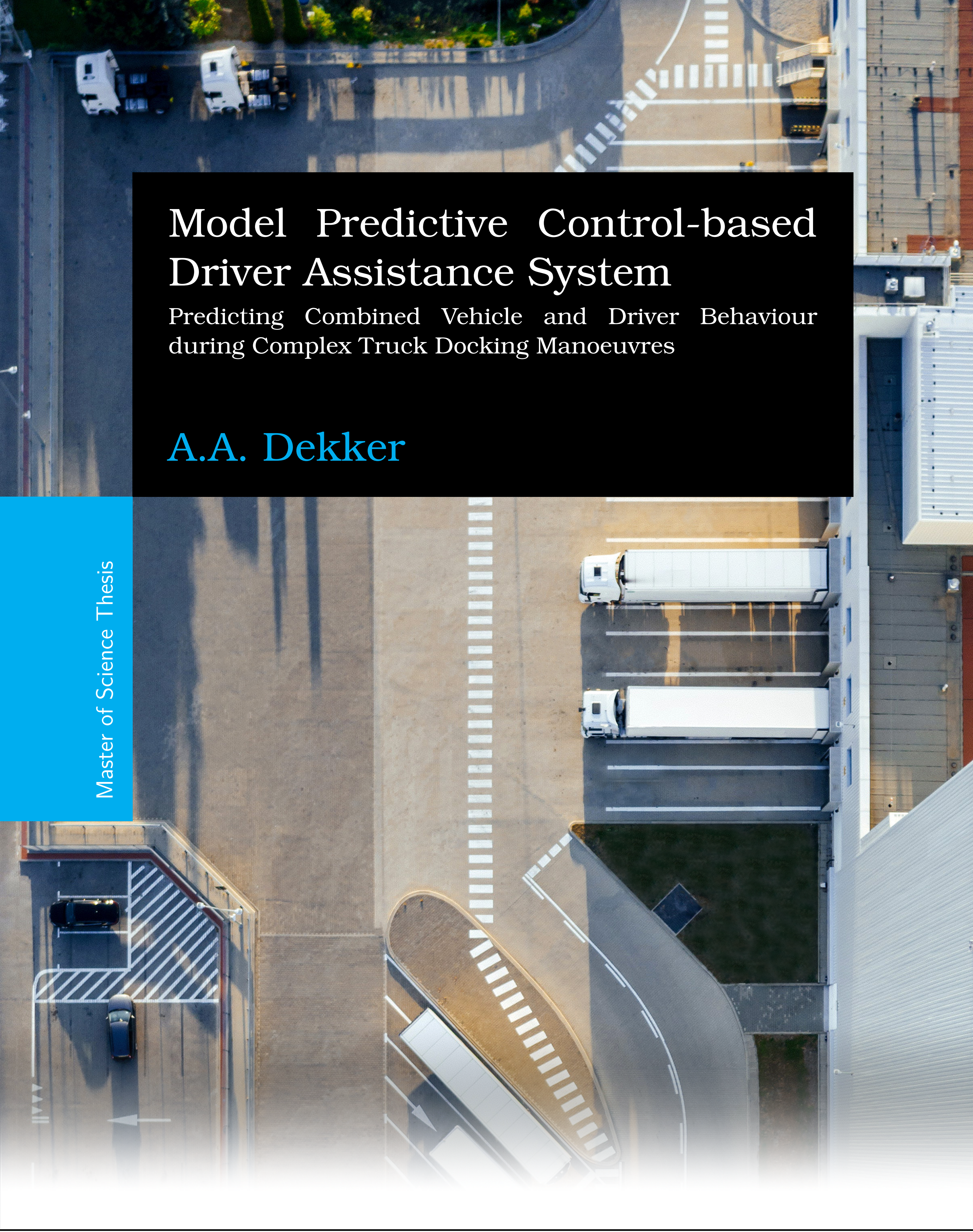


Model Predictive Control-based Driver Assistance System

Predicting Combined Vehicle and Driver Behaviour during Complex Truck Docking Manoeuvres

A.A. Dekker

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Model Predictive Control-based Driver Assistance System

Predicting Combined Vehicle and Driver Behaviour during Complex Truck Docking Manoeuvres

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

A.A. Dekker

February 10, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this Literature Survey was supported by Hogeschool van Arnhem en Nijmegen (HAN). Their cooperation is hereby gratefully acknowledged.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Model Predictive Control-based Driver Assistance System

by

A.A. Dekker

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: February 10, 2022

Supervisor(s):

Dr. L. Ferranti

Dr.-Ing. S. Wahls

Dr. K. Kural

J. Benders MSc

Reader(s):

Dr. R. Ferrari

Abstract

Due to the continuously increasing volume of road freight transportation, there is a need to aid or automate truck-trailer driving. Truck docking, the process of parking a truck-trailer combination at a loading dock, is one of the most difficult manoeuvres for professional drivers. In this work, we propose a Model Predictive Control (MPC)-based truck docking driver assistance system. The objective of the system is to support the truck driver while parking the vehicle by means of visual instructions. MPC is an advanced control method that can be used to control Multiple-Input and Multiple-Output (MIMO) systems based on a finite horizon optimization. The control structure allows one to formulate a multi-objective control strategy with explicit constraints. This work has been conducted within the scope of the VIsion Supported Truck docking Assistant (VISTA) project and the practical application of the proposed system is to experiment with an MPC-based approach for the VISTA system currently in development. To determine the optimal control action, the proposed MPC-based driver assistant makes use of a kinematic model describing the motion of the vehicle as well as a second-order linear time-invariant model representing the driver's behaviour. The system is examined by testing four performance categories: path tracking error, robustness, computational speed, and driver acceptance. The performance of the system is tested with professional truck drivers and people without a truck driver's licence in a Virtual Reality (VR) simulation. The results suggest that the proposed MPC-based driver assistant is able to perform well regarding reference path tracking and is robust against driver errors, with satisfactory computational speed. The feedback and comments from the professional drivers during testing also indicate definite possible advantages of using the system. As of now, the MPC-based solution is preferred over previous concepts of the VISTA system.

Table of Contents

List of Figures

List of Tables

Preface

This work is written as a part of my Master of Science graduation thesis. Finishing this thesis and subsequently completing the MSc program of Systems and Control is a milestone in my academic as well as my personal life. This past year, I was privileged to work on an interesting subject and to get a lot of support along the way.

I would like to sincerely thank my supervisors Dr. K. Kural and J. Benders MSc from the Hogeschool van Arnhem en Nijmegen (HAN). The ongoing pandemic meant most of the research and tests had to be conducted from home and the ideal test scenarios were not always an option. Thank you for your compassion and for providing the means to still conduct the research in the best way possible.

Additionally, I would like to thank my supervisors Dr. L. Ferranti and Dr.-Ing. S. Wahls from the Delft University of Technology (TU Delft) for their assistance during the entire process of the thesis. Their (bi-)weekly meetings kept me in touch with my fellow students and were a great way to discuss the current problems we were facing. Also, your feedback and comments on the work as it was progressing have kept me sharp and has contributed a great amount to the finalized work.

Finally, I would like to express my gratitude to my family and friends. All of them have kept the thesis process a joyous experience by providing me with the much needed regular distractions. Above all, I want to thank my parents for their unconditional love and support.

Delft, University of Technology A.A. Dekker
February 10, 2022

Chapter 1

Introduction

Road freight transportation has an important role in the economy [1] and the total volume of freight transport is expected to grow in the coming decades [2]. This will require more vehicles on the road, in order to satisfy the demand. Unfortunately, the human operation of this heavy machinery is prone to a lot of accidents [3]. *Truck docking* is the process of parking a truck-trailer combination towards a loading dock as shown in Figure 1-1. It is one of the most difficult manoeuvres for professional truck drivers, because it involves driving in reverse with an articulated vehicle, which is an unstable nonlinear process [4]. Articulated vehicles are vehicles consisting of multiple bodies connected via pivot joints, such as truck-trailer combinations. Currently, logistics companies suffer significant costs due to damages and delays in the logistic chain caused by collisions during truck docking manoeuvres [5]. Advancing technologies have enabled the development of increasingly complex automated and assistive driving systems to improve the safety and efficiency of automotive transportation.

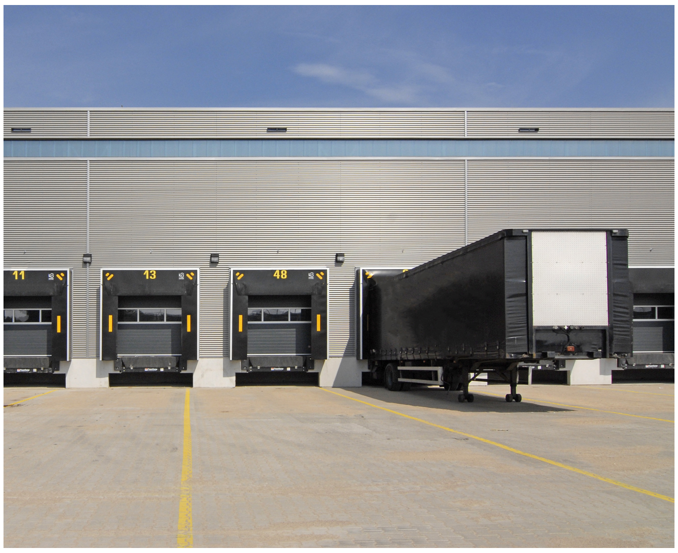


Figure 1-1: Trailer parked at a loading dock at a distribution center. Source: Adapted from [6].

1-1 Vision Supported Truck Docking Assistant Project

The VIsion Supported Truck docking Assistant (VISTA) project, a collaborative project co-financed by the European Union and led by the Automotive Research department of the Hogeschool van Arnhem en Nijmegen (HAN), aims to reduce collisions during these complex manoeuvres. The overarching goal of this project is to develop a system that supports truck drivers while docking in order to increase safety and productivity. The main requirement on the driver assistance system for the VISTA project is that it is usable without making modifications to the truck itself. This means no sensors or actuators are attached to the vehicle and the truck driver will perform the actual actions to control the movement of the vehicle. Even though the assistance system does not have full control over the actions of the driver, the entire system can be regarded as a control structure where the driver acts as an imperfect actuator. Therefore, techniques developed for autonomous driving can be used to determine the desired actions for the driver. Visualizing these actions to the driver will provide guidance during this complex manoeuvre. A conceptual overview of the assisted docking scenario is shown in Figure 1-2.

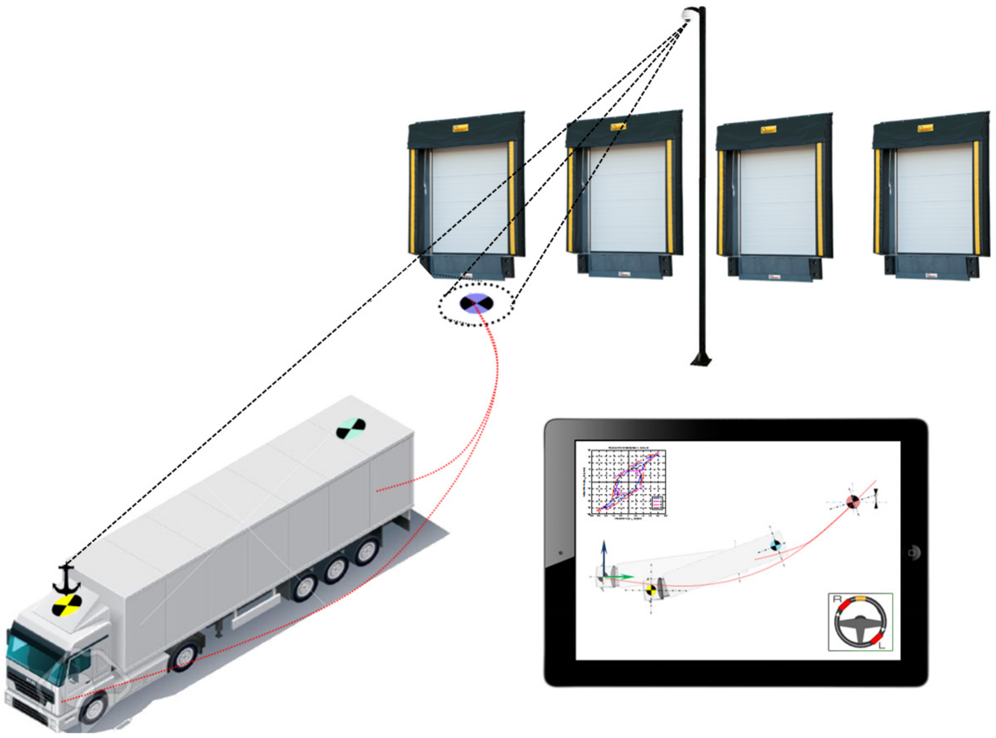


Figure 1-2: VISTA system concept with localization, path planning and driver support [7].

Figure 1-3 shows an overview of the control structure of the system. The assistance system comprises three distinct processes. First, the truck and trailer are localized by a camera-based Real Time Localization System (RTLS). With the use of cameras mounted on the distribution centre and machine learning algorithms, the position and orientation of all vehicle bodies are continuously determined. Secondly, the information from the RTLS is used by the motion planning module to compute a reference path. This reference path starts at the initial position of the vehicle and ends at the desired final position. Thirdly, the local feedback controller computes the input to the vehicle needed to park the truck-trailer combination at the loading dock. This computation is based on the reference path provided by the motion planner and the current location of the vehicle provided by the RTLS. The driver gets information on how to implement the input by means of an intuitive Human Machine Interface (HMI). This user interface is handed to the driver upon entering the Distribution Center (DC). There are multiple stages in the system which take significant time to execute. The main time-consuming tasks are: processing the images from the cameras, motion planning, computing optimal driver input, transferring the data to the human interface and a reaction delay from the driver. However, the driver requires timely updates on how to follow the reference path in real-time. Therefore, all delays should be taken into account and ensured to be sufficiently small.

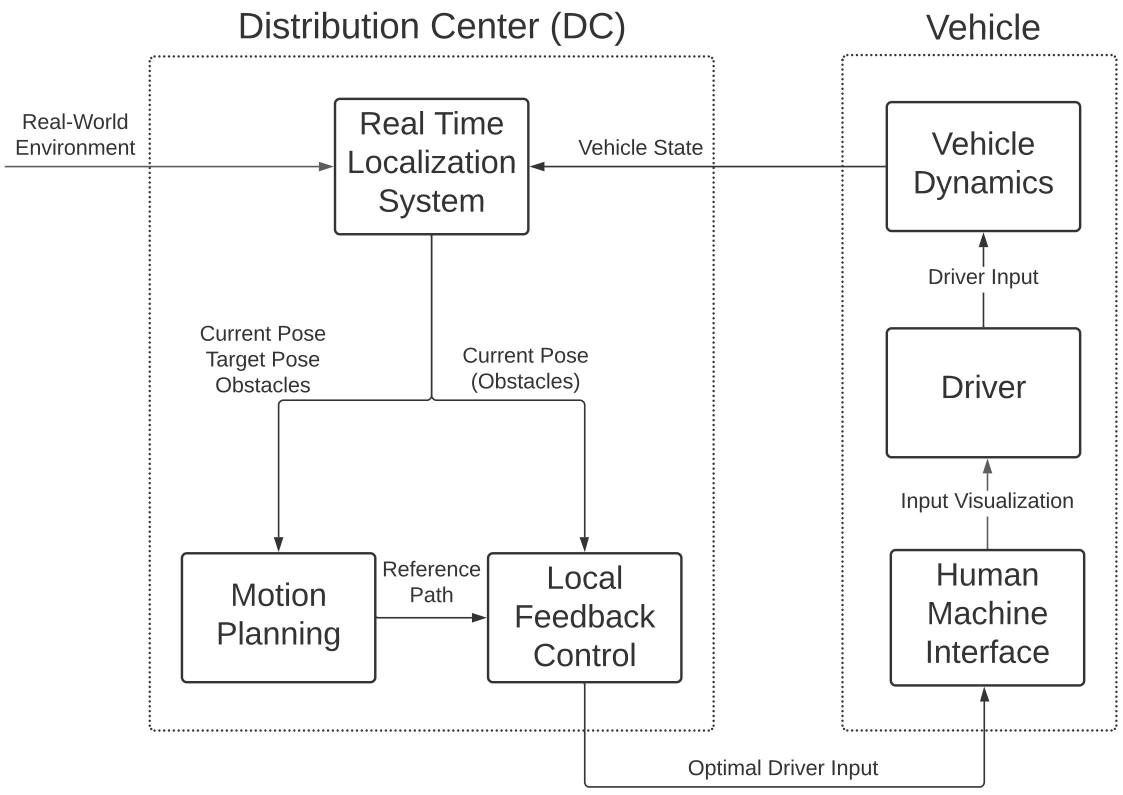


Figure 1-3: Overview of the VISTA control structure.

1-2 Research Objectives

Recent research in the VISTA project has resulted in a well-performing motion planning module. Unfortunately, the current controller structure does not show satisfactory behaviour yet with a human driver in the loop. The main shortcoming is the lack of robustness against driver errors. The goal of this research project is to investigate the implementation of an Model Predictive Control (MPC) scheme in the current system structure and the influence it has on the stability and performance of the truck docking assistant. Performance of the controller will be measured using Key Performance Indicator (KPI)s. These KPIs will include objective indicators, such as the accuracy of tracking the reference path and the computational speed of the controller. However, they will also include subjective indicators, such as driver comfort and the intuitiveness of the driver commands. With a mathematical model of the driver, the MPC scheme also allows to take the driver's behaviour into account. Unwanted behaviour, such as the delay caused by the driver's reaction time, can thus be anticipated by adjusting the control action accordingly. It is interesting to investigate whether implementation of a driver model shows to have a positive impact. Because MPC can be computationally expensive and the driver requires timely updates, it is important to ensure that the update frequency of the instructions to driver is always sufficient.

The main research question is:

How suitable is an MPC-based approach for the VISTA truck docking assistance system?

This question will be answered by the investigating the following sub-questions:

1. *What are the Key Performance Indicators of the VISTA system?*
2. *How does the MPC-based system perform regarding these Key Performance Indicators?*
3. *How can the implementation of a driver model within the MPC setup influence the performance of the VISTA system?*
4. *How can the configuration of the MPC controller influence the performance of the VISTA system?*

1-3 Thesis Outline

The main topics of this Thesis report are discussed in the following structure:

- Chapter 2 comprises a condensed version of the prior conducted literature research.
- Chapter 3 presents the MPC-based driver assistant.
- Chapter 4 explains the methods of testing the system.
- Chapter 5 presents the results of the tests.
- Chapter 6 contains the conclusion of the report and discusses future research possibilities.

Chapter 2

Preliminaries

The contents of this entire chapter is adapted from the preliminary literature review [8]. This literature study has led to the identification of interesting research directions and the main research objectives. The main topics covered are motion planning and control for articulated vehicles, modelling of a human driver and Model Predictive Control (MPC).

2-1 Motion Planning and Control for Articulated Vehicles

This section gives an overview of different motion planning methods and various control techniques which can be used for motion planning and control for articulated vehicles. Motion planning techniques are used to compute a path which will ensure that the vehicle reaches the desired goal. Motion control techniques in turn are used to determine the set of actions required for this motion plan. To deploy these motion planning and control methods, first a mathematical model of the vehicle is required. A model of the vehicle describes its behaviour when a certain input is applied to the system. In general, the choice of the model leads to a trade-off between accuracy and complexity. An intricate model may yield a better representation of the real world dynamics, but it also makes motion planning and control more complex.

2-1-1 Articulated Vehicle Model

A well-known method for modeling car-like vehicles is a kinematic bicycle model. Kinematic models are model that describe the movement of mechanical systems based on their geometry. In kinematic bicycle models, the axle groups are represented by a single wheel, resembling a bicycle. The major assumption of kinematic bicycle models is that the velocity vector at the centre point of each axle group is aligned with the orientation of the wheels of that axle group, in other words, there is no slip. For low speeds, this is a reasonable assumption, as the lateral force generated by the tires is small [9]. In Figure 2-1 the kinematic model for a truck-trailer combination used in [10] is shown.

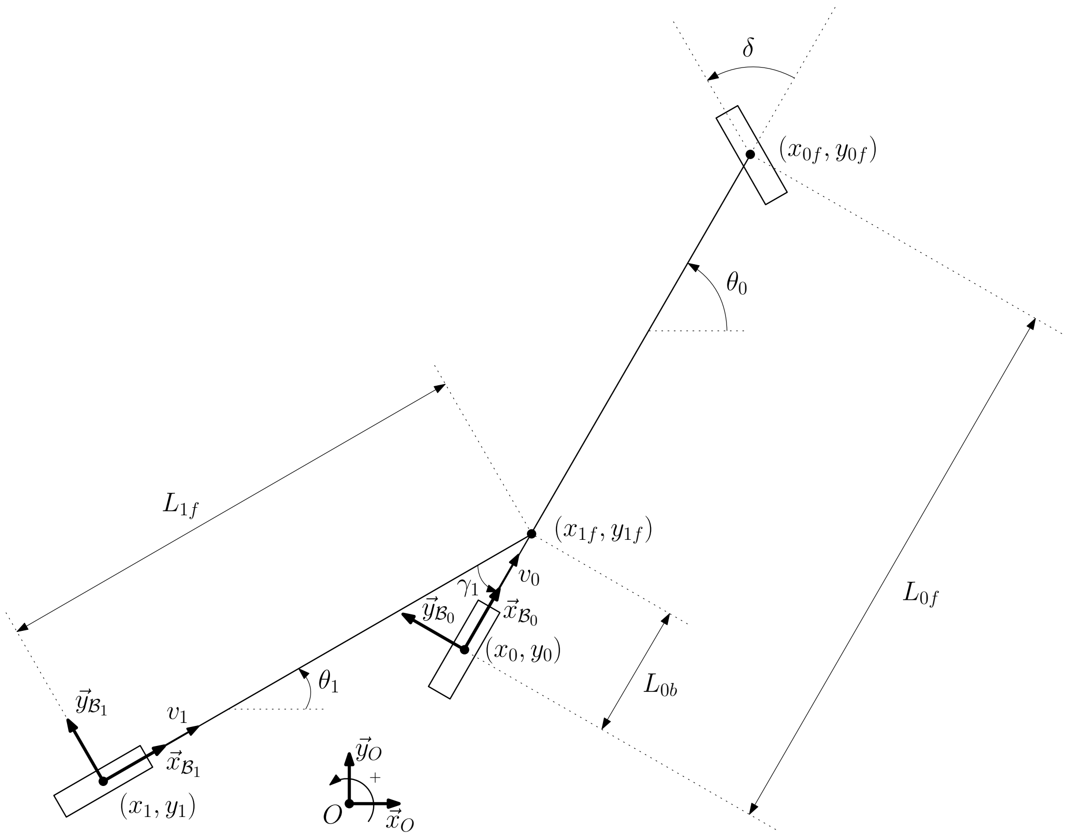


Figure 2-1: Kinematic model of truck-trailer combination. Source: Adapted from [10]. (The three tyres representing the three different axle groups are visualized as rectangles.)

The configuration of the vehicle is given by the (x_0,y_0) coordinates of the drive axle of the truck, the yaw angle of the truck θ_0 compared to the world frame and the yaw angle of the trailer θ_1 compared to the world frame. Here, $(\vec{x}_O,\vec{y}_O)$ denotes the coordinate system of the world frame, $(\vec{x}_{\mathcal{B}_0},\vec{y}_{\mathcal{B}_0})$ denotes the coordinate system of the trucks body frame, based at the drive axle, and $(\vec{x}_{\mathcal{B}_1},\vec{y}_{\mathcal{B}_1})$ denotes the coordinate system of the trailers body frame, based at the trailer's axle. The coordinates of the trucks steering axle are denoted by (x_{0f},y_{0f}). The distance between the trucks drive and steer axles is denoted by L_{0f}, v_0 is the longitudinal velocity of the truck and δ is the steering angle. As described in [10], the equations of motion prescribing the behaviour of this system are

$$\dot{x}_0 = v_0 \cos(\theta_0), \tag{2-1}$$

$$\dot{y}_0 = v_0 \sin(\theta_0), \tag{2-2}$$

$$\dot{\theta}_0 = \frac{v_0}{L_{0f}} \tan(\delta), \tag{2-3}$$

$$\dot{\theta}_1 = \frac{v_0}{L_{1f}} \sin(\gamma_1) + \frac{L_{0b}}{L_{1f}} \dot{\theta}_0 \cos(\gamma_1), \tag{2-4}$$

where γ_1 is the articulation angle between the truck and the trailer, L_{0b} is the distance between the drive axle and the articulation point and and L_{1f} is the distance between the trailer axle and the articulation point. The coordinates of the articulation point are (x_{1f},y_{1f}).

Kinematic bicycle models typically provide an appropriate representation of a vehicles motion at low driving speeds. For truck-trailer combinations they can be less suitable when the vehicle has multiple axles per axle group, as the assumption made in the kinematic model is that the axle groups can be represented as a single tyre. In reality, the dynamics of multiple axles on a vehicle body results in side-slip of the tyres [10]. Also, the mass of the vehicle and the current load are not considered when using a kinematic model. In [10], a dynamic multi-body model of a double articulated vehicle that includes tyre forces is presented. This more closely represents the actual movement of an articulated vehicle. However, this model is considerably more complex, which, in general, has the disadvantage of being computationally expensive to use [11].

2-1-2 Motion Planning for Articulated Vehicles

The motion planners task is to compute a feasible plan to manoeuvre the vehicle from its starting position to the goal position. A plan is feasible if it is physically possible for a vehicle to execute it, while avoiding all obstacles. The planned path must take the behaviour of the chosen model into account. In the case of a kinematic model, it means the path is kinematically feasible. To avoid obstacles, the configuration of the vehicle should stay within the set of allowed configurations $\mathcal{X}_{\text{free}}$. The configuration of the vehicle $\mathbf{c}\in\mathcal{X}$ defines the exact location and orientation of all relevant vehicle bodies. $\mathcal{X}$ is the *configuration space*, in other words the set of all possible configurations. The obstacle space $\mathcal{X}_{\text{obs}}$ is the set of all configurations where the vehicle collides with an obstacle. The relationship between the different sets is the following:

$$\mathcal{X}_{\text{free}} \cup \mathcal{X}_{\text{obs}} = \mathcal{X}, \tag{2-5}$$

$$\mathcal{X}_{\text{free}} \cap \mathcal{X}_{\text{obs}} = \emptyset. \tag{2-6}$$

The computed path prescribes the sequence of feasible vehicle configurations from a vehicles initial configuration to the goal region. The initial configuration is the given vehicle state at the very beginning of the manoeuvre. The goal region is the set of configurations which are declared as allowed configurations at the very end of the manoeuvre. Finding a feasible path can already be quite challenging. However, it is oftentimes also desirable to find a path that is optimal in a certain sense.

Graph Search Methods

There is rich literature on methods for finding a connection between a pair of nodes in a graph that yields the lowest total cost, better known as solving the shortest path problem [12]. A *graph* $G(V,E)$ is a mathematical structure describing the relations of different objects through nodes and edges. Each *edge* $e_{ij} \in E$ connects a pair of *nodes* $v_i \in V$ and $v_j \in V$. Each edge e_{ij} is associated with a weight c_{ij}, which can be viewed as the cost of travelling from node v_i to node v_j. Graph search methods can be used to find the shortest path from one node in the graph to another, where the shortest path is defined as the sequence of edges resulting in minimal total costs.

Dijkstra Dijkstra's algorithm, published in [13], is a well known method of finding the shortest path. The goal is to find the shortest path from the starting node v_s to final node v_t. The steps needed to find this path are shown in Algorithm 1 [14].

Algorithm 1 Dijkstra's Algorithm

```
     Input: $c_{ij}\ \forall v_i \in V, \forall v_j \in V$
     Output: $l(v)\ \forall v \in V$
 1: $l(v_s) \leftarrow 0$
 2: for all $v_i \neq v_s$ do
 3:     $l(v_i) \leftarrow \infty$
 4: end for
 5: $\mathcal{Q} \leftarrow v_s$
 6: while $\mathcal{Q} \neq \emptyset$ do
 7:     $v_i \leftarrow \arg\min \{l(v) | v \in \mathcal{Q}\}$
 8:     $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{v_i\}$
 9:     if $v_i = v_t$ then
10:         break
11:     end if
12:     for all neighbor $v_j$ of $v_i$ do
13:         if $l(v_j) > l(v_i) + c_{ij}$ then
14:             $l(v_j) \leftarrow l(v_i) + c_{ij}$
15:             $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{v_j\}$
16:         end if
17:     end for
18: end while
```

The first step of the algorithm is to initialize the cost $l(v)$ of reaching each node v from the starting node and to initialize the *open set* $\mathcal{Q}$. The second step is to chose node v_i from set $\mathcal{Q}$ that has the lowest cost $l(v)$. This node v_i is removed from set $\mathcal{Q}$. This means the value of $l(v_i)$ is fixed when node v_i is removed from the set of visited nodes $\mathcal{Q}$. Therefore, the algorithm is stopped when node v_t is removed from $\mathcal{Q}$ in this step. The third step is to update the cost $l(v_j)$ for all nodes v_j that are neighbouring nodes of v_i. The cost is only updated if the shortest path to node v_j contains node v_i. If $l(v_j)$ is updated, then node v_j has to be added to $\mathcal{Q}$ (unless v_j is already in $\mathcal{Q}$). The algorithm stops if $\mathcal{Q}$ is empty. If $\mathcal{Q}$ is non-empty, the algorithm is repeated from the second step on. If a feasible path from starting node to final node exists, goal node t will be or will have been in the set of visited nodes. In this case, Dijkstra's algorithm will not only find a feasible path, but also the optimal path [15].

A* Algorithm Even though Dijkstra's algorithm is proven to be optimal in the most general case of graph search, there are ways to improve the computational efficiency in some other cases. The A* algorithm presented in [16] uses a heuristic function $h(v)$ to guide the search. This is only possible if there is additional information available that can be exploited to make a better prioritization of nodes to explore. In the case of path planning, for example, a good heuristic function can be constructed by using the Euclidean distance between the nodes. The algorithm is fairly similar to Dijkstra's algorithm, but the difference lies in the way node v_i is chosen in the second step. In the A* algorithm, node $v_i \in \mathcal{Q}$ is selected such that $v_i = \arg\min \{l(v) + h(v) \mid v \in \mathcal{Q}\}$. Figure 2-2 shows the difference between Dijkstra's algorithm and the A* graph search algorithm. Both algorithms are used to try to find the shortest path from the red starting node in the bottom left of the map to the green node in the top right of the map. The light blue nodes represent the nodes in the open set and the color of the other nodes indicate their heuristics value. Also, there is a grey obstacle obstructing the direct path. Dijkstra's algorithm has a circular pattern when expanding nodes, while the A* algorithm needs to explore far less nodes by prioritizing the exploration of nodes closer to the goal node. Therefore, the A* algorithm is preferred over the Dijkstra algorithm if a heuristic is available.

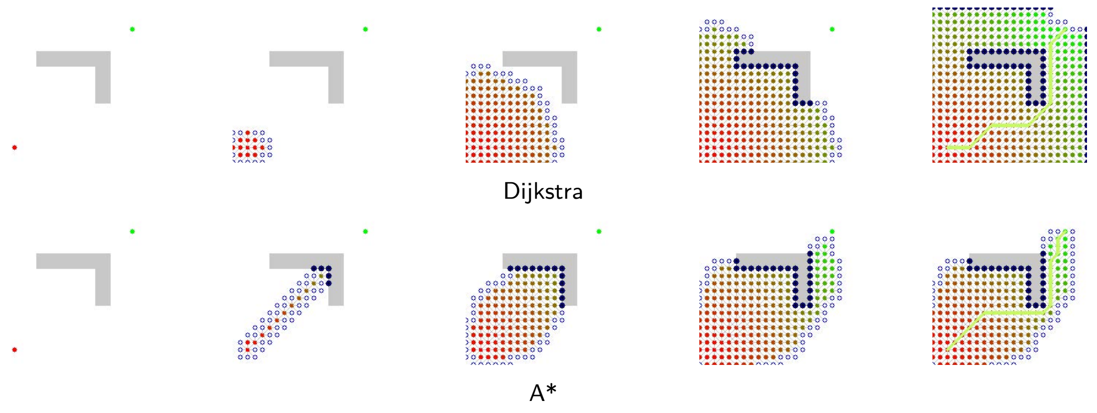
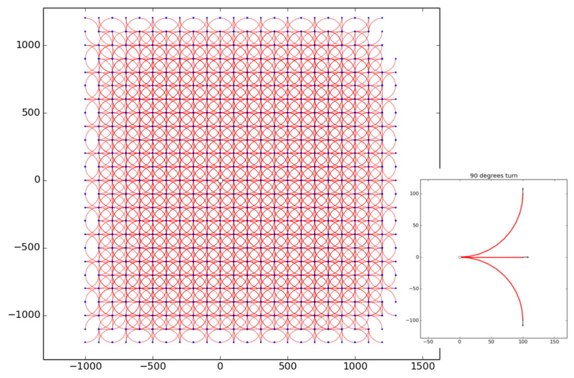
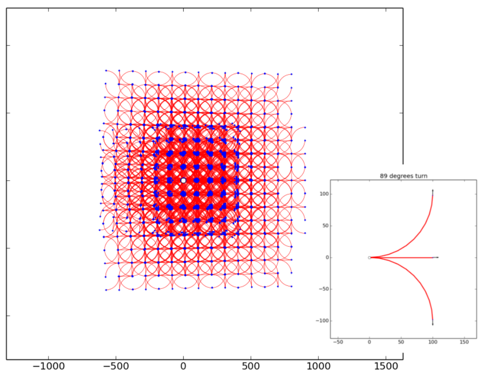


Figure 2-2: Graphical comparison of the Dijkstra and the A* search algorithms [15].

Motion primitives To use graph search methods for a motion plan, the configuration space needs to be represented as a graph. The graph representing the configuration space consists of a discrete set of selected configurations, these are the nodes of the graph. The edges with weights show the cost of moving from one configuration to another. One way of discretizing is by considering a grid map. The current location of any object in this real world can be approximated by the grid coordinates. When computing the shortest path between two cells in this grid, the result will most likely be a decent approximation of the shortest path in the real world. However, it will be a discontinuous path and the quality will highly depend on the resolution of the grid. A higher resolution leads to a higher-quality path, but also requires more computation because there are more nodes to explore. It is also not guaranteed that the vehicle is able to actually follow this path due to the constraints in the movement abilities of the vehicle. Another method is to generate a graph by recursively applying a set of motion primitives. *Motion primitives* are fixed manoeuvres used to discretize the configuration space of a vehicle. The motion primitives describe a feasible transistion from a certain configuration to another. A path consisting of a concatenation of motion primitives will therefore always be kinematically feasible. A graph search will find the optimal sequence of motion primitives to manoeuvre the vehicle from an initial configuration to the goal configuration. A *lattice graph* is graph of motion primitives that is designed to have an recursive pattern. This leads to a significant reduction of nodes in the graph. The recursive property of the graph is vizualized in Figure 2-3. Figure 2-3a shows the graph resulting from recursive application of three motion primitives, namely a 90° left circular arc, 90° right circular arc, and a straight line. Figure 2-3b shows the same graph with 89° circular arcs, clearly covering a smaller area.



(a) Lattice graph.



(b) Non-lattice graph.

Figure 2-3: Lattice and non-lattice graph, both with 5000 edges [12].

As a part of the VIsion Supported Truck docking Assistant (VISTA) project, [17] used this approach to compute a reference path. There, motion primitives of an articulated vehicle are used to build a lattice graph and the A* algorithm is used to find an optimal path. The existing framework of a bi-directional path planner was later improved in terms of computational timing and final pose error by using an optimized motion primitive library in [18].

2-1-3 Motion Control for Articulated Vehicles

Once a motion plan is constructed, the next step is to determine the actions needed for the vehicle to actually follow the reference path. It is possible to compute this set of actions a priori based on the vehicle model. This is called open-loop control and in most cases it will not yield the desired results in practice. This is due to simplifications made in the design of the model and disturbances present in the real-world environment. A *feedback controller* determines the input action in closed-loop to deal with these in uncertainties. In closed-loop control, the reference state and information from sensors about the current state of the system are used to determine the best actions in real-time. In general, the best action is the one that minimizes the error between the reference path and the actual vehicle configuration.

This should result in a collision free manoeuvre, because the motion planner already took the obstacles into account. However, it is beneficial for the controller to also mind the obstacles. Obstacle avoidance is then still ensured in the case that the obstacle space changes over time. This way the system is able to cope with moving obstacles and obstacles that were not yet identified by the perception module at the motion planning stage. Both scenarios cause the obstacle space to change over time. It is sensible to design a motion controller that is able to cope with these changes. In this section a variety of motion controllers will be discussed.

Pure Pursuit Control

One of the earliest developed strategies to control the lateral movement of a car-like vehicle is pure pursuit control, first discussed in [19]. The control law determines the steering angle based on a kinematic bicycle geometry and a fixed look-ahead distance l_d. One of the latest concepts of the control structure for the VISTA project is based on pure pursuit control.

The geometry of pure pursuit control is visualized in Figure 2-4 and it is described by the following equations:

$$\alpha = \tan^{-1}\left(\frac{y_{l_d} - y_0}{x_{l_d} - x_0}\right) - \theta_0, \tag{2-7}$$

$$R_P = \frac{l_d}{2\sin(\alpha)}, \tag{2-8}$$

where R_P is the radius of a specific circular arc. The geometry of this arc is defined by the look-ahead distance l_d and the yaw angle of the vehicle relative to the straight line from the vehicle's location to the location on the reference path at the look-ahead distance with coordinates (x_{l_d}, y_{l_d}).

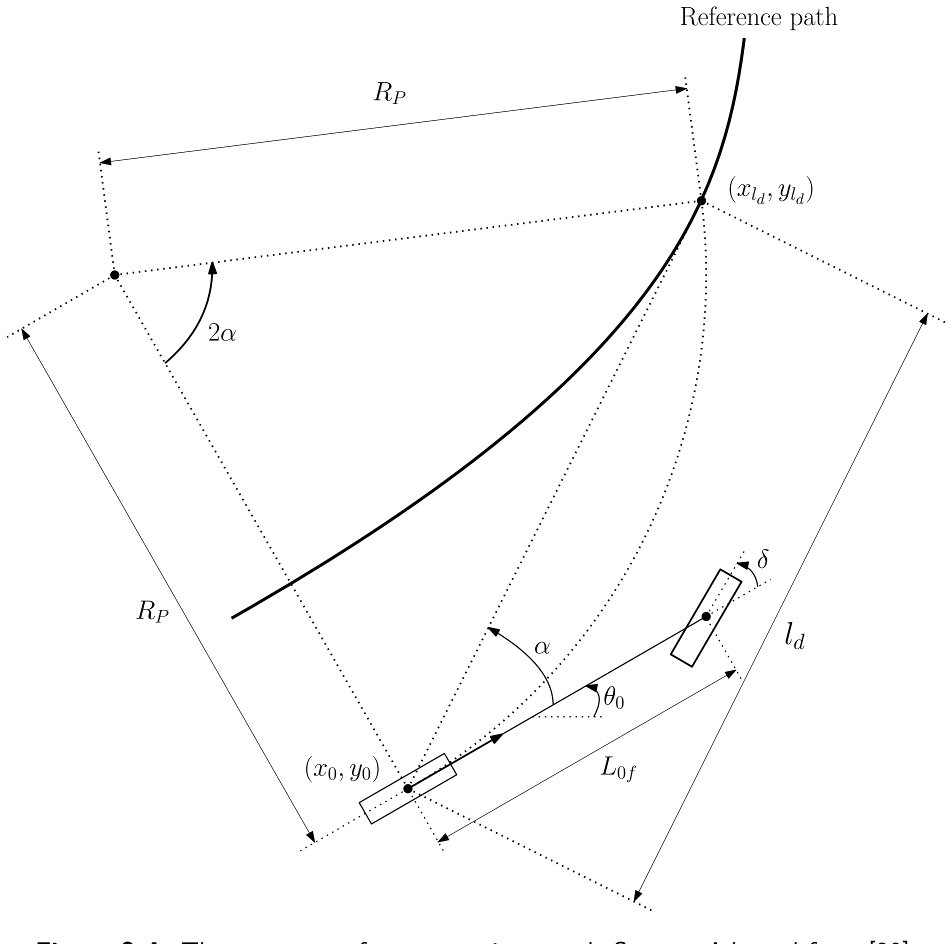


Figure 2-4: The geometry of pure pursuit control. Source: Adapted from [20].

The relationship between the steering angle δ and radius R_P is

$$\delta = \tan^{-1}\left(\frac{L_{0f}}{R_P}\right). \tag{2-9}$$

The idea behind this control law is to substitute R_P from (2-8) to (2-9). This results in the following pure pursuit control law:

$$\delta = \tan^{-1}\left(\frac{2L_{0f}\sin(\alpha)}{l_d}\right). \tag{2-10}$$

The main advantage of this control law is that it is straightforward and fairly easy to implement. However, this control structure also has some disadvantages. For one, this controller will only take the precomputed reference path into account, thus is not able to cope with dynamic obstacles. Other problems occur with large curvature changes in the reference path. To improve the performance of the controller one is able to tune the look-ahead distance. When the value is too small it will result in oscillatory behaviour, while a larger value will result in the vehicle cutting corners. A common approach is to define the look-ahead distance as a function of the vehicle speed. Also, the distance between the vehicle and the reference path is not allowed to be larger than l_d at all times. In this instance the control output would be undefined. The control input is purely based on the geometry of the vehicle and reference path. Therefore, it is also impossible to include online collision avoidance in this control scheme. The vehicle only avoids obstacles if the reference path is well planned.

Inverse Kinematics

One vital part of parking truck-trailer combinations is ensuring bi-directional functionality, meaning the vehicle can operate in a forward and a reversing motion. The most complicated of which, is to control the reverse motion of the articulated vehicle. In [10], a technique is used that exploits the concept of a virtual truck to simplify the motion control module. The idea is to view the rear most trailer as a virtual truck with a virtual steering wheel. This is visualized in Figure 2-5. A controller can determine the virtual steering angle δ^* which ensures the virtual truck to follow a reference path. The desired angular velocities of each body can then be determined with the use of the inverse kinematic model. The relationship between them is given by the following equations:

$$\dot{\theta}_1 = \frac{v_1}{L_{1f}} \tan(\delta^*), \tag{2-11}$$

$$v_0 = v_1 \cos(\gamma_1) + L_{1f}\dot{\theta}_1 \sin(\gamma_1), \tag{2-12}$$

$$\dot{\theta}_0 = -\frac{v_1}{L_{0b}} \sin(\gamma_1) + \frac{L_{1f}}{L_{0b}}\dot{\theta}_1 \cos(\gamma_1), \tag{2-13}$$

where v_1 is the longitudinal velocity of the trailer. Substituting (2-3) in these equations allows to express the desired steering angle δ in terms of the vehicle state, velocity and the virtual steering angle δ^* as follows:

$$\delta = \tan^{-1}\left(\frac{L_{0f}}{v_0}\left[-\frac{v_1}{L_{0b}} \sin(\gamma_1) + \frac{L_{1f}}{L_{0b}}\dot{\theta}_1 \cos(\gamma_1)\right]\right). \tag{2-14}$$

An advantage of this method is that the motion of the trailer can be controlled as if it was a single-body kinematic bicycle model. This is significantly easier than controlling a truck-trailer combination as a whole. However, by determining the required steering angle in this backwards manner, only the movement of the trailer is taken into account. This does seem to make sense for this intended purpose as the position of the trailer is the main priority. It is required that the rear of the trailer is connected to the loading dock in order to start the loading and unloading process. The exact position of the truck is of a far smaller importance. However, avoidance of collisions of the truck with any obstacles during the manoeuvre is not guaranteed when using the inverse kinematic model. One of the latest control concepts used for the VISTA driver assistance system uses a pure pursuit-based control strategy with inverse kinematics for reverse driving [5].

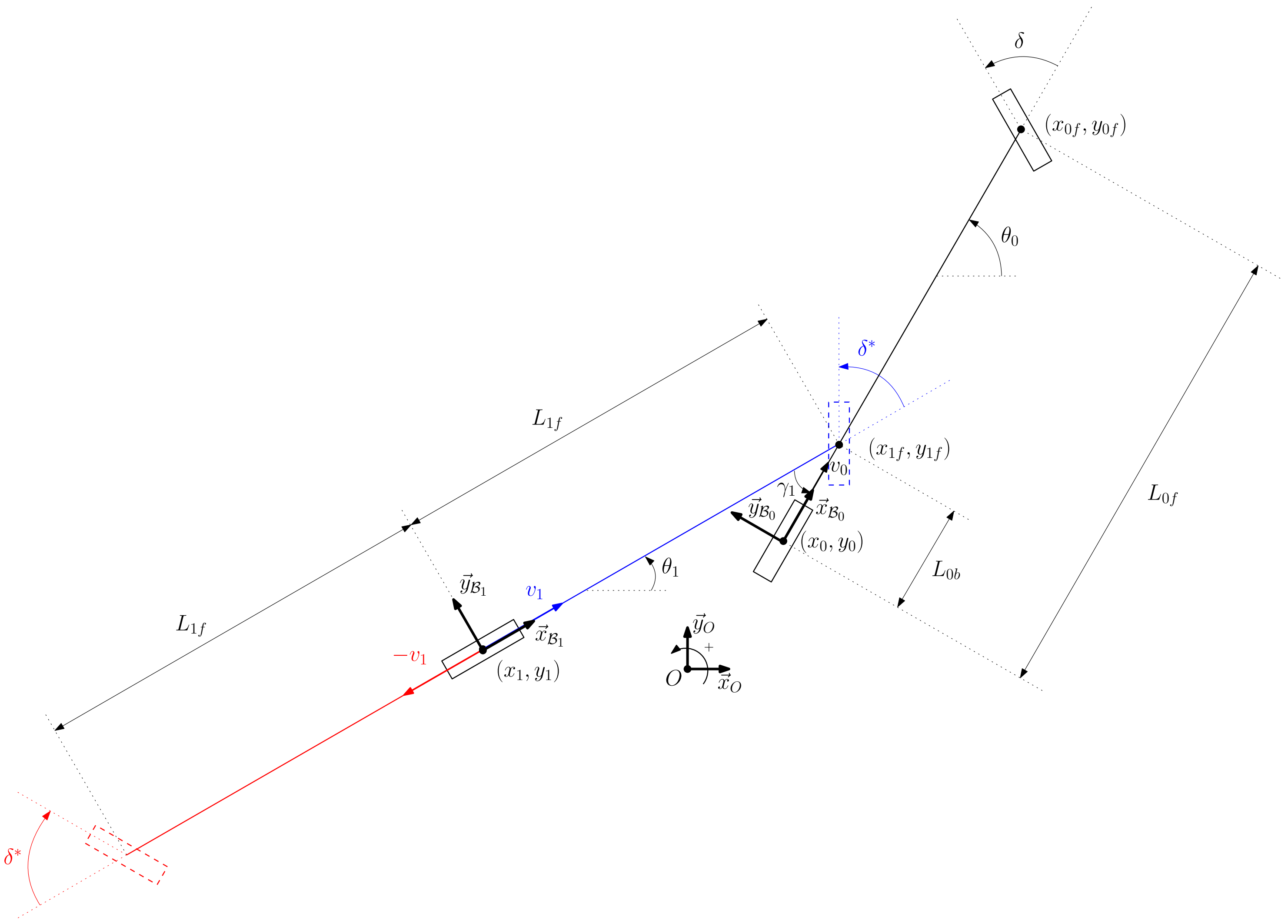


Figure 2-5: Inverse kinematic model of articulated vehicle combination. Source: Adapted from [10].
(The three tyres representing the three different axle groups are visualized as black rectangles. The red portion visualizes the virtual truck and the virtual steering angle. The blue portion visualizes the origin of the dimensions of the virtual truck.)

2-2 Modelling of a Human Driver

From a modelling perspective, the key components of the VISTA system are the vehicle and the driver. The kinematic vehicle model discussed in Section 2-1-1 describes the response of the vehicle when an input is given. It can also be used to compute the required input when a specific response is desired. A complication for the VISTA truck docking assistant is that the computed input is not directly applied to the system. First, the computed input is visualized to the driver and then the driver will try to execute it. The human behaviour inside the control loop adds a couple of complications. A human will exhibit a time delay when reacting. This is caused by the systems latency and the time it takes for the driver to process the information and react accordingly. This delay and any additional inaccuracies can be dealt with by predicting the driver's behaviour beforehand. The instructions to the driver can then be adjusted appropriately. To predict this behaviour, a suitable way to model the driver is required. A relatively simple model to represent the driver is given by the following transfer function:

$$H(s) = K_p \frac{(T_L s + 1)}{(T_l s + 1)(T_N s + 1)} e^{-\tau_r s}, \tag{2-15}$$

where K_p is the static gain, T_L is the lead time constant, T_l is the lag time constant, T_N is the neuromuscular lag and τ_r is the reaction time delay. This model was first presented in [21]. The idea behind this model is that the driver will try to anticipate on future events, while taking the past into account. The driver will thus act like a lead-lag compensator with a certain time delay. [22] states that humans are also able to adapt their control behaviour. Once adapted to the dynamics, humans can increase gain and decrease time delay, thereby influencing the properties of the total closed-loop system. This has led to idea of looking at the driver and vehicle as a combined system. It was observed that the driver will adapt himself in such a way that the combined system can be described with a single transfer function [23], independent of the specific input and output of the system. This formulation is known as the *crossover model*

$$H(s) \cdot G(s) = \frac{\omega_c}{s} e^{-s\tau_r}, \tag{2-16}$$

where the transfer function $H(s)$ again describes the dynamics of the human operator and transfer function $G(s)$ describes the dynamics of the vehicle. The assumption here is that the vehicle behaves as a linear system. However, drivers have also been observed to be capable of internalizing even nonlinear vehicle dynamics [24]. A disadvantage of the crossover model is that it only gives a suitable representation of the behaviour of the combined system within the immediate vicinity of the crossover frequency ω_c. More complex driver models exist, which try to capture the behaviour of the neuromuscular system [25]. This is most likely too detailed for the application in the VISTA project. However, it is interesting to notice that the results from modeling the neuromuscular system affirm the observation of the human ability to adapt to the physical environment they interact with. This means that the parameters of the driver model (2-15) depend on the specific use case. The parameters which most accurately describe the behaviour of a driver in the VISTA setup will therefore need to be determined in an experimental fashion.

2-3 Model Predictive Control

MPC is an advanced control technique, which is able to control Multiple-Input and Multiple-Output (MIMO) systems by successively solving constrained optimization problems. The controller uses a model of the system to predict its behaviour over a finite horizon. At each time-step, the optimal sequence of inputs is computed based on this open-loop prediction. The first element of this sequence is the control action which is applied in closed-loop to the system. The entire calculation is repeated at each subsequent time-step. This can be a very computational expensive task, especially with nonlinear models and when making predictions over a large *prediction horizon* N.

Therefore, MPC was primarily interesting for the control of slow systems, such as those that can be found in the process industry. In recent years however, an increase in computational resources is responsible for a rise in the popularity of MPC in other fields as well, such as fast real-time applications [12]. MPC can, for example, be used to control vehicles at high speed on slippery roads [26] and fast control of hexacopters [27], which are complex nonlinear systems. Research has also been conducted to use MPC for more practical vehicle parking scenarios by taking a signal delay into account [28].

2-3-1 Receding Horizon Principle

The iterative process of recomputing the optimal input over a fixed horizon at each time-step is called the *receding horizon principle*. At each time-step, an optimization problem is solved with to objective to find the sequence of control inputs, $\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(N-1)$, such that the total cost function is minimized. The total cost function is the sum of cost function $J(\mathbf{x}(k), \mathbf{u}(k))$ over all time-steps between and including the current time-step until the end of the prediction horizon N, while adhering to a set of constraints. The control action is then computed by solving the following optimization problem:

$$\min_{\mathbf{u}(0),\mathbf{u}(1),\ldots,\mathbf{u}(N-1)} \quad \sum_{k=0}^{N} J(\mathbf{x}(k), \mathbf{u}(k)), \tag{2-17}$$

$$\text{s.t.} \quad \mathbf{x}(0) = \mathbf{x}_{\text{init}}, \tag{2-18}$$

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)) \quad \forall k \in \{0, 1, \ldots, N-1\}, \tag{2-19}$$

$$g(\mathbf{x}(k), \mathbf{u}(k)) = 0 \quad \forall k \in \{0, 1, \ldots, N\}, \tag{2-20}$$

$$h(\mathbf{x}(k), \mathbf{u}(k)) \leq 0 \quad \forall k \in \{0, 1, \ldots, N\}. \tag{2-21}$$

The set of constraints ensure that, firstly, the predicted state of the system at the current time-step $\mathbf{x}(0)$ is equal to the latest provided state update $\mathbf{x}_{\text{init}}$. Secondly, they describe that the state of the system at each consecutive time-step is fully determined by the state of system and the input to the system at the previous time-step. The MPC controller predicts the state of a continuous system at a finite set of steps within the prediction horizon N. A continuous model, such as the model described by (2-1)-(2-4), therefore needs to be discretized with a sample time T_s. The discretized model of the system's dynamics is given by

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)) \quad \forall k. \tag{2-22}$$

Lastly, the set of constraints contain a subset of equality constraints $g(\mathbf{x}(k), \mathbf{u}(k)) = 0$ and inequality constraints $h(\mathbf{x}(k), \mathbf{u}(k)) \leq 0$ that describe the physical limitations of the system.

(Non)linear Model Predictive Control

The difficulty of solving this optimization problem greatly depends on the complexity of the cost function and the constraints. Existing MPC controllers can be divided in Linear MPC (LMPC) and Nonlinear MPC (NMPC) [29]. LMPC requires a linear or linearized model and is in general much faster than NMPC. However, a linearized model will lead to a rougher approximation of the real-world dynamics. A linear time-invariant model can be described by the well-known state-space representation:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k), \tag{2-23}$$

$$\mathbf{y}(k) = C\mathbf{x}(k) + D\mathbf{u}(k). \tag{2-24}$$

The state-space matrices A, B, C and D describe the evolution of the state $\mathbf{x}(k)$ and the output $\mathbf{y}(k)$ at time-step k as a result of input $\mathbf{u}(k)$. The output of the system at each time step over a horizon N can be reformulated by stacking the matrices as follows [30]:

$$Y_N = \mathcal{C}_N \mathbf{x}_{\text{init}} + \mathcal{T}_N U_N, \tag{2-25}$$

where $\mathcal{C}_N$, $\mathcal{T}_N$ and U_N are matrices constructed in the following manner:

$$Y_N = \begin{bmatrix} \mathbf{y}(1) \\ \mathbf{y}(2) \\ \mathbf{y}(3) \\ \vdots \\ \mathbf{y}(N) \end{bmatrix}, \quad \mathcal{C}_N = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^N \end{bmatrix}, \quad U_N = \begin{bmatrix} \mathbf{u}(0) \\ \mathbf{u}(1) \\ \mathbf{u}(2) \\ \vdots \\ \mathbf{u}(N-1) \end{bmatrix},$$

$$\mathcal{T}_N = \begin{bmatrix} CB & D & 0 & 0 & \cdots & 0 \\ CAB & CB & D & 0 & \cdots & 0 \\ CA^2B & CAB & CB & D & & 0 \\ \vdots & \vdots & & \ddots & \ddots & \\ CA^NB & CA^{N-1}B & CA^{N-2}B & \cdots & CB & D \end{bmatrix}.$$

Consider a cost function which aims to minimize the magnitude of the output and the magnitude of the control input. This leads to a reformulation of the cost function in (2-17):

$$\sum_{k=0}^{N} J(\mathbf{x}(k), \mathbf{u}(k)) = \sum_{k=0}^{N} \mathbf{y}^T(k) Q \mathbf{y}(k) + \sum_{k=0}^{N} \mathbf{u}^T(k) P \mathbf{u}(k), \tag{2-26}$$

where Q and P are positive definite weight matrices defining the relative importance of the reference output and the reference control input. Again by stacking matrices, this cost function can be rewritten as

$$\sum_{k=0}^{N} J(\mathbf{x}(k), \mathbf{u}(k)) = Y_N^T Q_N Y_N + U_N^T P_N U_N, \tag{2-27}$$

$$Q_N = \begin{bmatrix} Q & 0 & 0 & \cdots & 0 \\ 0 & Q & 0 & \cdots & 0 \\ 0 & 0 & Q & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & Q \end{bmatrix}, \qquad P_N = \begin{bmatrix} P & 0 & 0 & \cdots & 0 \\ 0 & P & 0 & \cdots & 0 \\ 0 & 0 & P & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & P \end{bmatrix}.$$

The objective function can now be written in the form of a quadratic programming function

$$\min_{U_N} \quad \frac{1}{2} U_N^T H U_N + \mathbf{f}^T U_N. \tag{2-28}$$

The quadratic term, given by symmetric matrix H, and the linear term, given by vector $\mathbf{f}$ can be derived as follows:

$$H = \mathcal{T}_N^T Q_N \mathcal{T}_N + P_N, \tag{2-29}$$

$$\mathbf{f} = \mathcal{T}_N^T Q_N^T \mathcal{C}_N \mathbf{x}_{\text{init}}. \tag{2-30}$$

A linearization based approach thus enables the problem to be rewritten in a quadratic programming problem which results in a reduction of the computation time [12]. A linearization of a system is always done at a given reference point. Different possibilities of reference points include the current operating point or closest point on the reference path. With a correct choice of reference point, the LMPC controller might be able to make predictions which highly resemble the real-world dynamics. However, for some cases an NMPC controller can be better suited for predicting the dynamics of a nonlinear system. Solving nonlinear optimization problems is unfortunately far less straightforward. The performance of the controller highly depends on the solver used. In [31], an overview of different solvers is given. Many software packages, such as FORCES PRO [32], rely on automatic code generation. This allows for efficient self-contained linear algebra routines, but reduces the flexibility of the controller. The big trade-off influencing the choice of solver is the one between flexibility, memory and speed.

2-3-2 Model Predictive Contouring Control

In the MPC objective function, it is possible to compute the actions which minimize the error between the reference path and the predicted trajectory of the vehicle. Often the reference path lacks information about the way the vehicle should progress along the path. In that case, the MPC could find that the optimal solution is for the vehicle to stop somewhere on the reference path, as the distance to the reference path is then still minimized. Therefore, also a certain velocity of the vehicle is desired. The Model Predictive Contouring Control (MPCC) scheme, presented in [33], is able to decouple these objectives. The objective of the contouring problem is to steer the position of the vehicle along a continuously differentiable and bounded two-dimensional geometric reference path. The control scheme relies on a path parameter s, which keeps track of the vehicle's progression along the reference path. The path parameter defines the location on the reference path which is closest to the current coordinates of the vehicle.

The closest location on the reference path is the point at which the distance between the path and the vehicle is minimal. This position is important, because it determines the current error between the vehicle and the reference path. The direct computation of this path parameter s is computationally expensive. However, a good approximation can be made by stating

$$s(k+1) = s(k) + v(k)T_s, \tag{2-31}$$

where $v(k)$ is the longitudinal velocity of the vehicle at time-step k. The variable s then represents an approximation of the traveled distance along the reference path. The approximation introduces two errors [34]. Firstly, a lag error, which is defined as the longitudinal error with respect to the path's abscissa along the path's tangent at the estimated position. Secondly, a contouring error, which is defined as the lateral deviation of the vehicle position from the estimated position projected onto the path normal. In the cost function of the MPCC problem, a tracking cost $J_{\text{t}}(\mathbf{x}(k), s(k))$ is included to penalize the both errors. Other terms of the cost function may include $J_{\text{s}}(\mathbf{x}(k), u(k))$, to penalize deviation from the reference velocity, and $J_{\text{i}}(\mathbf{u}(k))$, to penalize the magnitude of the inputs. The MPCC optimization problem will then take the following form [35]:

$$\begin{aligned}
\min_{\mathbf{u}(0),\mathbf{u}(1),\ldots,\mathbf{u}(N-1)} \quad & \sum_{k=0}^{N} J_{\text{t}}(\mathbf{x}(k), s(k)) + J_{\text{s}}(\mathbf{x}(k), \mathbf{u}(k)) + J_{\text{i}}(\mathbf{u}(k)), && && (2\text{-}32)\\
\text{s.t.} \quad & \mathbf{x}(0) = \mathbf{x}_{\text{init}}, && && (2\text{-}33)\\
& \mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)) && \forall k \in \{0, 1, \ldots, N-1\}, && (2\text{-}34)\\
& s(k+1) = s(k) + v(k)T_s && \forall k \in \{0, 1, \ldots, N-1\}, && (2\text{-}35)\\
& g(\mathbf{x}(k), \mathbf{u}(k)) = 0 && \forall k \in \{0, 1, \ldots, N\}, && (2\text{-}36)\\
& h(\mathbf{x}(k), \mathbf{u}(k)) \leq 0 && \forall k \in \{0, 1, \ldots, N\}. && (2\text{-}37)
\end{aligned}$$

In the VISTA project's practical application, the controller will be provided with a precomputed reference path. This reference path consists of a set of waypoints, which indicate a desired sequence of coordinates and a desired velocity for the vehicle at each waypoint. For the proposed MPC-based driver assistance system, the MPCC scheme is used to implement the tracking of a reference path and a reference velocity in a single cost function.

2-3-3 Obstacle Avoidance

An advantage of MPC is that it allows for real-time obstacle avoidance. Collision avoidance can be ensured by incorporating it as optimization constraints. A convenient formulation of collision avoidance with hard constraints is based on the following notion of distance [36]:

$$\text{dist}(\mathcal{E}(\mathbf{c}), \mathcal{O}) = \inf\{\|t\| \mid t \in \mathbb{R}^n, (\mathcal{E}(\mathbf{c}) + t) \cap \mathcal{O} \neq \emptyset\}, \tag{2-38}$$

where the region occupied by the vehicle $\mathcal{E}(\mathbf{c}) \subseteq \mathbb{R}^n$ is determined by the configuration of the vehicle $\mathbf{c}$. Each obstacle is represented by a convex set $\mathcal{O}^{(m)} \subseteq \mathbb{R}^n$, for $m = 1, \ldots, M$, where M is the total number of obstacles. The union of these convex sets describes the space occupied by obstacles $\mathcal{O}$. The inf function denotes the infimum function. Collision avoidance is ensured by requiring $\text{dist}(\mathcal{E}(\mathbf{c}), \mathcal{O}) > d_{\min}$. The distance $d_{\min} \geq 0$ is needed to provide a safety margin between the vehicle and the obstacles. Real-time collision avoidance is not yet incorporated in the proposed MPC-based driver assistant. Instead, obstacles are avoided by accurately tracking a collision-free reference path.

Chapter 3

Proposed MPC-based Driver Assistant

In this chapter, the Model Predictive Control (MPC) control scheme for a truck docking assistance system will be presented. The system is specifically designed for a scenario that comprises a driver performing an aided bi-directional parking manoeuvre with a truck-trailer combination. The driver is aided in the form of certain visual instructions. The controller's function is to compute instructions such that they actually offer support to the driver and ensure that he or she is able to follow a reference path. This reference path is computed offline before the start of the manoeuvre and consists of waypoints starting at the initial position of the vehicle and ending in the desired goal region.

3-1 System Overview

To control a system, it is important to analyse the system structure and the controller's position within the system. An overview of the structure is given in Figure 3-1. The current state of the system $\mathbf{x}$ and the reference output $\mathbf{y}_{\text{ref}}$ are fed to the MPC controller. In this case, the state of the system $\mathbf{x}$ comprises the configuration of the vehicle $\mathbf{c}$ and the state of the driver $\mathbf{x}_\text{d}$. The output of the system $\mathbf{y}$ is a subset of the system's state, namely the configuration of the vehicle $\mathbf{c}$. The reference output indicates the desired the position for trailer axle's x- and y-coordinate based on the reference path; this changes over time.

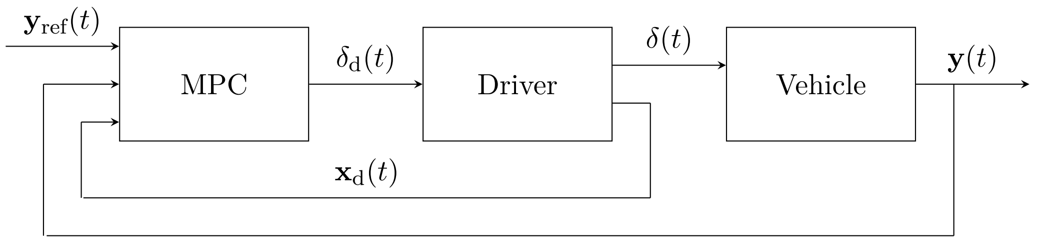


Figure 3-1: System structure of MPC-based truck docking assistant

The output of the controller is the steering angle instruction given to the driver δ_d. As a response to this instruction, the driver will set the actual steering angle of the vehicle δ. In turn, this steering angle influences the configuration of the vehicle, thus the output of the system. At this point, it is assumed that any additional disturbances are absent. However, the drivers behaviour will result in a difference between the given instruction they are given and the actual steering actions they perform. This behaviour of the driver is included in the design of the MPC controller in the form of a driver model.

The control action is determined by solving an optimization problem each time a new update of the current state of the system $\mathbf{x}_{\text{init}}$ is provided. In the real-life applications of the VIsion Supported Truck docking Assistant (VISTA) system, this state update is provided by measurements, but during simulations the state is assumed to be exactly known. The objective of the optimization problem is to find the optimal sequence of control inputs that minimizes the objective function, while adhering to the constraints of the system. This is described by the optimization problem in (2-17)-(2-21), in Section 2-3-1. The continuous nonlinear dynamics of the system, comprising the vehicle and the driver, are presented in Section 3-2. The discretization of these dynamics is used to predict the system's state evolution over a finite time horizon N. The structure of the cost function is explained in Section 3-3 and the constraints of the optimization problem are discussed in Section 3-4. In Section 3-5 the computation of the reference output $\mathbf{y}_{\text{ref}}$ is presented and the usage of online constraints is explained in Section 3-6. Lastly, the process of identification of the driver model is discussed in Section 3-7.

3-2 System Dynamics

The proposed MPC scheme makes use of the kinematic truck-trailer model as described in Section 2-1-1 and the driver model as shown in (2-15), which is described in Section 2-2. The states of this system are given in the following vector:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ \theta_1(t) \\ \gamma_1(t) \\ x_{\text{d}1}(t) \\ x_{\text{d}2}(t) \end{bmatrix}. \tag{3-1}$$

The states x_1, y_1, θ_1 and γ_1 are the x-coordinate of the trailer axle, the y-coordinate of the trailer axle, the yaw-angle of the trailer and the articulation angle, respectively. The states $x_{\text{d}1}$, $x_{\text{d}2}$ are the states of the driver model that arise when converting the transfer function from (2-15) to its state-space representation. The continuous time state-space representation of the driver model is

$$\dot{\mathbf{x}_{\text{d}}}(t) = A\mathbf{x}_{\text{d}}(t) + B\delta_{\text{d}}(t), \tag{3-2}$$
$$\delta(t) = C\mathbf{x}_{\text{d}}(t) + D\delta_{\text{d}}(t), \tag{3-3}$$

where δ_{d} is the steering angle instructed to the driver, δ is the actual steering angle of the vehicle and $\mathbf{x}_{\text{d}}(t) = \begin{bmatrix} x_{\text{d}1}(t) & x_{\text{d}2}(t) \end{bmatrix}^T$. The A, B, C and D matrices are constructed such that the state-space realization is in controllable canonical form:

$$A = \begin{bmatrix} -\frac{T_l+T_N}{T_l T_N} & -\frac{1}{T_l T_N} \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{3-4}$$

$$C = \begin{bmatrix} \frac{KT_L}{T_l T_N} & \frac{K}{T_l T_N} \end{bmatrix}, \quad D = 0. \tag{3-5}$$

The continuous time dynamics of the system with this combined vehicle and driver model, as visualized in Figure 3-1, are

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} v_0(t)\left(\cos\gamma_1(t)\cos\theta_1(t) - \frac{L_{0b}}{L_{0f}}\cos\theta_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\cos\gamma_1(t)\sin\theta_1(t) - \frac{L_{0b}}{L_{0f}}\sin\gamma_1(t)\sin\theta_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\frac{1}{L_{1f}}\sin\gamma_1(t) + \frac{L_{0b}}{L_{0f}L_{1f}}\cos\gamma_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\frac{1}{L_{0f}}\tan\delta(t) - \frac{1}{L_{1f}}\sin\gamma_1(t) - \frac{L_{0b}}{L_{0f}L_{1f}}\cos\gamma_1(t)\tan\delta(t)\right) \\ -\frac{T_l+T_N}{T_l T_N}x_{\mathrm{d}1}(t) - \frac{1}{T_l T_N}x_{\mathrm{d}2}(t) + \delta_{\mathrm{d}}(t) \\ x_{\mathrm{d}1}(t) \end{bmatrix}. \tag{3-6}$$

Here, the actual steering of the truck δ can be derived from (3-3) and (3-5):

$$\delta(t) = \frac{KT_L}{T_l T_N}x_{d1}(t) + \frac{K}{T_l T_N}x_{d2}(t). \tag{3-7}$$

The yaw angle of the truck can be computed by considering the articulation angle, as follows:

$$\theta_0(t) = \gamma_1(t) + \theta_1(t). \tag{3-8}$$

The control input that can be applied to the system is

$$\mathbf{u}(t) = \begin{bmatrix} \delta_{\mathrm{d}}(t) \\ v_0(t) \end{bmatrix}. \tag{3-9}$$

It can be noticed that v_0, the longitudinal velocity of the truck, is regarded as an input to the system. This means it is assumed that the MPC controller can directly determine the velocity of the truck. In reality, however, the velocity of the truck is fully determined by the actions of the driver and there is no continuous instruction regarding the optimal velocity send to the driver. The velocity of the truck is regarded as an input to the system, because some manoeuvres require a specific regulation in velocity in order to accurately track the reference path. The controller is then able to find solutions which include such manoeuvres. It is assumed that the human driver will intuitively regulate the velocity and reduce the velocity when needed. Therefore, it is needed that the optimal velocity computed by the MPC controller is similar to the actual velocity regulation of the driver.

The output of the system contains the first four states of the system. This can be written as

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ \theta_1(t) \\ \gamma_1(t) \end{bmatrix}. \tag{3-10}$$

The output consists of states with reference values that need to be tracked and states of which the constraints change online. The output does not appear in the optimization problem (2-17)-(2-21), as it is a subset of the system's state.

3-3 Cost Function

The total cost function to be minimized is a sum of the costs at each prediction stage in the control horizon. The costs at a specific stage are computed as

$$J(\mathbf{x}(k), \mathbf{u}(k)) = J_y(\mathbf{x}(k)) + J_u(\mathbf{u}(k)) + J_{\Delta u}(\mathbf{u}(k)). \tag{3-11}$$

This cost function consists of three terms. The terms J_y, J_u and $J_{\Delta u}$ are responsible for output reference tracking, input reference tracking and input change rate suppression, respectively. The relative importance of each term is determined by the tuning weights. All weights in the overall cost function will in their own way influence on the performance of the MPC controller. The choice of weights for the truck docking assistant application and the effect different choices of weights have on the overall performance is researched and discussed in Section 5-3.

3-3-1 Output reference tracking

The output reference tracking term is needed in the cost function (3-11) to minimize the error between the system outputs and the reference values for the output $\mathbf{y}_{\text{ref}}$. The reference output consists of the desired coordinates of the trailer axle (x_1, y_1) in order to follow the precomputed reference path. The term in the cost function responsible for output reference tracking cost function is computed as

$$J_y(\mathbf{x}(0)) = 0, \tag{3-12}$$

$$J_y(\mathbf{x}(k)) = \Big(w_{\mathbf{y}}[x_{1,\text{ref}}(k) - x_1(k)]\Big)^2 + \Big(w_{\mathbf{y}}[y_{1,\text{ref}}(k) - y_1(k)]\Big)^2 \quad \forall k \in \{1, 2, \dots, N\}, \tag{3-13}$$

where $x_{1,\text{ref}}(k)$ and $y_{1,\text{ref}}(k)$ are the reference values for trailer axle's x- and y-coordinate at the kth prediction horizon step, and $w_{\mathbf{y}}$ is the tuning weight for output reference tracking.

In an ideal scenario, the reference output is equal to the desired goal configuration of the parking manoeuvre. In that case, the desired goal coordinates are fed to controller and the controller tries to steer the vehicle to this location directly, without the use of a reference path. However, in most cases this is infeasible due to three reasons. Firstly, most parking manoeuvres will include multiple changes of direction and it is very difficult to find such a solution in the form of an optimization problem. Secondly, the controller does not take the obstacles into account when computing the control action. Therefore, collision avoidance is not guaranteed. Lastly, the probability for the controller to find feasible parking manoeuvres, with just a goal location given, would depend greatly on the prediction horizon. If the prediction capabilities of the MPC are limited, it is less probable for the controller to find any feasible solutions and will get stuck in local optima. Therefore, the bi-directional reference path provided by the motion planning model is used to compute the output reference values online. This reference path is guaranteed to be collision free, which means accurately tracking the reference path can guarantee collision avoidance.

The reference outputs $x_{1,\text{ref}}(k)$ and $y_{1,\text{ref}}(k)$ at each step k in the prediction horizon are computed online based on the current position of the vehicle and the provided reference path. The computation is derived from the Model Predictive Contouring Control (MPCC) scheme presented in Section 2-3-2. This is further discussed in Section 3-5.

3-3-2 Input reference tracking

The second term in the cost function (3-11) is needed to keep the control inputs close to the reference input. In this case, the reference input indicates a preference for a certain velocity. The input reference tracking term in the cost function could also specify a preference of steering angle instructed to the driver, as this is also an input to the system. However, this has little added value as there is no steering angle preference. This term is computed as:

$$J_u(\mathbf{u}(k)) = \Big(w_{v_0}[v_{0,\text{ref}}(k) - v_0(k)]\Big)^2 \qquad \forall k \in \{0, 1, \ldots, N-1\}, \tag{3-14}$$

$$J_u(\mathbf{u}(N)) = 0, \tag{3-15}$$

where $v_{0,\text{ref}}(k)$ is the reference value for truck's longitudinal velocity at the kth prediction horizon step and w_{v_0} is the tuning weight for truck's velocity control input. The reference value for the velocity is fully determined by the desired driving direction of the current segment of the reference path. If the direction of the current segment is indicated to be forwards

$$v_{0,\text{ref}}(k) = v_\text{f} \quad \forall k \in \{0, 1, \ldots, N-1\}, \tag{3-16}$$

and if the direction of the current segment is indicated to be backwards

$$v_{0,\text{ref}}(k) = v_\text{b} \quad \forall k \in \{0, 1, \ldots, N-1\}. \tag{3-17}$$

Here, v_f is the desired forward driving velocity of the truck and v_b is the desired velocity of the truck when driving in reverse. There is a need for a reference velocity to be specified, as this ensures that slow driving or stopping is penalized and progressing along the reference path is encouraged. However, it is still possible for the MPC controller to find solutions where the optimal velocity deviates from the reference velocity. This is beneficial as there is a limit to the steering speed. Therefore, a lower velocity can improve the reference path tracking, as the steering speed then increases relative to the driving velocity.

3-3-3 Input change rate suppression

The last term in the cost function (3-11) is needed to suppress the rate of change of the inputs. This indicates the preference for small adjustments of the steering angle instructions. The input change rate suppression term in the cost function could also specify a preference of small changes in the velocity of the truck. However, this is not necessary, as the velocity is already near constant and the computed optimal velocity is not actually send as an instruction to the driver. The input change rate suppression term of the cost function is computed as:

$$J_{\Delta u}(\mathbf{u}(k)) = \Big(w_{\Delta\delta}[\delta_\text{d}(k) - \delta_\text{d}(k-1)]\Big)^2 \qquad \forall k \in \{0, 1, \ldots, N-1\}, \tag{3-18}$$

$$J_{\Delta u}(\mathbf{u}(N)) = 0, \tag{3-19}$$

where $w_{\Delta\delta}$ is the tuning weight for steering rate of change suppression. This weight determines the penalization of large angular velocities of the steering wheel. Large penalization can be beneficial, because drivers prefer smoother steering actions. Very swift changes of the steering angle instructions are difficult for a driver to comprehend. Also, if the driver responds with heavy steering actions, it will have a significant negative impact on the tyre wear [37].

3-4 Constraints

The final component of the proposed MPC controller is the set of constraints representing the physical limitations of the system. These include limitations on the maneuverability of the truck-trailer combination as well as constraints ensuring that the vehicle ends up with the correct yaw angle. The constraints can be divided in constraints on the systems states, control inputs, the control inputs rate of change and the systems outputs.

3-4-1 State Constraints

Firstly, there is one state of the system that needs to be limited. This is the fourth state, the articulation angle γ_1. The articulation angle is limited as the truck and trailer would otherwise collide with each-other, which is called jackknifing. This constraint to a maximum absolute articulation angle $\gamma_{\max}$ can be written as

$$-\gamma_{\max} \leq \gamma_1(k) \leq \gamma_{\max} \qquad \forall k \in \{0, 1, \dots, N\}. \tag{3-20}$$

3-4-2 Input Constraints

Secondly, both inputs should be constrained, as there are limitations on the steering angle and the velocity of the truck. The steering angle is physically limited. Therefore, the steering angle instructions to the driver should also be limited. The velocity of the truck has, of course, a physical limitation, the top speed. However, as the truck-trailer combination will perform a parking manoeuvre, the velocity of the truck should be further limited. Also, the kinematic model used is only valid when driving at low velocities. The constraints to a maximum absolute steering angle $\delta_{\max}$, a minimum velocity of the truck $v_{\min}$ and a maximum velocity of the truck $v_{\max}$ are mathematically written as

$$-\delta_{\max} \leq \delta_{\mathrm{d}}(k) \leq \delta_{\max} \qquad \forall k \in \{0, 1, \dots, N-1\}, \tag{3-21}$$

$$v_{\min} \leq v_0(k) \leq v_{\max} \qquad \forall k \in \{0, 1, \dots, N-1\}. \tag{3-22}$$

3-4-3 Input Rate of Change Constraints

Thirdly, the rate of change of the steering angle instructions is constrained. This constraint is similar to the input rate change suppression term in the cost function that is presented in Section 3-3-3. The difference is that this is a hard constraint on the absolute rate of change of the steering angle instructions. The constraint on the rate of change of the instructions ensure that the changes are coherent with the physical capabilities of the driver. This constraint to a maximum absolute steering rate $\omega_{\max}$ can be written as

$$-T_s\omega_{\max} \leq \delta_{\mathrm{d}}(k+1) - \delta_{\mathrm{d}}(k) \leq T_s\omega_{\max} \qquad \forall k \in \{0, 1, \dots, N-1\}. \tag{3-23}$$

To limit the amount of change of the steering angle, it is required to include sample time T_s in the equation. The sample time is defined as the time between the different prediction stages. Therefore, the amount of change of the steering angle between subsequent prediction stages is constraint.

3-4-4 Output Constraints

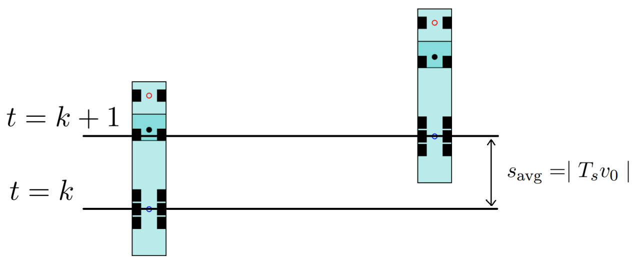
Lastly, it is possible to constrain the allowed outputs of the system. As the outputs of the system are a subset of the states of the system, these constraints are already taken account of, namely the constraint on the articulation angle. However, when nearing the end of the parking manoeuvre, an additional set of constraints was found to be desired. While performing the manoeuvre, the orientation of the trailer is not always aligned with the reference path and most of the time this is also not required. However, this needed at the end of the parking manoeuvre, because the trailer needs to line up with the loading dock. Also, the articulation angle needs to be constrained, such that the vehicle will not hinder subsequent vehicles from parking. To incorporate these constraints such that they only take effect when the manoeuvre is nearing its end, an online change of constraints needs to be utilized at the correct time. How this is done, is discussed in Section 3-6.

3-5 Reference Output

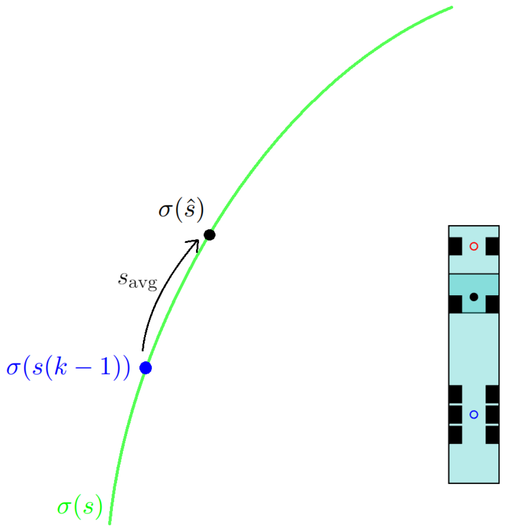
In the current VISTA concept, a Real Time Localization System (RTLS) will detect the initial location and orientation of the vehicle. A path planner will then compute a path from this starting pose to the desired parking spot. In this case, the reference paths are generated by the latest path planner specifically designed as a part of the VISTA project, which is presented in [18]. The generated reference paths are fed to the path tracking module as a set of waypoints that is split up in different segments, where each segment is indicated to be either a forward or a reversing manoeuvre. The waypoints indicate the desired sequence of x- and y-coordinates of the trailer axle (x_1, y_1). They are used to create the reference values for the MPC controller. The reference values are computed at each control interval on the basis of the current state of the system and the reference path.

The objective of the controller is to track a reference path and a reference velocity. Therefore, a setup similar to the MPCC scheme, as presented in [35], is implemented. MPCC relies on computing the path parameter s. The path parameter corresponds to the location on the path at which the distance between the path and current location of the vehicle is minimal. Computing the euclidean distance from every point on the path to the current location of the vehicle and then computing the minimum distance is a time-consuming procedure. Therefore, first an estimation of the current path parameter is made. It is assumed that the vehicle will track the reference path fairly closely. The path parameter will be approximately equal to the distance traveled by the vehicle along the path. At each time-step, the path parameter can thus be estimated with the sum of the previous path parameter and the distance traveled in the previous time-step. This formula is shown in (2-31). The exact path parameter can then easily be computed by just computing the euclidean distance to a selection of points on the path near the estimated path parameter.

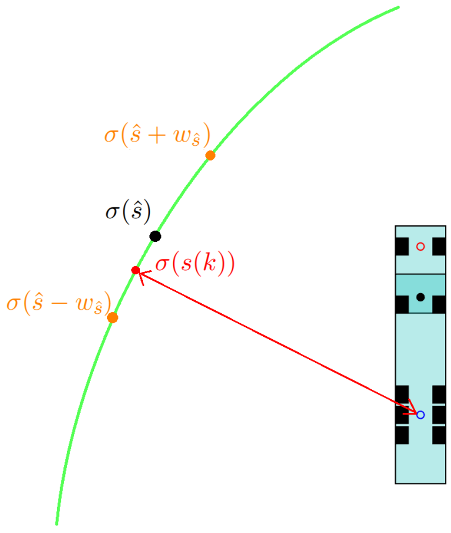
This path parameter is then used to compute the local output reference for the MPC controller at each time-step. The local reference output prescribes the desired output of the system at each stage of the current prediction horizon. This means the local output reference changes at each time-step as the vehicle progresses along the reference path. The local reference path can be computed as the part of the path between the current location of the vehicle and the location where the vehicle is predicted to be at the end of the prediction horizon N. It is assumed that the velocity of the vehicle is nearly constant. Therefore, the local reference path can be computed with just the current location of the vehicle. To aid this process, the reference path consisting of waypoints is first converted to a piece-wise polynomial structure $\sigma(s)$ defined by its break points, polynomial coefficients and dimension. The reference output at control interval k is computed as shown in Algorithm 2. The steps of the algorithm are visualized in Figure 3-2, where s_{avg} is the average travel distance between subsequent prediction intervals.



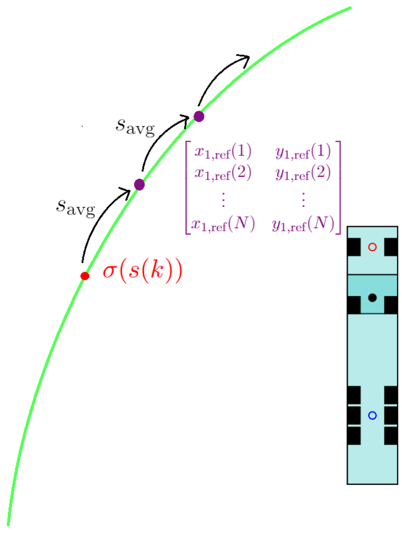
(a) Line 1.



(b) Line 2.


$\sigma(\hat{s} + w_{\hat{s}})$
$\sigma(\hat{s})$
$\sigma(s(k))$
$\sigma(\hat{s} - w_{\hat{s}})$

(c) Line 3.


s_{avg}
s_{avg}
$\begin{bmatrix} x_{1,\text{ref}}(1) & y_{1,\text{ref}}(1) \\ x_{1,\text{ref}}(2) & y_{1,\text{ref}}(2) \\ \vdots & \vdots \\ x_{1,\text{ref}}(N) & y_{1,\text{ref}}(N) \end{bmatrix}$
$\sigma(s(k))$

(d) Line 4-6.

Figure 3-2: Local reference output algorithm visualization.

Algorithm 2 Compute local reference output

```
    Input: $x_1(k)$, $y_1(k)$, $s(k-1)$, $w_{\hat{s}}$, $\sigma(s)$, $v_0$, $T_s$
    Output: $x_{1,\text{ref}}(1)$, $x_{1,\text{ref}}(2)$, $\ldots$, $x_{1,\text{ref}}(N)$, $y_{1,\text{ref}}(1)$, $y_{1,\text{ref}}(2)$, $\ldots$, $y_{1,\text{ref}}(N)$, $s(k)$
1: $s_{\text{avg}} \leftarrow \mid T_s v_0 \mid$
2: $\hat{s} \leftarrow s(k-1) + s_{\text{avg}}$
3: $s(k) \leftarrow \arg\min \{\|(x_1, y_1) - \sigma(s_i)\|_2 \mid s_i \in [\hat{s} - w_{\hat{s}}, \hat{s} + w_{\hat{s}}]\}$
4: for all $i \in \{1, 2, \ldots, N\}$ do
5:     $[x_{1,\text{ref}}(i), y_{1,\text{ref}}(i)] \leftarrow \sigma(s(k) + i s_{\text{avg}})$
6: end for
```

Here, $w_{\hat{s}}$ is the window of range where is looked for the exact path parameter, with respect to the estimated path parameter $\hat{s}$. The outputs $x_{1,\text{ref}}(i)$ and $y_{1,\text{ref}}(i)$ are the current reference values at each stage of the prediction horizon for coordinates x_1 and y_1, respectively. The x- and y-coordinates corresponding to a path parameter s are computed by evaluating the piece-wise polynomial structure $\sigma(s)$ using the MATLAB function `ppval`.

3-6 Online Constraints

In Section 3-4-4, it is mentioned that there is a requirement that some of the constraints only need to take effect when nearing the end of the parking manoeuvre. The output of the system consists of, among other things, the articulation angle and the yaw angle of the trailer. For a truck-trailer parking manoeuvre to be successful, it is important that the trailer lines up with the loading dock and that the articulation angle is considerably small. This means it is required to set explicit constraints on the yaw angle of the trailer and the articulation angle at the end of the parking manoeuvre. However, these constraints should not apply when the vehicle is still some distance away from the loading dock. Fortunately, the MPC setup allows to change the constraints online and even to specify the constraints for each specific stage in the prediction horizon.

Ideally, the online constraints are implemented such that they take effect at the prediction stage where the vehicle is predicted to reach the loading dock. In the previous section, it is explained how it is possible to compute the approximate current location of the vehicle on the reference path. With this knowledge about the progression along the reference path, it can be determined how much time is left to reach the end of the current path segment. It is therefore known if the end of the parking manoeuvre is likely to be reached within the prediction horizon. Additionally, it can be determined at which prediction stage the vehicle is predicted to reach this.

If the current segment is the last segment of the parking manoeuvre, the constraints are determined online. At the current control interval, $k = 0$, the vehicle is predicted to reach the parking spot within the prediction horizon if

$$s_{\text{end}} < s(0) + N \mid T_s v_0 \mid, \tag{3-24}$$

where s_{end} is the length of the final segment.

The predicted amount of stages left before reaching the desired end position can then be computed as:

$$N_{\text{end}} = \text{round}\left(\frac{s_{\text{end}} - s(0)}{N \mid T_s v_0 \mid}\right). \tag{3-25}$$

The online output constraints are written as:

$$\begin{aligned}
-\infty \leq \theta_1(k) &\leq \infty && \forall k \in \{0, 1, \ldots, N_{\text{end}} - 1\}, && (3\text{-}26)\\
\theta_{\text{end}} - \epsilon_\theta \leq \theta_1(k) &\leq \theta_{\text{end}} + \epsilon_\theta && \forall k \in \{N_{\text{end}}, N_{\text{end}} + 1, \ldots, N\}, && (3\text{-}27)\\
-\gamma_{\max} \leq \gamma_1(k) &\leq \gamma_{\max} && \forall k \in \{0, 1, \ldots, N_{\text{end}} - 1\}, && (3\text{-}28)\\
\gamma_{\text{end}} - \epsilon_\gamma \leq \gamma_1(k) &\leq \gamma_{\text{end}} + \epsilon_\gamma && \forall k \in \{N_{\text{end}}, N_{\text{end}} + 1, \ldots, N\}. && (3\text{-}29)
\end{aligned}$$

Here, γ_{end} and θ_{end} are the desired final articulation angle and yaw angle of the trailer, respectively. The values for ϵ_γ and ϵ_θ indicate the maximum allowed error for both constraints.

3-7 Driver Model Identification

As discussed in Section 3-1, and shown in Figure 3-1, a significant element of the proposed MPC-based driver assistant is the driver model used to predict the driver's behaviour. The structure of the model which is used is shown in (3-2)-(3-5). The defining parameters of this model are τ_r, T_N, K, T_L and T_l. Here, τ_r, the reaction time delay, and T_N, the neuromuscular lag, are considered physical attributes of the driver, while K, T_L and T_l are parameters associated with the controlled vehicle [38]. In order to include the driver model in the MPC setup, first the defining parameters have to be estimated. A driver model can be estimated with by recording and analysing driver input and output data. Here, the input data is considered to be the steering instruction given to the driver, and the output data is considered to be the actual steering angle actuated by the driver. The driver model is estimated with the use of the MATLAB `System Identification` application. The format of the model selected in the application is a process model with 2 poles, 1 zero and a constant delay. This corresponds to the presented driver model. The process model is estimated with an Interior-point optimization algorithm. The algorithm computes the parameters such that the difference between the output obtained by simulating the process model and the actual output is minimized.

3-7-1 Driver Reaction Time

It can be noticed that the state derivative function (3-6) used in the formulation of the MPC controller does not include the driver reaction time τ_r. This reaction time is modeled as a time-delay in the system, as seen in (2-15). The state derivative describes the continuous dynamics of the system in the time domain. Therefore, it only allows to describe the relationship between the state dynamics and the current control input. This means it is impossible to construct the controller such that it takes into account that the current state transformation is driven by a previous control input. However, the time-delay is an important part of the behaviour of the driver. Fortunately, the MPC structure offers another approach to tackle this problem.

As mentioned, the idea behind the MPC controller is to determine the control action by solving the optimization problem as presented in Section 3-1. The objective of this problem is to find a solution which minimizes the objective function. The solution provides an optimal sequence of inputs within the prediction horizon, $\hat{\mathbf{u}}(0), \hat{\mathbf{u}}(1), \ldots, \hat{\mathbf{u}}(N-1)$. In traditional MPC approaches, the first element of this sequence is then applied to the system. However, to cope with the time-delay present in the system, it is also possible to select a subsequent element as input to the system. This element is computed to be optimal at some future point, thus applying this input will counteract the delay introduced by the driver. The steering instruction send to the driver is determined by the prediction model sample time and the driver reaction time as follows:

$$\delta_{\mathrm{d}} = \hat{\mathbf{u}}\left(\mathrm{round}\left(\frac{\tau_r}{T_s}\right)\right) \tag{3-30}$$

The future input is optimal only according to the prediction made by the controller. Therefore, this approach is only valid if the prediction about the state of the system is close to the actual state of the system after this time-delay. The effectiveness of this technique to compensate the driver's delay is tested and discussed in Section 5-6.

3-7-2 Driver Behavioural Changes

It is expected that the performance of the assistance system will improve when the driver model is included. However, as discussed in Section 2-2, humans are very capable of adapting to the dynamics of the system. Their behaviour might therefore change in such a drastic way that it is impossible to capture their dynamics with one driver model. The driver's behaviour might also vary when a driver model is included or when different driver models are used, because this will actually change the overall assistant system's dynamics. Moreover, it is possible that the drivers behaviour changes due to a number of other causes. Firstly, the behaviour might change because the driver is first confused by the Human Machine Interface (HMI), but eventually gets used to it. Secondly, the alertness of the driver could vary over time, thus influencing their behaviour. Moreover, there could be a multitude of additional physical and mental conditions effecting the current behaviour of the driver. Therefore, it is important to test the driver's behaviour multiple times and estimate the model at different time instances. One important test includes observing whether an estimated driver model is similar to the driver model used at that test. This is necessary in order to check if the driver model is actually valid, because the driver's behaviour is then approximately consistent when that model is used.

Another important aspect to test is whether a driver model is interchangeable between different drivers. On one hand, it can be expected that the driving behaviour of two different professional drivers is fairly similar. All drivers have a certain shared know-how on how to operate such a vehicle and the average reaction time of different humans is, in general, quite similar [39]. On the other hand, it can be expected that there are always extreme cases in human behaviour. For example, a skilful, experienced driver might behave quite different from a less competent, inexperienced driver. Therefore, it is interesting to investigate if the behaviour is still similar enough to be captured in one driver model or if different driver models for different drivers are required. Moreover, the drivers behaviour might significantly differ when driving forward or when reversing. It is therefore also interesting to test if identifying and using distinct driver models for forward driving and reversing manoeuvres is beneficial.

Chapter 4

Test Setup

The proposed controller scheme is tested in various simulation environments, as immediate testing with a real truck-trailer combination is quite cumbersome. In this chapter, an extensive overview of the simulation environments and their usage for different test scenarios is presented. Additionally, the hardware and software used for the simulations are described.

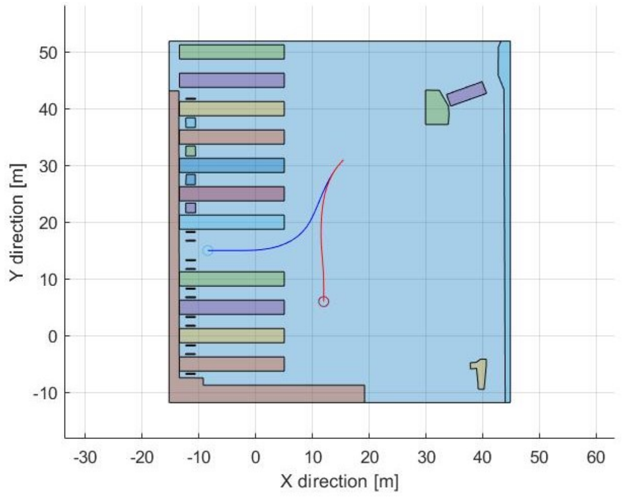
4-1 Test Reference Paths

Four reference paths have been generated to test the configuration of the proposed Model Predictive Control (MPC) driver assistant. All four test paths end at the same loading dock between two already parked vehicles. The paths are computed for different starting positions in a known environment with additional obstacles, based on a real-life Distribution Center (DC). It is advantageous that these reference paths are all unique to accurately test the performance of the overall setup in different scenarios. The four test paths can be found in Figure 4-1 to Figure 4-4. The figures show the relative location of each reference path in the DC and the location of the obstacles present in the DC. Additionally, the figures show the initial pose of the vehicle and the desired final pose of the vehicle.

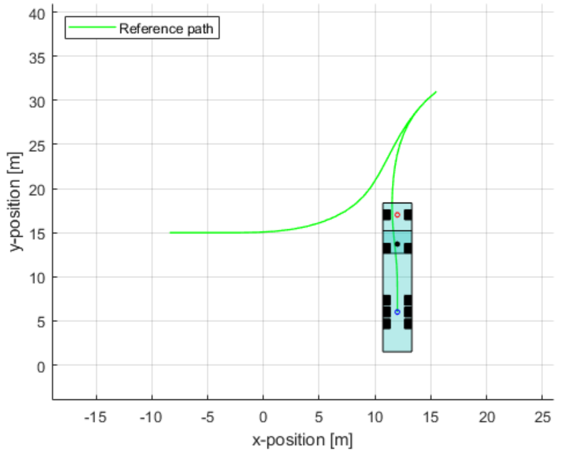
The vehicle dimensions used when generating these reference paths are shown in Table 4-1. These values will also be used for the kinematic model in the MPC setup for all tests with these reference paths.

Table 4-1: Vehicle dimensions.

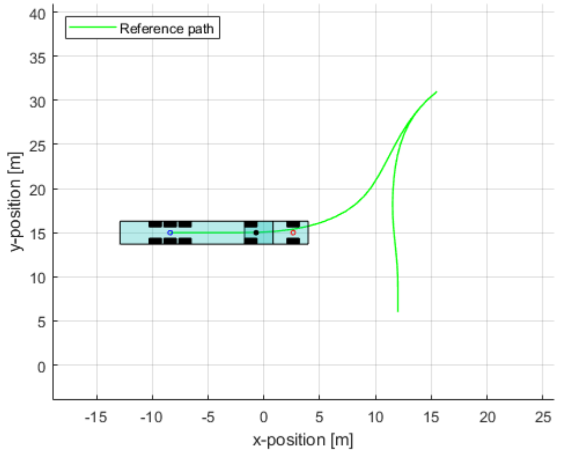
Symbol	Definition	Value
L_{0f}	Distance between truck axles	3.802 [m]
L_{0b}	Distance between drive axle and articulation point	0.485 [m]
L_{1f}	Distance between trailer axle and articulation point	7.702 [m]



(a) Path location in DC

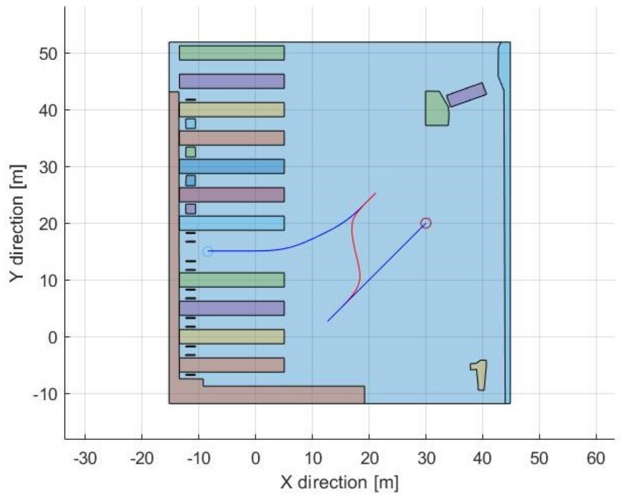


(b) Starting pose of vehicle.

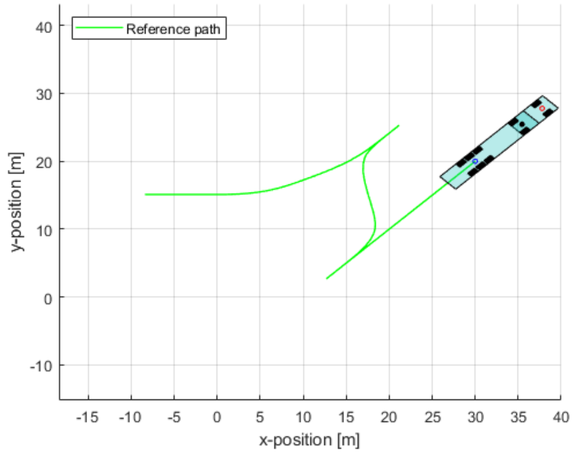


(c) Desired end pose of vehicle.

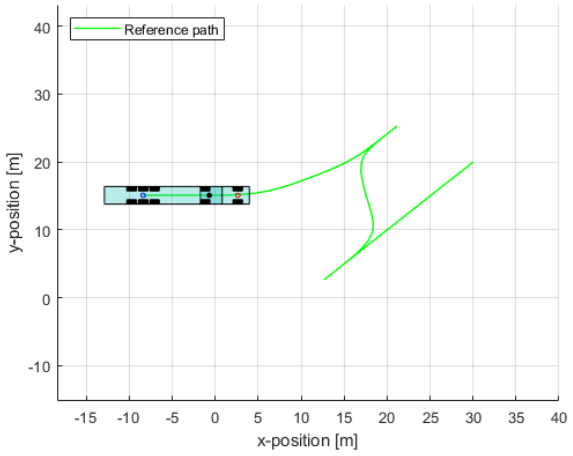
Figure 4-1: Test path 1.



(a) Path location in DC

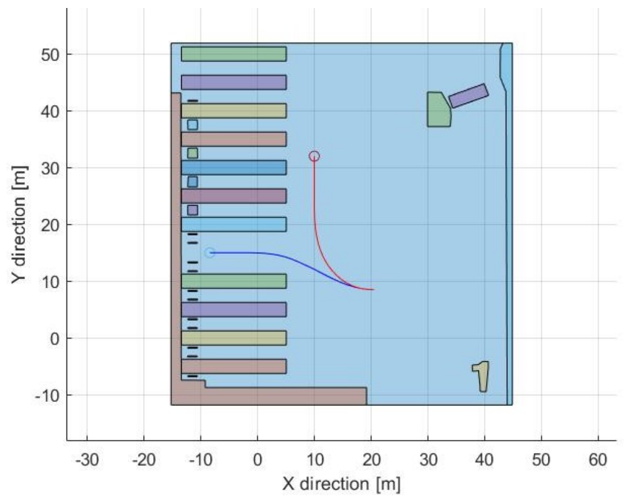


(b) Starting pose of vehicle.

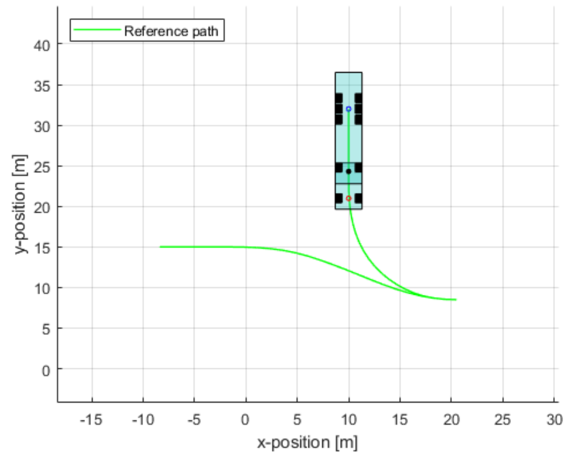


(c) Desired end pose of vehicle.

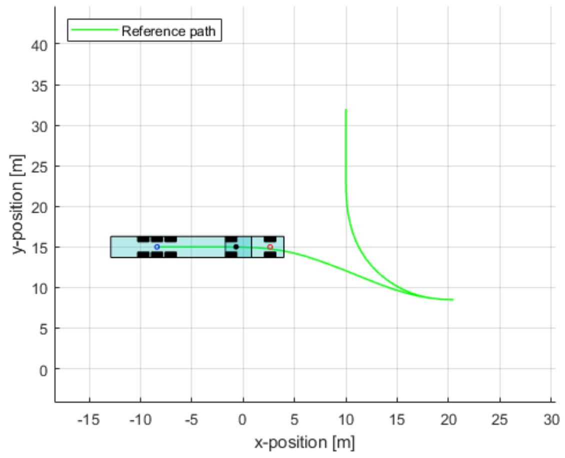
Figure 4-2: Test path 2.



(a) Path location in DC

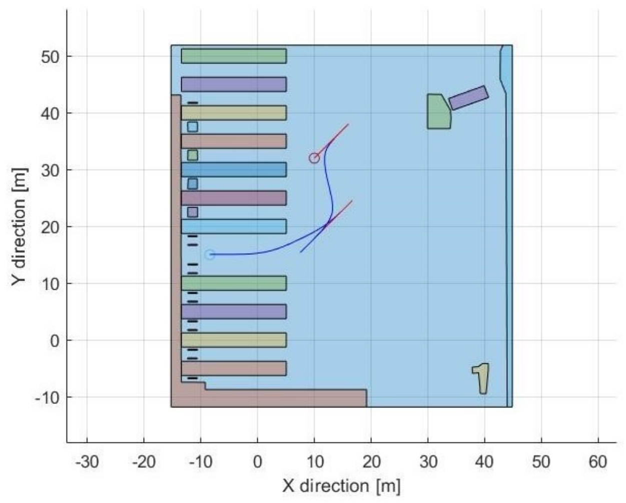


(b) Starting pose of vehicle.

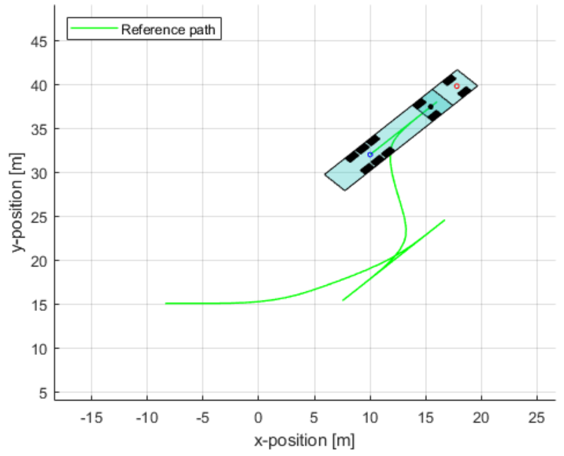


(c) Desired end pose of vehicle.

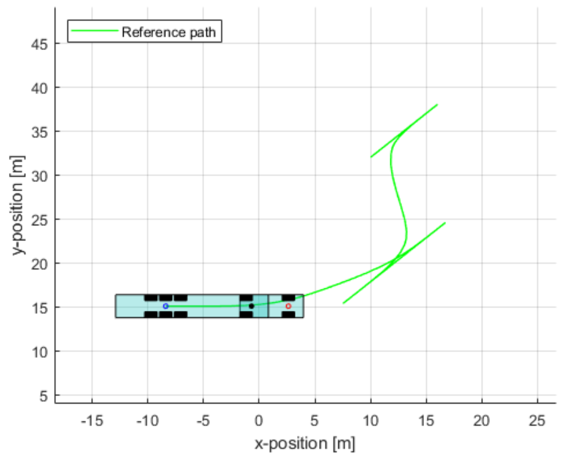
Figure 4-3: Test path 3.



(a) Path location in DC



(b) Starting pose of vehicle.



(c) Desired end pose of vehicle.

Figure 4-4: Test path 4.

4-2 Truck Docking Simulation without Human Driver

For the first basic tests of the controller, a simulation environment is used where only the dynamics of the vehicle are taken into account. A driver or a driver model are not involved at this point. This allows to test the basic functionalities of the proposed MPC controller. For this simulation, the current state of the vehicle is sent directly to the controller and the module which computes the local reference path. In the simulation, the information is sent and received without any added delay. The vehicle is now perceived as an fully autonomous vehicle. Therefore, the state of the system (3-1) is shortened by omitting the driver states:

$$\mathbf{x}_{\text{test}}(t) = \begin{bmatrix} x_1(t) \\ y_1(t) \\ \theta_1(t) \\ \gamma_1(t) \end{bmatrix}. \tag{4-1}$$

The continuous dynamics of the vehicle are described as follows:

$$\dot{\mathbf{x}}_{\text{test}}(t) = \begin{bmatrix} v_0(t)\left(\cos\gamma_1(t)\cos\theta_1(t) - \frac{L_{0b}}{L_{0f}}\cos\theta_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\cos\gamma_1(t)\sin\theta_1(t) - \frac{L_{0b}}{L_{0f}}\sin\gamma_1(t)\sin\theta_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\frac{1}{L_{1f}}\sin\gamma_1(t) + \frac{L_{0b}}{L_{0f}L_{1f}}\cos\gamma_1(t)\tan\delta(t)\right) \\ v_0(t)\left(\frac{1}{L_{0f}}\tan\delta(t) - \frac{1}{L_{1f}}\sin\gamma_1(t) - \frac{L_{0b}}{L_{0f}L_{1f}}\cos\gamma_1(t)\tan\delta(t)\right) \end{bmatrix}. \tag{4-2}$$

This simplified version of the setup, excluding the driver model, can be used to test the basic capabilities of the controller. The main goal of using this test setup is to review the performance of the controller under ideal conditions without a driver in the loop. Also, it can be used to research the effect of changing the weights in the cost function of the controller, as formulated in Section 3-3. Afterwards, the results can be used to make a choice for values which are suitable as weights for the subsequent test scenarios.

4-3 2D World Simulation

The next set of tests includes interaction with a human driver. Therefore, it is required to visualize the instructions and a current overview of the scenario for the driver. Additionally, the driver needs to be able to actuate the simulated vehicle. In this test environment, the driver is only able to see a 2D top-view overview of the scenario. Here, he or she can see the current location and orientation of the vehicle on a screen. An overview of this visualization is shown in Figure 4-5. The vertically oriented vehicle is the ego vehicle, while the other two vehicles represent two parked truck-trailer combinations, in other words, obstacles. The vehicle that is being controlled is called the ego vehicle to distinguish it from any other vehicles that are considered to be obstacles.

On a secondary screen, the driver sees the steering instructions and the current steering angle, both represented by circular gauges. This is visualized in Figure 4-6. In the simulation, the driver is capable of changing the steering angle and the velocity of the truck directly using the Logitech G29 Racing Wheel.

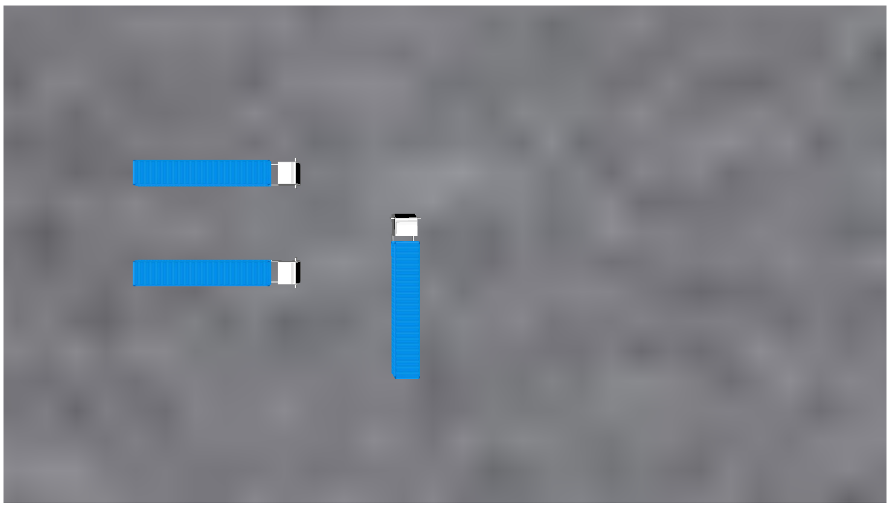


Figure 4-5: Top view of simulated docking manoeuvre.

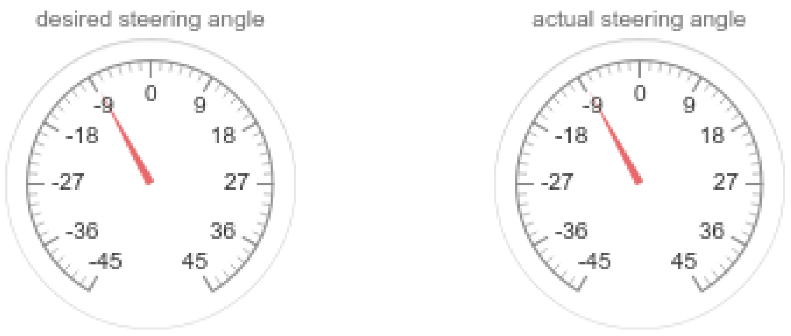


Figure 4-6: Driver instructions in 2D world simulation. (Left: Steering instruction δ_d. Right: Steering input from driver δ.)

This test setup will be used for initial research regarding the inclusion of the driver model in the controller scheme. Firstly, this setup can be utilized to identify a driver model with relative ease. Secondly, the objective of this test is to research whether the behaviour of the driver can be accurately captured by the presented driver model (2-15). This driver model can subsequently be used to simulate the behaviour of the driver. This eases the process of further testing the MPC controller. Some tests can now be conducted without requiring input from an actual driver. Therefore, the simulation can be sped up, which allows for additional testing opportunities. Lastly, the driving behaviour of the driver while receiving input from the controller that uses a driver model can be reviewed. This test setup can thus be used to thoroughly investigate the performance of the controller, with a driver model included, in a rudimentary simulated environment.

4-4 Virtual Reality Simulation

The final testing phase is done with a Virtual Reality (VR) simulator. VR is a simulated experience where the user, or driver in this case, is immersed in a computer-generated world. The driver is able to interact with the simulated environment with certain actions. These actions have the same effect in the simulated environment as they would have in the real world. This simulator is build in an actual truck cabin and configured such that the simulated vehicle can actually be controlled with the original steering wheel and pedals. The outside and inside of the cabin are shown in Figure 4-7 and Figure 4-8, respectively. Building the VR simulator in an actual cabin should contribute to the authenticity of the driving experience.

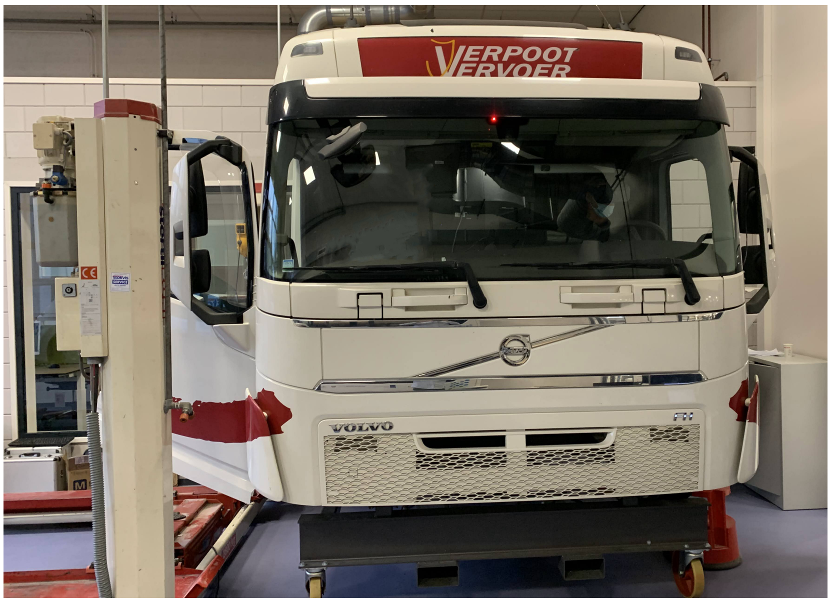


Figure 4-7: Outside view of cabin with Virtual Reality simulator.

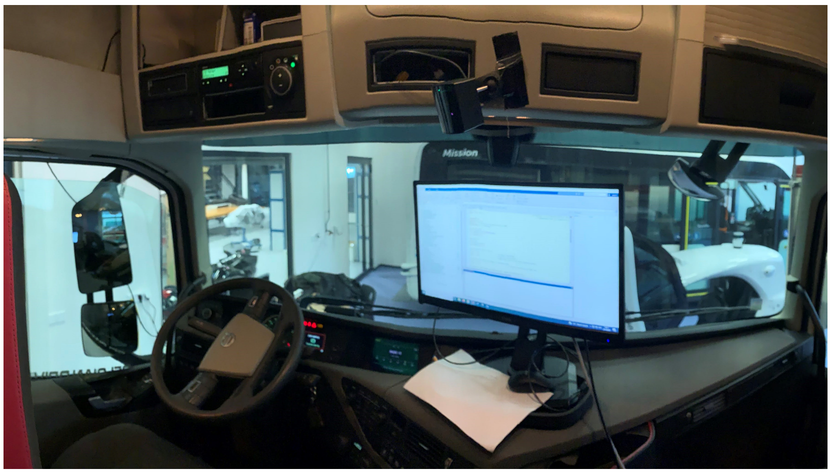


Figure 4-8: Inside view of cabin with Virtual Reality simulator.

While the driver is sitting in the driver's seat, they are able to view the simulated environment by means of VR glasses. This way, the driver is able to move and turn their head, and their point of view will change accordingly. Their point of view in the VR simulator while wearing the glasses is shown in Figure 4-9. Therefore, the VR environment gives the driver a very accurate representation of the real-world scenario. Testing is done in the VR simulator, as it facilitates a more convenient testing process, when compared to testing with an actual truck-trailer. In the simulator, the exact location and orientation of the vehicle are known at all times, which means the real-time localization module can be omitted at this point. Also, testing in the VR simulator enables quick changes to be made between different tests and to repeat tests fairly swiftly. For example, it is possible to easily change the starting position of the vehicle and the location of any obstacles. Moreover, small changes in the MPC setup can be made and tested with the exact same initial conditions as previous tests, while in real-life it is difficult to set the initial conditions equal for all tests. The VR simulator is used to test the final version of the proposed MPC-based driver assistant in an environment close to reality. Testing will be done with laymen as well as professional truck drivers.

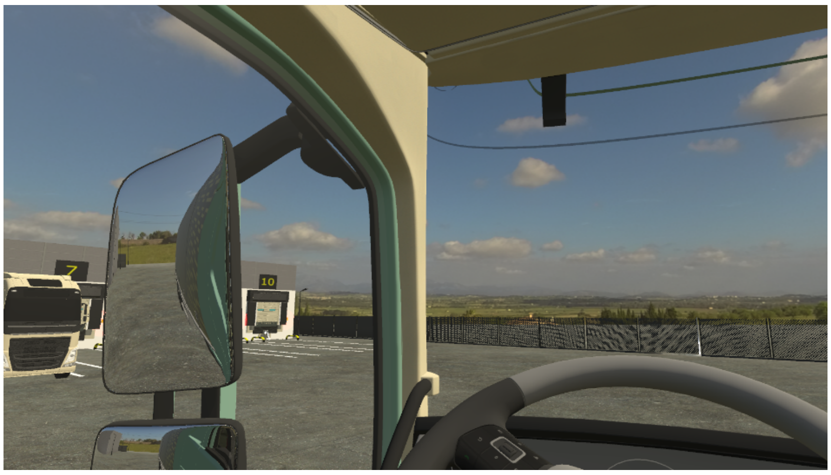
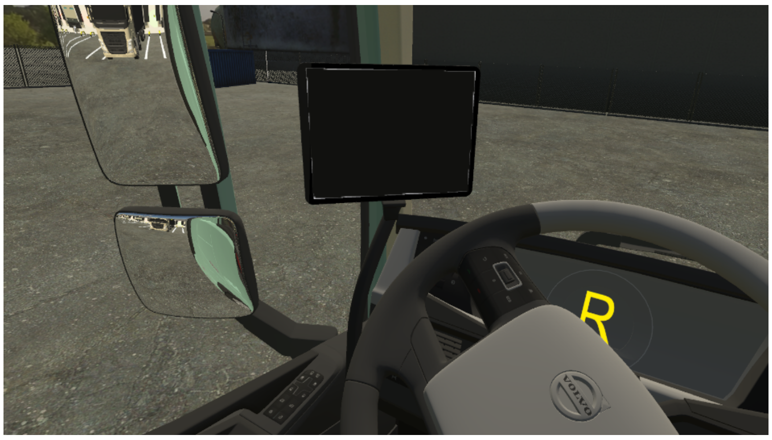


Figure 4-9: Point of view while wearing Virtual Reality glasses.

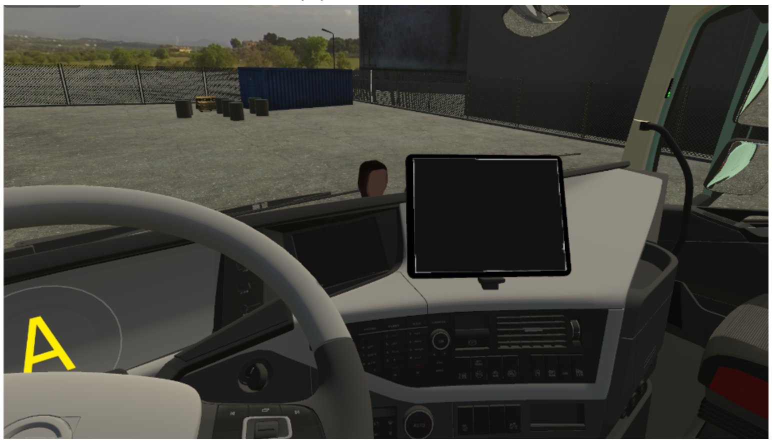
4-4-1 Human Machine Interfaces

An important module of the VIsion Supported Truck docking Assistant (VISTA) system is the Human Machine Interface (HMI), as shown in Figure 1-3. The choice HMI determines the way of delivering the instructions to the driver. Currently, there are various concepts for the HMI. The development of these concepts did not fall within the scope of this research. However, the exact way of delivering the instructions to the driver could have a significant influence on the intuitiveness of the system and the resulting performance of the system as a whole. Various combinations of HMI concepts will be used during testing. Therefore, the different concepts are presented here.

The main concept is to visualize the instructions by means of a tablet that is handed to driver upon entering the parking structure. Thereafter, the truck driver may mount the tablet somewhere in their cabin. Currently, two different locations for mounting the tablet are being considered. These are shown in Figure 4-10.



(a) Left of cabin.

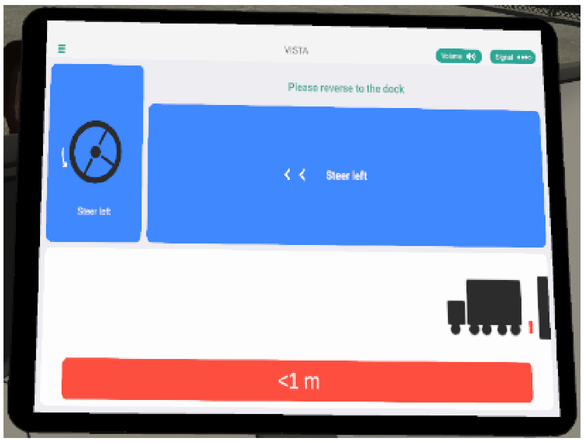


(b) Right of cabin.

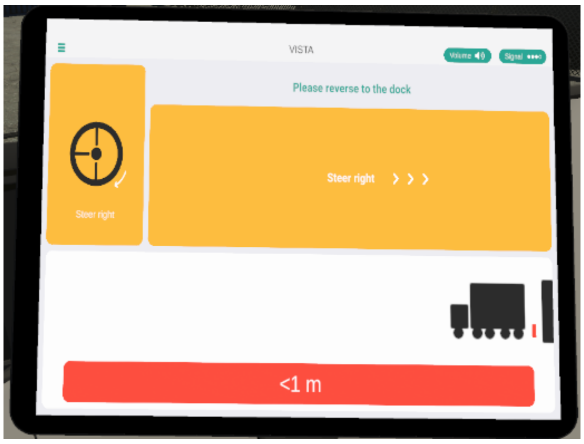
Figure 4-10: Different tablet locations for driver instructions.

An advantage of using a tablet is that this still leaves plenty of room for different ways of visualizing the instructions, as the design of the application on the tablet can vary. There are currently three latest developed application layouts. The first one is shown in Figure 4-11. The instructions are visualized by stating whether a left or right turning steering action is required. This is done by explicit written instructions and by the color of background. This color cue should benefit the reaction time of the driver. Additionally, the distance to the end of the current segment is shown.

The second concept is shown in Figure 4-12. This one is fairly similar to concept 1, with the most significant change being that the instruction indicating a turn to the left is on the left side of the screen, and the instruction indicating a turn to the right is on the right side of the screen. This way a driver can see at a glance whether to steer left or right. The third concept is shown in Figure 4-13. The most important feature of this concept is the live feed from the cameras. This could potentially be a huge asset for the truck driver, as the rear of the truck is difficult to see when inside the cabin. However, this is until now only been tested in the VR simulator and there might be practical implications when using this concept in real-life. For example, when the driver fully depends on the camera images while they are not up-to-date, which may result in a collision with an obstacle.

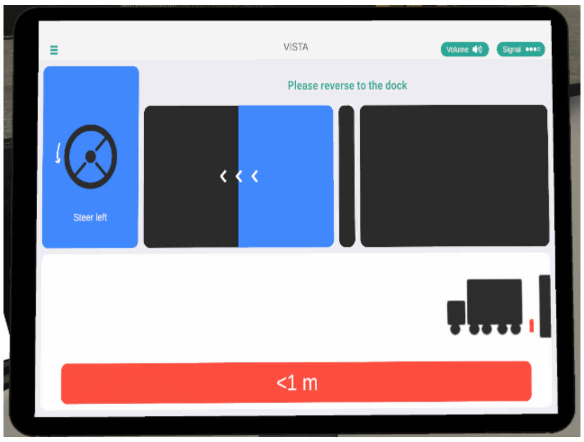


(a) Instruction indicating steering to left desired.

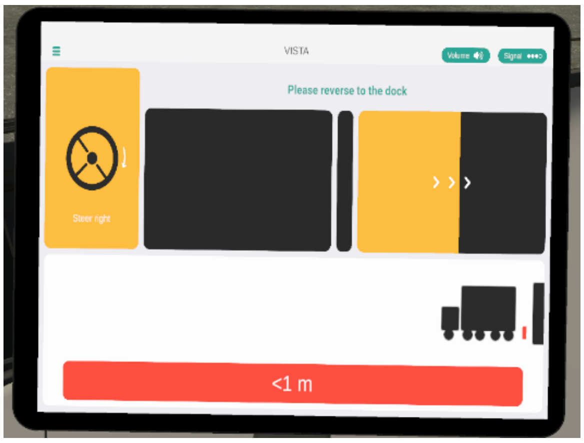


(b) Instruction indicating steering to right desired.

Figure 4-11: Human Machine Interface tablet concept no. 1

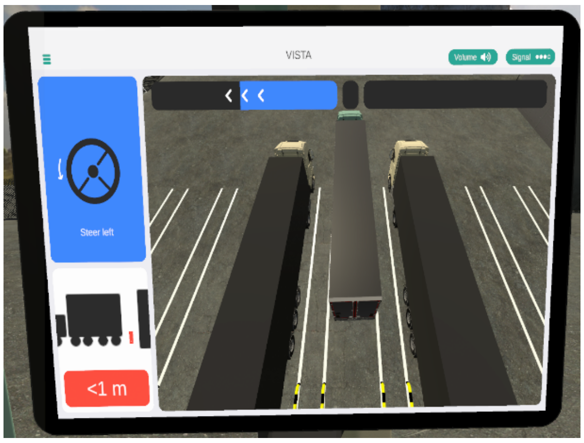


(a) Instruction indicating steering to left desired.

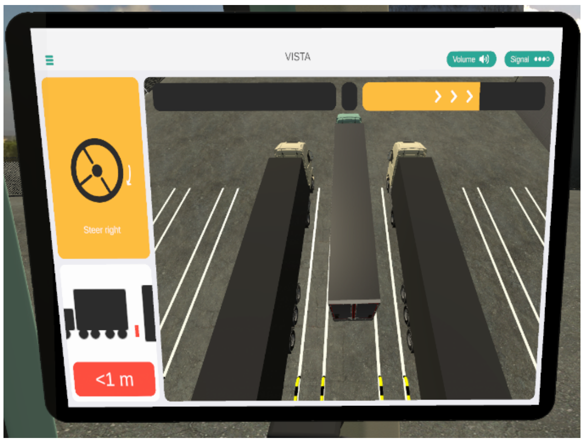


(b) Instruction indicating steering to right desired.

Figure 4-12: Human Machine Interface tablet concept no. 2



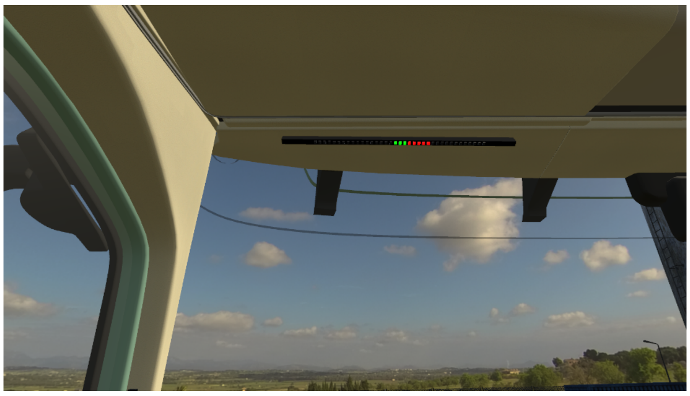
(a) Instruction indicating steering to left desired.



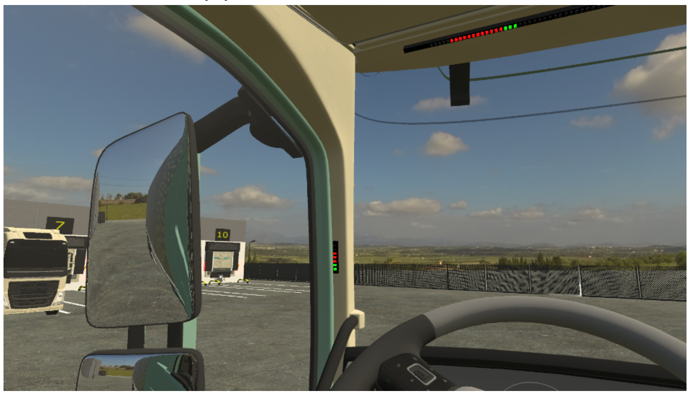
(b) Instruction indicating steering to right desired.

Figure 4-13: Human Machine Interface tablet concept no. 3

Another concept, deviating from the tablet concept, is to visualize the instructions with the use of LED strips. The idea behind this concept is that it condenses the information for the driver and the instruction is intuitive and easy to comprehend. Figure 4-14 shows the LED strip concept.



(a) View of horizontal LED.



(b) View of left vertical LED.

Figure 4-14: Human Machine Interface LED concept

4-4-2 Driver Instructions

The driver model describes the output of the driver when a certain input is given. The output of importance is the steering angle induced by the driver. The input given to the driver, when regarding it as a subsystem, is a steering angle instruction. The steering angle instruction could be regarded as the desired steering angle or the reference steering angle. The exact visualization of instruction depends on the chosen HMI. The choice of HMI results in a specific way of processing the received steering instruction, thus altering the overall systems behaviour.

One major alteration is that none of the concepts actually display the desired steering angle. Instead, they show the difference between the desired steering angle and the current steering angle. This way a driver can immediately see if a clockwise or counterclockwise steering action is asked of them. However, this approach only works under the assumption that the current steering angle is known. In the practical implementation of the VISTA system, this will most likely not be a valid assumption. The Real Time Localization System (RTLS) is able to estimate the location and orientation of the truck and the trailer. Unfortunately, it is, at this point, too difficult to get an accurate estimation of the orientation of the tyres. Therefore, other concepts of different HMIs are being devised, which display a different type of input to the driver.

One of these concepts is based on the idea to visualize the current orientation of the truck and the optimal orientation of the truck. This way, the driver has to determine the required steering actions such that the current orientation of the truck matches the optimal one. The controller will therefore need to compute the optimal orientation of the truck. The optimal input is computed by predicting the dynamics of the system with a mathematical model of the vehicle. Therefore, the optimal predicted states are also computed, thus the optimal orientation of the truck can be send to the driver. However, this will influence the drivers behaviour. They have to play a larger role in the determination of the optimal steering strategy. It is interesting to investigate the effect of the use of a driver model to predict this more complex behaviour.

4-5 Vehicle Model for Simulations

The different simulation environments described in the previous sections all require a vehicle model. This model is a formulation of the behaviour of the vehicle as a driver input is actuated on the simulated vehicle. This model may vary from the vehicle model used in the MPC setup. It can even be argued that a different, more accurate vehicle is preferred for the simulation, as this offers the opportunity to test if the kinematic model used in the MPC setup is actually valid. Moreover, this mismatch between the simulation model and the model used for the controller formulation allows to test the robustness of the controller. It is also possible to simulate an error in the estimation of the current state of the vehicle. All described testing environments will, in principle, utilize the same kinematic model as the MPC controller. The main difference between both models is that the controller uses a discretized version. The controller requires a discretized version, because it predicts the optimal input at discrete time-steps. As opposed to the model in the simulation, as this is run in continuous time. In Section 4-6, the discretization of the model will be further discussed. The validity of the kinematic model has been found to be sufficiently adequate for low-speed manoeuvring [10]. Moreover, first tests with the VR simulator are favorable to the utilization of the kinematic model. Professional drivers felt that the behaviour of the vehicle closely resembled to that of a real truck-trailer combination.

While simulations with this model already show promising results, there is also the possibility to simulate the vehicle's dynamics using a multi-body model. A multi-body model of a truck-trailer combination is presented in [10] and it is constructed with the use of the multi-body toolbox of MATLAB Simulink. The model is validated and includes tyre behaviour, mass distribution and other effects which are neglected in the kinematic model. This way, it gives a more realistic representation of the real-world dynamics of a truck-trailer combination. It is interesting to research whether the controller setup with the kinematic model is valid when testing with a higher-fidelity model.

4-6 Hardware and Software

All the initial simulations without the VR simulator are run on a single computer, hereafter called PC1. These are the simulations as described in Section 4-2 and Section 4-3. This is a computer running on Windows 10 with an Intel® Core™ i7-4710MQ @2.50GHz processor, 16.0GB RAM and an NVIDIA Quadro K1100M graphics card. The MATLAB-based graphical programming environment Simulink is used for modeling and simulating the dynamics of the system.

The VR simulations are executed on a different computer with more computation power, hereafter called PC2. This is a computer running on Windows 10 with an AMD Ryzen™ 7 3800X 8-core 3.89GHz, 32GB RAM and an NVIDIA GeForce RTX2080Ti. The VR environment is created with Unity 2019.1.14f1 and interacted with by a HTC Vive Pro.

Both simulation environments use MATLAB Simulink version R2019b to register and process the driver input and to simulate the dynamics of the vehicle. In addition, this program is also used to execute the controller's solver. The solver is responsible for solving the optimization problem, resulting in the optimal control action. At both simulation environments, the solver is executed on PC1. For the VR test scenario, as discussed in Section 4-4, this means the current state of the system needs to be send from PC2 to PC1 and the output of the solver needs to be send from PC1 to PC2. This is done via UDP signals on a local area network.

The solver is generated with the Embotech FORCES Pro software [40]. It is specifically designed to quickly solve the nonlinear optimization problem using efficient interior-point methods [32]. The solver is generated using the MPC setup as presented in the previous chapter, including the dynamics discussed in Section 3-2. It can be noticed that these dynamics are given in continuous time. However, the MPC module predicts the behaviour at discrete time instances. Therefore, the solver generation is preceded by a discretization of the dynamics using the implicit Runge-Kutta method of order 2. This is done with the use of the CasADi AD tool [41]. It can be interesting to research whether other nonlinear programming methods, such as sequential quadratic programming, and/or different discretization methods, such as the explicit Runge-Kutta method of order 4, might improve the overall performance of the controller. However, this is outside of the scope of this thesis.

Chapter 5

Test Results

In this chapter, the performance of the Model Predictive Control (MPC)-based truck docking assistant system is discussed. First, the quantifying properties that define the performance of the VIsion Supported Truck docking Assistant (VISTA) are presented. These properties are measured with various sets of tests in the simulation environments as described in the previous chapter. The results of the tests are presented and discussed.

5-1 Key Performance Indicators

The working of the MPC-based driver assistance system is reviewed with a Key Performance Indicator (KPI) evaluation. A set of different KPIs is selected, such that they demonstrate the success of the system. They are divided in different categories that specify their contribution in the overall system. This is shown in Table 5-1.

Table 5-1: Key Performance Indicators

Category	Key Performance Indicator
Reference path tracking	• Mean tracking error [m]
Robustness	• Maximum allowed driver delay [s]
Computational speed	• Average computation time [s] • Peak computation time [s]
Driver acceptance	• Average steering instruction rate [rad/s] • Peak steering instruction rate [rad/s] • A professional driver's opinion

5-1-1 Reference path tracking

The first KPI indicates the ability for the system to perform its most fundamental function. This is the ability of tracking a precomputed reference path. It is measured by the error of the driven path, or path tracking error. The path tracking error should be as low as possible, because it is assumed that the reference path is optimal and ensures collision avoidance. The path tracking error is determined by computing the difference between the reference path and actual path driven path.

The precomputed reference path is provided as a set of R waypoints, indicating the reference coordinates. The path tracking error is based on the euclidean distance between these waypoints and the actual driven path. This distance is measured for each of the reference waypoints, by first locating the closest point on the actual driven path. The driven path is stored as a set of Y points. These points represent the evolution of the coordinates of the rear axle of the trailer. The coordinates are sampled at a fixed frequency. A high frequency is required, in order to have a near continuous capture of the output of the system. This way the smallest distance between a reference waypoint and the actual driven path can accurately be measured. The distance to the driven path at each way point can thus be computed as

$$d_{\text{ref}}(i) = \min_j \sqrt{(x_r(i) - x_d(j))^2 + (y_r(i) - y_d(j))^2} \quad \forall i \in \{1, 2, .., R\},\ \forall j \in \{1, 2, .., Y\}, \tag{5-1}$$

where $x_r(i)$ and $y_r(i)$ are the x- and y-coordinate of reference waypoint i, and $x_d(j)$ and $y_d(j)$ are the j-th sampled x-and y-coordinate of the actual driven path. The path tracking error is then measured as the mean difference between the reference path path and the actual path driven path, as follows:

$$\text{Mean Tracking Error} = \frac{1}{R}\sum_{i=1}^{R} d_{\text{ref}}(i). \tag{5-2}$$

Figure 5-1 shows an example of a comparison between reference waypoints and a simulated driven path. Moreover, it shows a zoomed in view of the local path tracking error at individual waypoints. A lower mean tracking error is preferred.

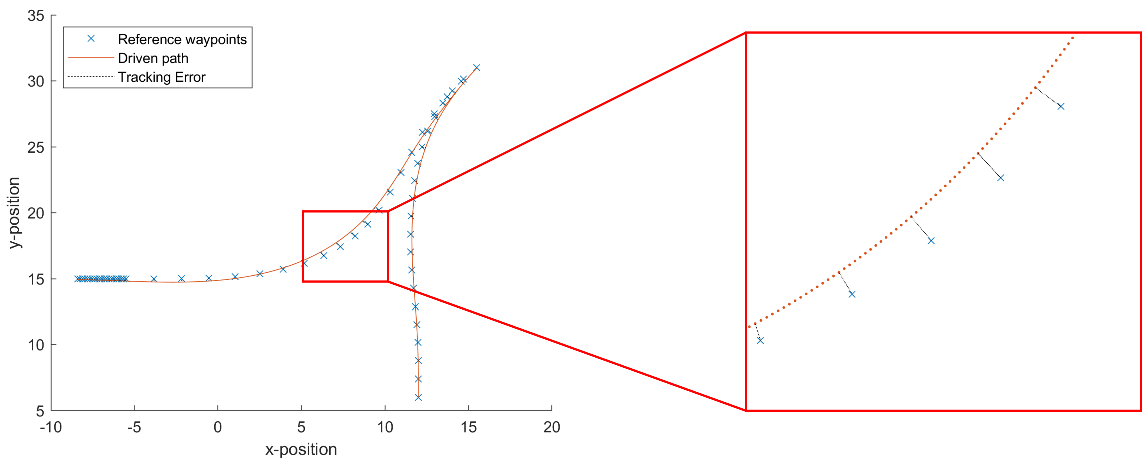


Figure 5-1: Path tracking error

5-1-2 Robustness

The robustness of the system is measured by the maximum delay allowed for the system to still function. The delay simulates a driver inaccuracy that is probable to occur in real-life. This is simulated as a constant delay present at the transmission of the steering angle instruction to the driver. Measuring this KPI is a time-consuming task as the approach is fairly repetitive. The same parking manoeuvre is simulated repeatedly and the duration of the delay is increased at each iteration. This is repeated until the controller is seemingly unable to handle the disturbance, which means the simulation does not end with the truck-trailer combination reaching the desired parking destination. This may happen due to the controller being unable to solve the optimization problem, or due to the vehicle colliding with an obstacle. A higher delay, which the system is still able to cope with, is preferred.

5-1-3 Computational speed

As mentioned, it is vital that the driver receives timely updated instructions. However, MPC is known to be a computationally expensive control method. It is therefore interesting to track the time needed for the solver to find the optimal control input at each time-step. Therefore, the time is measured from the moment that the solver is executed up to the point where it has found a solution. The two KPIs identifying the performance regarding the computational speed are the average computation time and the peak computation time of an entire parking manoeuvre. Obviously, the computation time also highly depends on the hardware and software used. The absolute values of the computation time might therefore seem less important. However, these KPIs do give an insight in the influence that certain parameters have on the computational speed. A lower average computation time and a lower peak computation time are preferred.

5-1-4 Driver acceptance

The driver acceptance is measured by driver comfort and driver satisfaction. For driver comfort it is important that the instructions are intuitive and smooth. Therefore, the rate of change of the steering instructions is measured. This is done by computing the derivative of the steering angle instructions given to the driver. The derivative can be both positive or negative, indicating a left or right turning action. It is therefore important to look at the absolute value of the steering instruction rate. The two KPIs are the average absolute steering instruction rate and the peak absolute steering instruction rate. A lower average steering instruction rate and a lower peak steering instruction rate are preferred.

Lastly, the driver satisfaction is determined by an evaluation of the system by a professional driver. This KPI is subjective opinion of the driver about the parking assistant. Ideally this is measured for different drivers in different scenarios. It is preferred for the professional driver(s) to have a high opinion of the system.

5-2 Testing Overview

In the following sections of this chapter, the results of various sets of tests are discussed. Each set of tests is conducted with a different MPC configuration in order to research their effect on the performance. In table Table 5-2, an overview is given of the configuration used for each test set. Also the driver prediction model and the driver input is given. The driver prediction model is the model used in the MPC scheme to predict the behaviour of the driver. This can either be no model, as discussed in Section 4-2, or one of the models presented in Section 5-4. The driver input represents the actual behaviour of the driver during the tests. This can be a direct input, meaning the optimal input computed by the MPC controller is directly applied to simulated vehicle. Additionally, this driver behaviour can be simulated with one of the driver model, or the behaviour is modeled as a pure time delay. Lastly, the driver behaviour can be of an actual human driver.

Table 5-2: Overview of MPC configuration for each test set.

Test set	Tracking weight $w_{\mathbf{y}}$	Steering suppr. weight $w_{\Delta\delta}$	Horizon N	Sample time T_s	Driver prediction model	Driver input
MPC Weights testing (Section 5-3)	1, 3, 10	1, 3, 10	50	0.2	No model	Direct input
Driver model testing (Section 5-4)	2	1	50	0.2	No model, 1st model, 2nd model	Amateur driver
Prediction horizon & sample time testing (Section 5-5)	2	1	25, 50, 100	0.1, 0.2, 0.3, 0.4	2nd model	Simulated with 2nd model
Delay compensation testing (Section 5-6)	2	1	100	0.3	No model	Time delay
VR simulator testing (Section 5-7)	2	1	100	0.3	No model	Amateur driver, Professional drivers

5-3 MPC Weights Testing

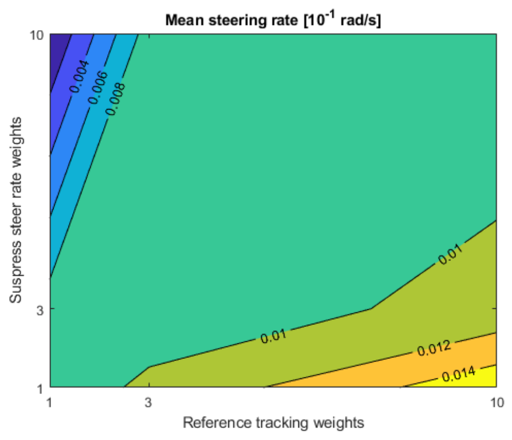
The first tests are conducted to investigate the effect of the choice of weights in the objective function of the optimization problem (3-11), discussed in Section 3-3. The testing scenario for these tests is described in Section 4-2. A simplified version of the controller, excluding the driver model, is used and the steering angle computed by the controller is directly applied to the vehicle. As discussed in Section 4-1, the reference paths are specifically designed for a truck-trailer combination with the dimensions as presented in Table 4-1. Therefore, it should be possible track this path with complete accuracy. However, the objective function used for the MPC controller does take additional objectives into account besides reference tracking. One major influencing component is the term in the cost function that indicates a preference for suppressing the steering rate. When a large change in steering angle is needed in order to track the reference path, a trade-off will be made between steering rate suppression and tracking accuracy. In theory, a high weight can be selected for both output reference tracking and the suppression of steering rate. However, this might cause the controller to severely limit the velocity of the vehicle when such a manoeuvre is needed in order to track the reference path. In order to effectively test the choice of weights for output reference tracking and the suppression of steering rate, the velocity of the truck is therefore considered constant for this test scenario. This means, for this test set only, $v_0(t) = v_f$ for forward segments, and $v_0(t) = v_b$ for reverse segments. Also, the driver model is omitted at this point, as the steering angle is directly applied to the simulated vehicle. This results in input of the system, in this case, being

$$\mathbf{u}_{\text{test}} = \begin{bmatrix} \delta \end{bmatrix}. \tag{5-3}$$

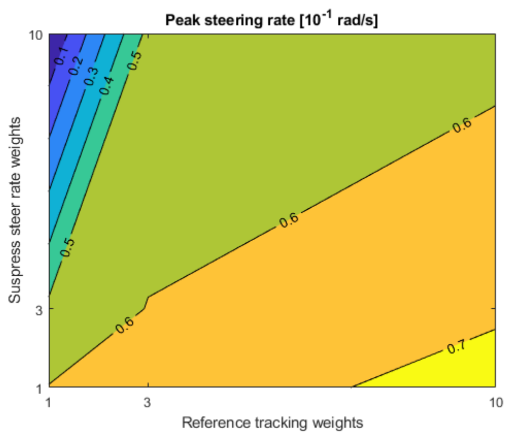
The main goal of using this test setup is to research the effect of different choices for the input change rate suppression weight $w_{\Delta\delta}$ and output reference tracking weights $w_{\mathbf{y}}$. Afterwards, the results can be used to make a choice for values which are suitable for the subsequent test sets. The parking scenario is simulated with a prediction horizon N of 50 stages and a prediction model sample time T_s of 0.2 seconds. The output reference tracking weights and the suppress steer rate weights are both set at either 1, 3 or 10. This makes for a total of 9 different combinations of weights to test. Each combination of weights is tested for all four of the test paths shown in Section 4-1.

5-3-1 Results

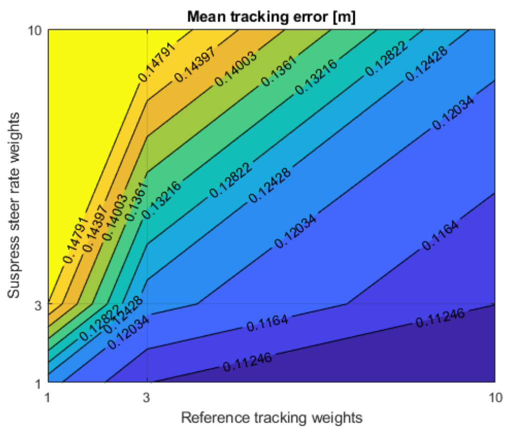
For the relevant KPIs, the mean values of all four test paths are shown in Figure 5-2. It can be seen that the results are fairly similar for the cases where the output reference tracking weight and the suppress steer rate weight are equal. This is to be expected, as the ratio between weights is more important than their absolute value. For the upcoming tests, the output reference tracking weight $w_{\mathbf{y}}$ is chosen to be twice as high as the suppress steer rate weight $w_{\Delta\delta}$. In most of the upcoming test scenario's, a human input is required and the results indicate that this choice of weights will result in a fairly low tracking error, while still requiring relatively smooth steering actions from the driver. With the velocity reference tracking weight w_{v_0} set at 1, the most promising results were shown with the output reference tracking $w_{\mathbf{y}}$ set at 2 and the suppress steer rate weight $w_{\Delta\delta}$ set at 1.



(a) Mean steering rate.



(b) Peak steering rate.



(c) Mean tracking error.

Figure 5-2: Different weights testing results.

This exact combination of weights has not been tested yet. Therefore, this choice of weights is validated by simulating the scenario with an output reference tracking weight $w_{\mathbf{y}}$ of 2, and a suppress steer rate weight $w_{\Delta\delta}$ of 1. In Figure 5-3 to Figure 5-6, the vehicle's movement while executing the parking manoeuvre is shown for test path 1 to test path 4, respectively. The scenario is simulated without a driver model and with the input directly applied to the simulated vehicle. The prediction horizon is still 50 stages and the prediction model sample time T_s is still 0.2 seconds. The figures show the truck-trailer combination represented as two light blue rectangles, the tyres of the vehicle are represented as black rectangles, where the steer axle consists of two tyres, the drive axle also consists of two tyres and the trailer axles consists of six tyres. Each figure also contains a black dashed line, which indicates the current prediction of the MPC controller. This prediction consists of a sequence of x- and y-coordinates of the trailer axle (x_1, y_1).

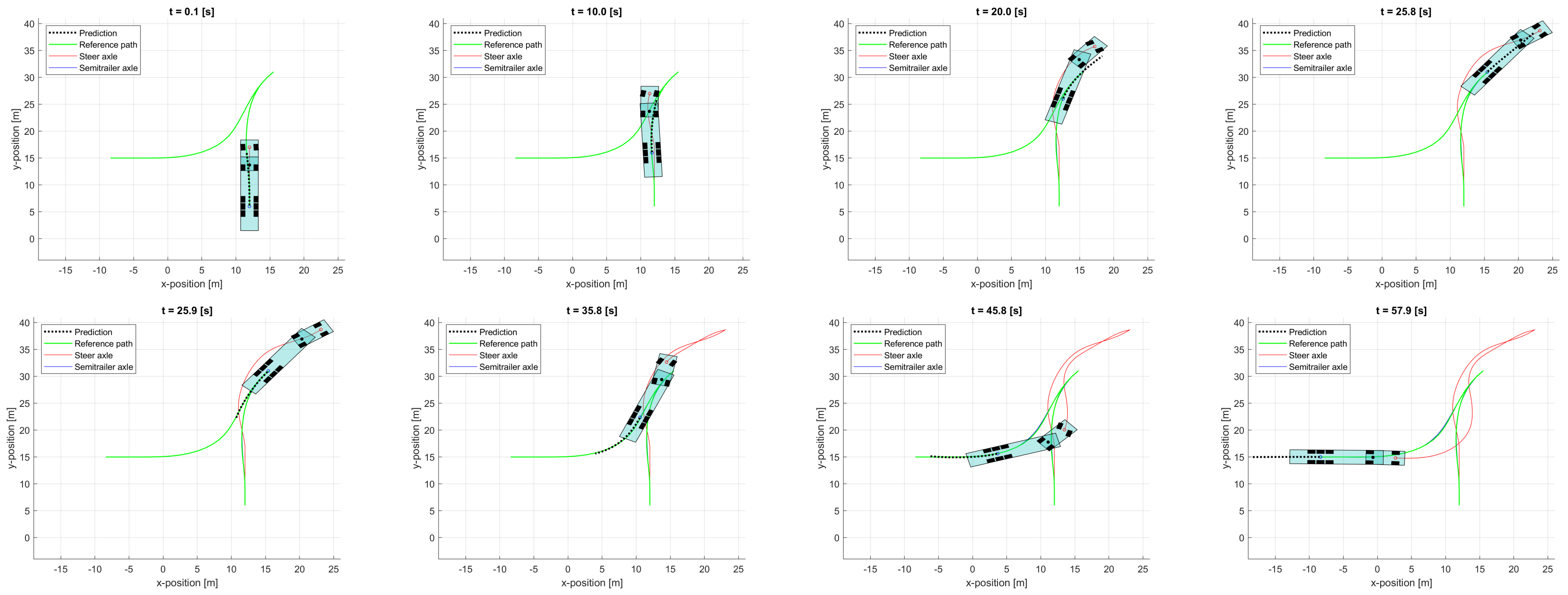


Figure 5-3: Vehicle's movement while executing parking manoeuvre - test path 1.

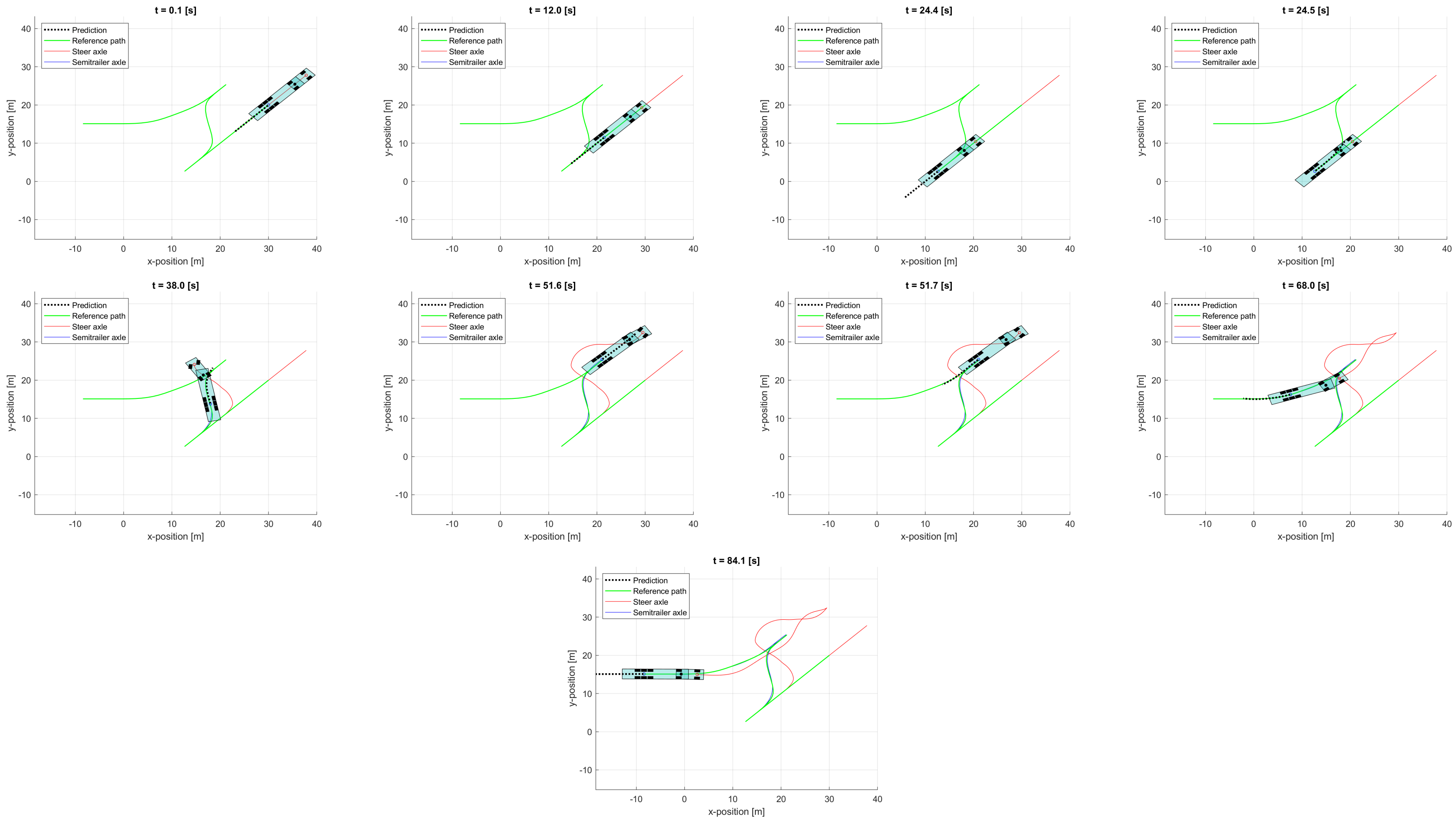


Figure 5-4: Vehicle's movement while executing parking manoeuvre - test path 2.

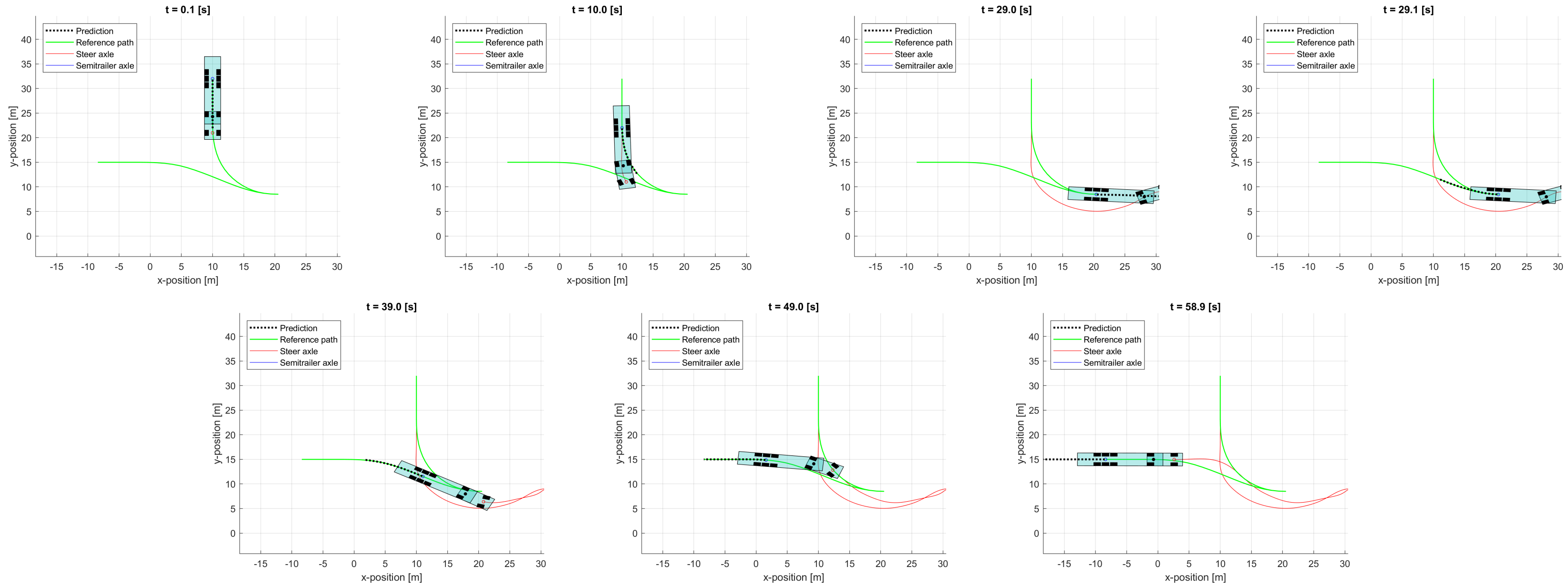


Figure 5-5: Vehicle's movement while executing parking manoeuvre - test path 3.

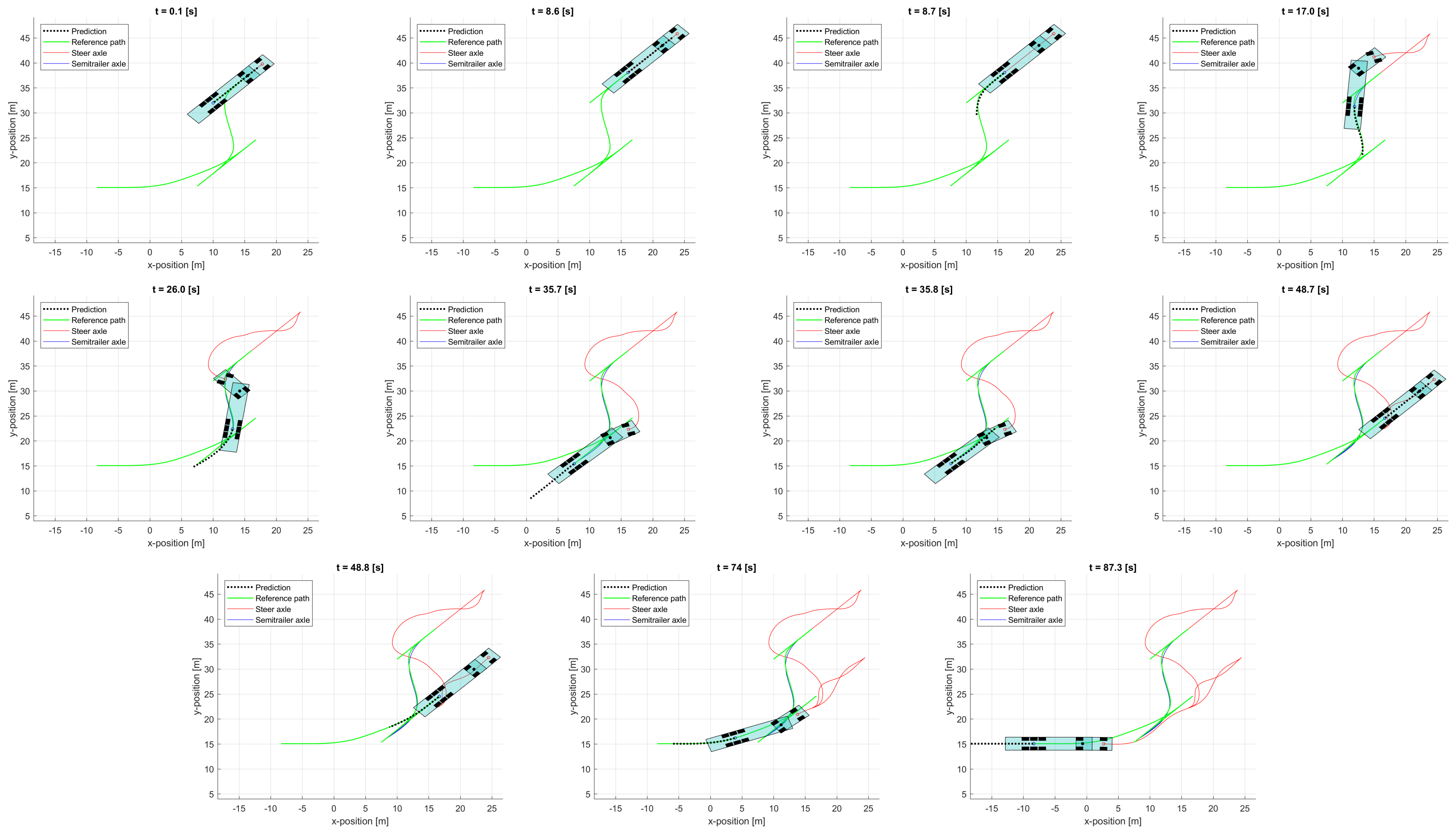


Figure 5-6: Vehicle's movement while executing parking manoeuvre - test path 4.

5-4 Driver Model Testing

After the tests regarding the choice of weights for the MPC controller, the next step is to analyze the system with the driver model, as described by equations (3-2)-(3-5), included. Therefore, first the parameters of the model have to be identified. In this section, the results of the first tests regarding the identification of these driver models are presented. The driver's behaviour is tested in the environment as described in Section 4-3. This environment consists of a 2D top-view of the parking scenario and two circular gauges indicating the desired steering action for the driver. The subsequent model identification is conducted in the manner as described in Section 3-7. The identification is based on data of the steering actions performed by the driver when given certain steering instructions.

There are four main points of interest while conducting these tests. First and foremost, it will give a first insight in the performance of the proposed MPC-based driver assistant with an actual human in the loop. Secondly, these tests will demonstrate the possible benefits of the use of a driver model in the MPC formulation. Thirdly, it will show the level of robustness of the system against the behavioural changes of the driver, as discussed in Section 3-7-2. Lastly, the identified driver model can be used for the upcoming tests where the focus also lies on the computation time of the solver and the full system, with the driver model included, is required to be used.

The parking scenario is simulated with a prediction horizon N of 50 stages, a prediction model sample time T_s of 0.2 seconds, the output reference tracking weight $w_{\mathbf{y}}$ set at a value of 2, and the suppress steer rate weight $w_{\Delta\delta}$ set at a value of 1. Each MPC setup is tested with an amateur driver performing the parking manoeuvre on all four test paths. This person is in possession of a regular driver's license, but has no experience with truck driving whatsoever.

5-4-1 Results

First, a test is conducted with the MPC setup without a driver model. The driver's behaviour has been analyzed during these manoeuvres and this is used to identify the first driver model. Then, the manoeuvres are repeated, but with the MPC structure that includes the first identified driver model. Again, the driver's behaviour is analyzed and used to identify a second driver model. A last round of tests has been conducted with the MPC structure including this second identified driver model. Table 5-3 shows the parameters of the driver models.

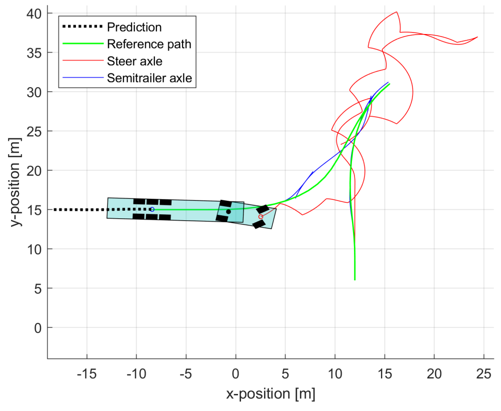
Table 5-3: Parameters of identified driver models.

Parameter	1st Model	2nd Model
K	1	1
T_L	0.0227	0.0763
T_l	0.3	0.4938
T_N	0.8094	1.164
τ_r	0.2	0.2

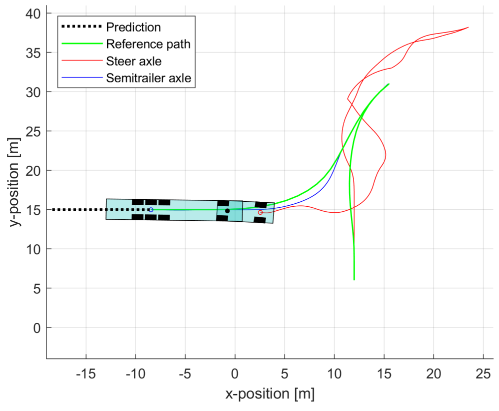
In Table 5-4, the relevant KPIs are presented for the system using different controller configurations. Figure 5-7 shows the resulting paths compared to the reference path for test path 1. Also, it shows the final location and orientation of the vehicle after the manoeuvre is executed. Figure 5-8, shows a more detailed insight in the tracking error on test path 1. In Figure 5-9, the absolute path tracking error at each specific waypoint on path 1 is shown. This is only visualized for test path 1, but the test is conducted for all four test paths.

Table 5-4: KPIs using different driver models (mean of 4 test paths).

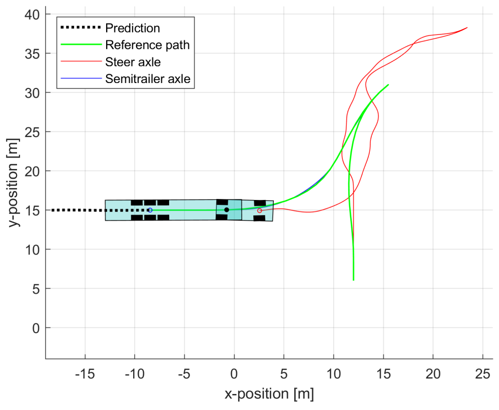
KPI	No model	1st Model	2nd Model
Mean Tracking Error [m]	0.281	0.148	0.096
Peak comp. time [s]	0.058	0.026	0.0204
Mean comp. time [s]	0.011	0.010	0.010
Peak steer rate [rad/s]	1.381	1.282	1.015
Mean steer rate [rad/s]	0.200	0.121	0.137



(a) Without driver model.

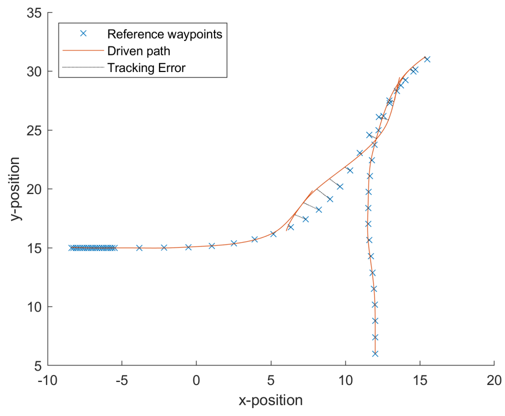


(b) With driver model 1.

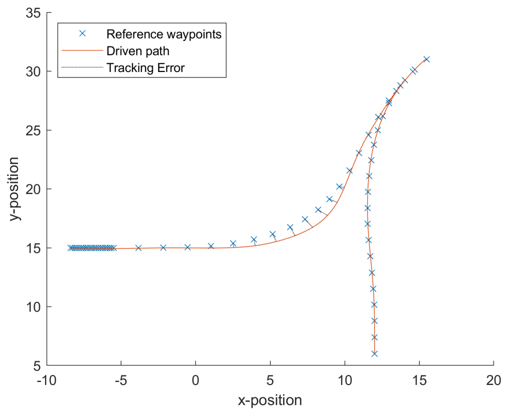


(c) With driver model 2.

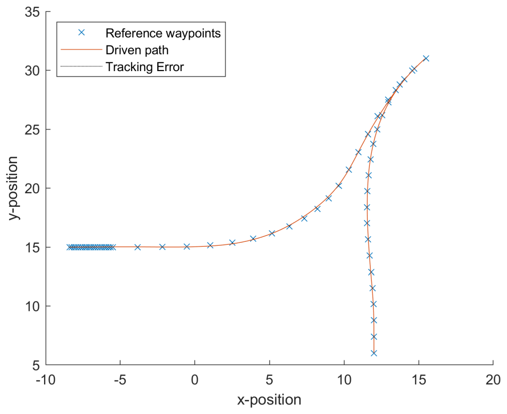
Figure 5-7: Results of using different driver models on test path 1.



(a) Without driver model.



(b) With driver model 1.



(c) With driver model 2.

Figure 5-8: Tracking error on test path 1 using different driver models.

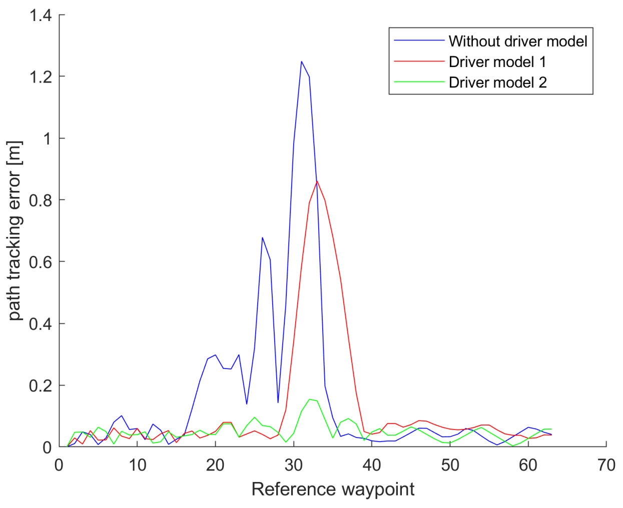


Figure 5-9: Path tracking error on path 1 using different driver models.

From these results, using the test environment as described in Section 4-3, it can be seen that the tracking error is reduced by using a driver model. Also, the steering actions are smoother at each iteration and the final pose of the vehicle is better after each attempt. Overall, the results show a significant improvement when including a driver model in the MPC configuration. It mainly seems to have a positive effect on counteracting the delay introduced by the driver.

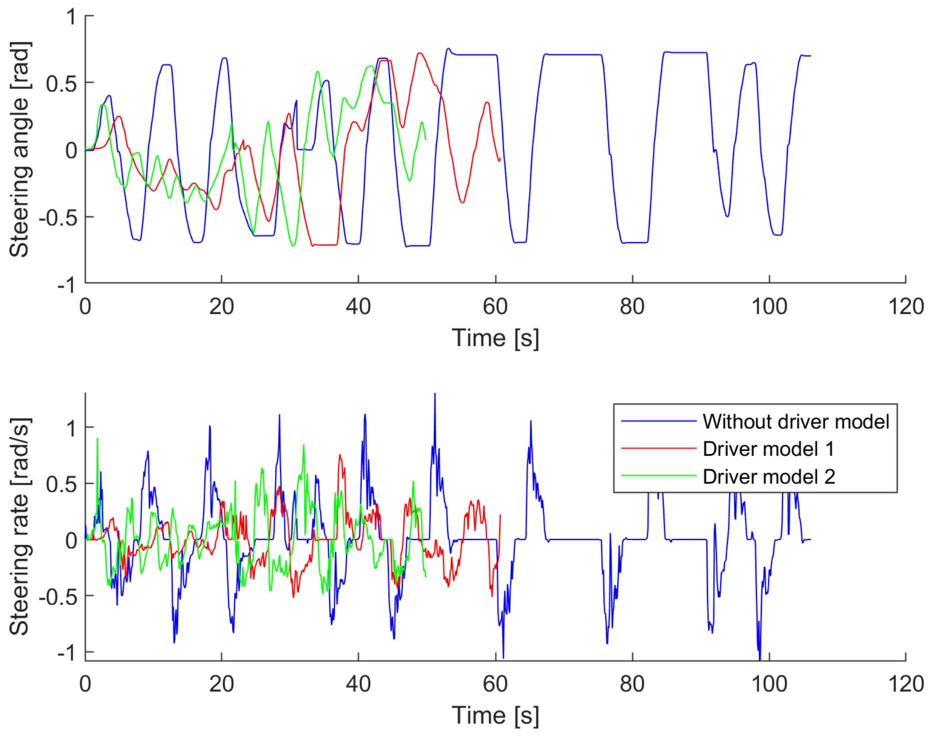


Figure 5-10: Steering actions on path 1 using different driver models.

Figure 5-10 shows the steering actions on path 1 using different driver models. This indicates that the inclusion of a driver model has a positive effect on the steering behaviour of the driver. The absolute steering angle is, on average, relatively smaller and the magnitude of the steering rate is also lower. Moreover, it shows that the total duration of the manoeuvre is considerably shorter when a driver model is used.

Also, an interesting event occurred while conducting these tests. Namely, at one point during the reverse manoeuvre of the first test without a driver model, the solver found that the optimal solution for tracking the reference path was to drive forward again. This was due to the large deviations from the reference path, as can be seen in Figure 5-7 **(a)** and Figure 5-8 **(a)**. As of now, the optimal steering angle is continuously instructed to the driver, and the change of forward or reverse motion is only instructed to the driver upon reaching the end of the current segment. However, this event indicates that it can be advantageous to also continuously instruct an optimal velocity to the driver, or at least a continuous indication of the optimal driving direction (forward or reverse). This could greatly improve the robustness of the overall system.

5-5 Prediction Horizon and Sample Time Testing

The subsequent tests are conducted to investigate how different prediction horizons and prediction model sample times influence the performance of the MPC controller. The computation time of the MPC controller is a vital performance indicator. As the controller is based on solving an optimization problem, the most important factor on the computation time is the prediction horizon. Additionally, the choice for the prediction horizon length goes hand in hand with the choice for the prediction model sample time, where the sample time is the time between each prediction stage. The product of the prediction horizon and the sample time can therefore be regarded as the prediction time. As discussed before, the velocity of the vehicle is assumed to be nearly constant. This means that the driven distance between the current location of the vehicle and the predicted location of the vehicle at the end of the prediction horizon is largely defined by the prediction horizon and the prediction model sample time. Therefore, a few considerations have to be made.

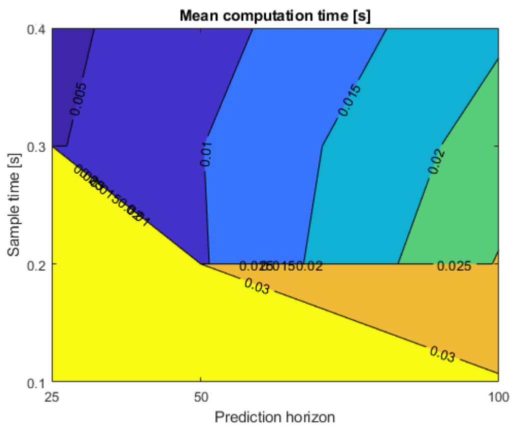
A larger prediction horizon will lead to an increase of decision variables in the optimization problem. Therefore, it gets increasingly more difficult to predict the states at every prediction stage as the prediction horizon increases. A too large sample time will also have a negative impact on the performance. The solutions will be sub-optimal and coarse if the time between different stages is too large, because system input can only change once per stage. However, a too small prediction horizon or sample time may cause the controller to be too short-term focused. This results in aggressive steering actions or infeasibility when a turn needs to be made, as the controller does not anticipate for this manoeuvre.

For these tests, the computation time is one of the major properties being measured. It is important that the MPC setup is similar to the way it operates in the real-life scenario. Therefore, the system will be tested in the simulation environment where the driver model is included in the MPC setup. The driver model used for these tests is the second identified driver model presented in Section 5-4. Also, the same driver model is used to simulate the behaviour of the driver. Therefore, no input from a human is required while conducting these tests, thus the tests can be conducted in an accelerated manner. The controller will in that case probably perform relatively better than in real-world scenarios, as the prediction driver model is equal to the driver model used to simulate the behaviour of the driver.

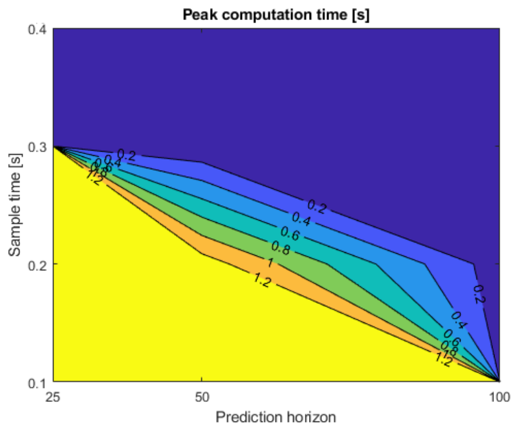
The parking scenario is simulated with the output reference tracking weight $w_{\mathbf{y}}$ set at a value of 2, and the suppress steer rate weight $w_{\Delta\delta}$ set at a value of 1. The prediction horizon is set at either 25, 50 or 100 stages and the prediction model sample time is set at either 0.1, 0.2, 0.3 or 0.4 seconds. This makes for a total of 12 different combinations of prediction horizon and sample time to test, which are all tested for all four of the test paths.

5-5-1 Results

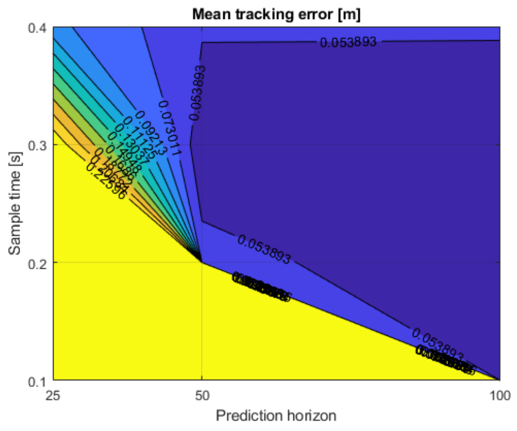
For the relevant KPIs, the mean values of all four test paths are shown in Figure 5-11. The yellow lower left part of all three graphs indicate an infinitely high number. This is caused by the combination of prediction horizon and prediction model sample time resulting in the controller being unable to successfully track the reference path, thus infinitely high computation time and tracking error. Furthermore, it can be seen that an increase of the prediction horizon leads to an increase of the mean computation time. In contrary to the peak computation time, as this seems to be mainly influenced by the sample time. Also, a high prediction horizon has a great positive effect on the path tracking error. For the following tests, a prediction horizon of 100 steps and a prediction model sample time of 0.3 is chosen. The results indicate that this will result in a fairly low tracking error, while still ensuring timely updated instructions for the driver, due to a relatively low computation time.



(a) Mean computation time.



(b) Peak computation time.

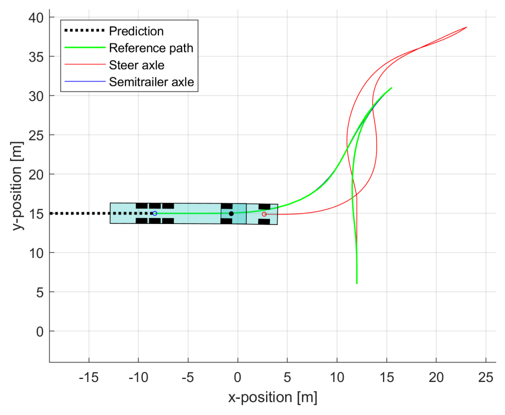


(c) Mean tracking error.

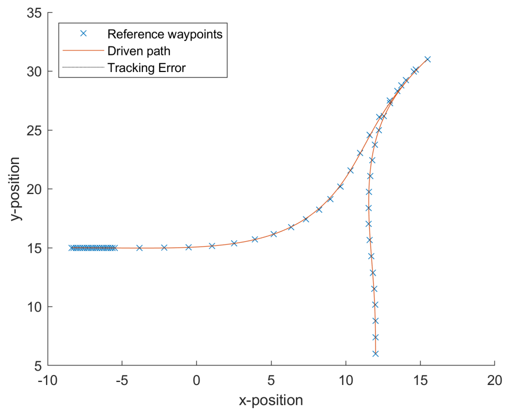
Figure 5-11: Different horizon and sample time test results.

5-6 Driver Delay Compensation Testing

The previous tests have indicated a suitable choice for weights, prediction horizon and prediction model sample time. These choices are validated by testing the performance of the MPC-based system under ideal circumstances without driver behaviour and without a driver model. Figure 5-12 to Figure 5-15 show the results of the docking manoeuvre under ideal circumstances with a prediction horizon N of 100 stages, a prediction model sample time T_s of 0.3 seconds, an output reference tracking weight $w_{\mathbf{y}}$ set at a value of 2, and a suppress steer rate weight $w_{\Delta\delta}$ set at a value of 1. The figures show that the controller is able to track the reference path with high accuracy.

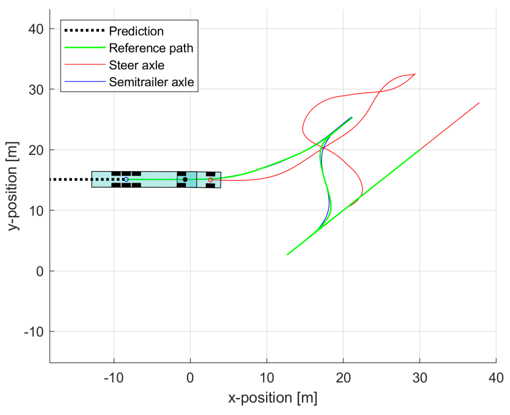


(a) Parking manoeuvre results.

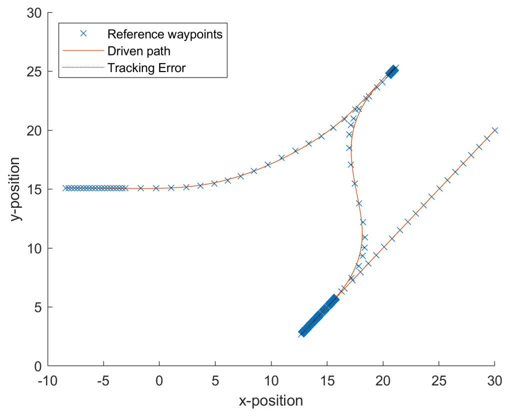


(b) Tracking error.

Figure 5-12: Truck docking results under ideal circumstances - test path 1.

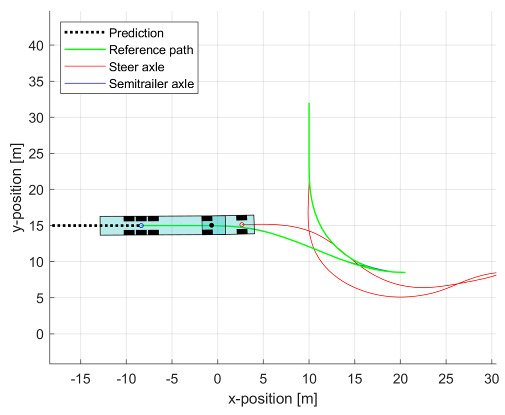


(a) Parking manoeuvre results.

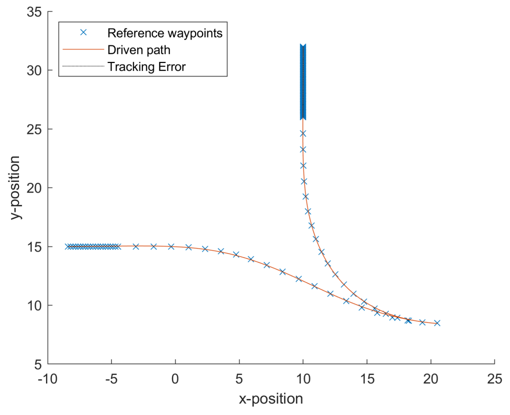


(b) Tracking error.

Figure 5-13: Truck docking results under ideal circumstances - test path 2.

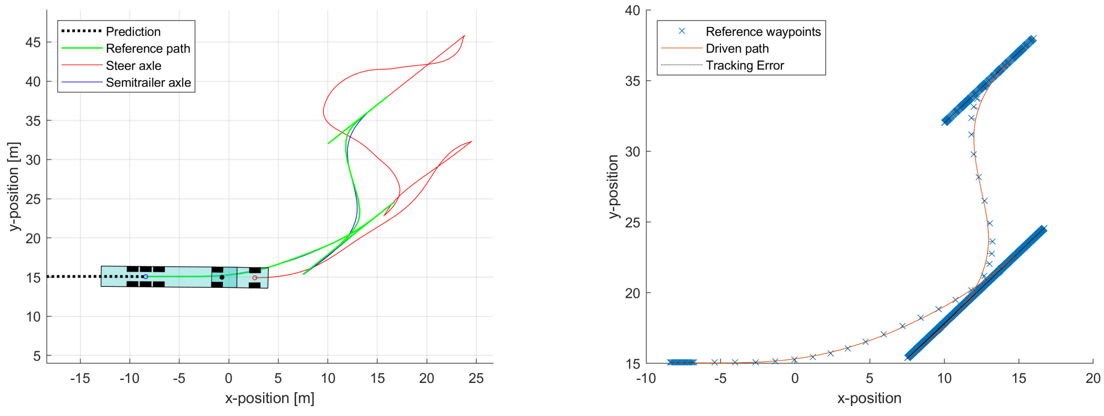


(a) Parking manoeuvre results.



(b) Tracking error.

Figure 5-14: Truck docking results under ideal circumstances - test path 3.



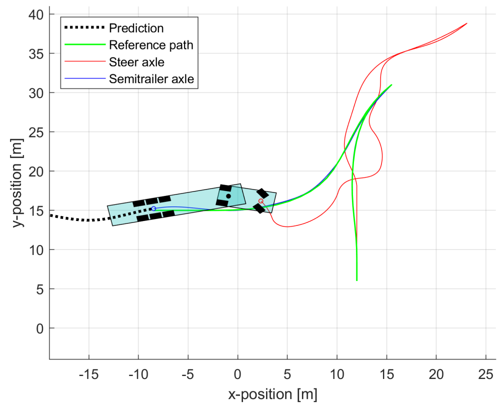
(a) Parking manoeuvre results. **(b)** Tracking error.

Figure 5-15: Truck docking results under ideal circumstances - test path 4.

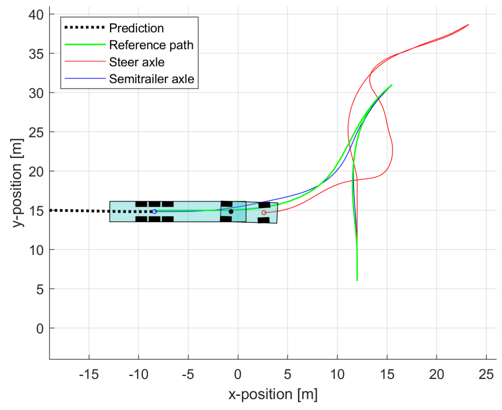
Additionally, the robustness of the system when subject to disturbances can be tested. As described in Section 3-7-1, the prediction of the MPC controller is utilized to compensate for the reaction time of the driver. In these tests, the effectiveness of this approach will be investigated. First, the maximum allowed delay is measured for the MPC setup without driver delay compensation. Here, the controller uses the first element of the optimal input sequence as control input. Afterwards, the same is done for the MPC setup with driver delay compensation, to compare. In this case, the controller uses a subsequent element of the optimal input sequence as control input to compensate for the driver delay. The maximum allowed delay is found by repeating the simulation and increasing the delay with increments of 0.1 seconds, until a collision occurs or the controller is unable to find a solution.

5-6-1 Results

Figure 5-16 to Figure 5-19 show the results of the robustness test for all four test paths. The left side of each figure shows the result of the maximum allowed driver delay without driver compensation. The right side of each figure shows the result of the maximum allowed driver delay with driver compensation. First of all, the results show that the MPC-based system is fairly robust against driver delays, even without explicitly compensating it. The system seems to still function with simulated driver delays which are far higher then mean reaction times of humans [39]. Besides, it can be seen that the technique used to compensate the driver delay shows to be effective. The maximum allowed driver delay is almost twice as high for all four test paths compared to the maximum allowed delay without compensation. This offers a greater margin of error for practical implementations of the system. Moreover, the results show that, in almost all cases, the trailer axle still tracks the reference path with high accuracy, albeit at a cost of heavy steering actions. However, it can be seen that the steering actions are reduced when actively compensating the delay, even though the delay is much higher. In these tests, the simulated delay is constant and the compensation is based on the exact value of this delay. In real-life, the driver delay will variate over time and it will have to be estimated. Therefore, it is interesting to also investigate the robustness of the MPC-based system for variable time-delay.

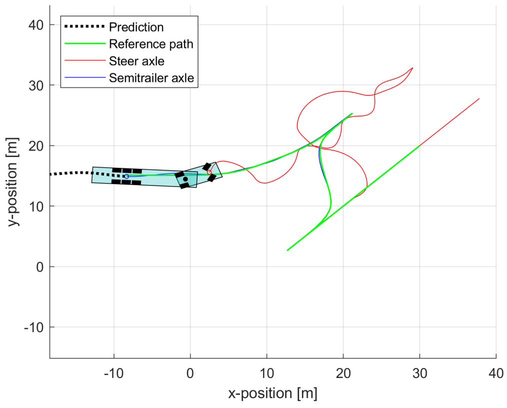


(a) Max. driver delay without compensation $\tau_r = 1.1[s]$.

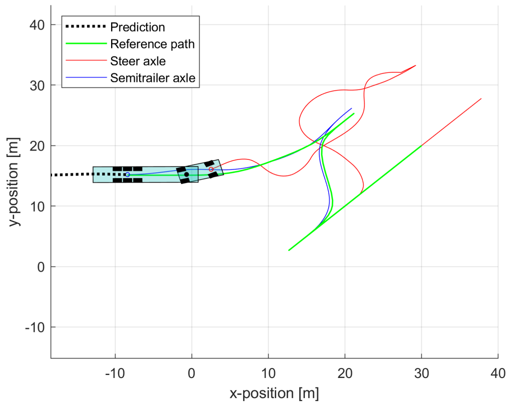


(b) Max. driver delay with compensation $\tau_r = 1.9[s]$.

Figure 5-16: Truck docking results with and without driver delay compensation - test path 1.

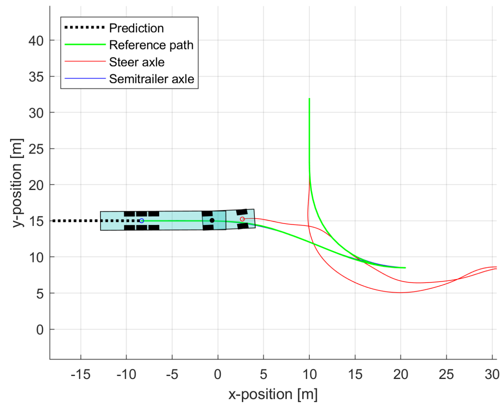


(a) Max. driver delay without compensation $\tau_r = 1.0[s]$.

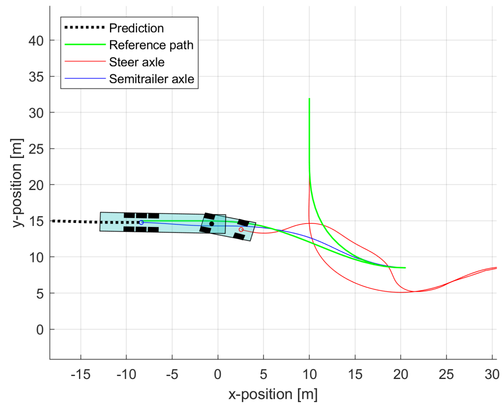


(b) Max. driver delay with compensation $\tau_r = 2.0[s]$.

Figure 5-17: Truck docking results with and without driver delay compensation - test path 2.

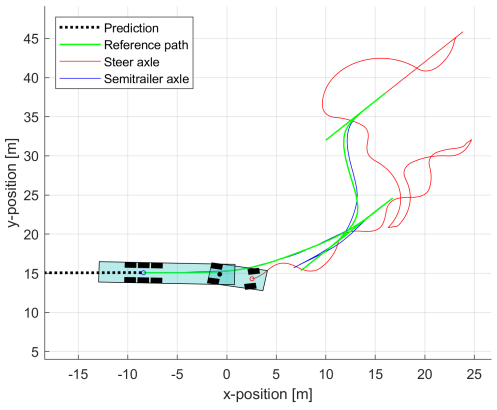


(a) Max. driver delay without compensation $\tau_r = 1.0[s]$.

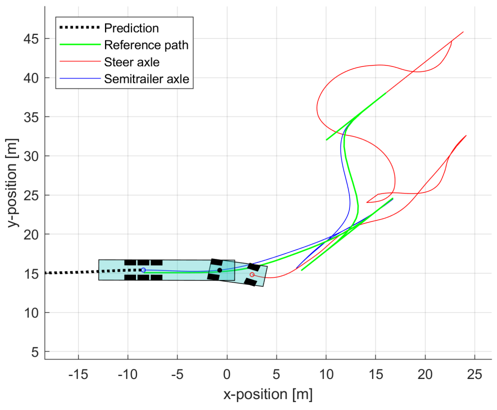


(b) Max. driver delay with compensation $\tau_r = 1.9[s]$.

Figure 5-18: Truck docking results with and without driver delay compensation - test path 3.



(a) Max. driver delay without compensation $\tau_r = 0.9[s]$.



(b) Max. driver delay with compensation $\tau_r = 1.6[s]$.

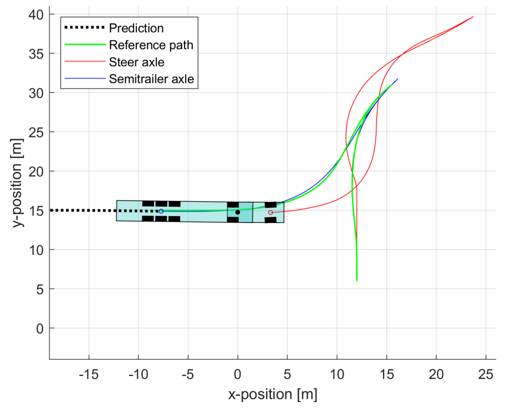
Figure 5-19: Truck docking results with and without driver delay compensation - test path 4.

5-7 Testing in Virtual Reality Simulator

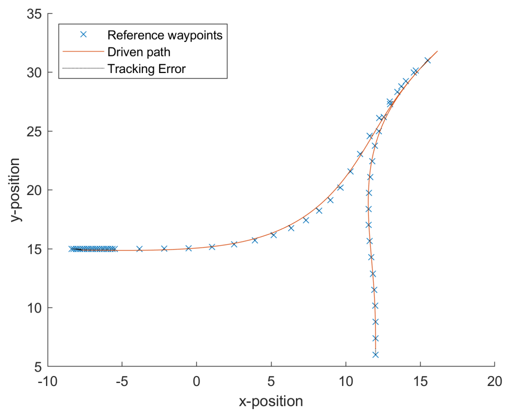
The final set of tests are conducted in the Virtual Reality (VR) simulator, which is the simulation environment explained in Section 4-4. During the tests, two Human Machine Interface (HMI) concepts are used simultaneously. This allowed the drivers to utilize the HMI that they were more comfortable with. The used HMIs are the tablet with application concept two, as visualized in Figure 4-12 and LED concept, as visualized in Figure 4-14. First, the parking manoeuvre is executed by an amateur driver. The person is asked to park the truck-trailer combination between two parked trucks, while being aided with the MPC-based VISTA system. The MPC configuration is setup without a driver model, as the previously identified driver models were from a different simulation environment. Unfortunately, it was impossible to already experiment with driver model identification in the VR simulator due to practical time management limitations. The main goal of these tests is to investigate the performance of the constructed MPC-based driver assistant in a real-world-like scenario. The tests are executed for all four test paths.

5-7-1 Amateur Driver Results

Figure 5-20 to Figure 5-23 show the results of the assisted parking manoeuvre on test path 1 to test path 4, respectively. The results are very promising, as the path tracking error for all paths is minimal. Also, the red path, indicating the traveled path of the steer axle, is fairly smooth. This directly translates to smooth steering actions, which is also desired. However, the amateur driver solely depends on the instructions of the VISTA system. Therefore, the driver is less concerned with keeping track of their surroundings. The professional drivers are expected to pay more attention to this and also make up their own mind on the optimal way of performing the parking manoeuvre. It is interesting to investigate whether the system still performs well when the driver is more involved in the planning of the parking manoeuvre. The driver may than choose to ignore the instructions, resulting in the vehicle diverging from the reference path. This could demonstrate the robustness of the MPC-based driver assistant.

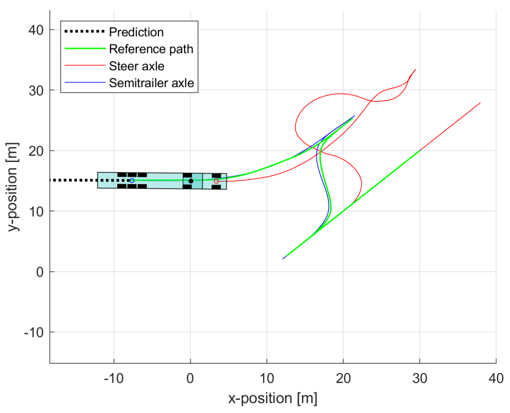


(a) Parking manoeuvre results.

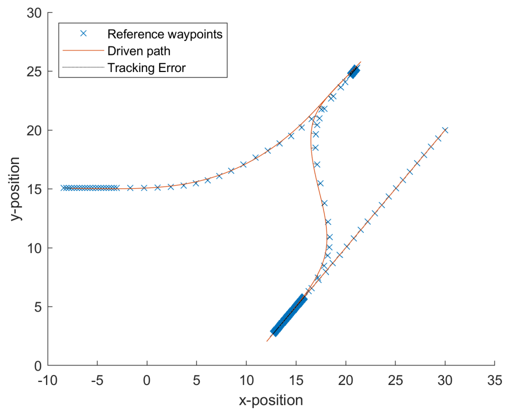


(b) Tracking error.

Figure 5-20: Amateur driver in VR - path 1.

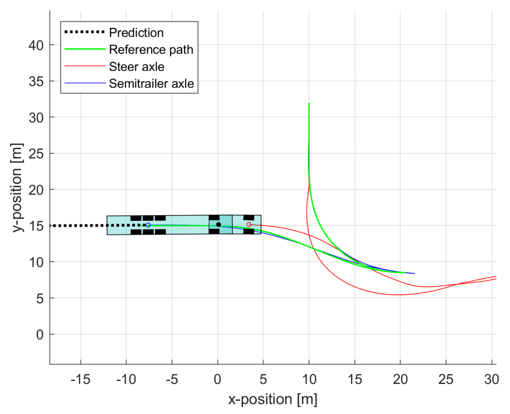


(a) Parking manoeuvre results.

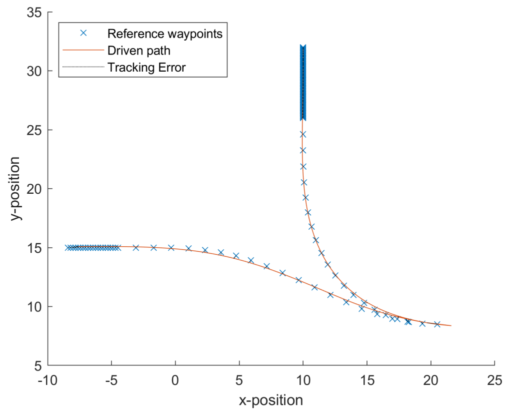


(b) Tracking error.

Figure 5-21: Amateur driver in VR - path 2.

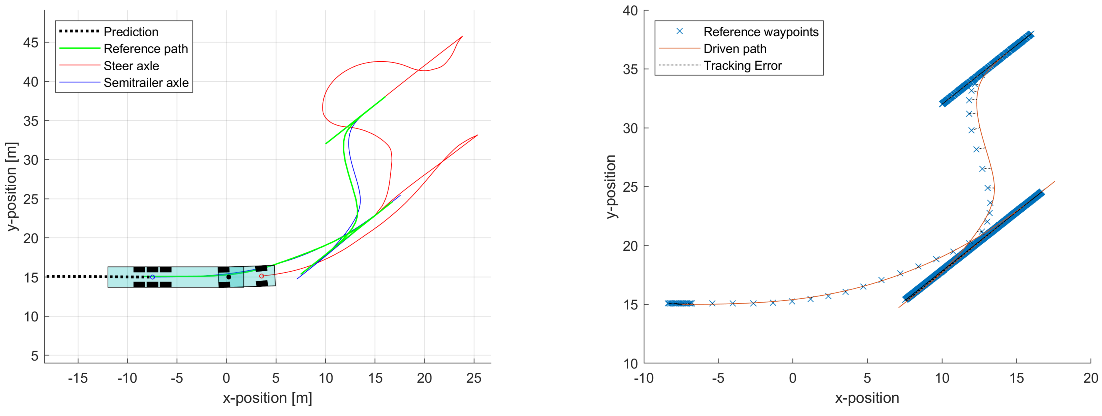


(a) Parking manoeuvre results.



(b) Tracking error.

Figure 5-22: Amateur driver in VR - path 3.



(a) Parking manoeuvre results. **(b)** Tracking error.

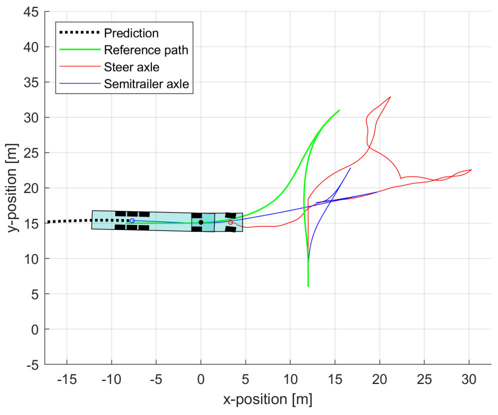
Figure 5-23: Amateur driver in VR - path 4.

5-7-2 Professional Drivers Results

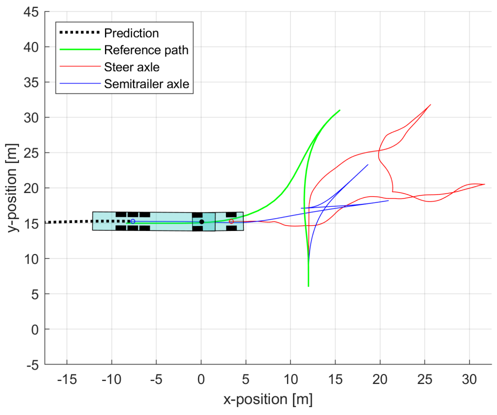
Finally, a set of tests is conducted with 2 professional drivers in the VR simulator. These drivers are experienced in operating truck-trailer combinations. However, they are unfamiliar with the VR environment and the VISTA system. Fortunately, the drivers noted that the behaviour and the dynamics of the simulated truck-trailer closely resembled the real-life dynamics of an articulated vehicle. The drivers were asked to park the vehicle between two parked vehicles in a simulated Distribution Center (DC).

First, the drivers were supposed to execute this manoeuvre without any aid. Thereafter, they were asked to repeat the manoeuvre while given instructions from the MPC-based VISTA system. Multiple tests have been conducted with the drivers. It quickly became apparent that most of the test paths were not suited for this test case. These reference paths felt very unintuitive for the professional driver and did not at all resemble the paths they would have chosen for the specific start and desired end positions. The most insightful tests are the four attempts of the first driver on test path 1 and the three attempts of the second driver on test path 3. Additionally, it would have been interesting to let both drivers test the system using the same reference path. However, this was non-viable as elongated periods of time in the VR simulator are quite exhausting.

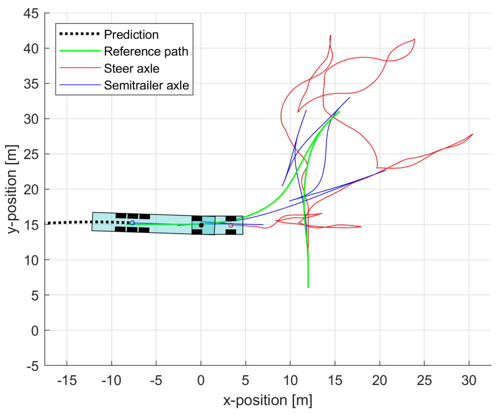
Figure 5-24 shows the results of the four attempts of professional driver 1 on test path 1. The first attempt **(a)** was done without any help from the assistance system. It can be seen that the driven path is very different from the reference path generated by the path planner. When shown the comparison between the two paths, the drivers noted that a professional driver would always make use of the free space directly in front of the intended dock. This allows to already straighten the vehicle before starting the reverse manoeuvre, which results in improved vision for the driver. It can be noticed that the first attempt with instructions from the VISTA system **(b)**, the driver chose to ignore them. For the second attempt with instructions **(c)**, the driver was asked to more actively utilize the instructions given. The driver had a hard time to do so, as the instructions felt unintuitive. At some point during the reversing segment of the reference path, the driver felt his truck-trailer combination came unnecessarily close to colliding with one of the parked vehicle. Also, at this point a blind spot was created, which the professional driver disliked. Therefore, the driver chose to change between forward and reverse motion multiple times in order to correct the orientation of the truck and trailer. For the last attempt **(d)**, the driver was instructed to fully rely on the assistance system. This did result in the vehicle closely tracking the reference path. However, this is undesirable for the intended use of the VISTA system, as it is meant to be a merely assisting system, where the driver still keeps track of their surroundings and makes use of their own driving capabilities.



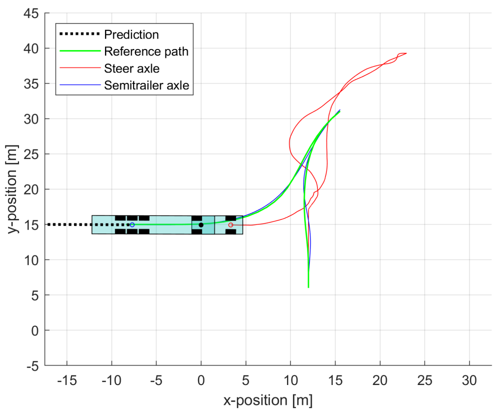
(a) Without VISTA.



(b) MPC-based VISTA try 1.



(c) MPC-based VISTA try 2.



(d) MPC-based VISTA try 3.

Figure 5-24: Professional driver in VR path 1.

Figure 5-25 shows the steering angle and steering rate executed by the driver for all four attempts. It can be noticed that the outliers in the steering rate graph are all from the attempts where the driver was given instructions. This indicates that the instructions included harsher steering actions than the driver would naturally choose. The graphs also give insight in the total duration of the manoeuvre. Additionally, the duration of each attempt is shown in Table 5-5. This shows that the driver seems to be able to perform the manoeuvre faster without the MPC-based VISTA system. This is due to the extra cautious driving, especially at the blind spot location.

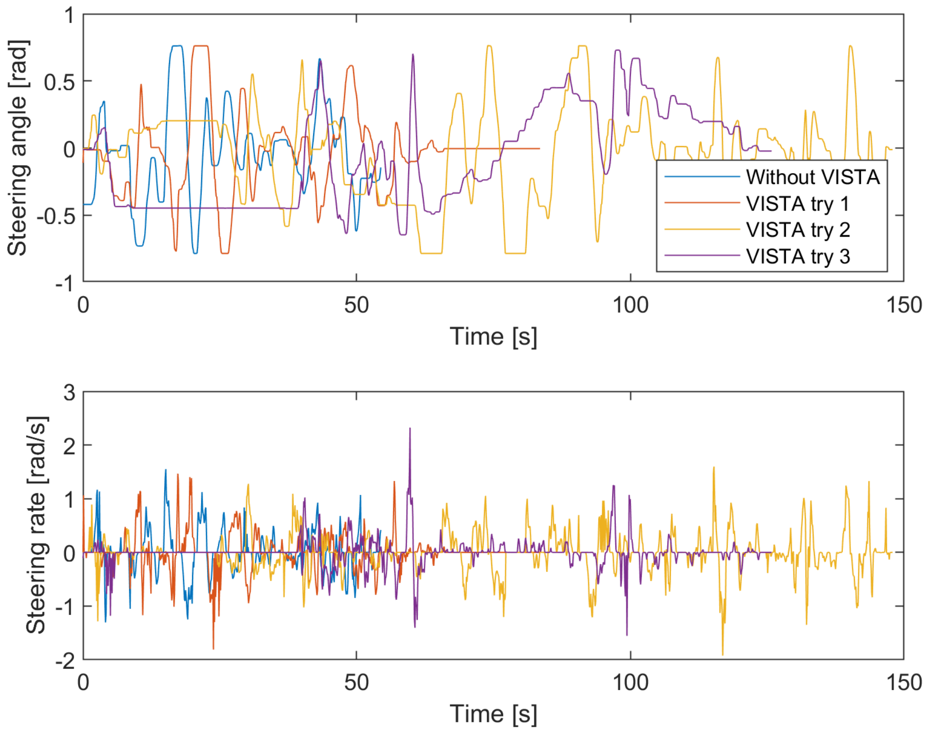
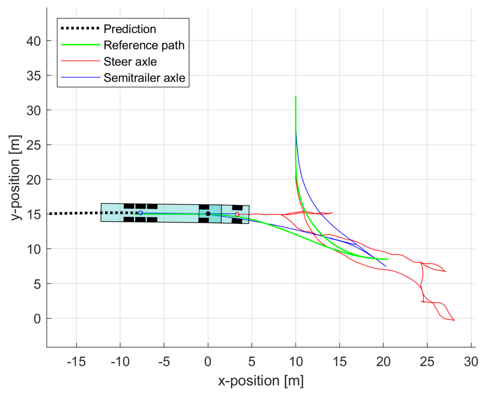


Figure 5-25: Steering actions on path 1 by professional driver.

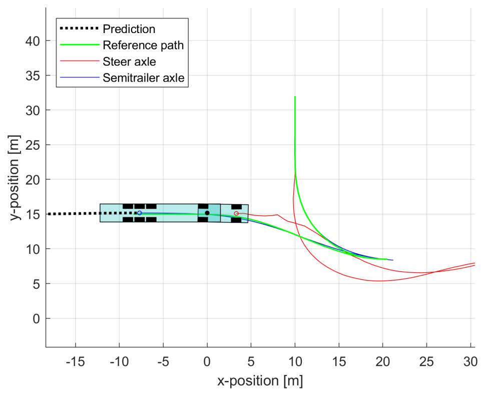
Table 5-5: Duration of parking manoeuvre path 1.

Professional driver attempt	Duration
Without VISTA	54.4 [s]
MPC-based VISTA try 1	83.5 [s]
MPC-based VISTA try 2	147.9 [s]
MPC-based VISTA try 3	125.8 [s]

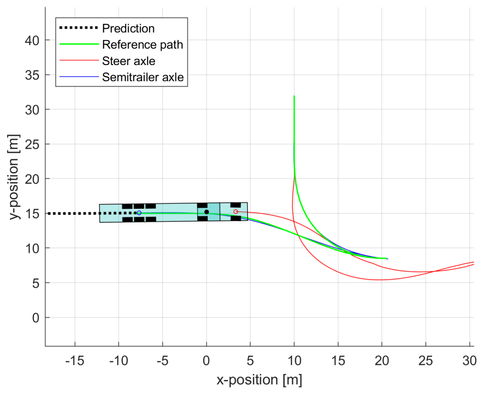
Figure 5-26 shows the results of the three attempts of professional driver 2 on test path 3. To the professional drivers, test path 3 was very different to the other test paths. This is because test path 3 includes a blind side backing action. Blind side backing is a reversing manoeuvre where the view of the trailer is obstructed. This happens when a truck with the steering wheel on the left side, such as the truck in the simulator, approaches from the right side. This is a significantly harder operation for the drivers. The first attempt **(a)** was done without any help from the assistance system. In contrary to test path 1, this test path does seem to closely resemble the path chosen by a professional driver. The two subsequent attempts **(b)**, **(c)** did include instructions from the MPC-based VISTA system. The instructions allowed the driver to execute the parking action with only one change of forward and reverse motion and a minimal tracking error. The driver was very pleased with the result, as he felt that the assistance system could have a great positive effect for these complex blind side backing manoeuvres.



(a) Without instructions.



(b) MPC-based VISTA try 1.



(c) MPC-based VISTA try 2.

Figure 5-26: Professional driver in VR path 3.

Figure 5-27 shows the steering angle and steering rate executed by the driver for all three attempts. It is very interesting that the magnitude of the steering actions of the driver are significantly reduced due to the instructions given. The graph also shows that the steering actions of both attempts with the MPC-based VISTA system are fairly similar. This indicates that the driver has quickly familiarized himself with the system. The graphs also give insight in the total duration of the manoeuvre. Additionally, the duration of each attempt is shown in Table 5-6. The duration of each attempt is very similar. The results show that the MPC-based VISTA system already works very well when the reference path is similar to the path a professional driver would naturally plan.

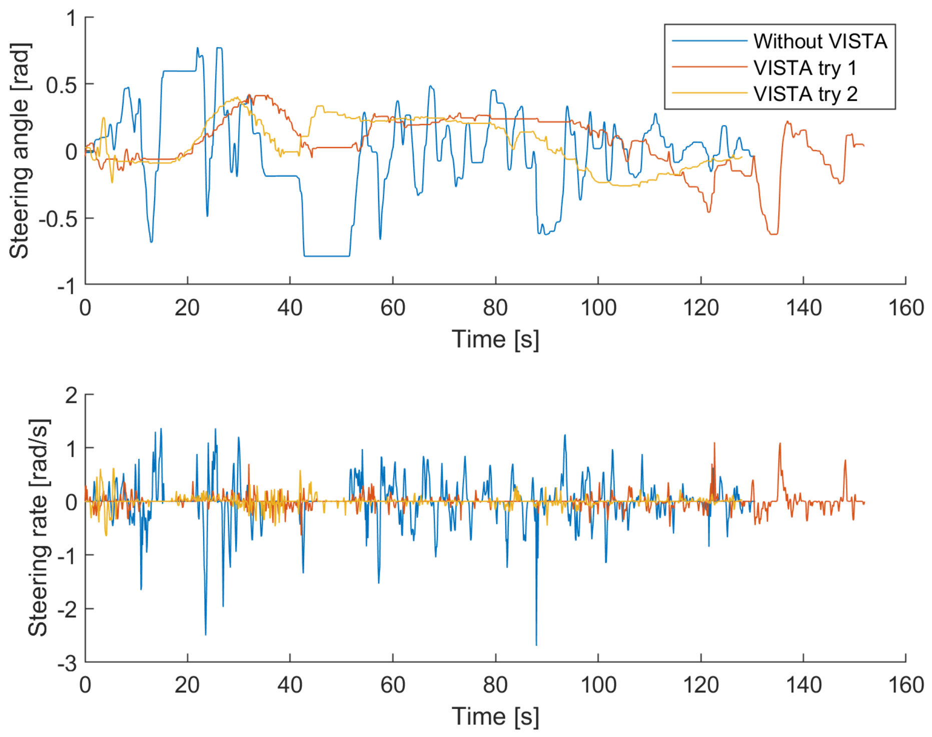


Figure 5-27: Steering actions on path 3 by professional driver.

Table 5-6: Duration of parking manoeuvre path 3.

Professional driver attempt	Duration
Without VISTA	130.6 [s]
MPC-based VISTA try 1	151.8 [s]
MPC-based VISTA try 2	128.1 [s]

Chapter 6

Conclusion

The research objective of this thesis, as stated in Section 1-2, was to investigate how suitable an Model Predictive Control (MPC)-based approach is for the VIsion Supported Truck docking Assistant (VISTA) truck docking assistance system. In this work, a MPC-based driver assistant was presented. The controller uses a kinematic vehicle model to predict the behaviour of the truck-trailer combination, and a second-order linear time-invariant system model to predict the behaviour of the driver. The implementation of said controller in the VISTA framework and the tests in several simulation environments, among which a Virtual Reality (VR) environment, are explained in this work. As of now, the MPC-based solution is preferred over previous concepts of the VISTA system and this work has contributed to the extended paper abstract submitted to the Advanced Vehicle Control (AVEC) Symposium. This extended abstract can be found in Appendix Section A.

The performance of the MPC-based driver assistant was quantified by a Key Performance Indicator (KPI) evaluation. The set of KPIs was determined based on literature, on input given by experienced members of the VISTA project team and from properties deemed important by professional drivers. In each set of tests, the focus was on measuring these KPIs, and on the way these properties could be influenced by different configurations.

The results suggest that an MPC-based system is suitable for assisting truck drivers while performing truck docking manoeuvres. In the hypothetical ideal scenario where the controller is able to directly steer the vehicle, the controller is able to track reference paths with high accuracy. Also, when the driver model is correctly identified, a driver is able to swiftly and accurately execute the entire parking manoeuvre with help of the driver assistant. Furthermore, the MPC-based system shows robustness against driver errors and is able to find solutions when the vehicle has diverged from the reference path. The drawback is that the results demonstrate that the system often only performs well when the driver fully depends on the instructions given. This can be seen by comparing the results of the amateur driver and the professional drivers in the VR simulator. The amateur driver is unable to execute such a complex manoeuvre without the help of the VISTA system. The professional drivers, on the other hand, are very well capable of executing the manoeuvre on their own.

The results also suggest that the usage of a mathematical model to predict the behaviour of the driver is beneficial. The system is able to compensate for the delay of the driver, which resulted in great improvements during the tests with the 2D world simulation environment. Furthermore, the test results confirm that the specific configuration of the MPC controller can influence the performance of the system. The choice of weights directly effects the path tracking results and the comfort of the driver. The drivers are more comfortable with relatively calm and smooth instructions. Certain combination have shown desired results, while other combinations have resulted in the system being unable to effectively performing the manoeuvre. Moreover, the combination of prediction horizon and sample time was demonstrated to be very important.

6-1 Recommendations

The results of the test scenarios have indicated that the use of a driver model to predict the behaviour of the human can improve the overall performance of the system. The first step in using a driver model, is to identify this model using system identification techniques. For this thesis, a program is used to fit the experimental data to a model found in literature. This has allowed to conduct first tests regarding the contribution of the driver model. However, a more in-depth analysis of suitable system identification techniques could be beneficial. Additionally, first tests have only be conducted with one particular driver model, a second-order linear time-invariant system. However, it is known that human drivers are capable of changing their behaviour. Therefore, it could be advantageous to capture the behaviour of the driver continuously with use of adaptive models. My recommendation is to research how the performance of the system can be improved with different system identification techniques and with different driver models.

To add to that, the current identified driver model captures the steering behaviour of the driving when they receive instructions about the desired steering angle. The usage of this model assumes that the actual steering angle of the truck can always be measured or estimated. However, the Real Time Localization System (RTLS) will most likely be unable to precisely measure the current steering angle. As the current steering angle is unknown, it is impossible to instruct the driver with the optimal steering actions. Therefore, a different desired entity, which can be measured by the RTLS, needs to be instructed to the driver. My recommendation is to research how using the desired yaw angle of the truck as instruction to the driver influences the performance of the system, and how the behaviour of the driver can then still be captured by a mathematical model.

The intended users of the finalized system are professional truck drivers. Unfortunately, the tests with professional drivers show that some parts of the system feel unintuitive for the drivers. The drivers noted that most of the reference paths were unlike the paths they would chose given the same scenario. What stood out most was the desire of the drivers to use the free space to line the truck up as much as possible with the dock before driving in reverse. They dislike to approach the dock with a large articulation angle. These comments suggest that the path planner is perhaps missing a component, which drivers deem important when planning a reference path. My recommendation is to research how the performance of the reference path planner can be improved by incorporating more aspects deemed important by professional drivers.

The formulation of the proposed MPC controller involves an optimization of the steering angle as well as the velocity of the vehicle. The latter is considered an optimization variable as this has a great positive effect on the ability of the solver to find feasible solutions. The expectation was that the optimal velocity as computed by the solver would still be approximately constant and equal to the reference velocity. This has also been the case in most of the test scenarios. However, at specific moments, when the vehicle diverged from the reference path with a great amount, the optimal solution found by the solver included driving in the opposite direction as the reference velocity indicated. Indicating this direction change to the driver could improve the robustness of the system. As of now, the optimal velocity is not instructed to the driver. My recommendation is to research whether conveying the optimal velocity as additional instruction to the driver can improve the performance of the system.

The system has shown to be robust against driver errors and is able to track the reference path. When the vehicle has diverged from the reference path, the MPC-based system is able to find solutions that steer the vehicle back to the reference path. Unfortunately, it cannot be guaranteed that the solution found by the solver is collision free when the vehicle has diverged from the reference path. A first step is made in investigating the possibility of including obstacle avoidance in the MPC formulation. My recommendation is to research how the implementation of collision avoidance constraints in the MPC formulation can improve the performance of the system.

The current test scenarios used in order to examine the performance of the system were subject to certain assumptions. One of which, is that the configuration of the vehicle is exactly known. In reality, the dimensions of each truck-trailer combinations will vary. Therefore, the exact dimensions of the vehicle are unknown for the path planner and the MPC controller. My recommendation is to research how the performance of the system is influenced when the exact dimensions of the vehicle are unknown and how the MPC-based system can be configured to properly deal with uncertain vehicle dimensions.

6-1-1 Real-Time Collision Avoidance

As discussed in Section 2-3-3, a clear advantage of using an MPC controller is that it offers the option for implementing real-time collision avoidance by incorporating it as online constraints. During this research, a first attempt was made to implement real-time obstacle avoidance within the MPC-based driver assistance system. However, this was still too primitive for practical implementations within the VISTA project. This first endeavour included adding an additional number of outputs, which represent the distance from the vehicle to all present obstacles. It is then possible to guarantee collision avoidance by setting a minimum distance from the vehicle to the obstacles. Collision avoidance can thus be achieved by constraining the optimization problem, such that solutions are only considered feasible if they are free of collisions. In this case, the region occupied by the ego vehicle $\mathcal{E}(\mathbf{c})$ was represented as a union of M convex sets:

$$\mathcal{E}(\mathbf{c}) = \bigcup_{m=1}^{M} \mathcal{E}^{(m)}(\mathbf{c}) = \mathcal{E}^{(1)}(\mathbf{c}) \cup \cdots \cup \mathcal{E}^{(M)}(\mathbf{c}). \tag{6-1}$$

The region occupied by the sets are defined by the configuration of the vehicle $\mathbf{c}$, which is equal to the output of the system as seen in (3-10).

The region occupied by the obstacles $\mathcal{O}$ is represented as a union of P sets:

$$\mathcal{O} = \bigcup_{p=1}^{P} \mathcal{O}^{(p)} = \mathcal{O}^{(1)} \cup \cdots \cup \mathcal{O}^{(P)}. \tag{6-2}$$

The location and orientation of these sets is considered to be static and to be known in advance. Collision avoidance is ensured by requiring dist($\mathcal{E}(\mathbf{c})$,$\mathcal{O}$)$> d_{\text{min}}$, which is based on (2-38) and d_{min} is the minimum distance used as a safety margin. A valid set of constraints that adhere to this requirement is

$$\text{dist}(\mathcal{E}^{(m)}(\mathbf{c}), \mathcal{O}^{(p)}) > d_{\text{min}} \qquad \forall m \in \{1, 2, \dots, M\},\ \forall p \in \{1, 2, \dots, P\}. \tag{6-3}$$

This means the minimum distance from each convex subset of $\mathcal{E}(\mathbf{c})$ to each convex subset of $\mathcal{O}$ is at least d_{min}. Some scenarios have been analyzed in order to test an initial attempt for incorporating real-time collision avoidance in the proposed MPC scheme. In these cases, the truck-trailer combination is represented as a union of M circular sets, and the obstacles are represented as a union of P circular sets. A union of circular regions, or disks, is somewhat limited in the capability of representing the vehicle and the obstacles. For example, the shape of the truck and the trailer are both fairly rectangular. To accurately represent these shapes as a combination of multiple disks, one would need a great amount of infinitesimal small circular regions. Another option is to let the area covered by the disks be slightly larger than the actual area covered by the vehicle. This means it is possible to construct the set such that collision avoidance is still guaranteed, albeit with an additional safety margin. The biggest advantage of this representation, is the possibility it provides for a more straightforward formulation of the collision avoidance constraints. Therefore, this is a reasonable approach to start implementing real-time collision avoidance.

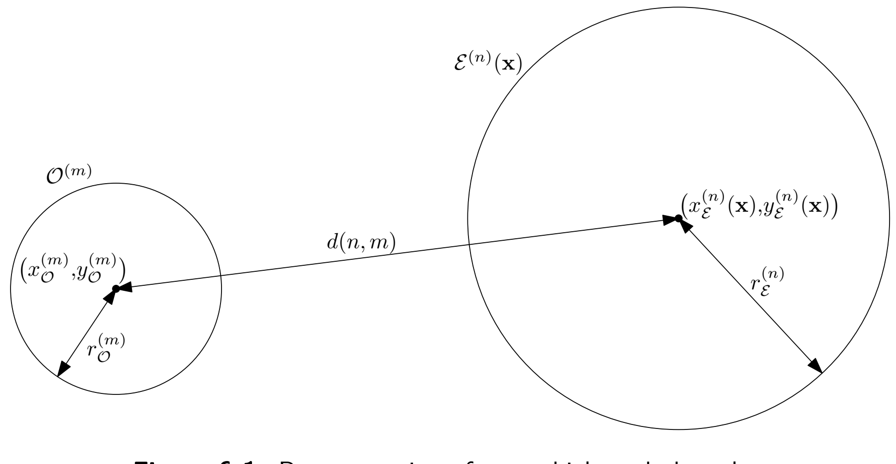

$\mathcal{E}^{(n)}(\mathbf{x})$
$\mathcal{O}^{(m)}$
$(x_{\mathcal{E}}^{(n)}(\mathbf{x}),y_{\mathcal{E}}^{(n)}(\mathbf{x}))$
$d(n,m)$
$(x_{\mathcal{O}}^{(m)},y_{\mathcal{O}}^{(m)})$
$r_{\mathcal{E}}^{(n)}$
$r_{\mathcal{O}}^{(m)}$

Figure 6-1: Representation of ego vehicle and obstacles.

In Figure 6-1 the chosen representation is visualised. Every set $\mathcal{E}^{(m)}(\mathbf{c})$ and $\mathcal{O}^{(p)}$ is a disk defined by the coordinates of its center and its radius. Where the radius is constant for all, while the coordinates are only fixed for every set $\mathcal{O}^{(p)}$. For every set $\mathcal{E}^{(m)}(\mathbf{c})$, the coordinates depend on the current location and orientation of the vehicle. The x-coordinate and y-coordinate of the center of set $\mathcal{E}^{(m)}(\mathbf{c})$ are denoted as $x_{\mathcal{E}}^{(m)}(\mathbf{c})$ and $y_{\mathcal{E}}^{(m)}(\mathbf{c})$, respectively. The radius of this disk is denoted as $r_{\mathcal{E}}^{(m)}$. The center of set $\mathcal{O}^{(p)}$ is defined by the coordinates $x_{\mathcal{O}}^{(p)}$ and $y_{\mathcal{O}}^{(p)}$ and its radius is denoted as $r_{\mathcal{O}}^{(p)}$. The distance from the center of a set $x_{\mathcal{E}}^{(m)}(\mathbf{c})$ to the center of a set $\mathcal{O}^{(p)}$ is denoted as $d(m,p)$ and can be computed by the euclidean distance as follows:

$$d(m,p) = \sqrt{\left(x_{\mathcal{E}}^{(m)}(\mathbf{c}) - x_{\mathcal{O}}^{(p)}\right)^2 + \left(y_{\mathcal{E}}^{(m)}(\mathbf{c}) - y_{\mathcal{O}}^{(p)}\right)^2}. \tag{6-4}$$

The collection of constraints from (6-3) can be rewritten in order to ensure that the distance between the edge of the disks is always larger than $d_{\min}$. This results in the following set of constraints:

$$d(m,p) < d_{\min} + r_{\mathcal{E}}^{(m)} + r_{\mathcal{O}}^{(p)} \qquad \forall m \in \{1,2,\ldots,M\},\ \forall p \in \{1,2,\ldots,P\}. \tag{6-5}$$

As mentioned, an additional safety distance is included as a result of representing the vehicle as a union of disks. This is visualized in Figure 6-2. The additional safety could have a negative impact on the controller. This is due to occurrences where the MPC controller discards certain manoeuvres where the union of disks representing the vehicle would collide with the area occupied by the obstacles, while in reality the vehicles could have performed such a manoeuvre without colliding. Therefore, it could be troublesome to represent the vehicle as a larger area than the actual area it covers.

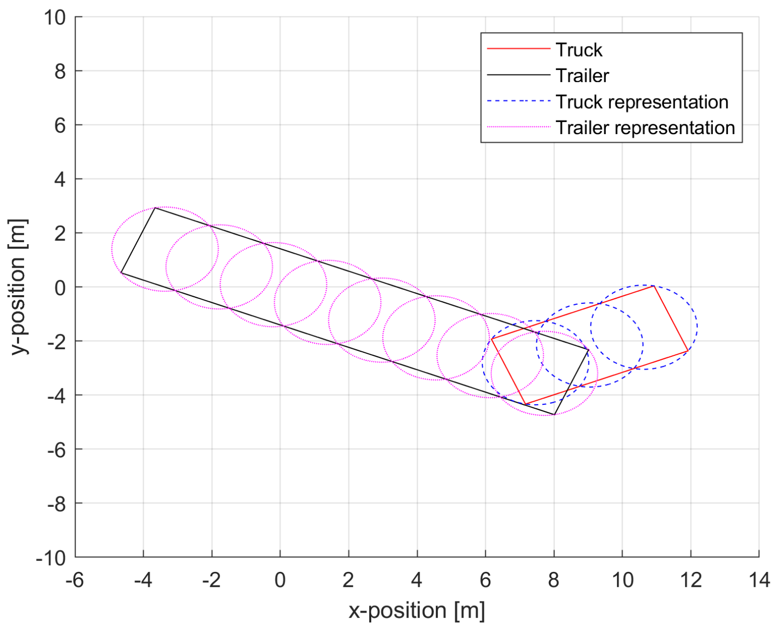


Figure 6-2: Representation of vehicle with multiple disks.

My recommendation is to continue this research regarding the incorporation of real-time collision avoidance in MPC-based VISTA system to improve the safety of the driver assistant.

Appendix A

Extended Paper Abstract

The following two pages comprise the extended abstract submitted to the International Symposium on Advanced Vehicle Control (AVEC). Acceptance for submission of the full paper to the AVEC symposium is pending.

Model Predictive Control based Driver Support for Docking of Articulated Vehicles at Logistics Areas

Abram Dekker [1]), Jason van Kolfschoten [2]), Karel Kural [2]), Laura Ferranti [1]), Jan Benders [2])
[1])Delft University of Technology, Netherlands
[2])HAN University of Applied Sciences, Netherlands
E-mail: jan.benders@han.nl

Bi-directional maneuvering of articulated vehicles at distribution centers is a complex task even for an experienced driver given *(i)* the unstable nature of the vehicle combination whilst reversing, *(ii)* a limited field of view, and *(iii)* a constrained maneuverability space. To support the driver, a novel driver-assist system is established, which consists of a computer vision-based localization module, a vehicle navigation system, and a human machine interface (HMI). This paper focuses on a fundamental module for the vehicle navigation system, that is, the design of a model-predictive-control(MPC)-based tracking controller. This controller is responsible for providing an input for the HMI, based on a known reference path and the actual vehicle pose resulting from driver-vehicle interaction. The controller is validated in a virtual reality simulator with human-drivers in the loop.

Topics: Advanced Driver Assistance Systems, Driver-Vehicle Systems, Testing and Validation

1. INTRODUCTION

The volume of cargo in Europe transported on the road is continuously increasing over the past decade. In the future, it may be expected that more vehicles on the roads will be needed to satisfy the transport demand [1], creating challenges on distribution centers and yards, where the vehicle combination needs to be parked towards the loading dock.

Although the automatization inside the warehouses and distribution centers already took place decades ago, the automation outside, at the parking areas, has not emerged so far. The docking of the vehicle combination towards the loading dock is still done manually by the drivers alike decades ago, even though safety risks exist when operating the vehicle combination.

As confirmed by the measurements with human drivers during bi-directional low-speed maneuvering with articulated vehicle combinations [2], the driver primarily suffers from a lack of view from the cabin, which is limited to the frontal outlook and the rear mirrors. Moreover, the driver is challenged to control the naturally unstable vehicle combination during reversing at an area which is typically limited in space. To address these challenges, the VIsion Supported Truck docking Assistant (VISTA) is being developed [3]. The framework consists of a computer vision-based localization module, a vehicle navigation system consisting of a path planner and a path tracking controller, and a human machine interface (HMI) to support the driver [4]. The functionality of the system is being extensively tested in a Virtual Reality (VR) simulator.
Compared to the framework presented in [4], this paper focuses on the design of a novel MPC-based path tracking controller to improve the VISTA vehicle navigation system.

2. PROBLEM DEFINITION

The major role of the path tracking controller is to minimize the tracking error between a reference path (provided by a path planner) and the center of the semitrailer axle group whilst actuating the steering angle of the hauling unit represented by the tractor. Subsequently, the steering angle is being fed as an input for the HMI, which transforms the required steering angle to the audio/visual advice for the driver, who acts as the actuator of the steering angle. In the context of VISTA, the controller should consider the presence of a human driver. In our work, we model the driver as an imperfect actuator introducing noise and delays in the control loop. In addition, the controller needs to be functional bi-directionally (i.e. for both forward and reversing directions).

In this work, we rely on Model Predictive Control (MPC). MPC optimizes the navigation objectives and the behavior of the vehicle over a finite time window by relying on online numerical optimization tools. This allows the controller to compensate for deviations from the reference path due to dynamical limitations. Additionally, MPC allows one to incorporate a driver model and consequently to compensate for possible delays in the driver’s reactions.

3. RESEARCH APPROACH

MPC is an advanced control technique which can be used to control Multiple-Input and Multiple-Output (MIMO) systems. A (nonlinear) model is used to predict the state evolution $\boldsymbol{x}$ over a finite time horizon N. At each time step, when new measurement $\boldsymbol{x}_{\text{init}}$ are provided by the sensors, an optimization problem is solved to compute the optimal sequence of control inputs $\boldsymbol{u_N}$:

$$\begin{aligned}
&\min_{\boldsymbol{u_N}} \sum_{k=t}^{t+N} J(\boldsymbol{u_N}) \\
&\text{s.t.} \quad \boldsymbol{x}(\text{t}) = \boldsymbol{x}_{\text{init}} \\
&\qquad \boldsymbol{x}(k+1) = f\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big) \qquad \forall k \in [t, N-1] \\
&\qquad g\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big) = 0 \qquad \forall k \in [t, N] \\
&\qquad h\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big) \leq 0 \qquad \forall k \in [t, N]
\end{aligned}$$

The objective is to minimize the sum of the stage costs $J(\boldsymbol{u_N})$. In our context, the cost consists of three terms which aim at minimizing the tracking error, while following a desired reference velocity and suppressing large adjustments of control inputs (this term allows the controller to provide smoother steering instructions to the driver). Furthermore, the controller combines the vehicle [5] and driver [6] dynamics $f\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big)$ and vehicle limitations inequality constraints. These describe the physical limitations $h\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big) \leq 0$, such as the maximum steering and articulation angles.

The six entities defining the state of the system $\boldsymbol{x} = [x_1, y_1, \theta_1, \gamma_1, x_{\text{d}1}, x_{\text{d}2}]^{\boldsymbol{T}}$ are the coordinates of the trailers rear axle, the yaw angle of the truck, the articulation angle and two driver states, respectively. With the vehicle's dynamics represented by the kinematic vehicle model for a truck-trailer combination as presented in [5], and the drivers behavior represented by the second order driver model as presented in [6], the continuous dynamics of the total system are:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} v_0\left(\cos\gamma_1 \cos\theta_1 - \frac{L_{0b}}{L_{0f}} \cos\theta_1 \tan\delta\right) \\ v_0\left(\cos\gamma_1 \sin\theta_1 - \frac{L_{0b}}{L_{0f}} \sin\gamma_1 \sin\theta_1 \tan\delta\right) \\ \frac{v_0}{L_{1f}}\left(\frac{L_{0b}}{L_{0f}} \cos\gamma_1 \cos\theta_1 - \frac{L_{0b}}{L_{0f}} \cos\theta_1 \tan\delta\right) \\ \frac{v_0}{L_{0f}L_{1f}}\left(L_{1f} \tan\delta - L_{0f} \sin\gamma_1 - L_{0b} \cos\gamma_1 \tan\delta\right) \\ x_{\text{d}2} \\ -\frac{1}{T_l T_N} x_{\text{d}1} - \frac{T_l + T_N}{T_l T_N} x_{\text{d}2} + \delta_{\text{d}} \end{bmatrix}$$

The control inputs to the system are the velocity of the truck v_0 and the steering angle instruction given to the driver δ_{d}. The actual steering angle of the truck δ is determined by the driver states as follows:

$$\delta = \frac{K}{T_l T_N} x_{\text{d}1} + \frac{K T_L}{T_l T_N} x_{\text{d}2}$$

The parameters L_{0b}, L_{0f} and L_{0b} are the relevant dimensions of the truck and trailer, while T_l, T_N and T_L are the parameters defining the behavior of the driver.

4. FIRST RESULTS

The controller setup is tested by simulating the system in the MATLAB Simulink environment. This test environment consists of a reference path, which is generated using motion primitives. The reference path describes the reference x- and y- coordinates of the rear axle of the trailer. The figures below show a docking maneuver using the MPC-based controller, taking driver behavior into account. The actual driven path closely matches the reference path.

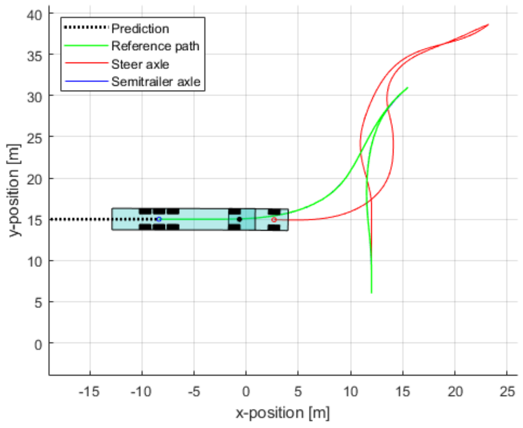


Fig. 1 Reference path tracking – top view

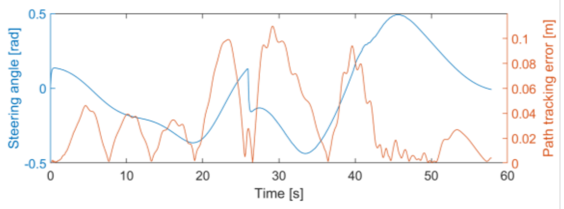


Fig. 2 Path tracking results – steering angle and path tracking error

5. RESEARCH OUTLOOK AND CONCLUSION

The MPC-based controller proves to be a valuable component of the VISTA framework. In first evaluations using the VR-simulator, the controller appeared to be more forgiving to driver-introduced steering angle deviations compared to previously developed controllers, i.e. desired steering angle dynamics were perceived as more naturalistic. In the full paper we will provide more details about the control design and an extensive evaluation of our method in the VR-simulator.

REFERENCES

[1] European Commission, Impact Assessment - accompanying document to the White Paper, pp. 137, Brussels, 2011.

[2] Kural, K., Besselink, I.J.M., Pauwelussen, J.P., Nijmeijer, H. & Patel, K., Analysis of driver behavior during reverse driving of double articulated vehicles, Proceedings HVTT13, San Luis, Argentina, 2014

[3] www.vista-project.eu

[4] Ribeiro, P., et al., VR Truck Docking Simulator Platform for Developing Personalized Driver Assistance, Applied Sciences, 2021, 11, 8911. https://doi.org/10.3390/ app11198911

[5] Kural, K., Analysis of high capacity vehicles for Europe: application of performance based standards and improving manoeuvrability. PhD thesis, Department of Mechanical Engineering, 2019.

[6] McRuer, D. & Krendel, S., Mathematical models of human pilot behaviour. AGARDograph AGARD-AG-188, Advisory Group for Aerospace Research & Development, 1974.

Bibliography

[1] Eurostat, "Share of road in inland freight transport on the rise." https://ec.europa.eu/eurostat/product?code=DDN-20200904-1, Sep 2020.

[2] European Commission, Brussels, *Impact Assessment - accompanying document to the White Paper*, 2011. pp. 137.

[3] European Commission. Statistical Office of the European Union., *Energy, transport and environment statistics: 2020 edition.* Publications Office, 2020.

[4] C. Altafini, A. Speranzon, and B. Wahlberg, "A feedback control scheme for reversing a truck and trailer vehicle," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 915–922, 2001.

[5] P. Ribeiro, A. F. Krause, P. Meesters, K. Kural, J. van Kolfschoten, M.-A. Büchner, J. Ohlmann, C. Ressel, J. Benders, and K. Essig, "A vr truck docking simulator platform for developing personalized driver assistance," *Applied Sciences*, vol. 11, no. 19, 2021.

[6] M. LS, "Modern loading dock." https://commons.wikimedia.org/wiki/File:Modern_loading_dock.jpg, May 2011. License: CC BY-SA 4.0.

[7] VISTA, "Home - vista - han automotive." https://vistaproject.eu/. Accessed: 10-02-2022.

[8] A. Dekker, "Motion planning and control for parking manoeuvres with articulated vehicles and a driver in the loop: Literature survey," Master's thesis, Delft University of Technology, 2021.

[9] R. Rajamani, *Vehicle dynamics and control.* Springer Science & Business Media, 2011.

[10] K. Kural, *Analysis of high capacity vehicles for Europe: application of performance based standards and improving manoeuvrability.* PhD thesis, Department of Mechanical Engineering, oct 2019. Proefschrift.

[11] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, 2015.

[12] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, Dec 1959.

[14] K. Erciyes, *Distributed Graph Algorithms for Computer Networks.* Springer London, 2013.

[15] N. Evestedt, "Sampling based motion planning for heavy duty autonomous vehicles." Licentiate dissertation, Linköping University Electronic Press, Linköping, 2016.

[16] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] D. Devasia, "Motion planning with obstacle avoidance for autonomous docking of single articulated vehicle," Master's thesis, HAN University of Applied Sciences, 2019.

[18] M. Kannan, "Automated docking maneuvering of an articulated vehicle in the presence of obstacles," Master's thesis, HAN University of Applied Sciences, 2021.

[19] R. Wallace, A. T. Stentz, C. Thorpe, H. Moravec, W. R. L. Whittaker, and T. Kanade, "First results in robot road-following," in *Proceedings of 9th International Joint Conference on Artificial Intelligence (IJCAI '85)*, vol. 2, pp. 1089 – 1095, August 1985.

[20] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Tech. Rep. CMU-RI-TR-92-01, Carnegie Mellon University, Pittsburgh, PA, January 1992.

[21] D. T. McRuer and E. S. Krendel, "Mathematical models of human pilot behavior," AGARDograph AGARD-AG-188, Advisory Group for Aerospace Research and Development, 01 1974.

[22] D. T. McRuer and H. R. Jex, "A review of quasi-linear pilot models," *IEEE Transactions on Human Factors in Electronics*, vol. HFE-8, no. 3, pp. 231–249, 1967.

[23] C. C. Macadam, "Understanding and modeling the human driver," *Vehicle System Dynamics*, vol. 40, no. 1-3, pp. 101–134, 2003.

[24] C. C. MacAdam, "Frequency domain methods for analyzing the stability and maneuverability of driver/vehicle systems," in *Proceedings of the International Conference on Modern Vehicle Design Analysis*, International Association for Vehicle Design, 1983.

[25] A. J. Pick and D. J. Cole, "A mathematical model of driver steering control including neuromuscular dynamics," *Journal of dynamic systems, measurement, and control*, vol. 130, no. 3, 2008.

[26] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical model predictive control framework for autonomous ground vehicles," in *2008 American Control Conference*, pp. 3719–3724, 2008.

[27] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1398–1404, 2016.

[28] J. H. Jhang and F. L. Lian, "An autonomous parking system of optimally integrating bidirectional rapidly-exploring random trees* and parking-oriented model predictive control," *IEEE Access*, vol. 8, pp. 163502–163523, 2020.

[29] G. Bai, Y. Meng, L. Liu, W. Luo, Q. Gu, and L. Liu, "Review and comparison of path tracking based on model predictive control," *Electronics*, vol. 8, no. 10, 2019.

[30] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach.* Cambridge University Press, 2012.

[31] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados: a modular open-source framework for fast embedded optimal control," 2020.

[32] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, vol. 93, no. 1, pp. 13–29, 2020.

[33] D. Lam, C. Manzie, and M. C. Good, "Model predictive contouring control for biaxial systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 2, pp. 552–559, 2013.

[34] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2018.

[35] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," 2020.

[36] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[37] J. Lépine, X. Na, and D. Cebon, "An empirical tire-wear model for heavy-goods vehicles," *Tire Science and Technology*, 06 2021.

[38] M. Plöchl and J. Edelmann, "Driver models in automobile dynamics application," *Vehicle System Dynamics*, vol. 45, no. 7-8, pp. 699–741, 2007.

[39] C. A. Nagler and W. M. Nagler, "Reaction time measurements," *Forensic Science*, vol. 2, pp. 261–274, 1973.

[40] A. Domahidi and J. Jerez, “Forces professional.” Embotech AG, `https://embotech.com/FORCES-Pro`, 2014–2019.

[41] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, In Press, 2018.

Glossary

List of Acronyms

AVEC	Advanced Vehicle Control
DC	Distribution Center
HAN	Hogeschool van Arnhem en Nijmegen
HMI	Human Machine Interface
KPI	Key Performance Indicator
LMPC	Linear Model Predictive Control (MPC)
MIMO	Multiple-Input and Multiple-Output
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
NMPC	Nonlinear MPC
RTLS	Real Time Localization System
TU Delft	Delft University of Technology
VISTA	VIsion Supported Truck docking Assistant
VR	Virtual Reality

List of Symbols

δ	Steering angle
δ^*	Virtual steering angle
$\delta_{\max}$	Maximum absolute steering angle
δ_d	Steering angle instruction given to the driver
ϵ_γ	Allowed error on articulation angle constraint
ϵ_θ	Allowed error on truck yaw angle constraint

γ_1	Articulation angle
γ_{end}	Desired final articulation angle
γ_{max}	Maximum absolute articulation angle
ω_{max}	Maximum absolute steering rate
ω_c	Crossover frequency
θ_0	Truck yaw angle
θ_1	Trailer yaw angle
θ_{end}	Desired final yaw angle of the trailer
$\hat{s}$	Estimated path parameter
$\mathbf{c}$	Vehicle configuration
$\mathbf{u}$	Input of the system
$\mathbf{x}$	State of the system
$\mathbf{y}$	Output of the system
$\mathbf{y}_{\text{ref}}$	Reference output
$\mathcal{E}(\mathbf{c})$	Region occupied by the vehicle
$\mathcal{O}$	Region occupied by obstacles
$\mathcal{Q}$	Set of open nodes during graph search
$\mathcal{X}$	Configuration space
$\mathcal{X}_{\text{free}}$	Free configuration space
$\mathcal{X}_{\text{obs}}$	Obstacle space
$\sigma(s)$	Piece-wise polynomial reference path structure
τ_r	Driver reaction time delay
$\mathbf{x}_{\text{d}}$	Driver state
$\mathbf{x}_{\text{init}}$	Most recent state update
$\vec{x}_O$	X-axis world frame
$\vec{x}_{\mathcal{B}_0}$	X-axis truck body frame
$\vec{x}_{\mathcal{B}_1}$	X-axis trailer body frame
$\vec{y}_O$	Y-axis world frame
$\vec{y}_{\mathcal{B}_0}$	Y-axis truck body frame
$\vec{y}_{\mathcal{B}_1}$	Y-axis trailer body frame
A	State-space matrix A
B	State-space matrix B
C	State-space matrix C
c_{ij}	Weight of edge e_{ij}
D	State-space matrix D
d_{min}	Minimum distance to obstacle
E	Set of edges
e	Edge connecting a pair of nodes
G	Graph
K_p	Driver static gain

l_d	Pure pursuit look-ahead distance
L_{0b}	Distance between drive axle and articulation point
L_{0f}	Distance between truck axles
L_{1f}	Distance between trailer axle and articulation point
N	Prediction horizon
R	Number of waypoints
R_P	Pure pursuit circular arc radius
s	Path parameter
s_{avg}	Average travel distance between prediction intervals
s_{end}	Length of final segment
T_L	Driver lead time constant
T_l	Driver lag time constant
T_N	Driver neuromuscular lag
T_s	Sample time
V	Set of nodes
v	Node in graph
v_0	Longitudinal velocity of truck
v_1	Longitudinal velocity of trailer
v_{max}	Maximum velocity of the truck
v_{min}	Minimum velocity of the truck
$v_{0,\text{ref}}$	Truck velocity reference value
v_{b}	Desired backward driving velocity of truck
v_{f}	Desired forward driving velocity of truck
$w_{\Delta\delta}$	Tuning weight for steering rate of change suppression
$w_{\hat{s}}$	Path parameter search window
$w_{\mathbf{y}}$	Tuning weight for output reference tracking
w_{v_0}	Tuning weight for truck velocity control input
x_0	X-coordinate truck drive axle
x_{d1}	Driver state 1
x_{d2}	Driver state 2
x_{0f}	X-coordinate truck steer axle
$x_{1,\text{ref}}$	Trailer axle x-coordinate reference value
x_{1f}	X-coordinate articulation point
x_{l_d}	X-coordinate on reference path at look-ahead distance
Y	Number of sample points in driven path
y_0	Y-coordinate truck drive axle
y_{0f}	Y-coordinate truck steer axle
$y_{1,\text{ref}}$	Trailer axle y-coordinate reference value
y_{1f}	Y-coordinate articulation point
y_{l_d}	Y-coordinate on reference path at look-ahead distance