee of no collision, a second or multiple layers of capture sets could be used. These capture sets can be made with a percentage of maximum input on the Ego vehicle. The problem with this method is that it increases the amount of calculations and this can eventually become a problem for the application in real-time. Another, more simple option to implement, is using a variable input on the system in the capture slice generation. Using zero torque input on the first step and increasing this every time step, results in larger capture slices. This method has as disadvantage that the collision avoidance system only works, when the trajectory is at the border of the capture set. For this method, the vehicle has to follow exactly the predicted trajectory generated in the capture slice. Otherwise, the trajectory might be bouncing on and off the capture set, which results in continuously changing of the throttle and brake settings.

Another interesting option is making the control output depending on the distance towards the capture set. To make this work, first, the actual area of the capture set has to be generated. The collision avoidance system, at this stage, only checks if the system is in one of the two capture slices, but it does not generate the actual capture set. For the generation of the capture set, it might be possible to do this geometrically by making use of zonotopes and polytopes or one of the available reachability analysis tools. To be sure that the system still runs real-time, it might be an option to increase the step-size in

the capture slice generation and make use of an overestimation of reachable sets.

From testing with the autonomous-human driven case, it was found that the Ego vehicle became more conservative. The acceleration and deceleration ranges in the generation of the capture slices were causing this, because they were not variable. This can be made variable by adding heuristics on the system, for example based on the type of intersection that is being approached or on the distance of the other vehicle to the intersection, the size of the capture set can be reduced. With the current system, if the other vehicle is approaching the intersection from the left and has to give priority to the Ego vehicle, the Ego vehicle keeps considering that the other might not give priority and that the other might accelerate to force an accident. Therefore, the Ego vehicle slows down to make sure no collision occurs. If a human would be driving the vehicle, it would assume the other to give priority and that the other was going to start braking. The human driver would enter the intersection without considering that the other vehicle might try to force a collision.

The decision point of the mode estimator combined and the typical accelerations and disturbance also have to be tested with different vehicles and different human drivers. Because of time limitations, it was not possible to perform these kind of tests during the project. Therefore, it was made sure that the boundaries of the accelerations were larger than the full braking and full throttle values that the other vehicle was able to produce. In reality, nobody is applying these kind of accelerations at intersections. By taking more realistic values for the typical accelerations and disturbance, the size of the capture set reduces dramatically and this makes the Ego vehicle much less conservative.

It might be interesting to see if machine learning can be used to recognize typical driver behavior and use the results for the generation of the reachable sets. This can also decrease the size of the capture set and, consequently, make the vehicle less conservative. It can also be interesting to use some method for a better mode estimator. The mode estimator is at the moment only checking the acceleration at the last meters before the intersection. Based on distance towards intersection, velocity, acceleration and priority rules, it has to be possible to create a better estimate for the driver behavior of the other vehicle.

6.2.3. Other suggestions

In the testing and designing of the controllers, only one other road user was assumed. In reality, often multiple road users approach the intersection at the same time. Therefore, the current controller is only usable in just a few situations. To make the system work in the scenarios with more road users, several options are available. It is possible to add an extra layer, like a behavioral layer, in the system to make the collision avoidance work in this scenario. This can be done by making use of a logic switches, where the collision avoidance only works with the car that is most critical for a collision. A second method could be to develop a multiple dimensions capture set for all vehicles. This introduces new problems as a new control strategy has to be developed. The decision about which input to apply on the vehicle now becomes a optimization problem.

During the research, the main objective was to develop a controller that made sure that there were no collisions at the intersection. In future research, also the human factors have to be considered. If the improvements of the previous subsection are not implemented, the controller is not changing the controls smoothly. Otherwise, the ride might be uncomfortable for the occupants. Secondly, the system has to be tested more with humans as the system might be too conservative. If the occupant would feel that the controller is too conservative, it might decide to turn the system off. People prefer to spend as less time as possible in the vehicle and if the vehicle decides to give priority too often to the other road user, the travelling time increases. An increasing travelling time is undesired and will lead to people switching of the automated driving functionality.

Bibliography

[1] Matthias Althoff. Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars. *Fakultät für Elektrotechnik und Informationstechnik*, page 221, 2010.

[2] R Craig Coulter. Implementation of the Pure Pursuit Path Tracking Algorithm. *Communication*, (January), 1992.

[3] D Del Vecchio, M Malisoff, and R Verma. A Separation Principle for Hybrid Automata on a Partial Order. *Proc. American Control Conference*, pages 3638–3643, 2009.

[4] Domitilla Del Vecchio. Observer-based control of block-triangular discrete time hybrid automata on a partial order. *International Journal of Robust and Nonlinear Control*, 19(14):1581–1602, sep 2009.

[5] V Desaraju, H. C. Ro, M Yang, E Tay, S Roth, and D. Del Vecchio. Partial order techniques for vehicle collision avoidance: Application to an autonomous roundabout test-bed. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 82–87. IEEE, may 2009.

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–271, dec 1959.

[7] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The Dynamic Window Approach t o Collision Avoidance. *Robotics & Automation Magazine, …*, (March):1–23, 1997.

[8] Michael R. Hafner, Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. Cooperative collision avoidance at intersections: Algorithms and experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1162–1175, 2013.

[9] M.R. Hafner and D. Del Vecchio. Computation of safety control for uncertain piecewise continuous systems on a partial order. *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 0:1671–1677, 2009.

[10] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[11] SAE international. Automated Driving - Levels of Driving Automation are Defined in New SAE International Standandard J 3016, 2016. URL `https://www.sae.org/news/3544/`.

[12] Sooren Kammel, Julius Ziegler, Benjamin Pitzer, Tobias Gindele, Daniel Jagzent, Joachim Schr, and Felix Von Hundelshausen. Team AnnieWAY ' s Autonomous System for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

[13] S Koenig and M Likhachev. D* Lite. *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 476–483, 2002.

[14] Sascha Kolski, Dave Fergusont, Mario Bellino, and Roland Siegwart. Autonomous Driving in Structured and Unstructured Environments. *Intelligent Vehicles Symposium*, 2006.

[15] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, Olivier Koch, Yoshiaki Kuwata, David Moore, Edwin Olson, Steve Peters, Justin Teo, Robert Truax, Matthew Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, oct 2008.

[16] Alexis C. Madrigal. The Trick That Makes Google's Self-Driving Cars Work - The Atlantic. URL https://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/.

[17] Sikandar Moten, Francesco Celiberti, Marco Grottoli, Anne van der Heide, and Yves Lemmens. X-in-the-loop advanced driving simulation platform for the design, development, testing and validation of ADAS. Unpublished, 2018.

[18] Hiroki Ohta, Naoki Akai, Eijiro Takeuchi, Shinpei Kato, and Masato Edahiro. Pure pursuit revisited: Field testing of autonomous vehicles in urban areas. *Proceedings - 4th IEEE International Conference on Cyber-Physical Systems, Networks, and Applications, CPSNA 2016*, pages 7–12, 2016.

[19] Hans B. Pacejka. *Tire and Vehicle Dynamics*. 2012. ISBN 9780080970172.

[20] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles. pages 1–27, 2016.

[21] B Shyrokau and M Ali Arat. Vehicle Dynamics A Lecture Slides, 2015.

[22] Jarrod M Snider. Automatic Steering Methods for Autonomous Automobile Path Tracking. *Work*, (February):1–78, 2009.

[23] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 3, pages 3310–3317. IEEE Comput. Soc. Press, 1994.

[24] D. Stewart. A Platform with Six Degrees of Freedom. *Proceedings of the Institution of Mechanical Engineers*, 180(1):371–386, 1965.

[25] Tesla. Full Self-Driving Hardware on All Cars. URL https://www.tesla.com/autopilot.

[26] C Thorpe and H Durrant-Whyte. *The 2005 DARPA Grand Challenge*, volume 36 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-73428-4.

[27] Rajeev Verma and Domitilla Del Vecchio. Semiautonomous multivehicle safety. *IEEE Robotics and Automation Magazine*, 18(3):44–54, 2011.

[28] Rajeev Verma and Domitilla Del Vecchio. Development and experimental validation of a semi-autonomous cooperative active safety system. *Proceedings of the IEEE Conference on Decision and Control*, pages 4849–4854, 2011.

[29] Rajeev Verma and Domitilla Del Vecchio. Safety in Semi-autonomous Multi-vehicle Systems: A Hybrid Control Approach. *IEEE Robotics and Automation Magazine*, 2011.

[30] Felix von Hundelshausen, Michael Himmelsbach, Falk Hecker, Andre Mueller, and Hans-Joachim Wuensche. Driving with tentacles: Integral structures for sensing and motion. *Journal of Field Robotics*, 25(9):640–673, sep 2008.

A

Vehicle Models

This chapter elaborates on the used vehicle models in the controllers. In the controllers a very simplified model of the vehicles is used. This chapter starts from a more general known vehicle model and shows all assumptions that lead to the used models.

A.1. Longitudinal motion

For longitudinal motion, a model that is often used is the quarter-car model. A free-body diagram of this model is shown in figure A.1.

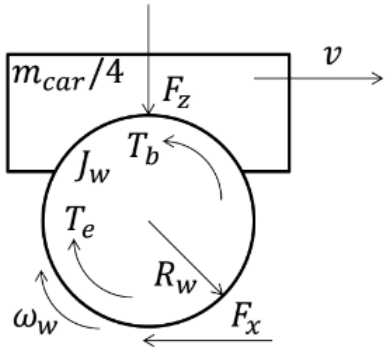

Figure A.1: Quarter-car model [21]

In this model, the longitudinal motion is coupled with the rotational motion by the contact between the tire and the road. The equation that can be derived from this free-body diagram are:

$$\begin{cases} I_w \dot{\omega}_w = F_x R_w - T_w \\ \frac{m}{4} \dot{v} = -F_x \end{cases} \tag{A.1}$$

In these equations and FBD, I_w is the inertia of the wheel, ω_w is the angular velocity of the wheel, R_w is the radius of the wheel and m is the mass of the vehicle. The longitudinal force generated in the contact point between the tire and the road is given by F_x. In the equations, T_w is the resulting torque at the wheel.

The contact force is determined by the friction coefficient μ and the gravitational force F_z:

$$F_x = F_z \mu(\kappa) \tag{A.2}$$

The friction coefficient depends on many factors, like the type of tire and the type of surface of the road. Next to the tire and road characteristics, the friction coefficient also depends on the longitudinal slip, which is the velocity difference in the contact surfaces of the road and the tire. The longitudinal slip is given by:

$$\kappa = \frac{|v - \omega_w R_w|}{\max(v, \omega_w R_w)} \tag{A.3}$$

The relationship between the longitudinal slip and the longitudinal force of the tireis shown in the second graph of figure A.2.

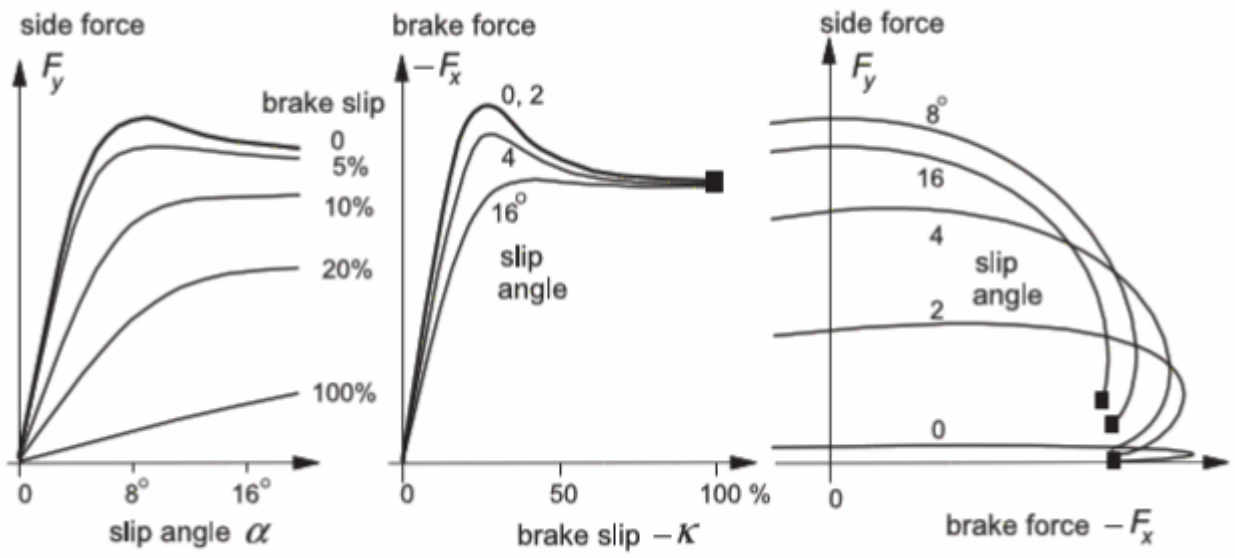

Figure A.2: The relations of tire contact forces with the slip angles [19]

To simplify the dynamics, the slip can be assumed to be zero. In this case, the longitudinal velocity is equal to the angular velocity of the wheel times the radius of the wheel. This relates the angular acceleration with the longitudinal velocity as:

$$\dot{\omega} = \frac{\dot{v}}{R_w} \tag{A.4}$$

For the next step, the equations of equation A.1 are combined and for all four wheels:

$$I_w \frac{\dot{v}}{R_w} = -m\dot{v}R_w - T_w \tag{A.5}$$

This can be rewritten to:

$$\left(\frac{I_w}{R_w} + mR_w\right)\dot{v} = -T_w \tag{A.6}$$

$$\dot{v} = -\frac{R_w}{I_w + mR_w^2} T_w \tag{A.7}$$

Now, it is possible to replace the wheel torque by its components and assuming the forces act on the center of the wheel results in the following equation:

$$\dot{v} = \frac{R_w}{I_w + mR_w^2}(T_e - T_b - (f_a - f_r - f_{\text{road}})R_w \tag{A.8}$$

Where T_e is the powertrain torque, T_b is the braking torque, f_a the aerodynamic forces, f_r the rolling resistance and f_{road} is the resistance coming from the road dynamics. The road dynamics will be assumed as zero, because the vehicle is moving on a flat road.

Now, it is possible to write the dynamics in the following shape:

$$\ddot{p} = au + b - cv^2 \tag{A.9}$$

Where $u = T_e - T_b$ is the input determined by the engine torque and the brake torque and $a = R_w/(I_w + mR_w^2)$ the rolling resistance returns in $b = -R_w^2/(I_w + mR_w^2)f_r$ and the last component consists of the aerodynamic drag. The aerodynamic drag is usually given by:

$$D = c_d \frac{1}{2} \rho v^2 S \tag{A.10}$$

In which c_d is the drag coefficient, ρ the air density and S is a reference surface area, usually the frontal area. Now, the last parameter of equation A.9 can be defined as:

$$c = \frac{c_d \rho S R_w^2}{2(I_w + mR_w^2)} \tag{A.11}$$