A path planning and path-following control framework for a general 2-trailer with a car-like tractor

Oskar Ljungqvist, Niclas Evestedt, Daniel Axehill, Marcello Cirillo and Henrik Pettersson

The self-archived postprint version of this journal article is available at Linköping University Institutional Repository (DiVA):
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-161845

N.B.: When citing this work, cite the original publication.
Ljungqvist, O., Evestedt, N., Axehill, D., Cirillo, M., Pettersson, H., (2019), A path planning and path-following control framework for a general 2-trailer with a car-like tractor, *Journal of Field Robotics*. https://doi.org/10.1002/rob.21908

Original publication available at:
https://doi.org/10.1002/rob.21908

Copyright: Wiley (12 months)
http://eu.wiley.com/WileyCDA/

Tweet

LINKÖPING UNIVERSITY

A path planning and path-following control framework for a general 2-trailer with a car-like tractor

Oskar Ljungqvist[†*], Niclas Evestedt[‡], Daniel Axehill[†], Marcello Cirillo[◇] and Henrik Pettersson[◇]

[†]Department of Automatic Control, Linköping University, Linköping, Sweden

[‡]Embark Trucks Inc. San Francisco, USA

[◇]Autonomous Transport Solutions, Scania CV, Södertälje, Sweden

[*]E-mail: oskar.ljungqvist@liu.se

Abstract

Maneuvering a general 2-trailer with a car-like tractor in backward motion is a task that requires significant skill to master and is unarguably one of the most complicated tasks a truck driver has to perform. This paper presents a path planning and path-following control solution that can be used to automatically plan and execute difficult parking and obstacle avoidance maneuvers by combining backward and forward motion. A lattice-based path planning framework is developed in order to generate kinematically feasible and collision-free paths and a path-following controller is designed to stabilize the lateral and angular path-following error states during path execution. To estimate the vehicle state needed for control, a nonlinear observer is developed which only utilizes information from sensors that are mounted on the car-like tractor, making the system independent of additional trailer sensors. The proposed path planning and path-following control framework is implemented on a full-scale test vehicle and results from simulations and real-world experiments are presented.

1 Introduction

A massive interest for intelligent and fully autonomous transport solutions has been seen from industry over the past years as technology in this area has advanced. The predicted productivity gains and the relatively simple implementation have made controlled environments such as mines, harbors, airports, etc., interesting areas for commercial launch of such systems. In many of these applications, tractor-trailer systems are used for transportation and therefore require fully automated control. Reversing a semitrailer with a car-like tractor is known to be a task that require lots of training to perfect and an inexperienced driver usually encounter problems already when performing simple tasks, such as reversing straight backwards. To help the driver in such situations, trailer assist systems have been developed and released to the passenger car market [30, 69]. These systems enable the driver to easily control the semitrailer's curvature though a control knob. An even greater challenge arise when reversing a general 2-trailer (G2T) with a car-like tractor. As

seen in Figure 1, this system is composed of three interconnected vehicle segments: a front-wheel steered tractor, an off-axle hitched dolly and an on-axle hitched semitrailer. The word general refers to that the connection between the vehicle segments are of mixed hitching types [1].

Compared to a single semitrailer, the dolly introduces an additional degree of freedom into the system, making it very difficult to stabilize the semitrailer and the joint angles in backward motion. A daily challenge that many truck drivers encounter is to perform a reverse maneuver in, *e.g.*, a parking lot or a loading/off-loading site. In such scenarios, the vehicle is said to operate in an unstructured environment because no clear driving path is available. To perform a parking maneuver, the driver typically needs to plan the maneuver multiple steps ahead, which often involves a combination of driving forwards and backwards. For an inexperienced driver, these maneuvers can be both time-consuming and mentally exhausting. To aid the driver in such situations, this work presents a motion planning and path-following control framework for a G2T with a car-like tractor that is targeting unstructured environments. It is shown through several experiments that the framework can be used to automatically perform advanced maneuvers in different environments.

The framework can be used as a driver assist system to relieve the driver from performing complex tasks or as part of a motion planning and feedback control layer within an autonomous system architecture. The motion planner is based on the state-lattice motion planning framework [18, 19, 54] which has been tailored for this specific application in our previous work in [40]. The lattice planner efficiently computes kinematically feasible and collision-free motion plans by combining a finite number of precomputed motion segments. During online planning, challenging parking and obstacle avoidance maneuvers can be constructed by deploying efficient graph search algorithms [36]. To execute the motion plan, a path-following controller based on our previous work in [38] is used to stabilize the lateral and angular path-following error states during the execution of the planned maneuver. Finally, a nonlinear observer based on an extended Kalman filter (EKF) is proposed to obtain full state information of the system. Based upon request from our commercial partner and since multiple trailers are usually switched between during daily operation, the observer is developed so that it only uses information from sensors that are mounted on the tractor.

The proposed path planning and path-following control framework summarizes and extends our previous work in [38–40]. Here, the complete system is implemented on a full-scale test vehicle and results from both simulations and real-world experiments are presented to demonstrate its performance. To the best of the author's knowledge, this paper presents the first path planning and path-following control framework for a G2T with a car-like tractor that is implemented on a full-scale test vehicle.

The remainder of the paper is structured as follows. In Section 2, the responsibility of each module in the path planning and path-following control framework is briefly explained and an overview of related work is provided. In Section 3, the kinematic vehicle model of the G2T with a car-like tractor and the problem formulations are presented. The lattice-based path planner is presented in Section 4 and the hybrid path-following controller in Section 5. In Section 6, the nonlinear observer that is used for state estimation is presented. Implementation details are covered in Section 7 and simulation results as well as results from real-world experiments are presented in

Figure 1: The full-scale test vehicle that is used as a research platform. The car-like tractor is a modified version of a Scania R580 6x4 tractor.

Section 8. The paper is concluded in Section 9 by summarizing the contributions and discusses directions for future work.

2 Background and related work

The full system is built from several modules and a simplified system architecture is illustrated in Figure 2, where the integration and design of state estimation, path planning and path-following control are considered as the main contributions of this work. Below, the task of each module is briefly explained and for clarity, related work for each module is given individually.

2.1 Perception and localization

The objective of the perception and localization layer is to provide the planning and control layer with a consistent representation of the surrounding environment and an accurate estimation of where the tractor is located in the world. A detailed description of the perception layer is outside the scope of this paper, but a brief introduction is given for clarity.

Precomputed maps and onboard sensors on the car-like tractor (RADARs, LIDARs, a global positioning system (GPS), inertial measurement units (IMUs) and cameras) are used to construct an occupancy grid map [24] that gives a probabilistic representation of drivable and non-drivable areas. Dynamic objects are also detected and tracked but they are not considered in this work. Standard localization techniques are then used to obtain an accurate position and orientation estimate of the car-like tractor within the map [35, 47, 63]. Together, the occupancy grid map and the tractor's position and orientation provide the environmental representation in which motion planning and control is performed.

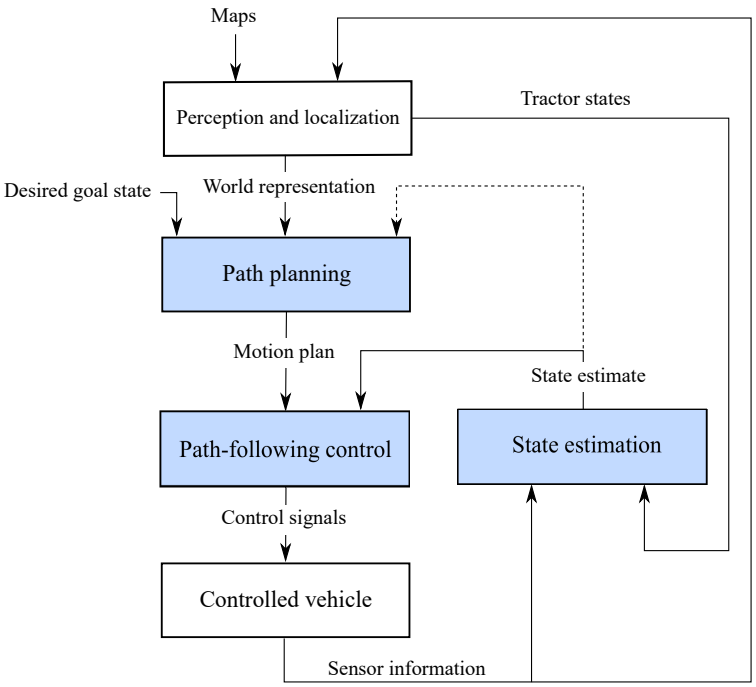


Figure 2: A schematic illustration of the proposed system architecture where the blue subsystems: path planning, path-following control and state estimation, are considered in this work.

2.2 State estimation

To control the G2T with car-like tractor, accurate and reliable state estimation of the semitrailer's position and orientation as well as the two joint angles of the system need to be obtained. An ideal approach would be to place sensors at each hitch connection to directly measure each joint angle [26,30,42] and equip the semitrailer with a similar localization system as the tractor (*e.g.*, IMU and a high precision GPS). However, commercial trailers are often exchanged between tractors and a high-performance navigation system is very expensive, making it an undesirable solution for general applications. Furthermore, no standardized communication protocol between different trailer and tractor manufacturers exists.

Different techniques for estimating the joint angle for a tractor with a semi-trailer and for a car with a trailer using wide-angle cameras are reported in [61] and [16], respectively. In [61], an image bank with images taken at different joint angles is first generated and during execution used to compare and match against the current camera image. Once a match is found, the corresponding joint angle is given from the matched image in the image bank. The work in [16] exploits symmetry of the trailer's drawbar in images to estimate the joint angle between a car and the trailer. In [28], markers with known locations are placed on the trailer's body and then tracked with a camera to estimate the joint angles of a G2T with car-like tractor. The proposed solution is tested on a small-scale vehicle in a lab environment.

Even though camera-based joint angle estimation would be possible to utilize in practice, it is unclear how it would perform in different lighting conditions, *e.g.*, during nighttime. The concept

for angle estimation used in this work was first implemented on a full-scale test vehicle as part of the master's thesis [50] supervised by the authors of this work. Instead of using a rear-view camera, a LIDAR sensor is mounted in the rear of the tractor. The LIDAR sensor is mounted such that the body of the semitrailer is visible in the generated point cloud for a wide range of joint angles. The semitrailer's body is assumed to be rectangular and by iteratively running the random sample consensus (RANSAC) algorithm [27], the visible edges of the semitrailer's body can be extracted from the point cloud. Virtual measurements of the orientation of the semitrailer and the lateral position of the midpoint of its front with respect to the tractor are then constructed utilizing known geometric properties of the vehicle. These virtual measurements together with information of the position and orientation of the tractor are used as observations to an EKF for state estimation.

In [9], the proposed iterative RANSAC algorithm is benchmarked against deep-learning techniques to compute the estimated joint angles directly from the LIDAR's point cloud or from camera images. That work concludes that for trailers with rectangular bodies, the LIDAR and iterative RANSAC solution outperforms the other tested methods in terms of accuracy and robustness which makes it a natural choice for state estimation in this work.

2.3 Path planning

Motion planning for car-like vehicles is a difficult problem due to the vehicle's nonholonomic constraints and the non-convex environment the vehicle is operating in [34]. Motion planning for tractor-trailer systems is even more challenging due to the vehicle's complex kinematics, its relatively large dimensional state-space and its structurally unstable joint angle kinematics in backward motion. The standard N-trailer (SNT) which only allows on-axle hitching, is differentially flat and can be converted into chained form when the position of the axle of the last trailer is used as the flat output [64]. This property of the SNT is explored in [49, 65] to develop efficient techniques for local trajectory generation. In [65], simulation results for the one and two trailer cases are presented but obstacles as well as state and input constraints are omitted. A well-known issue with flatness-based trajectory generation is that it is hard to incorporate constraints, as well as minimizing a general performance measure while computing the motion plan. Some of these issues are handled in [62] where a motion planner for unstructured environments with obstacles for the S2T is proposed. In that work, the motion planning problem is split into two phases where a holonomic path that violates the vehicle's nonholonomic constraints is first generated and then iteratively replaced with a kinematically feasible trajectory by converting the system into chained form. A similar hierarchical motion planning scheme is proposed in [33] for a unicycle-like tractor with a single trailer which is also experimentally demonstrated on a small scale platform.

An important contribution in this work is that most of the approaches presented above only consider the SNT-case with on-axle hitching, despite that most practical applications have both on-axle and off-axle hitching. The off-axle hitching makes the kinematics for the general N-trailer (GNT) more complicated [1]. To include the G2T with car-like tractor, we presented a probabilistic motion planner in [25]. Even though the proposed motion planner is capable of solving several hard problems, the framework lacks all completeness and optimality guarantees that are given by the

approach developed in this work.

The family of motion planning algorithms that belong to the lattice-based motion planning family, can guarantee resolution optimality and completeness [54]. In contrast to probabilistic methods, a lattice-based motion planner requires a regular discretization of the vehicle's state-space and is constrained to a precomputed set of feasible motions which, combined, can connect two discrete vehicle states. The precomputed motions are called motion primitives and can be generated offline by solving several optimal control problems (OCPs). This implies that the vehicle's nonholonomic constraints already have been considered offline and what remains during online planning is a search over the set of precomputed motions. Due to its deterministic nature and real-time capabilities, lattice-based motion planning has been used with great success on various robotic platforms [7, 19, 51, 54, 67] and is therefore the chosen motion planning strategy for this work.

Other deterministic motion planning algorithms rely on input-space discretization [12, 23] in contrast to state-space discretization. A model of the vehicle is used during online planning to simulate the system for certain time durations, using constant or parametrized control signals. In general, the constructed motions do not end up at specified final states. This implies that the search graph becomes irregular and results in an exponentially exploding frontier during online planning [54]. To resolve this, the state-space is often divided into cells where a cell is only allowed to be explored once. A motion planning algorithm that uses input-space discretization is the hybrid A* [23]. In [12], a similar motion planner is proposed to generate feasible paths for a car-like tractor with a semitrailer with active trailer steering. A drawback with motion planning algorithms that rely on input-space discretization, is that they lack completeness and optimality guarantees. Moreover, input-space discretization is in general not applicable for unstable systems, unless the online simulations are performed in closed-loop with a stabilizing feedback controller [25].

A problem with lattice-based approaches is the curse of dimensionality, *i.e.*, exponential complexity in the dimension of the state-space and in the number of precomputed motions. In [40], we circumvented this problem and developed a real-time capable lattice-based motion planner for a G2T with a car-like tractor. By discretizing the state-space of the vehicle such that the precomputed motions always move the vehicle from and to a circular equilibrium configuration, the dimension of the state lattice remained sufficiently low and made real-time use of classical graph search algorithms tractable. Even though the dimension of the discretized state-space is limited, the motion planner was shown to efficiently solve difficult and practically relevant motion planning problems.

In this work, the work in [40] is extended by better connecting the cost functional in the motion primitive generation and the cost function in the online motion planning problem. Additionally, the objective functional in backward motion is adjusted such that it reflects the difficulty of executing a maneuver. To avoid maneuvers in backward motion that in practice have a large risk of leading to a jack-knife state, a quadratic penalty on the two joint angles is included in the cost functional.

2.4 Path-following control

During the past decades, an extensive amount of feedback control techniques for different tractor-trailer systems for both forward and backward motion have been proposed. The different control tasks include path-following control (see *e.g.*, [4, 10, 13, 44, 59]), trajectory-tracking and set-point control (see *e.g.*, [22, 43, 46, 60]). Here, the focus will be on related path-following control solutions.

For the SNT, its flatness property can be used to design path-following controllers based on feedback linearization [59] or by converting the system into chained form [60]. A car-like tractor with a semitrailer is still differentially flat using a certain choice of flat outputs [56]. However, the flatness property does not hold when two consecutive trailers are off-axle hitched [45, 56]. In [13], this issue is circumvented by introducing a simplified reference vehicle which has equivalent stationary behavior but different transient behavior. Similar concepts have also been proposed in [48, 72]. Input-output linearization techniques are used in [2] to stabilize the GNT around paths with constant curvature, where the path-following controller minimizes the sum of the lateral offsets to the path. The proposed approach is however limited to forward motion since the introduced zero-dynamics become unstable in backward motion. A closely related approach is presented in [37], where the objective of the path-following controller is to minimize the swept path of a car-like tractor with a semitrailer along paths in backward and forward motion.

Tractor-trailer vehicles that have pure off-axle hitched trailers, are referred to as non-standard N-trailers (nSNT) [17, 45]. For these systems, scalable cascade-like path-following control techniques are presented in [42, 44]. Compared to many other path-following control approaches, these controllers do not need to find the closest distance to the nominal path and the complexity of the feedback controllers scales well with increasing number of trailers. By introducing artificial off-axle hitches, the proposed controller can also be used for the GNT-case [44]. However, as experimental results illustrate, the path-following controller becomes sensitive to measurement noise when an off-axle distance approaches zero.

A hybrid linear quadratic (LQ) controller is proposed in [4] to stabilize the G2T with car-like tractor around different equilibrium configurations corresponding to straight lines and circles, and a survey in the area of control techniques for tractor-trailer systems can be found in [20]. Inspired by [4], a cascade control approach for stabilizing the G2T with car-like tractor in backward motion around piecewise linear reference paths is proposed in [26]. An advantage of this approach is that the controller can track reference paths that are not necessarily kinematically feasible. However, if a more detailed reference path with full state information is available, this method is only using a subset of the available information and the control accuracy might be reduced. A similar approach for path tracking is also proposed in [55] for reversing a G2T with a car-like tractor which has been successfully demonstrated in practice.

Most of the path-following approaches presented above consider the problem of following a path defined in the position and orientation of the last trailer's axle. In this work, the nominal path obtained from the path planner is composed of full state information as well as nominal control signals. Furthermore, in a motion planning and path-following control architecture, it is crucial that

all nominal vehicle states are followed to avoid collision with surrounding obstacles. To utilize all information in the nominal path, we presented a state-feedback controller with feedforward action in [38]. The proposed path-following controller is proven to stabilize the path-following error kinematics for the G2T with a car-like tractor in backward motion around a set of admissible paths. The advantage of this approach is that the nominal path satisfies the vehicle kinematics making it, in theory, possible to follow exactly. However, the developed stability result in [38] fails to guarantee stability in continuous-time for motion plans that are combining forward and backward motion segments [39]. In [39], we proposed a solution to this problem and presented a framework that is exploiting the fact that a lattice planner is combining a finite number of precomputed motion segments. Based on this, a framework is proposed for analyzing the behavior of the path-following error, how to design the path-following controller and how to potentially impose restrictions on the lattice planner to guarantee that the path-following error is bounded and decays towards zero. Based on this, the same framework is used in this work, where results from real-world experiments on a full-scale test vehicle are also presented.

3 Kinematic vehicle model and problem formulations

The G2T with a car-like tractor considered in this work is schematically illustrated in Figure 3. This system has a positive off-axle connection between the car-like tractor and the dolly and an on-axle connection between the dolly and the semitrailer. The state vector $x = [x_3 \ y_3 \ \theta_3 \ \beta_3 \ \beta_2]^T \in \mathbb{R}^5$ is used to represent a configuration of the vehicle, where (x_3, y_3) is the position of the center of the semitrailer's axle, θ_3 is the orientation of the semitrailer, β_3 is the joint angle between the semitrailer and the dolly and β_2 is the joint angle between the dolly and the car-like tractor[1]. The length L_3 represent the distance between the axle of the semitrailer and the axle of the dolly, L_2 is the distance between the axle of the dolly and the off-axle hitching connection at the car-like tractor, $M_1 > 0$ is the length of the positive off-axle hitching, and L_1 denotes the wheelbase of the car-like tractor. The car-like tractor is front-wheeled steered and assumed to have perfect Ackerman geometry. The control signals to the system are the steering angle α and the longitudinal velocity v of the rear axle of the car-like tractor. A recursive formula derived from nonholonomic and holonomic constraints for the GNT vehicle is presented in [1]. Applying the formula for this

[1] All angles are defined positive counter clockwise.

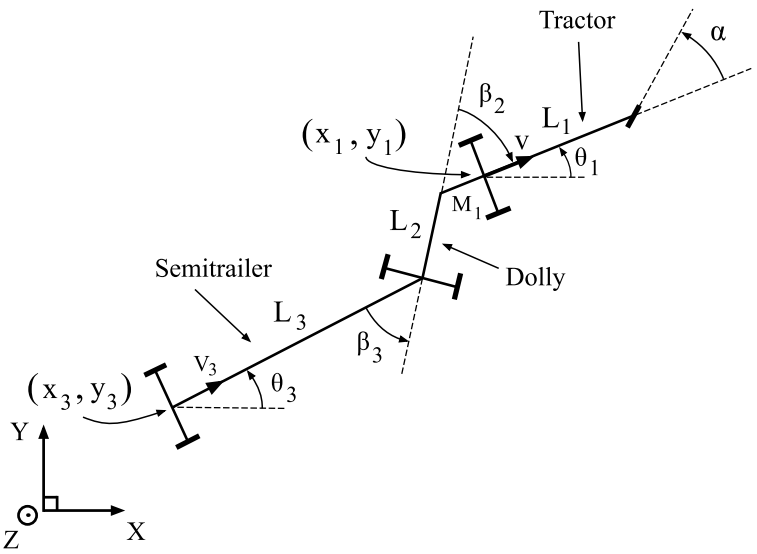


Figure 3: Definition of the geometric lengths, states and control signals that are of relevance for modeling the general 2-trailer with a car-like tractor.

specific G2T with a car-like tractor results in the following vehicle model [3]:

$$\dot{x}_3 = v\cos\beta_3 C_1(\beta_2, \tan\alpha/L_1)\cos\theta_3, \tag{1a}$$

$$\dot{y}_3 = v\cos\beta_3 C_1(\beta_2, \tan\alpha/L_1)\sin\theta_3, \tag{1b}$$

$$\dot{\theta}_3 = v\frac{\sin\beta_3}{L_3} C_1(\beta_2, \tan\alpha/L_1), \tag{1c}$$

$$\dot{\beta}_3 = v\left(\frac{1}{L_2}\left(\sin\beta_2 - \frac{M_1}{L_1}\cos\beta_2\tan\alpha\right) - \frac{\sin\beta_3}{L_3} C_1(\beta_2, \tan\alpha/L_1)\right), \tag{1d}$$

$$\dot{\beta}_2 = v\left(\frac{\tan\alpha}{L_1} - \frac{\sin\beta_2}{L_2} + \frac{M_1}{L_1 L_2}\cos\beta_2\tan\alpha\right), \tag{1e}$$

where $C_1(\beta_2, \kappa)$ is defined as

$$C_1(\beta_2, \kappa) = \cos\beta_2 + M_1\sin\beta_2\kappa. \tag{2}$$

By performing the input substitution $\kappa = \frac{\tan\alpha}{L_1}$, the model in (1) can be written on the form $\dot{x} = vf(x, \kappa)$. Define

$$g_v(\beta_2, \beta_3, \kappa) = \cos\beta_3 C_1(\beta_2, \kappa), \tag{3}$$

which describes the relationship, $v_3 = vg_v(\beta_2, \beta_3, \kappa)$, between the longitudinal velocity of the axle of the semitrailer, v_3 and the longitudinal velocity of the rear axle of the car-like tractor, v. When $g_v(\beta_2, \beta_3, \kappa) = 0$, the system in (1) is uncontrollable which practically implies that the position of the axle of the dolly or the semitrailer remain in stationarity even though the tractor moves. To avoid these vehicle configurations, it is assumed that $g_v(\beta_2, \beta_3, \kappa) > 0$, which implies that the joint angles has to satisfy $|\beta_3| < \pi/2$ and $|\beta_2| < \pi/2$, respectively, and that $C_1(\beta_2, \kappa) > 0$. These

imposed restrictions are closely related to the segment-platooning assumption defined in [42] and does not limit the practical usage of the model since structural damage could occur on the semi-trailer or the tractor, if these limits are exceeded.

The model in (1) is derived based on no-slip assumptions and the vehicle is assumed to operate on a flat surface. Since the intended operational speed is quite low for our use case, these assumptions are expected to hold. The direction of motion is essential for the stability of the system (1), where the joint angle kinematics are structurally unstable in backward motion ($v < 0$), where it risks to fold and enter what is called a jack-knife state [3]. In forward motion ($v > 0$), these modes are stable.

Since the longitudinal velocity v enters linearly into the model in (1), time-scaling [58] can be applied to eliminate the dependence on the longitudinal speed $|v|$. Define $s(t)$ as the distance traveled by the rear axle of the tractor, *i.e.*, $s(t) = \int_0^t |v(\tau)| \mathrm{d}\tau$. By substituting time with $s(t)$, the differential equation in (1) can be written as

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \operatorname{sign}(v(s)) f(x(s), \kappa(s)). \tag{4}$$

Since only the sign of v enters into the state equation, it implies that the traveled path is independent of the tractor's speed $|v|$ and the motion planning problem can be formulated as a path planning problem [34], where the speed is omitted. Therefore, the longitudinal velocity v is, without loss of generality, assumed to take on the values $v = 1$ for forward motion and $v = -1$ for backward motion, when path planning is considered.

In practice, the vehicle has limitations on the steering angle $|\alpha| \leq \alpha_{\max} < \pi/2$, the steering angle rate $|\omega| \leq \omega_{\max}$ and the steering angle acceleration $|u_\omega| \leq u_{\omega,\max}$. These constraints have to be considered in the path planning layer in order to generate feasible paths that the physical vehicle can execute.

3.1 Problem formulations

In this section, the path planning and the path-following control problems are defined. In order to make sure the planned path avoid uncontrollable regions and the nominal steering angle does not violate any of its physical constraints, an augmented state-vector $z = [x^T \ \alpha \ \omega]^T \in \mathbb{R}^7$ is used during path planning. The augmented model of the G2T with a car-like tractor (1) can be expressed in the following form

$$\frac{\mathrm{d}z}{\mathrm{d}s} = f_z(z(s), u_p(s)) = \begin{bmatrix} v(s) f(x(s), \tan\alpha(s)/L_1) \\ \omega(s) \\ u_\omega(s) \end{bmatrix}, \tag{5}$$

where its state-space $\mathbb{Z} \subset \mathbb{R}^7$ is defined as follows

$$\mathbb{Z} = \left\{ z \in \mathbb{R}^7 \mid |\beta_3| < \pi/2, \ |\beta_2| < \pi/2, \ |\alpha| \leq \alpha_{\max}, \ |\omega| \leq \omega_{\max}, \ C_1(\beta_2, \tan\alpha/L_1) > 0 \right\}, \tag{6}$$

where $C_1(\beta_2, \tan\alpha/L_1)$ is defined in (2). During path planning, the control signals are $u_p = [v \ u_\omega]^T \in \mathbb{U}_p$, where $\mathbb{U}_p = \{-1, 1\} \times [-u_{\omega,\max}, u_{\omega,\max}]$. Here, u_ω denotes the steering angle acceleration and the longitudinal velocity v is constrained to ± 1 and determines the direction of motion.

It is assumed that the perception layer provides the path planner with a representation of the surrounding obstacles $\mathbb{Z}_{\text{obs}}$. In the formulation of the path planning problem, it is assumed that $\mathbb{Z}_{\text{obs}}$ can be described analytically (*e.g.*, circles, ellipsoids, polytopes or other bounding regions [34]). Therefore, the free-space where the vehicle is not in collision with any obstacles can be defined as $\mathbb{Z}_{\text{free}} = \mathbb{Z} \setminus \mathbb{Z}_{\text{obs}}$.

Given an initial state $z_I = [x_I^T \;\; \alpha_I \;\; 0]^T \in \mathbb{Z}_{\text{free}}$ and a desired goal state $z_G = [x_G^T \;\; \alpha_G \;\; 0]^T \in \mathbb{Z}_{\text{free}}$, a feasible solution to the path planning problem is an arc-length parametrized control signal $u_p(s) \in \mathbb{U}_p$, $s \in [0, s_G]$ which results in a nominal path in $z(s)$, $s \in [0, s_G]$ that is feasible, collision-free and moves the vehicle from its initial state z_I to the desired goal state z_G. Among all feasible solutions to this problem, the optimal solution is the one that minimizes a specified cost functional J. The optimal path planning problem is defined as follows.

Definition 1 (The optimal path planning problem)**.** *Given the 5-tuple z_I, z_G, $\mathbb{Z}_{free}$, $\mathbb{U}_p$ and L, find the path length $s_G \in \mathbb{R}_+$ and an arc-length parametrized control signal $u_p(s) = [v(s) \;\; u_\omega(s)]^T$, $s \in [0, s_G]$ that minimizes the following OCP:*

$$\underset{u_p(\cdot),\, s_G}{\text{minimize}} \quad J = \int_0^{s_G} L(x(s), \alpha(s), \omega(s), u_\omega(s))\, \mathrm{d}s \tag{7a}$$

$$\text{subject to} \quad \frac{\mathrm{d}z}{\mathrm{d}s} = f_z(z(s), u_p(s)), \tag{7b}$$

$$z(0) = z_I, \quad z(s_G) = z_G, \tag{7c}$$

$$z(s) \in \mathbb{Z}_{free}, \quad u_p(s) \in \mathbb{U}_p, \tag{7d}$$

where $L : \mathbb{R}^5 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is the cost function.

The optimal path planning problem in (7) is a nonlinear OCP which is often, depending on the shape of $\mathbb{Z}_{\text{free}}$, highly non-convex. Thus, the OCP in (7) is in general hard to solve by directly invoking a numerical optimal control solver [11, 73] and sampling-based path planning algorithms are commonly employed to obtain an approximate solution [34, 52]. In this work, a lattice-based path planner [19, 54] is used and the framework is presented in Section 4.

For the path-following control design, a nominal path that the vehicle is expected to follow is defined as $(x_r(s), u_r(s)), s \in [0, s_G]$, where $x_r(s)$ is the nominal vehicle state and $u_r(s) = [v_r(s) \;\; \kappa_r(s)]^T$ is the nominal velocity and curvature control signals. The objective of the path-following controller is to locally stabilize the vehicle around this path in the presence of disturbances and model errors. When path-following control is considered, it is not crucial that the vehicle is located at a specific nominal state in time, rather that the nominal path is executed with a small and bounded path-following error $\tilde{x}(t) = x(t) - x_r(s(t))$. The path-following control problem is formally defined as follows.

Definition 2 (The path-following control problem)**.** *Given a controlled G2T with a car-like tractor* (1) *and a feasible nominal path $(x_r(s), u_r(s))$, $s \in [0, s_G]$. Find a control-law $\kappa(t) = g(s(t), x(t))$ with $v(t) = v_r(s(t))$, such that the solution to the closed-loop system $\dot{x}(t) = v_r(s(t)) f(x(t), g(s(t), x(t)))$ satisfies the following locally around the nominal path: For all $t \in \{t \in \mathbb{R}_+ \mid 0 \leq s(t) \leq s_G\}$, there exist positive constants r, ρ and ε such that*

1. $||\tilde{x}(t)|| \leq \rho||\tilde{x}(t_0)||e^{-\varepsilon(t-t_0)}, \quad \forall||\tilde{x}(t_0)|| < r,$

2. $\dot{s}(t) > 0.$

If the nominal path would be infinitely long ($s_G \to \infty$), Definition 2 coincides with the definition of local exponential stability of the path-following error model around the origin [31]. In this work, the path-following controller is designed by first deriving a path-following error model. This derivation as well as the design of the path-following controller are presented in Section 5.

3.2 System properties

Some relevant and important properties of the model in (1) that will be exploited for path planning are presented below.

3.2.1 Circular equilibrium configurations

Given a constant steering angle α_e there exists a circular equilibrium configuration where $\dot{\beta}_2$ and $\dot{\beta}_3$ are equal to zero, as illustrated in Figure 4. In stationarity, the vehicle will travel along circles with radii determined by α_e [4]. The equilibrium joint angles, β_{2e} and β_{3e}, are related to α_e through

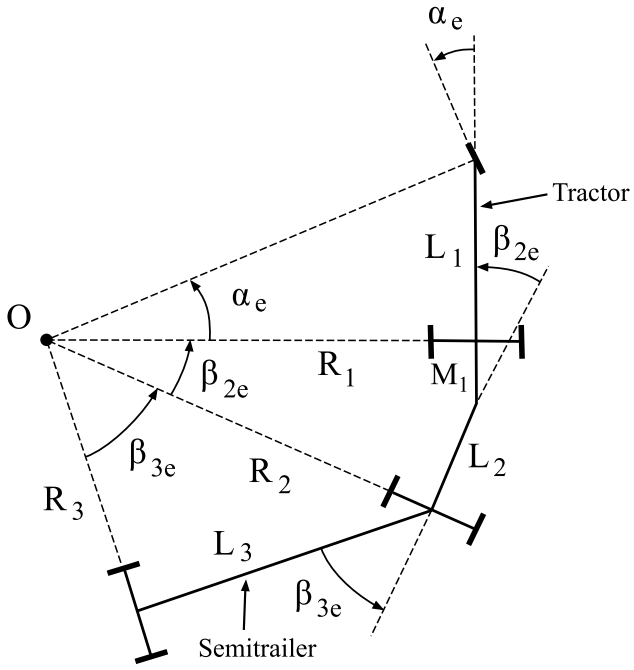


Figure 4: Illustration of a circular equilibrium configuration for the G2T with a car-like tractor. Given a constant steering angle α_e, there exists a unique pair of joint angles, $\beta_{2,e}$ and $\beta_{3,e}$, where $\dot{\beta}_2 = \dot{\beta}_3 = 0$.

the following equations

$$\beta_{3e} = \arctan\left(\frac{L_3}{R_3}\right), \tag{8a}$$

$$\beta_{2e} = \left(\arctan\left(\frac{M_1}{R_1}\right) + \arctan\left(\frac{L_2}{R_2}\right)\right), \tag{8b}$$

where the absolute values of the signed radii are $|R_1| = L_1/|\tan\alpha_e|$, $|R_2| = (R_1^2 + M_1^2 - L_2^2)^{1/2}$ and $|R_3| = (R_2^2 - L_3^2)^{1/2}$.

3.2.2 Symmetry

A feasible path $(z(s), u_p(s))$, $s \in [0, s_G]$ to (5) that moves the system from an initial state $z(0)$ to a final state $z(s_G)$, is possible to reverse in distance and revisit the exact same points in x and α by a simple transformation of the control signals. The result is formalized in Lemma 1 which is provided in Appendix A. Note that the actual state $x(\cdot)$ and steering angle $\alpha(\cdot)$ paths of the system (5) are fully distance-reversed and it is only the path of the steering angle rate $\omega(\cdot)$ that changes sign. Moreover, if $\omega(0)$ and $\omega(s_G)$ are equal to zero, the initial and final state constraints coincide. The practical interpretation of the result in Lemma 1 is that any path taken by the G2T with a car-like tractor (4) with $|\alpha(\cdot)| \leq \alpha_{\max}$ is feasible to follow in the reversed direction. Now, define the reverse optimal path planning problem to (7) as

$$\underset{\bar{u}_p(\cdot),\, \bar{s}_G}{\text{minimize}} \quad \bar{J} = \int_0^{\bar{s}_G} L(\bar{x}(\bar{s}), \bar{\alpha}(\bar{s}), \bar{\omega}(\bar{s}), \bar{u}_\omega(\bar{s}))\, \mathrm{d}\bar{s} \tag{9a}$$

$$\text{subject to} \quad \frac{\mathrm{d}\bar{z}}{\mathrm{d}\bar{s}} = f_z(\bar{z}(\bar{s}), \bar{u}_p(\bar{s})), \tag{9b}$$

$$\bar{z}(0) = z_G, \quad \bar{z}(\bar{s}_G) = z_I, \tag{9c}$$

$$\bar{z}(\bar{s}) \in \mathbb{Z}_{\text{free}}, \quad \bar{u}_p(\bar{s}) \in \mathbb{U}_p. \tag{9d}$$

Note that the only difference between the OCPs defined in (7) and (9), respectively, is that the initial and goal state constraints are switched. In other words, (7) defines a path planning problem from z_I to z_G and (9) defines a path planning problem from z_G to z_I. It is possible to show that also the optimal solutions to these OCPs are related through the result established in Lemma 1.

Assumption 1. *For all $z \in \mathbb{Z}_{free}$ and $u_p \in \mathbb{U}_p$, the cost function L in* (7) *satisfies $L(x, \alpha, \omega, u_\omega) = L(x, \alpha, -\omega, u_\omega)$.*

Assumption 2. $z = [x^T \;\; \alpha \;\; \omega]^T \in \mathbb{Z}_{free} \Leftrightarrow \bar{z} = [x^T \;\; \alpha \;\; -\omega]^T \in \mathbb{Z}_{free}$.

Theorem 1. *Under Assumption 1–2, if $(z^*(s), u_p^*(s))$, $s \in [0, s_G^*]$ is an optimal solution to the optimal path planning problem* (7) *with optimal objective functional value J^*, then the distance-reversed path $(\bar{z}^*(\bar{s}), \bar{u}_p^*(\bar{s}))$, $\bar{s} \in [0, \bar{s}_G^*]$ given by* (57)–(58) *with $\bar{s}_G^* = s_G^*$, is an optimal solution to the reverse optimal path planning problem* (9) *with optimal objective functional value $\bar{J}^* = J^*$.*

Proof. See Appendix A. □

Theorem 1 shows that if an optimal solution to the optimal path planning problem in (7) or the reversed optimal path planning problem in (9) is known, an optimal solution to the other one can immediately be derived using the invertible transformation defined in (57)–(58) and $\bar{s}_G = s_G$.

4 Lattice-based path planner

As previously mentioned, the path planning problem defined in (1) is hard to solve by directly invoking a numerical optimal control solver. Instead, it can be combined with classical search algorithms and a discretization of the state-space to build efficient algorithms to solve the path planning problem. By discretizing the state-space $\mathbb{Z}_d$ of the vehicle in a regular fashion and constraining the motion of the vehicle to a lattice graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, which is a directed graph embedded in an Euclidean space that forms a regular and repeated pattern, classical graph-search techniques can be used to traverse the graph and compute a path to the goal [19, 54]. Each vertex $v[k] \in \mathcal{V}$ represents a discrete augmented vehicle state $z[k] \in \mathbb{Z}_d$ and each edge $e_i \in \mathcal{E}$ represents a motion primitive m_i, which encodes a feasible path $(z^i(s), u_p^i(s))$, $s \in [0, s_f^i]$ that moves the vehicle from one discrete state $z[k] \in \mathbb{Z}_d$ to a neighboring state $z[k+1] \in \mathbb{Z}_d$, while respecting the vehicle model and its physically imposed constraints. For the remainder of this text, state and vertex will be used interchangeably.

Each motion primitive m_i is computed offline and stored in a library containing a set $\mathcal{P}$ of precomputed feasible motion segments that can be used to connect two vertices in the graph. In this work, an OCP solver is used to generate the motion primitives and the vehicle's nonholonomic constraints are in this way handled offline, and what remains during online planning is a search over the set of precomputed motions. Performing a search over a set of precomputed motion primitives is a well known technique and is known as lattice-based path planning [19, 54].

Let $z[k+1] = f_p(z[k], m_i)$ represent the state transition when m_i is applied from $z[k]$, and let $J_p(m_i)$ denote the stage-cost associated with this transition. The complete set of motion primitives $\mathcal{P}$ is computed offline by solving a finite set of OCPs to connect a set of initial states with a set of neighboring states in an obstacle-free environment. The set $\mathcal{P}$ is constructed from the position of the semitrailer at the origin and since the G2T with a car-like tractor (1) is position-invariant, a motion primitive $m_i \in \mathcal{P}$ can be translated and reused from all other positions on the grid. The cardinality of the complete set of motion primitives is $|\mathcal{P}| = M$, where M is a positive integer-valued scalar. In general, all motion primitives in $\mathcal{P}$ cannot be used from each state $z[k]$ and the set of motion primitives that can be used from $z[k]$ is denoted $\mathcal{P}(z[k]) \subseteq \mathcal{P}$. The cardinality of $\mathcal{P}(z[k])$ defines the number of motion primitives that can be used from a given state $z[k]$ and the average $|\mathcal{P}(z[k])|$ defines the branching factor of the search problem. Therefore, a trade off between planning time and maneuver resolution has to be made when designing the motion primitive set. Having a large library of diverse motions gives the lattice planner more flexibility, however, the planning time will increase exponentially with the size of $|\mathcal{P}(z[k])|$. As the branching factor increases, a well-informed heuristic function becomes more and more important in order to maintain real-time performance during online planning [19, 32]. The heuristic function estimates the true

cost-to-go from a state $z[k] \in \mathbb{Z}_d$ to the goal state z_G, and is used as guidance for the online graph search to expand the most promising vertices [19, 32, 34]. It is desired that the heuristic function is admissible to maintain optimality guarantees, and close to the true cost-to-go for efficient online planning. For nonholonomic systems, the Euclidean distance to the goal is known to severely underestimate the true cost-to-go in many situations and precomputed free-space heuristic look-up tables (HLUTs) are often used to improve the online planning time [19, 32].

The nominal path taken by the vehicle when motion primitive $m_i \in \mathcal{P}$ is applied from $z[k]$, is declared collision-free if it does not collide with any obstacles $c(m_i, z[k]) \in \mathbb{Z}_{\text{free}}$, otherwise it is declared as in collision $c(m_i, z[k]) \notin \mathbb{Z}_{\text{free}}$. Define $u_q : \mathbb{Z}_+ \to \{1, \ldots, M\}$ as a discrete and integer-valued decision variable that is selected by the lattice planner, where $u_q[k]$ specifies which motion primitive that is applied a stage k. By specifying the set of allowed states $\mathbb{Z}_d$ and precomputing the set of motion primitives $\mathcal{P}$, the continuous-time optimal path planning problem (7) is approximated by the following discrete-time OCP:

$$
\begin{aligned}
\underset{\{u_q[k]\}_{k=0}^{N-1},\, N}{\text{minimize}} \quad & J_{\text{D}} = \sum_{k=0}^{N-1} J_p(m_{u_q[k]}) \\
\text{subject to} \quad & z[0] = z_I, \quad z[N] = z_G, \\
& z[k+1] = f_p(z[k], m_{u_q[k]}), \\
& m_{u_q[k]} \in \mathcal{P}(z[k]), \\
& c(m_{u_q[k]}, z[k]) \in \mathbb{Z}_{\text{free}}.
\end{aligned} \tag{10}
$$

The decision variables to this problem are the integer-valued sequence $\{u_q[k]\}_{k=0}^{N-1}$ and its length N. A feasible solution is an ordered sequence of collision-free motion primitives $\{m_{u_q[k]}\}_{k=0}^{N-1}$, *i.e.*, a nominal path $(z(s), u_p(s))$, $s \in [0, s_G]$, that connect the initial state $z(0) = z_I$ and the goal state $z(s_G) = z_G$. Given the set of all feasible solutions to (10), the optimal solution is the one that minimizes the cost function J_{D}.

During online planning, the discrete-time OCP in (10) is solved using the anytime repairing A* (ARA*) search algorithm [36]. ARA* is based on standard A* but initially performs a greedy search with the heuristic function inflated by a factor $\gamma \geq 1$. This provides a guarantee that the found solution has a cost J_D that satisfies $J_D \leq \gamma J_D^*$, where J_D^* denotes the optimal cost to (10). When a solution with a guaranteed bound of γ-suboptimality has been found, γ is gradually decreased until an optimal solution with $\gamma = 1$ is found or if a maximum allowed planning time is reached. With this search algorithm, both real-time performance and suboptimality bounds for the produced solution can be guaranteed.

In (10), it is assumed that $z_I \in \mathbb{Z}_d$ and $z_G \in \mathbb{Z}_d$ to make the problem well defined. If $z_I \notin \mathbb{Z}_d$ or $z_G \notin \mathbb{Z}_d$, they have to be projected to their closest neighboring state in $\mathbb{Z}_d$ using some distance metric. Thus, the discretization of the vehicle's state-space restricts the set of possible initial states the lattice planner can plan from and desired goal states that can be reached. Even though not considered in this work, these restrictions could be alleviated by the use of numerical optimal control [68] as a post-processing step [8, 34, 51].

Workflow 1 The lattice-based path planning framework for the G2T with a car-like tractor

Step 1 – State lattice construction:

a) ***State-space discretization:*** Specify the resolution of the discretized state-space $\mathbb{Z}_d$.

b) ***Motion primitive selection:*** Specify the connectivity in the state lattice by selecting pairs of discrete states $\{z_s^i, z_f^i\}$, $i = 1, \ldots, M$, to connect.

c) ***Motion primitive generation:*** Design the cost functional J_p and compute the set of motion primitives $\mathcal{P}$ that moves the vehicle between $\{z_s^i, z_f^i\}$, $i = 1, \ldots, M$.

Step 2 – Efficiency improvements:

a) ***Motion primitive reduction:*** Systematically remove redundant motion primitives from $\mathcal{P}$ to reduce the branching factor of the search problem and therefore enhance the online planning time.

b) ***Heuristic function:*** Precompute a HLUT by calculating the optimal cost-to-go in an obstacle-free environment.

Step 3 – Online path planning:

a) ***Initialization:*** Project the vehicle's initial state z_I and desired goal state z_G to $\mathbb{Z}_d$.

b) ***Graph search:*** Solve the discrete-time OCP in (10) using ARA*.

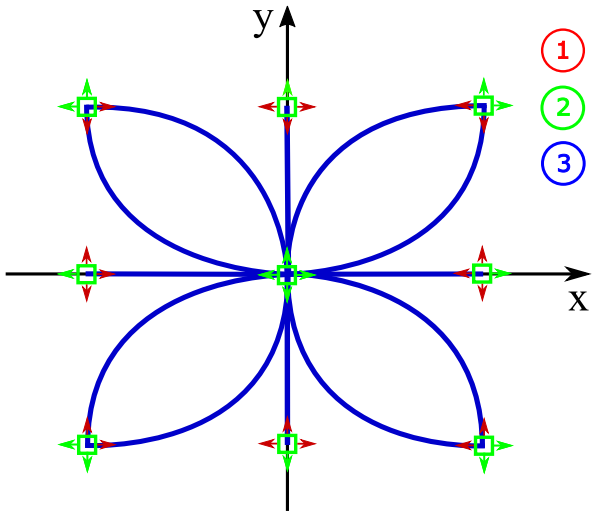
c) ***Return:*** Send the computed solution to the path-following controller or report failure.

The main steps of the path planning framework used in this work are summarized in Workflow 1 (see also Figure 5) and each step is now explained more thoroughly.

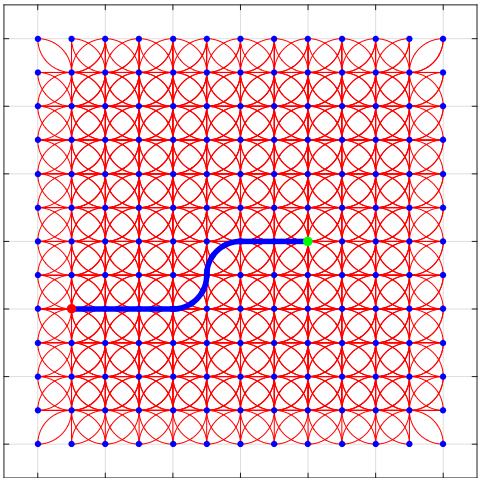
4.1 State lattice construction

The offline construction of the state lattice can be divided into three steps, as illustrated in Figure 5a. First, the state-space of the vehicle is discretized with a certain resolution. Second, the connectivity in the state lattice is decided by specifying a finite amount of pairs of discrete vehicle states $\{z_s^i, z_f^i\}$, $i = 1, \ldots, M$, to connect. Third, the motion primitives connecting each of these pairs of vehicle states are generated by the use of numerical optimal control [68]. Together, these three steps define the resolution and the size of the lattice graph $\mathcal{G}$ and needs to be chosen carefully to maintain a reasonable search time during online planning, while at the same time allowing the vehicle to be flexible enough to maneuver in confined spaces.

To obtain a tractable search space for the online graph search, the vehicle's augmented vector $z[k] = \begin{bmatrix} x[k]^T & \alpha[k] & \omega[k] \end{bmatrix}^T$ is discretized into circular equilibrium configurations (8) at each state in the state lattice. This implies that the joint angles, $\beta_2[k]$ and $\beta_3[k]$, are implicitly discretized since they are uniquely determined by the equilibrium steering angle $\alpha[k]$ through the relationships in (8). However, in between two discrete states in the state lattice, the system is not restricted to circular equilibrium configurations. The steering angle rate $\omega[k]$ is constrained to zero at each ver-



(a) Illustration of the three steps that are performed offline to generate a state lattice. The blue paths illustrate the resulting motion primitives $m_i \in \mathcal{P}$ between the states.



(b) The resulting state lattice and an example of a solution to a path planning problem (blue path) from the red dot to the green dot.

Figure 5: In (a), an illustration of the three steps that are performed to generate the state lattice. (1) Discretize the state-space, (2) select which pair of states to connect, (3) compute optimal paths (motion primitives) between each pair of states. In (b), the resulting state lattice together with a solution (blue path) to a graph-search problem.

tex in the state lattice to make sure that the steering angle is continuously differentiable, even when multiple motion primitives are combined during online planning. The position of the axle of the semitrailer $(x_3[k], y_3[k])$ is discretized to a uniform grid with resolution $r = 1$ m and the orientation of the semitrailer $\theta_3[k]$ is discretized irregularly[2] into $|\Theta| = 16$ different orientations [54]. This discretization of $\theta_3[k]$ is used to make it possible to construct short straight paths, compatible with the chosen discretization of the position from every orientation $\theta_3[k] \in \Theta$. Finally, the equilibrium steering angle $\alpha_e[k]$ is discretized into $|\Phi| = 3$ different angles, where $\Phi = \{-0.1, 0, 0.1\}$. With the proposed state-space discretization, the actual dimension of the discretized state-space $\mathbb{Z}_d$ is four. Of course, the proposed discretization imposes restriction to the path planner, but is motivated to enable fast and deterministic online planning.

4.2 Motion primitive generation

The motion primitive set $\mathcal{P}$ is precomputed offline by solving a finite set of OCPs that connect a set of initial states $z_s^i \in \mathbb{Z}_d$ to a set of neighboring states $z_f^i \in \mathbb{Z}_d$ in a bounded neighborhood in an obstacle-free environment.

Unlike our previous work in [40], the objective functional used during motion primitive generation coincides with the online planning stage-cost $J_p(m_i)$. This enables the resulting motion plan to be as close as possible to the optimal one and desirable behaviors can be favored in a systematic

[2] Θ is the the set of unique angles $-\pi < \theta_3 \leq \pi$ that can be generated by $\theta_3 = \arctan 2(i, j)$ for two integers $i, j \in \{-2, -1, 0, 1, 2\}$.

way. To promote and generate less complex paths that are easier for a path-following controller to execute, the cost function L in (7) is chosen as

$$L(z, u_\omega) = 1 + \left\| \begin{bmatrix} \beta_3 & \beta_2 \end{bmatrix}^T \right\|_{\mathbf{Q_1}}^2 + \left\| \begin{bmatrix} \alpha & \omega & u_\omega \end{bmatrix}^T \right\|_{\mathbf{Q_2}}^2, \tag{11}$$

where the matrices $\mathbf{Q}_1 \succeq 0$ and $\mathbf{Q}_2 \succeq 0$ are design parameters that are used to trade off between simplicity of executing the maneuver and the path distance s_f. By tuning the weight matrix $\mathbf{Q}_1$, maneuvers in backward motion with large joint angles, β_2 and β_3, that have a higher risk to enter a jack knife state, can be penalized and therefore avoided during online planning if less complex motion primitives exist. In forward motion, the modes corresponding to the two joint angles β_2 and β_3 are stable and therefore not penalized.

To guarantee that the motion primitives in $\mathcal{P}$ move the vehicle between two discrete states in the state lattice, they are constructed by selecting initial states $z_s^i \in \mathbb{Z}_d$ and final states $z_f^i \in \mathbb{Z}_d$ that lie on the grid. A motion primitive in forward motion from the initial state $z_s^i = [x_s^i \;\; \alpha_s^i \;\; 0]^T$ to the final state $z_f^i = [x_f^i \;\; \alpha_f^i \;\; 0]^T$ is computed by solving the following OCP:

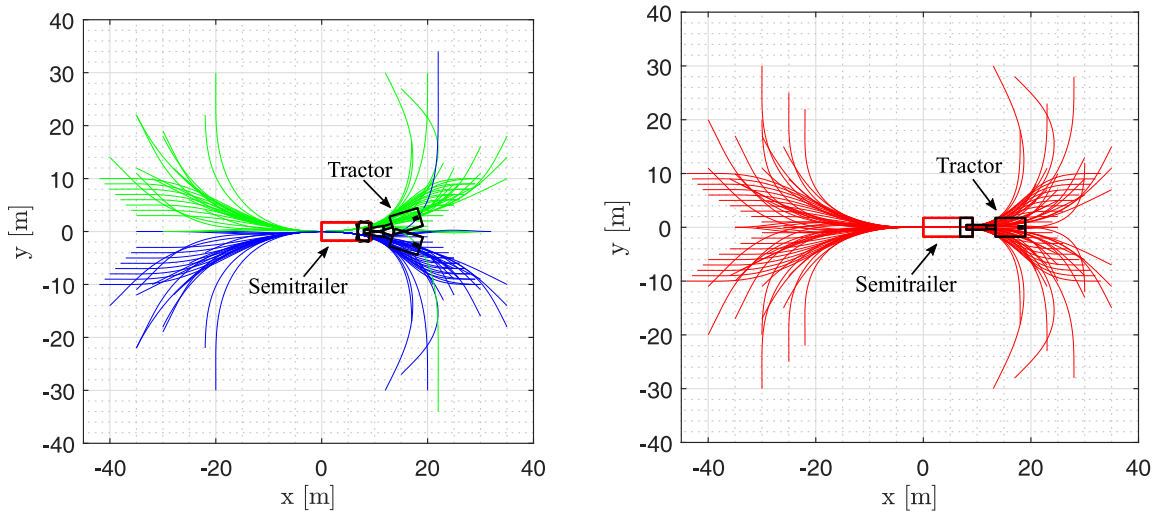
$$\begin{aligned}
\underset{u_\omega^i(\cdot),\, s_f^i}{\text{minimize}} \quad & J_p(m_i) = \int_0^{s_f^i} L(z^i(s), u_\omega^i(s))\, \mathrm{d}s \\
\text{subject to} \quad & \frac{\mathrm{d}z^i}{\mathrm{d}s} = \begin{bmatrix} f(x^i(s), \tan\alpha^i(s)/L_1) \\ \omega^i(s) \\ u_\omega^i(s) \end{bmatrix}, \\
& z^i(0) = z_s^i, \quad z^i(s_f) = z_f^i, \\
& z^i(s) \in \mathbb{Z}, \quad |u_\omega^i(s)| \leq u_{\omega,\max}.
\end{aligned} \tag{12}$$

Note the similarity of OCP in (12) with the optimal path planning problem (7). Here, the obstacle imposed constraints are neglected and the vehicle is constrained to only move forwards. The established results in Lemma 1 and Theorem 1 are exploited to generate the motion primitives for backward motion. Here, each OCP is solved from the final state z_f^i to the initial state z_s^i in forward motion and the symmetry result in Lemma 1 is applied to recover the backward motion segment. Theorem 1 guarantees that the optimal solution $(z^i(s), u_p^i(s))$, $s \in [0, s_f^i]$ and the optimal objective functional value $J_p(m_i)$ remain unaffected. This technique is used to avoid the structurally unstable joint-angle kinematics in backward motion that can cause numerical problems for the OCP solver.

In this work, the OCP in (12) is solved by deploying the state-of-the-art numerical optimal control solver CasADi [6], combined with the primal-dual interior-point solver IPOPT [68]. Each generated motion primitive is represented as a distance sampled path in all vehicle states and control signals. Finally, since the system is orientation-invariant, rotational symmetries of the system are exploited[3] to reduce the number of OCPs that need to be solved during the motion primitive generation [19, 54].

Even though the motion primitive generation is performed offline, it is not feasible to make an exhaustive generation of motion primitives to all grid points due to computation time and the

[3]Essentially, it is only necessary to solve the OCPs from the initial orientations $\theta_{3,s} = 0$, $\arctan(1/2)$ and $\pi/4$. The motion primitives from the remaining initial orientations $\theta_{3,s} \in \Theta$ can be generated by mirroring the solutions.



(a) The set of motion primitives from $(\theta_{3,s}, \alpha_s) = (0, 0.1)$ (green) and $(\theta_{3,s}, \alpha_s) = (0, -0.1)$ (blue) to different final states $z_f \in \mathbb{Z}_d$.

(b) The set of motion primitives from $(\theta_{3,s}, \alpha_s) = (0, 0)$ to different final states $z_f \in \mathbb{Z}_d$.

Figure 6: The set of motion primitives from initial position of the semitrailer at the origin with orientation $\theta_{3,s} = 0$ for different initial equilibrium configurations to different final states $z_f \in \mathbb{Z}_d$. The colored paths are the paths taken by the center of the axle of the semitrailer (x_3, y_3) during the different motions.

high risk of creating redundant and undesirable segments. Instead, for each initial state $x_s^i \in \mathbb{Z}_d$ with position of the semitrailer at the origin, a careful selection of final states $x_f^i \in \mathbb{Z}_d$ is performed based on system knowledge and by visual inspection. The OCP solver is then only generating motion primitives from this specified set of OCPs. For our full-scale test vehicle, the set of motion primitives from all initial states with $\theta_{3,s} = 0$, is illustrated in Figure 6. The following can be noted regarding the manual specification of the motion primitive set:

a) A motion primitive $m_i \in \mathcal{P}$ is either a straight motion, a heading change maneuver or a parallel maneuver.

b) The motion primitives in forward motion are more aggressive compared to the ones in backward motion, *i.e.*, a maneuver in forward motion has a shorter path distance compared to a similar maneuver in backward motion.

c) The final position $(x_{3,f}^i, y_{3,f}^i)$ of motion primitive m_i is selected such that the ratio between the stage-cost $J_p(m_i)$ and the path distance s_f^i is sufficiently small, *i.e.*, such that the nominal path in all vehicle states and controls are sufficiently smooth to be executed by a path-following controller.

d) While starting in a nonzero equilibrium configuration, the final position of the semitrailer $(x_{3,f}^i, y_{3,f}^i)$ is mainly restricted to the first and second quadrants for $\alpha_s^i = 0.1$ and to the third and fourth quadrants for $\alpha_s^i = -0.1$.

4.3 Efficiency improvements and online path planning

To improve the online planning time, the set of motion primitives $\mathcal{P}$ is reduced using the reduction technique presented in [19]. A motion primitive $m_i \in \mathcal{P}$ with stage-cost $J_p(m_i)$ is removed if its state transition $z[k+1] = f_p(z[k], m_i)$ in free-space can be obtained by a combination of the other motion primitives in $\mathcal{P}$ with a combined total stage-cost J_{comb} that satisfies $J_{\text{comb}} \leq \eta J_p(m_i)$, where $\eta \geq 1$ is a design parameter. This procedure can be used to reduce the size of the motion primitive set by choosing $\eta > 1$, or by selecting $\eta = 1$ to verify that redundant motion primitives do not exist in $\mathcal{P}$.

As previously mentioned, a heuristic function is used to guide the online search in the state lattice. The goal of the heuristic function is to perfectly estimate the cost-to-go at each vertex in the graph. In this work, we rely on a combination of two admissible heuristic functions: Euclidean distance and a free-space HLUT [32]. The HLUT is generated offline using the techniques presented in [32]. It is computed by solving several obstacle free path planning problems from all initial states $z_I \in \mathbb{Z}_d$ with position of the semitrailer at the origin, to all final states $z_G \in \mathbb{Z}_d$ with a specified maximum cut-off cost J_{cut}. As explained in [32], this computation step can be done efficiently by running a Dijkstra's algorithm from each initial state. During each Dijkstra's search, the optimal cost-to-come from explored vertices are simply recorded and stored in the HLUT. Moreover, in analogy to the motion primitive generation, the size of the HLUT is kept small by exploiting the position and orientation invariance properties of $\mathcal{P}$ [19, 32]. The final heuristic function value used during the online graph search is the maximum of these two heuristics. As shown in [32], a HLUT significantly reduces the online planning time, since it takes the vehicle's nonholonomic constraints into account and enables perfect estimation of cost-to-go in free-space scenarios with no obstacles.

5 Path-following controller

The motion plan received from the lattice planner is a nominal path $(x_r(s), u_r(s))$, $s \in [0, s_G]$ satisfying the time-scaled model of the G2T with a car-like tractor (4):

$$\frac{\mathrm{d}x_r}{\mathrm{d}s} = v_r(s) f(x_r(s), \kappa_r(s)), \quad s \in [0, s_G], \tag{13}$$

where $x_r(s)$ is the nominal vehicle states for a specific s and $u_r(s) = [v_r(s) \;\; \kappa_r(s)]^T$ is the nominal velocity and curvature control signals. The nominal path satisfies the system kinematics, its physically imposed constraints and moves the vehicle in free-space from the vehicle's initial state $x_r(0) = x_I$ to a desired goal state $x_r(s_G) = x_G$. Here, the nominal path is parametrized in s, which is the distance traveled by the rear axle of the car-like tractor. When backward motion tasks are considered and the axle of the semitrailer is to be controlled, it is more convenient to parameterize the nominal path in terms of distance traveled by the axle of the semitrailer $\tilde{s}$. Using the ratio $g_v > 0$ defined in (3), these different path parameterizations are related as $\tilde{s}(s) = \int_0^s g_v(\beta_{2,r}(\tau), \beta_{3,r}(\tau), \kappa_r(\tau))\mathrm{d}\tau$ and the nominal path (13) can equivalently be represented

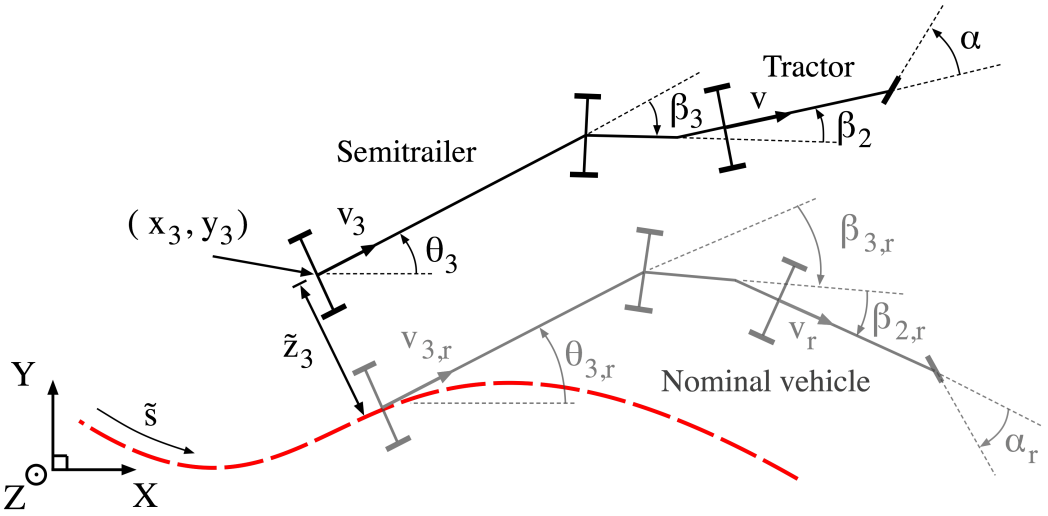


Figure 7: An illustrative description of the Frenet frame with its moving coordinate system located at the orthogonal projection of the center of the axle of the semitrailer onto the reference path (dashed red curve) in the nominal position of the axle of the semitrailer $(x_{3,0}(\tilde{s}), y_{3,0}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$. The black tractor-trailer system is the controlled vehicle and the gray tractor-trailer system is the nominal vehicle, or the desired vehicle configuration at this specific value of $\tilde{s}(t)$.

as

$$\frac{\mathrm{d}x_r}{\mathrm{d}\tilde{s}} = \frac{v_r(\tilde{s})}{g_v(\beta_{2,r}(\tilde{s}), \beta_{3,r}(\tilde{s}), \kappa_r(\tilde{s}))} f(x_r(\tilde{s}), \kappa_r(\tilde{s})), \quad \tilde{s} \in [0, \tilde{s}_G], \tag{14}$$

where $\tilde{s}_G$ denotes the total distance of the nominal path taken by the axle of the semitrailer. According to the problem definition in Definition 2, the objective of the path-following controller is to stabilize the G2T with a car-like tractor (1) around this nominal path. It is done by first describing the controlled vehicle (1) in terms of deviation from the nominal path generated by the system in (14), as depicted in Figure 7. During path execution, $\tilde{s}(t)$ is defined as the orthogonal projection of center of the axle of the semitrailer $(x_3(t), y_3(t))$ onto its nominal path $(x_{3,r}(\tilde{s}), y_{3,r}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$ at time t:

$$\tilde{s}(t) = \underset{\tilde{s} \in [0, \tilde{s}_G]}{\arg\min} \left|\left| \begin{bmatrix} x_3(t) - x_{3,r}(\tilde{s}) \\ y_3(t) - y_{3,r}(\tilde{s}) \end{bmatrix} \right|\right|_2. \tag{15}$$

Using standard geometry, the curvature $\kappa_{3,r}(\tilde{s})$ of the nominal path taken by the axle of the semitrailer is given by

$$\kappa_{3,r}(\tilde{s}) = \frac{\mathrm{d}\theta_{3,r}}{\mathrm{d}\tilde{s}} = \frac{\tan \beta_{3,r}(\tilde{s})}{L_3}, \quad \tilde{s} \in [0, \tilde{s}_G]. \tag{16}$$

Define $\tilde{z}_3(t)$ as the signed lateral distance between the center of the axle of the semitrailer $(x_3(t), y_3(t))$ and its projection to the nominal path in $(x_{3,r}(\tilde{s}), y_{3,r}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$ at time t. Introduce the controlled curvature deviation as $\tilde{\kappa}(t) = \kappa(t) - \kappa_r(\tilde{s}(t))$, define the orientation error of the semitrailer as $\tilde{\theta}_3(t) = \theta_3(t) - \theta_{3,r}(\tilde{s}(t))$ and define the joint angular errors as $\tilde{\beta}_3(t) = \beta_3(t) - \beta_{3,r}(\tilde{s}(t))$

and $\tilde{\beta}_2(t) = \beta_2(t) - \beta_{2,r}(\tilde{s}(t))$, respectively. Define $\Pi(a,b) = \{t \in \mathbb{R}_+ \mid a \leq \tilde{s}(t) \leq b\}$ as the time-interval when the distance traveled along the nominal path is between $a \in \mathbb{R}_+$ and $b \in \mathbb{R}_+$, where $0 \leq a \leq b \leq \tilde{s}_G$. Then, using the Frenet-Serret formula, the distance traveled $\tilde{s}(t)$ along the nominal path and the signed lateral distance $\tilde{z}_3(t)$ to the nominal path can be modeled as:

$$\dot{\tilde{s}} = v_3 \frac{v_r \cos\tilde{\theta}_3}{1 - \kappa_{3,r}\tilde{z}_3}, \quad t \in \Pi(0, \tilde{s}_G), \tag{17a}$$

$$\dot{\tilde{z}}_3 = v_3 \sin\tilde{\theta}_3, \qquad t \in \Pi(0, \tilde{s}_G), \tag{17b}$$

where $v_3 = v g_v(\tilde{\beta}_2 + \beta_{2,r}, \tilde{\beta}_3 + \beta_{3,r}, \tilde{\kappa} + \kappa_r)$ and the dependencies of $\tilde{s}$ and t are omitted for brevity. This transformation is valid in a tube around the nominal path in $(x_{3,r}(\tilde{s}), y_{3,r}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$ for which $\kappa_{3,r}\tilde{z}_3 < 1$. The width of this tube depends on the semitrailer's nominal curvature $\kappa_{3,r}$. When the nominal curvature tends to zero (a straight nominal path), $\tilde{z}_3$ can vary arbitrarily. Essentially, to avoid the singularities in the transformation, we must have that $|\tilde{z}_3| < |\kappa_{3,r}^{-1}|$, when $\tilde{z}_3$ and $\kappa_{3,r}$ have the same sign. Note that $v_r \in \{-1, 1\}$ is included in (17a) to make $\tilde{s}(t)$ a monotonically increasing function in time during tracking of nominal paths in both forward and backward motion. Here, it is assumed that the longitudinal velocity of the tractor $v(t)$ is chosen such that $\text{sign}(v(t)) = v_r(\tilde{s}(t))$ and it is assumed that the orientation error of the semitrailer satisfies $|\tilde{\theta}_3| < \pi/2$. With the above assumptions, $\dot{\tilde{s}}(t) > 0$ during path following of nominal paths in both forward and backward motion.

The models for the remaining path-following error states $\tilde{\theta}_3(t)$, $\tilde{\beta}_3(t)$ and $\tilde{\beta}_2(t)$ are derived by applying the chain rule, together with equations (1)–(3), (14) and (17a):

$$\dot{\tilde{\theta}}_3 = v_3 \left(\frac{\tan(\tilde{\beta}_3 + \beta_{3,r})}{L_3} - \frac{\kappa_{3,r}\cos\tilde{\theta}_3}{1 - \kappa_{3,r}\tilde{z}_3} \right), \qquad t \in \Pi(0, \tilde{s}_G), \tag{18a}$$

$$\begin{aligned}\dot{\tilde{\beta}}_3 = v_3 \Bigg(& \frac{\sin(\tilde{\beta}_2 + \beta_{2,r}) - M_1 \cos(\tilde{\beta}_2 + \beta_{2,r})(\tilde{\kappa} + \kappa_r)}{L_2 \cos(\tilde{\beta}_3 + \beta_{3,r}) C_1(\tilde{\beta}_2 + \beta_{2,r}, \tilde{\kappa} + \kappa_r)} - \frac{\tan(\tilde{\beta}_3 + \beta_{3,r})}{L_3} \\ & - \frac{\cos\tilde{\theta}_3}{1 - \kappa_{3,r}\tilde{z}_3} \left(\frac{\sin\beta_{2,r} - M_1 \cos\beta_{2,r}\kappa_r}{L_2 \cos\beta_{3,r} C_1(\beta_{2,r}, \kappa_r)} - \kappa_{3,r} \right) \Bigg), \quad t \in \Pi(0, \tilde{s}_G),\end{aligned} \tag{18b}$$

$$\begin{aligned}\dot{\tilde{\beta}}_2 = v_3 \Bigg(& \left(\frac{\tilde{\kappa} + \kappa_r - \frac{\sin(\tilde{\beta}_2 + \beta_{2,r})}{L_2} + \frac{M_1}{L_2}\cos(\tilde{\beta}_2 + \beta_{2,r})(\tilde{\kappa} + \kappa_r)}{\cos(\tilde{\beta}_3 + \beta_{3,r}) C_1(\tilde{\beta}_2 + \beta_{2,r}, \tilde{\kappa} + \kappa_r)} \right) \\ & - \frac{\cos\tilde{\theta}_3}{1 - \kappa_{3,r}\tilde{z}_3} \left(\frac{\kappa_r - \frac{\sin\beta_{2,r}}{L_2} + \frac{M_1}{L_2}\cos\beta_{2,r}\kappa_r}{\cos\beta_{3,r} C_1(\beta_{2,r}, \kappa_r)} \right) \Bigg), \qquad t \in \Pi(0, \tilde{s}_G).\end{aligned} \tag{18c}$$

A more detailed derivation of (18) is provided in Appendix A. Together, the differential equations in (17) and (18) describe the model of the G2T with a car-like tractor (1) in terms of deviation from the nominal path generated by the system in (14).

When path-following control is considered, the speed at which the nominal path (13) is executed is not considered, but only that it is followed with a small path-following error. This means

that the distance traveled $\tilde{s}(t)$ along the nominal path is not explicitly controlled by the path-following controller. However, the dependency of $\tilde{s}$ in (17b) and (18) makes the nonlinear system distance-varying. Define the path-following error states as $\tilde{x}_e = [\tilde{z}_3 \;\; \tilde{\theta}_3 \;\; \tilde{\beta}_3 \;\; \tilde{\beta}_2]^T$, where its model is given by (17b)–(18). By replacing v_3 with v using the relationship in (3), the path-following error model (17b)–(18) and the progression along the nominal path (17a), can compactly be expressed as (see Appendix A)

$$\dot{\tilde{s}} = v f_{\tilde{s}}(\tilde{s}, \tilde{x}_e), \qquad t \in \Pi(0, \tilde{s}_G), \tag{19a}$$

$$\dot{\tilde{x}}_e = v \tilde{f}(\tilde{s}, \tilde{x}_e, \tilde{\kappa}), \quad t \in \Pi(0, \tilde{s}_G), \tag{19b}$$

where $\tilde{f}(\tilde{s}, 0, 0) = 0$, $\forall t \in \Pi(0, \tilde{s}_G)$, *i.e.*, the origin $(\tilde{x}_e, \tilde{\kappa}) = (0, 0)$ is an equilibrium point. Since v enters linearly in (19), in analogy to (4), time-scaling [58] can be applied to eliminate the speed dependence $|v|$ from the model. Therefore, without loss of generality, it is hereafter assumed that the longitudinal velocity of the rear axle of the tractor is chosen as $v(t) = v_r(\tilde{s}(t)) \in \{-1, 1\}$ which implies that $\dot{\tilde{s}}(t) > 0$. Moreover, from the construction of the set of motion primitives $\mathcal{P}$, each motion primitive $m_i \in \mathcal{P}$ encodes a forward or backward motion segment (see Section 4.2).

5.1 Local behavior around a nominal path

The path-following error model in (17b) and (18) can be linearized around the nominal path $(x_r(\tilde{s}), u_r(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$ by equivalently linearizing (19b) around the origin $(\tilde{x}_e, \tilde{\kappa}) = (0, 0)$. The origin is by construction an equilibrium point to (19b) and hence a first-order Taylor series expansion yields

$$\dot{\tilde{x}}_e = vA(\tilde{s}(t))\tilde{x}_e + vB(\tilde{s}(t))\tilde{\kappa}, \quad t \in \Pi(0, \tilde{s}_G). \tag{20}$$

For the special case when the nominal path moves the system either straight forwards or backwards, the matrices A and B simplify to

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{L_3} & 0 \\ 0 & 0 & -\frac{1}{L_3} & \frac{1}{L_2} \\ 0 & 0 & 0 & -\frac{1}{L_2} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ -\frac{M_1}{L_2} \\ \frac{L_2 + M_1}{L_2} \end{bmatrix}, \tag{21}$$

and the characteristic polynomial is

$$\det(\lambda I - vA) = v^2 \lambda^2 \left(\lambda + \frac{v}{L_3}\right)\left(\lambda + \frac{v}{L_2}\right). \tag{22}$$

Thus, around a straight nominal path, the linearized system in (20) is marginally stable in forward motion ($v > 0$) because of the double integrator and unstable in backward motion ($v < 0$), since the system has two poles in the right half plane. Due to the positive off-axle hitching $M_1 > 0$, the linearized system has a zero in some of the output channels [4, 45]. In forward motion, the system has non-minimum phase properties since the zero is located in the right half-plane (see [45] for an

extensive analysis). In backward motion, this zero is located in the left half-plane and the system is instead minimum phase.

In the sequel, we focus on stabilizing the path-following error model (19b) in some neighborhood around the origin $(\tilde{x}_e, \tilde{\kappa}) = (0,0)$. This is done by utilizing the framework presented in [39], where the closed-loop system consisting of the controlled vehicle and the path-following controller, executing a nominal path computed by a lattice planner, is first modeled as a hybrid system. The framework is tailored for the lattice-based path planner considered in this work and is motivated because it is well-known from the theory of hybrid systems that switching between stable systems in an inappropriate way can lead to instability of the switched system [21, 53].

5.2 Connection to hybrid systems

The nominal path (14) is computed online by the lattice planner and is thus a priori unknown. However, it is composed of a finite sequence of precomputed motion primitives $\{m_{u_q[k]}\}_{k=0}^{N-1}$ of length N. Each motion primitive m_i is chosen from the set of M possible motion primitives, *i.e.*, $m_i \in \mathcal{P}$. Along motion primitive $m_i \in \mathcal{P}$, the nominal path is represented as $(x_r^i(\tilde{s}), u_r^i(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_f^i]$ and the path-following error model (19b) becomes

$$\dot{\tilde{x}}_e = v_r(\tilde{s})\tilde{f}_i(\tilde{s}, \tilde{x}_e, \tilde{\kappa}), \quad t \in \Pi(0, \tilde{s}_f^i). \tag{23}$$

From the fact that the sequence of motion primitives is selected by the lattice planner, it follows that the system can be descried as a hybrid system. Define $q : [0, \tilde{s}_G] \to \{1, \ldots, M\}$ as a piecewise integer-valued decision variable that is selected by the lattice planner. Then, the path-following error model can be written as a distance-switched continuous-time hybrid system

$$\dot{\tilde{x}}_e = v_r(\tilde{s})\tilde{f}_{q(\tilde{s})}(\tilde{s}, \tilde{x}_e, \tilde{\kappa}), \quad t \in \Pi(0, \tilde{s}_G). \tag{24}$$

This hybrid system is composed of M different subsystems, where only one subsystem is active for each $\tilde{s} \in [0, \tilde{s}_G]$. Here, $q(\tilde{s})$ is assumed to be right-continuous and from the construction of the motion primitives, it holds that there are finitely many switches in finite distance [21, 53]. We now turn to the problem of designing the hybrid path-following controller $\tilde{\kappa} = g_{q(\tilde{s})}(\tilde{x}_e)$, such that the path-following error is upper bounded by an exponentially decaying function during the execution of each motion primitive $m_i \in \mathcal{P}$, individually.

5.3 Design of the hybrid path-following controller

The synthesis of the path-following controller is performed separately for each motion primitive $m_i \in \mathcal{P}$. The class of hybrid path-following controllers is limited to piecewise linear state-feedback controllers with feedforward action. Denote the path-following controller dedicated for motion primitive $m_i \in \mathcal{P}$ as $\kappa(t) = \kappa_r(\tilde{s}(t)) + K_i\tilde{x}_e(t)$. When applying this control law to the path-following error model in (23), the nonlinear closed-loop system can, in a compact form, be written as

$$\dot{\tilde{x}}_e = v_r(\tilde{s})\tilde{f}_i(\tilde{s}, \tilde{x}_e, K_i\tilde{x}_e) = v_r(\tilde{s})\tilde{f}_{cl,i}(\tilde{s}, \tilde{x}_e), \quad t \in \Pi(0, \tilde{s}_f^i), \tag{25}$$

where $\tilde{x}_e = 0$ is an equilibrium point, since $f_{cl,i}(\tilde{s},0) = \tilde{f}_i(\tilde{s},0,0) = 0$, $\forall \tilde{s} \in [0,\tilde{s}_f^i]$. The state-feedback controller $\tilde{\kappa} = K_i\tilde{x}_e$ is intended to be designed such that the path-following error is locally bounded and decays towards zero during the execution of $m_i \in \mathcal{P}$. This is guaranteed by Theorem 2.

Assumption 3. *Assume $\tilde{f}_{cl,i} : [0,\tilde{s}_f^i] \times \tilde{\mathbb{X}}_e \to \mathbb{R}^4$ is continuously differentiable with respect to $\tilde{x}_e \in \tilde{\mathbb{X}}_e = \{\tilde{x}_e \in \mathbb{R}^4 \mid \|\tilde{x}_e\|_2 < r\}$ and the Jacobian matrix $[\partial f_{cl,i}/\partial \tilde{x}_e]$ is bounded and Lipschitz on $\tilde{\mathbb{X}}_e$, uniformly in $\tilde{s} \in [0,\tilde{s}_f^i]$.*

Theorem 2 ([39])**.** *Consider the closed-loop system in* (25)*. Under Assumption 3, let*

$$A_{cl,i}(\tilde{s}) = v_r(\tilde{s})\frac{\partial \tilde{f}_{cl,i}}{\partial \tilde{x}_e}(\tilde{s},0). \tag{26}$$

If there exist a common matrix $P_i \succ 0$ and a positive constant ε that satisfy

$$A_{cl,i}(\tilde{s})^T P_i + P_i A_{cl,i}(\tilde{s}) \preceq -2\varepsilon P_i \quad \forall \tilde{s} \in [0,\tilde{s}_f^i]. \tag{27}$$

Then, the following inequality holds

$$||\tilde{x}_e(t)|| \leq \rho_i ||\tilde{x}_e(0)|| e^{-\varepsilon t}, \quad \forall t \in \Pi(0,\tilde{s}_f^i), \tag{28}$$

where $\rho_i = Cond(P_i)$ is the condition number of P_i.

Proof. See, *e.g.*, [31]. □

Theorem 2 guarantees that if the feedback gain K_i is designed such that there exists a quadratic Lyapunov function $V_i(\tilde{x}_e) = \tilde{x}_e^T P_i \tilde{x}_e$ for (25) around the origin satisfying $\dot{V}_i \leq -2\varepsilon V_i$, then a small disturbance in the initial path-following error $\tilde{x}_e(0)$ results in a path-following error state trajectory $\tilde{x}_e(t)$ whose norm is upper bounded by an exponentially decaying function. In analogy to [39], the condition in (27) can be reformulated as a controller synthesis problem using linear matrix inequality (LMI) techniques. By using the chain rule, the matrix $A_{cl,i}(\tilde{s})$ in (26) can be written as

$$A_{cl,i}(\tilde{s}) = v_r(\tilde{s})\frac{\partial \tilde{f}_i}{\partial \tilde{x}}(\tilde{s},0,0) + v_r(\tilde{s})\frac{\partial \tilde{f}_i}{\partial \tilde{\kappa}}(\tilde{s},0,0)K_i \triangleq A_i(\tilde{s}) + B_i(\tilde{s})K_i. \tag{29}$$

Furthermore, assume the pairs $[A_i(\tilde{s}),B_i(\tilde{s})]$ lie in the convex polytope $\mathbb{S}_i$, $\forall \tilde{s} \in [0,\tilde{s}_f^i]$, where $\mathbb{S}_i$ is represented by its L_i vertices

$$[A_i(\tilde{s}),B_i(\tilde{s})] \in \mathbb{S}_i = \mathbf{Co}\left\{[A_{i,1},B_{i,1}],\ldots,[A_{i,L_i},B_{i,L_i}]\right\}, \tag{30}$$

where **Co** denotes the convex hull. Now, condition (27) in Theorem 2 can be reformulated as [15]:

$$(A_{i,j} + B_{i,j}K_i)^T P_i + P_i(A_{i,j} + B_{i,j}K_i) \preceq -2\varepsilon P_i, \quad j = 1,\ldots,L_i. \tag{31}$$

This matrix inequality is not jointly convex in P_i and K_i. However, if $\varepsilon > 0$ is fixed, using the bijective transformation $Q_i = P_i^{-1} \succ 0$ and $Y_i = K_i P_i^{-1} \in \mathbb{R}^{1\times 4}$, the matrix inequality in (31) can be rewritten as an LMI in Q_i and Y_i [71]:

$$Q_i A_{i,j}^T + Y_i^T B_{i,j}^T + A_{i,j} Q_i + B_{i,j} Y_i + 2\varepsilon Q_i \preceq 0, \quad j = 1,\ldots,L_i. \tag{32}$$

Hence, it is an LMI feasibility problem to find a linear state-feedback controller that satisfies condition (27) in Theorem 2. If Q_i and Y_i are feasible solutions to (32), the quadratic Lyapunov function is $V_i(\tilde{x}) = \tilde{x}^T Q_i^{-1} \tilde{x}$ and the linear state-feedback controller is $\tilde{\kappa} = Y_i Q_i^{-1} \tilde{x}_e$. As in [39], the LMI feasibility problem in (32) is reformulated as a semidefinite programming (SDP) problem

$$\begin{aligned} \underset{Y_i, Q_i}{\text{minimize}} \quad & \|Y_i - K_{\text{nom}}^i Q_i\| \\ \text{subject to} \quad & (32) \text{ and } Q_i \succeq I, \end{aligned} \tag{33}$$

where K_{nom}^i is a nominal feedback gain that depends on $m_i \in \mathcal{P}$. Here, two nominal feedback gains are used; $K_{\text{nom}}^i = K_{\text{fwd}}$ for all forward motion primitives $m_i \in \mathcal{P}_{\text{fwd}}$ and $K_{\text{nom}}^i = K_{\text{rev}}$ for all backward motion primitives $m_i \in \mathcal{P}_{\text{rev}}$. The motivation for this choice of objective function in (33) is that it is desired that the path-following controller inherits the nominal controller's properties. It is also used to reduce the number of different feedback gains, while not sacrificing desired convergence properties of the path-following error along the execution of each motion primitive. The nominal feedback gains K_{fwd} and K_{rev} are designed using infinite-horizon LQ-control [5] where the path-following error model has been linearized around a straight nominal path in backward and forward motion, respectively. In these cases, the Jacobian linearization is given by the matrices A and B defined in (21). After these nominal feedback gains have been designed, the optimization problem in (33) can be solved separately for each motion primitive, e.g., using YALMIP [41]. In this specific application, for all $m_i \in \mathcal{P}$, the optimal value of the objective function in (33) is zero, which implies that $K_i = K_{\text{nom}}^i$ since $Q_i \succ 0$. Thus, for this specific set of motion primitives $\mathcal{P}$ (see Figure 6), the hybrid path-following controller $\tilde{\kappa} = K_{q(\tilde{s})} \tilde{x}_e$ simplifies to

$$\kappa(t) = \kappa_r(\tilde{s}) + \begin{cases} K_{\text{fwd}} \tilde{x}_e(t), & m_i \in \mathcal{P}_{\text{fwd}}, \\ K_{\text{rev}} \tilde{x}_e(t), & m_i \in \mathcal{P}_{\text{rev}}, \end{cases} \tag{34}$$

where $\kappa_r(\tilde{s})$ is the feedforward computed by the lattice planner. Note that if a common quadratic Lyapunov function exists that satisfies (32) $\forall m_i \in \mathcal{P}$ (*i.e.*, $Q_i = Q$, but Y_i can vary), then the path-following error is guaranteed to exponentially decay towards zero under an arbitrary sequence of motion primitives [15, 21]. This is however not possible since the path-following error model (24) is underactuated[4] and the Jacobian linearization takes on the form in (20).

Theorem 3 ([39]). *Consider the switched linear system*

$$\dot{x} = vAx + vBu, \quad v \in \{-1, 1\}, \tag{35}$$

[4]Here, a system is defined underactuated if the number of control signals is strictly less than the dimension of its configuration space [45].

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. When $\text{rank}(B) < n$, there exists no hybrid linear state-feedback control law in the form

$$u = \begin{cases} K_1 x, & v = 1 \\ K_2 x, & v = -1 \end{cases}, \tag{36}$$

where $K_1 \in \mathbb{R}^{m\times n}$ and $K_2 \in \mathbb{R}^{m\times n}$, such that the closed-loop system is quadratically stable with a quadratic Lyapunov function $V(x) = x^T P x$, $\dot{V}(x) < 0$ and $P \succ 0$.

Proof. See [39]. □

From Theorem 3, it is clear that it is not possible to design a hybrid path-following controller $\tilde{\kappa} = K_{q(\tilde{s})}\tilde{x}_e$ such that the closed-loop path-following error system is locally quadratically stable along nominal paths that are composed of forward and backward motion primitives. In the next section, a systematic method is presented for analyzing the behavior of the distance-switched continuous-time hybrid system (24), when the hybrid path-following controller already has been designed.

5.4 Convergence along a combination of motion primitives

Consider the path-following error model in (24) with the hybrid path-following controller $\tilde{\kappa} = K_{q(\tilde{s})}\tilde{x}_e$ that has been designed following the steps presented in Section 5.3. Assume motion primitive $m_i \in \mathcal{P}$ is switched in at distance $\tilde{s}_k$, *i.e.*, $q(\tilde{s}(t)) = i$, for all $t \in \Pi(\tilde{s}_k, \tilde{s}_k + \tilde{s}_f^i)$. We are now interested in analyzing the evolution of the path-following error $\tilde{x}_e(t)$ during the execution of this motion primitive. Since the longitudinal velocity of the tractor is selected as $v(t) = v_r(\tilde{s}(t))$, then $\dot{\tilde{s}}(t) > 0$ and it is possible to eliminate the time-dependency in the path-following error model (19b). By applying the chain rule, we get $\frac{\mathrm{d}\tilde{x}_e}{\mathrm{d}\tilde{s}} = \frac{\mathrm{d}\tilde{x}_e}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tilde{s}} = \frac{\mathrm{d}\tilde{x}_e}{\mathrm{d}t}\frac{1}{\dot{\tilde{s}}}$. Hence, using (19a), the distance-based version of the path-following error model (19b) can be represented as

$$\frac{\mathrm{d}\tilde{x}_e}{\mathrm{d}\tilde{s}} = \frac{f_{cl,i}(\tilde{s}, \tilde{x}_e(\tilde{s}))}{f_s(\tilde{s}, \tilde{x}_e(\tilde{s}))}, \quad \tilde{s} \in [\tilde{s}_k, \tilde{s}_k + \tilde{s}_f^i], \tag{37}$$

where $\tilde{x}_e(\tilde{s}_k)$ is given. The evolution of the path-following error $\tilde{x}_e(\tilde{s})$ becomes

$$\tilde{x}_e(\tilde{s}_k + \tilde{s}_f^i) = \tilde{x}_e(\tilde{s}_k) + \int_{\tilde{s}_k}^{\tilde{s}_k + \tilde{s}_f^i} \frac{f_{cl,i}(\tilde{s}, \tilde{x}_e(\tilde{s}))}{f_s(\tilde{s}, \tilde{x}_e(\tilde{s}))} \mathrm{d}\tilde{s} \triangleq T_i(\tilde{x}_e(\tilde{s}_k)), \tag{38}$$

where $\tilde{x}_e(s_k)$ denotes the path-following error when motion primitive $m_i \in \mathcal{P}$ is started and $\tilde{x}_e(\tilde{s}_k + \tilde{s}_f^i)$ denotes the path-following error when the execution of m_i is finished. The solution to the integral in (38) has no analytical expression. However, numerical integration can be used to compute a local approximation of the evolution of $\tilde{x}_e(\tilde{s})$ between the two switching points $\tilde{s}_k$ and $\tilde{s}_k + \tilde{s}_f^i$. A first-order Taylor series expansion of (38) around the origin $\tilde{x}_e(\tilde{s}_k) = 0$ yields

$$\tilde{x}_e(\tilde{s}_k + \tilde{s}_f^i) = T_i(0) + \underbrace{\left.\frac{\mathrm{d}T_i(\tilde{x}_e(\tilde{s}_k))}{\mathrm{d}\tilde{x}_e(\tilde{s}_k)}\right|_{(0)}}_{=F_i} \tilde{x}_e(\tilde{s}_k). \tag{39}$$

The term $T_i(0) = 0$, since $\tilde{f}_{cl,i}(\tilde{s},0) = 0$, for all $\tilde{s} \in [\tilde{s}_k, \tilde{s}_k + \tilde{s}_f^i]$. Denote $\tilde{x}_e[k] = \tilde{x}_e(\tilde{s}_k)$, $\tilde{x}_e[k+1] = \tilde{x}_e(\tilde{s}_k + \tilde{s}_f^i)$ and $u_q[k] = q(\tilde{s}_k) = i$. By, *e.g.*, the use of finite differences, the evolution of the path-following error (38) after motion primitive $m_i \in \mathcal{P}$ has be executed can be approximated as a linear discrete-time system

$$\tilde{x}_e[k+1] = F_i \tilde{x}_e[k]. \tag{40}$$

Repeating this procedure for all M motion primitives, a set of M transition matrices $\mathbb{F} = \{F_1, \ldots, F_M\}$ can be computed. Then, the local evolution of the path-following error (38) between each switching point can be described as a linear discrete-time switched system

$$\tilde{x}_e[k+1] = F_{u_q[k]} \tilde{x}_e[k], \quad u_q[k] \in \{1, \ldots, M\}, \tag{41}$$

where the motion primitive sequence $\{u_q[k]\}_{k=0}^{N-1}$ and its length N are unknown at the time of the analysis. Exponential decay of the solution $\tilde{x}[k]$ to (41) is guaranteed by Theorem 4.

Theorem 4 ([39])**.** *Consider the linear discrete-time switched system in* (41)*. If there exist a matrix $S \succ 0$ and a $\eta \geq 1$ that satisfy*

$$I \preceq S \preceq \eta I, \tag{42a}$$

$$F_j^T S F_j - S \preceq -\mu S, \quad \forall j \in \{1, \ldots, M\}, \tag{42b}$$

where $0 < \mu < 1$ is a constant. Then, under arbitrary switching for $k \geq 0$ the following inequality holds

$$\|\tilde{x}_e[k]\| \leq \|\tilde{x}_e[0]\| \eta^{1/2} \lambda^k, \tag{43}$$

where $\lambda = \sqrt{1-\mu}$ and $\eta = Cond(S)$ denotes the condition number of S.

Proof. See [39]. □

Note that for a fixed μ, (42) is a set of LMIs in the variables S and η. The result in Theorem 4 establishes that the upper bound on the path-following error at the switching points exponentially decays towards zero. Thus, the norm of the initial path-following error $\|\tilde{x}_e(\tilde{s}_k)\|$, when starting the execution of a new motion primitive, will decrease as k grows. Moreover, combining Theorem 2 and Theorem 4, this implies that the upper bound on the continuous-time path-following error $\|\tilde{x}_e(t)\|$ will exponentially decay towards zero. This result is formalized in Corollary 1.

Corollary 1 ([39])**.** *Consider the hybrid system in* (24) *with the path-following controller $\tilde{\kappa} = K_{q(\tilde{s})} \tilde{x}_e$. Assume the conditions in Theorem 2 are satisfied for each mode $i \in \{1, \ldots, M\}$ of* (24) *and assume the conditions in Theorem 4 are satisfied for the resulting discrete-time switched system* (41)*. Then, $\forall k \in \mathbb{Z}_+$ and $t \in \Pi(\tilde{s}_k, \tilde{s}_k + \tilde{s}_f^i)$ with $q(\tilde{s}(t)) = i$, the continuous-time path-following error $\tilde{x}_e(t)$ satisfies*

$$\|\tilde{x}_e(t)\| \leq \|\tilde{x}_e(t_0)\| \eta^{1/2} \rho_i^{1/2} \lambda^k, \tag{44}$$

where $0 < \lambda < 1$, $\eta = Cond(S)$ and $\rho_i = Cond(P_i)$.

Proof. See [39]. □

The practical interpretation of Corollary 1 is that the upper bound on the continuous-time path-following error is guaranteed to exponentially decay towards zero as a function of the number of executed motion primitives. The analysis method presented in this section will be later used for this specific application in Section 8.1.

In this application, none of the vehicle states are directly observed from the vehicle's onboard sensors and we instead need to rely on dynamic output feedback [57], *i.e.*, the hybrid state-feedback controller $\tilde{\kappa} = K_{q(\tilde{s})}\tilde{x}_e$ is operating in series with a nonlinear observer. Naturally, the observer is operating in a discrete-time fashion and we make the assumption that the observer is operating sufficiently fast and estimates the state $\hat{x}(t_k)$ with good accuracy. This means that it is further assumed that the separation principle of estimation and control holds. That is, the current state estimate from the observer $\hat{x}(t_k)$ is interpreted as the true vehicle state $x(t_k)$, which is then used to construct the path-following error $\tilde{x}_e(t_k)$ used by the hybrid path-following controller.

6 State observer

The state-vector $x = [x_3 \;\; y_3 \;\; \theta_3 \;\; \beta_3 \;\; \beta_2]^T$ for the G2T with a car-like tractor is not directly observed from the sensors on the car-like tractor and therefore needs to be inferred using the available measurements, the vehicle model (1) and the geometry of the vehicle.

High accuracy measurements of the position of the rear axle of the car-like tractor (x_1, y_1) and its orientation θ_1 are obtained from the localization system that was briefly described in Section 2.1. To obtain information about the joint angles β_2 and β_3, a LIDAR sensor is mounted in the rear of the tractor as illustrated in Figure 8. This sensor provides a point-cloud from which the y-coordinate L_y, given in the tractor's local coordinate system, of the midpoint of the semitrailer's front and the relative orientation ϕ between the tractor and semitrailer can be extracted[5]. To estimate L_y and ϕ, an iterative RANSAC algorithm [27] is first used to find the visible edges of the semitrailer's body. Logical reasoning and the known width b of the semitrailer's front are used to classify an edge to the front, the left or the right side of the semitrailer's body. Once the front edge and its corresponding corners are found, L_y and ϕ can easily be calculated [9, 50].

The measurements $y_k^{\text{loc}} = [x_{1,k} \;\; y_{1,k} \;\; \theta_{1,k}]^T$ from the localization system and the constructed measurements $y_k^{\text{ran}} = [L_{y,k} \;\; \phi_k]^T$ from the iterative RANSAC algorithm are treated as synchronous observations with different sampling rates. These observations are fed to an EKF to estimate the full state vector $\hat{x}$ of the G2T with car-like tractor (1).

6.1 Extended Kalman filter

The EKF algorithm performs two steps, a time update where the next state $\hat{x}_{k|k-1}$ is predicted using a prediction model of the vehicle and a measurement update that corrects $\hat{x}_{k|k-1}$ to give a filtered

[5] Other features could be extracted from the point cloud, but using L_y and ϕ have shown to yield good performance in practice.

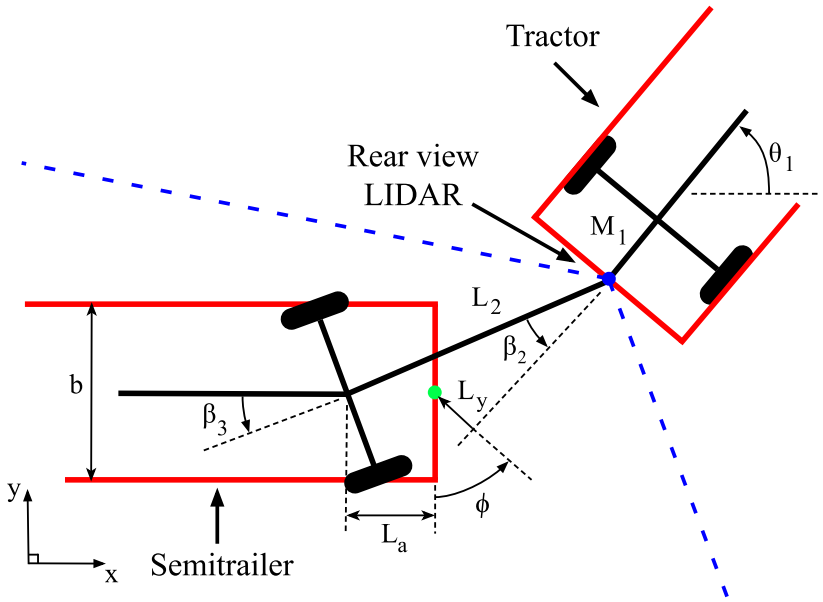


Figure 8: A bird's-eye view of the connection between the car-like tractor and the semitrailer, as well as the geometric properties of the semitrailer that are used by the nonlinear observer. The green dot represents the midpoint of the front of the semitrailer's body, where L_y is the y-coordinate in the tractor's local coordinate system. The LIDAR sensor is mounted at the blue dot and the dashed blue lines illustrate the LIDAR's field of view.

estimate $\hat{x}_{k|k}$ using the available measurements [29].

To construct the prediction model, the continuous-time model of the G2T with a car-like tractor (1) is discretized using Euler forward with a sampling time of T_s seconds. The control signals to the prediction model are the longitudinal velocity v of the car-like tractor and its curvature κ. Given the control signals $u_k = [v_k \;\; \kappa_k]^T$, the vehicle state x_k and a process noise model w_k with covariance Σ^w, the prediction model for the G2T with a car-like tractor can be written as

$$x_{k+1} = \hat{f}(x_k, u_k, w_k), \quad w_k \sim \mathcal{N}(0, \Sigma^w). \tag{45}$$

Since the observations y_k^{loc} and y_k^{ran} are updated at different sampling rates, independent measurement equations for each observation are derived. Assuming measurements with normally distributed zero mean noise, the measurement equation for the observation from the iterative RANSAC algorithm $y_k^{\text{ran}} = [L_{y,k} \;\; \phi_k]^T$ can be written as

$$y_k^{\text{ran}} = h^{\text{ran}}(x_k) + e_k^{\text{ran}}, \quad e_k^{\text{ran}} \sim \mathcal{N}(0, \Sigma_{\text{ran}}^e), \tag{46}$$

where e_k^{ran} is the measurement noise with covariance matrix Σ_{ran}^e and $h^{\text{ran}}(x_k)$ defines the relationship between the states and the measurements. From Figure 8, the two components of $h^{\text{ran}}(x_k)$ can be derived as

$$L_{y,k} = h_1^{\text{ran}}(x_k) = L_2 \sin\beta_{2,k} - L_a \sin(\beta_{2,k} + \beta_{3,k}), \tag{47a}$$

$$\phi_k = h_2^{\text{ran}}(x_k) = \beta_{2,k} + \beta_{3,k}. \tag{47b}$$

The second measurement equation, corresponding to the observation from the localization system $y_k^{\text{loc}} = [x_{1,k} \;\; y_{1,k} \;\; \theta_{1,k}]^T$ is given by

$$y_k^{\text{loc}} = h^{\text{loc}}(x_k) + e_k^{\text{loc}}, \quad e_k^{\text{loc}} \sim \mathcal{N}(0, \Sigma_{\text{loc}}^e), \tag{48}$$

where the components of $h^{\text{loc}}(x_k)$ can be derived from Figure 3 as

$$x_{1,k} = h_1^{\text{loc}}(x_k) = x_{3,k} + L_3 \cos\theta_{3,k} + L_2 \cos(\theta_{3,k} + \beta_{3,k}) + M_1 \cos(\theta_{3,k} + \beta_{3,k} + \beta_{2,k}), \tag{49a}$$

$$y_{1,k} = h_2^{\text{loc}}(x_k) = y_{3,k} + L_3 \sin\theta_{3,k} + L_2 \sin(\theta_{3,k} + \beta_{3,k}) + M_1 \sin(\theta_{3,k} + \beta_{3,k} + \beta_{2,k}), \tag{49b}$$

$$\theta_{1,k} = h_3^{\text{loc}}(x_k) = \theta_{3,k} + \beta_{3,k} + \beta_{2,k}, \tag{49c}$$

and e_k^{loc} is the the measurement noise with covariance matrix Σ_{loc}^e. The standard EKF framework is now applied using the prediction model in (45) and the measurement equations in (46) and (48).

The process noise w_k is assumed to enter additively with u_k into the prediction model in (45) and the time update of the EKF is performed as follows

$$\hat{x}_{k+1|k} = \hat{f}(\hat{x}_{k|k}, u_k, 0), \tag{50a}$$

$$\Sigma_{k+1|k}^x = F_k \Sigma_{k|k}^x F_k^T + G_{w,k} \Sigma^w G_{w,k}^T, \tag{50b}$$

where $F_k = \hat{f}_x'(\hat{x}_{k|k}, u_k, 0)$ and $G_{w,k} = \hat{f}_u'(\hat{x}_{k|k}, u_k, 0)$ are the linearization of the prediction model around the current state estimate $\hat{x}_{k|k}$ with respect to x and u, respectively.

Since the observations y_k^{loc} and y_k^{ran} are updated at different sampling rates, the measurement update of the state estimate $\hat{x}_{k|k}$ and the covariance matrix $P_{k|k}$ is performed sequentially for y_k^{loc} and y_k^{ran}. Let H_k be defined as the block matrix

$$H_k = \begin{bmatrix} H_{1,k} \\ H_{2,k} \end{bmatrix} = \begin{bmatrix} \left(\dfrac{\partial h^{\text{ran}}(x_{k|k-1})}{\partial x} \right)^T & \left(\dfrac{\partial h^{\text{loc}}(x_{k|k-1})}{\partial x} \right)^T \end{bmatrix}^T. \tag{51}$$

Each time an observation from the localization system y_k^{loc} is available, the following measurement update is performed

$$K_k = \Sigma_{k|k-1}^x H_{2,k}^T \left(\Sigma_{\text{loc}}^e + H_{2,k} \Sigma_{k|k-1}^x H_{2,k}^T \right)^{-1}, \tag{52a}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \left(y_k^{\text{loc}} - h^{\text{loc}}(\hat{x}_{k|k-1}) \right), \tag{52b}$$

$$\Sigma_{k|k}^x = \Sigma_{k|k-1}^x - K_k H_{2,k} \Sigma_{k|k-1}^x, \tag{52c}$$

where K_k is the Kalman gain [29]. Similarly, when the observation y_k^{ran} is updated, the same measurement update (52) is performed with Σ_{loc}^e, $H_{2,k}$, y_k^{loc} and h^{loc} replaced with Σ_{ran}^e, $H_{1,k}$, y_k^{ran} and h^{ran}, respectively.

To decrease the convergence time of the estimation error, the EKF is initialized as follows. Define the combined measurement equation of y_k^{ran} and y_k^{loc} as $y_k = h(x_k)$. Assuming noise-free

observations and that $|\beta_{2,k}| \leq \pi/2$, this system of equations has a unique solution $x_k = h^{-1}(y_k)$ given by

$$\beta_{2,k} = \arcsin\left(\frac{L_{y,k} + L_a \sin\phi_k}{L_2}\right) = h^{-1}_{\beta_{2,k}}(y_k), \tag{53a}$$

$$\beta_{3,k} = \phi_k - h^{-1}_{\beta_{2,k}}(y_k) = h^{-1}_{\beta_{3,k}}(y_k), \tag{53b}$$

$$\theta_{3,k} = \theta_{1,k} - \phi_k = h^{-1}_{\theta_{3,k}}(y_k), \tag{53c}$$

$$\begin{aligned} x_{3,k} = x_{1,k} &- L_3 \cos\left(h^{-1}_{\theta_{3,k}}(y_k)\right) - L_2 \cos\left(h^{-1}_{\theta_{3,k}}(y_k) + h^{-1}_{\beta_{3,k}}(y_k)\right) \\ &+ M_1 \cos\left(h^{-1}_{\theta_{3,k}}(y_k) + h^{-1}_{\beta_{3,k}}(y_k) + h^{-1}_{\beta_{2,k}}(y_k)\right), \end{aligned} \tag{53d}$$

$$\begin{aligned} y_{3,k} = y_{1,k} &- L_3 \sin\left(h^{-1}_{\theta_{3,k}}(y_k)\right) - L_2 \sin\left(h^{-1}_{\theta_{3,k}}(y_k) + h^{-1}_{\beta_{3,k}}(y_k)\right) \\ &+ M_1 \sin\left(h^{-1}_{\theta_{3,k}}(y_k) + h^{-1}_{\beta_{3,k}}(y_k) + h^{-1}_{\beta_{2,k}}(y_k)\right). \end{aligned} \tag{53e}$$

This relationship is used to initialize the EKF with the initial state estimate $\hat{x}_{1|0} = h^{-1}(y_0)$, the first time both measurements are obtained. The state covariance matrix is at the same time initialized to $\Sigma^x_{1|0} = \Sigma^x_0$, where $\Sigma^x_0 \succeq 0$ is a design parameter. Since no ground truth is available for all vehicle states, the filter cannot be individually evaluated but will be seen as part of the full system and thus be evaluated through the overall system performance.

7 Implementation details: Application to full-scale tractor-trailer system

The path planning and path-following control framework has been deployed on a modified version of a Scania G580 6x4 tractor that is shown in Figure 1. The car-like tractor is equipped with a sensor platform as described in Section 2, including a real time kinematic GPS (RTK-GPS), IMUs and a rear view LIDAR sensor with 140 degrees field of view in the horizontal scan field. The tractor is also equipped with a servo motor for automated control of the steering column and additional computation power compared to the commercially available version. The triple axle semitrailer and the double axle dolly are both commercially available and are not equipped with any sensors that are used by the system. The vehicle lengths and the physical parameters for the car-like tractor are summarized in Table 1, where it is assumed that the rotational centers are located at the longitudinal center for each axle pair and triple, respectively. The total distance from the front axle of the car-like tractor to the center of the axle of the semitrailer is approximately 18.4 m. In the remainder of this section, implementation details for each module within the path planning and path-following control framework are presented.

7.1 Lattice planner

The lattice planner is implemented in C++ and the motion primitive set is calculated offline using the numerical optimal control solver CasADi [6], together with the primal-dual interior-point

solver IPOPT [68]. The resulting paths are represented as discretely sampled points containing full state information including the control signals. For generation of the set of backward motion primitives $\mathcal{P}_{\text{rev}}$, the weight matrices $\mathbf{Q}_1 \succeq 0$ and $\mathbf{Q}_2 \succeq 0$ in the cost function (11) are chosen as

$$\mathbf{Q}_1 = \begin{bmatrix} 11 & -10 \\ -10 & 11 \end{bmatrix}, \quad \mathbf{Q}_2 = \text{diag}\left(\begin{bmatrix} 1 & 10 & 1 \end{bmatrix}\right),$$

giving the integrand $|| \begin{bmatrix} \beta_3 & \beta_2 \end{bmatrix}^T ||^2_{\mathbf{Q_1}} = \beta_3^2 + \beta_2^2 + 10(\beta_3 - \beta_2)^2$. This means that large joint angles with opposite signs are highly penalized during backward motion, which is directly related to motion plans that have an increased risk of leading to a jack-knife state during path execution. For the set of forward motion primitives $\mathcal{P}_{\text{fwd}}$, the weight $\mathbf{Q}_1$ is chosen as $\mathbf{Q}_1 = 0_{2\times 2}$. During motion primitive generation, the physical limitation on steering angle $\alpha_{\max}$ is additionally 20 % tightened to enable the path-following controller to reject disturbances during plan execution. The complete set of motion primitives from the initial orientation $\theta_{3,i} = 0$ is presented in Figure 6. The generated motion primitive set $\mathcal{P}$ was then reduced using the reduction technique described in Section 4.3, with $\eta = 1.2$, yielding a reduction factor of about 7 %. The size of the reduced motion primitive set was $|\mathcal{P}'| = 3888$, with between 66–111 different state transitions from each discrete state $z[k] \in \mathbb{Z}_d$. For the reduced motion primitive set $\mathcal{P}'$, a free-space HLUT [19, 32] was precomputed using a Dijkstra's search with cut-off cost $J_{\text{cut}} = 170$. The surrounding environment is represented by an occupancy gridmap [24] and efficient collision checking is performed using grid inflation and circle approximations for the semitrailer's and tractor's bounding boxes [34].

In the experiments, the lattice planner is given a desired goal state z_G that can be specified by an operator or selected by an algorithm. At the goal state, the system is constrained to end up in a straight vehicle configuration where all vehicle segments are lined up, *i.e.*, the steering angle and the joint angles are constrained to zero at the goal state. When a desired goal state z_G has been specified, the vehicle's initial state $z(0)$ is first projected down to its closest neighboring state in $\mathbb{Z}_d$. The ARA* search algorithm is initialized with heuristic inflation factor $\gamma = 2$ and then iteratively decreased by 0.1 in every subsequent iteration. If γ reaches 1 or if a specified maximum allowed planning time is reached and a motion plan with a proven γ-suboptimality cost has been found, portions of the resulting motion plan are sent to the path-following controller for path execution.

Table 1: Vehicle parameters for the research platform used for the real-world experiments.

Vehicle Parameters	Value
The tractor's wheelbase L_1	4.62 m
Maximum steering angle $\alpha_{\max}$	$42\pi/180$ rad
Maximum steering angle rate $\omega_{\max}$	0.6 rad/s
Maximum steering angle acceleration $u_{\omega,\max}$	40 rad/s^2
Length of the off-hitch M_1	1.66 m
Length of the dolly L_2	3.87 m
Length of the semitrailer L_3	8.00 m
Length of the semitrailer's overhang L_a	1.73 m
Width of the semitrailer's front b	2.45 m

Table 2: Design parameters for the EKF and the path-following controller during the real-world experiments.

EKF parameters	Value
Process noise Σ^w	$10^{-3} \times \text{diag}\left(\begin{bmatrix}1 & 1\end{bmatrix}\right)$
Measurement noise Σ^e_{loc}	$10^{-3} \times \text{diag}\left(\begin{bmatrix}1 & 1 & 0.5\end{bmatrix}\right)$
Measurement noise Σ^e_{ran}	$10^{-3} \times \text{diag}\left(\begin{bmatrix}0.5 & 0.1\end{bmatrix}\right)$
Initial state covariance Σ^x_0	$0.5 \times \text{diag}\left(\begin{bmatrix}1 & 1 & 0.1 & 0.1 & 0.1\end{bmatrix}\right)$
EKF frequency	100 Hz
Controller parameters	**Value**
Nominal LQ weight $\tilde{Q}_{\text{fwd}}$	$0.05 \times \text{diag}\left(\begin{bmatrix}0.8 & 6 & 8 & 8\end{bmatrix}\right)$
Nominal LQ weight $\tilde{Q}_{\text{rev}}$	$0.05 \times \text{diag}\left(\begin{bmatrix}0.3 & 6 & 7 & 5\end{bmatrix}\right)$
Controller frequency	50 Hz

7.2 Path-following controller

The framework presented in Section 5.3 is deployed to synthesize the path-following controller for this specific application. First, a feedback gain K_i and a corresponding Lyapunov function $V_i(\tilde{x}_e) = \tilde{x}_e^T P_i \tilde{x}_e$ is computed for each motion primitive $m_i \in \mathcal{P}$, separately. The convex polytope $\mathbb{S}_i$ in (30) is estimated by evaluating the Jacobian linearization (29) of the path-following error model (25) at each sampled point of the nominal path. Each resulting pair $[A_{i,j}, B_{i,j}]$ of the linearization is assumed to be a vertex of the convex polytope $\mathbb{S}_i$ in (30). The nominal feedback gains are designed using infinite-horizon LQ-control [5] where the path-following error model has been linearized around a straight nominal path in backward and forward motion, respectively. In these cases, the Jacobian linearization is given by the matrices A and B defined in (21). The weight matrices $\tilde{Q}_{\text{fwd}}$ and $\tilde{Q}_{\text{rev}}$ that are used in the LQ-design are listed in Table 2. By choosing the penalty on the curvature deviation as $\tilde{R}_{\text{rev}} = \tilde{R}_{\text{fwd}} = 1$, the nominal feedback gains are

$$K_{\text{rev}} = \begin{bmatrix}-0.12 & 1.67 & -1.58 & 0.64\end{bmatrix}, \tag{54a}$$

$$K_{\text{fwd}} = -\begin{bmatrix}0.20 & 2.95 & 1.65 & 1.22\end{bmatrix}, \tag{54b}$$

where positive feedback is assumed. Here, K_{fwd} and K_{rev} are dedicated for the set of forward $\mathcal{P}_{\text{fwd}}$ and backward $\mathcal{P}_{\text{rev}}$ motion primitives, respectively. Using these nominal feedback gains, the SDP problem in (33) with decay rate $\varepsilon = 0.01$ is solved separately for each motion primitive $m_i \in \mathcal{P}$ using YALMIP [41]. In this specific application, $\forall m_i \in \mathcal{P}$, the optimal value of the objective function in (33) is zero, which implies that $K_i = K^i_{\text{nom}}$. Thus, for this specific set of motion primitives $m_i \in \mathcal{P}$ (see Figure 6), the hybrid path-following controller can be written as in (34). However, the continuous-time quadratic Lyapunov functions are not equal $\forall m_i \in \mathcal{P}$.

In this setup, it is possible to find a common quadratic Lyapunov function $V_{\text{fwd}}(\tilde{x}_e)$ with decay-rate $\varepsilon = 0.01$ and path-following controller $\tilde{\kappa} = K_{\text{fwd}}\tilde{x}_e$, for all forward motion primitives $\mathcal{P}_{\text{fwd}}$. It is also possible to find a common quadratic Lyapunov function $V_{\text{rev}}(\tilde{x}_e)$ with decay-rate $\varepsilon = 0.01$ and path-following controller $\tilde{\kappa} = K_{\text{rev}}\tilde{x}_e$, for all backward motion primitives $\mathcal{P}_{\text{rev}}$. It is however

not possible to find a common quadratic Lyapunov function $V(\tilde{x}_e)$ with a decay-rate $\varepsilon > 0$ and hybrid path-following controller $\tilde{\kappa} = K_i \tilde{x}_e$, $i \in \{1, \ldots, M\}$, for the complete set of forward and backward motion primitives $\mathcal{P}$. This follows directly from Theorem 3. Thus, if the lattice planner is constrained to only compute nominal paths using either $\mathcal{P}_{\text{fwd}}$ or $\mathcal{P}_{\text{rev}}$, it is possible to guarantee that the continuous-time path-following error exponentially decays towards zero. To guarantee similar properties for the path-following error when the motion plan is composed by a combination of forward and backward motion primitives, the analysis method presented in Section 5.4 needs to be applied. This analysis is presented in Section 8.1.

The hybrid path-following controller (34) was implemented in Matlab/Simulink and C-code was then auto-generated where the path-following controller was specified to operate at 50 Hz. During the real-world experiments, the tractor's cruise controller was used for longitudinal control with $v = 1$ m/s along forward motion primitives and $v = -0.8$ m/s along backward motion primitives.

7.3 State observer

The design parameters for the EKF are summarized in Table 2, which were tuned using collected data from manual tests with the vehicle. This data was then used offline to tune the covariance matrices in the EKF and to calibrate the position and orientation of the rear view LIDAR sensor. The pitch angle of the rear view LIDAR sensor was adjusted such that that body of the semitrailer was visible in the LIDAR's point-cloud for all vehicle configurations that are of relevance for this application.

The EKF and the iterative RANSAC algorithm [9, 50] was implemented in Matlab/Simulink and C-code was then auto-generated. The EKF was specified to operate at 100 Hz and the measurements from the localization system is updated at the same sampling rate. The observation from the iterative RANSAC algorithm is received at a sampling rate of 20 Hz. The iterative RANSAC algorithm is specified to extract at most two edges of the semitrailer's body and 500 random selections of data pairs are performed for each edge extraction with an inlier threshold of 5 centimeters.

8 Results

In this section, we first analyze the behavior of the closed-loop system consisting of the controlled G2T with a car-like tractor and the path-following controller when executing a nominal path computed by the lattice planner. Then, the planning capabilities of the lattice planner and the ideal tracking performance of the path-following controller are evaluated in simulations. Finally, the complete framework is evaluated in three different real-world experiments on the full-scale test vehicle that can be seen in Figure 1.

8.1 Analysis of the closed-loop hybrid system

To verify that the path-following error $\tilde{x}_e(t)$ is bounded and decays toward zero when the nominal path is constructed by any sequence of motion primitives, backward as well as forward ones, the method presented in Section 5.4 is applied. The closed-loop system in (37) is implemented in MATLAB/Simulink. Central differences with step size $\delta = 0.01$ is used to compute the state-transition matrix F_i for the linear discrete-time system (40) that describes the evolution of the path-following error (38) when motion primitive $m_i \in \mathcal{P}$ is executed. Since there are four path-following error states, eight simulations are performed in order to generate each transition matrix F_i. This numerical differentiation is performed for all $m_i \in \mathcal{P}$ and M state-transition matrices are produced, *i.e.*, $\mathbb{F} = \{F_1, \ldots, F_M\}$.

The matrix inequalities in (42) are solved to show that the norm of the path-following error for the discrete-time switched system in (41), exponentially decays towards zero at the switching instants $\tilde{x}_e[k]$. By selecting $0 < \mu < 1$, the following SDP problem can be solved

$$\begin{aligned} \underset{\eta, S}{\text{minimize}} \quad & \eta \\ \text{subject to} \quad & F_j^T S F_j - S \preceq -\mu S, \quad j = 1, \ldots, M \\ & I \preceq S \preceq \eta I, \end{aligned} \tag{55}$$

where the condition number of S is minimized to compute a guaranteed upper bound (43) of the path-following error that is as tight as possible. It turns out that it is not possible to select $0 < \mu < 1$ such that a feasible solution to (55) exists for the original motion primitive set $\mathcal{P}$. This is because there exist short motion primitives of about 1 m that moves the vehicle either straight forwards or backwards. If these motion primitives are switched between, it is not possible to guarantee that the norm of the path-following error at the switching points will exponentially decay towards zero. In order to resolve this, the short motion primitives were extended to about 18 m (as the length of the tractor-trailer system) and their corresponding state-transition matrices F_i were again computed. With this adjusted motion primitive set $\mathcal{P}_{\text{adj}}$ and $\mu = 0.3$, the optimization problem in (55) is feasible to solve using YALMIP [41] and the optimal solutions are $\eta = 51.58$ and

$$S = \begin{bmatrix} 1.04 & 1.29 & 0.29 & 0.34 \\ 1.29 & 50.54 & -0.22 & 6.62 \\ 0.29 & -0.23 & 51.09 & 2.58 \\ 0.34 & 6.62 & 2.58 & 5.16 \end{bmatrix}. \tag{56}$$

Extending the short motion primitives manually is equivalent to adding constraints on the switching sequence $\{u_q[k]\}_{k=0}^{N-1}$ in the lattice planner. For this case, when a short motion primitive $p_i \in \mathcal{P}$ is activated, $u_q[k]$ needs to remain constant for a certain amount of switching instances. This constraint can easily be added within the lattice planner. To illustrate the phenomenon, Figure 9 shows the behavior of the closed-loop path-following error system when switching thirty times between a straight forward and backward motion primitive of three different path lengths $D = 1$ m, 10 m and 18 m, with the initial path-following error state $\tilde{x}_e[0] = [1 \text{ m } \ 0.1 \text{ rad } \ -0.1 \text{ rad } \ 0.1 \text{ rad}]^T$. In the figure, the path-following error at each switching instance is evaluated using $V_d(\tilde{x}_e[k]) =$

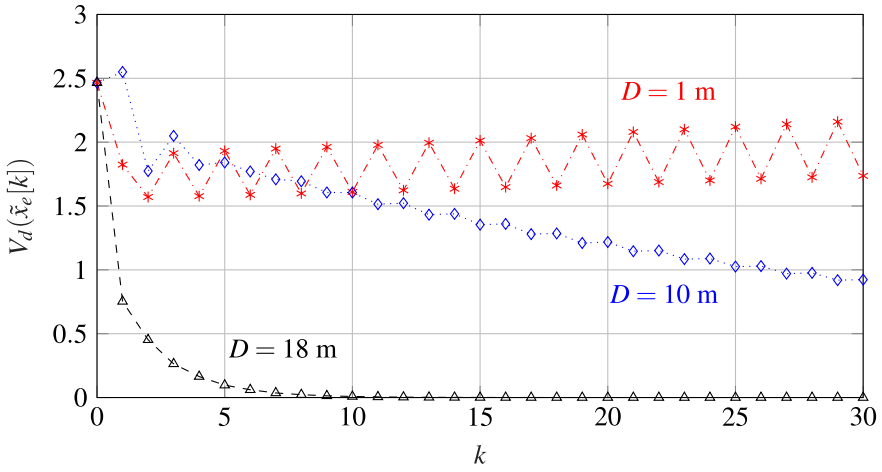


Figure 9: Simulation results of the closed-loop path-following error system when switching thirty times between a straight forward and backward motion primitive of different path lengths D m. The path-following error at each switching instance $\tilde{x}_e[k]$ is evaluated using the discrete-time Lyapunov function $V_d(\tilde{x}_e[k]) = \tilde{x}_e^T[k]S\tilde{x}_e[k]$, where S is given in (56) for $D = 1$ m (red stars), $D = 10$ m (blue diamonds) and $D = 18$ m (black triangles).

$\tilde{x}_e^T[k]S\tilde{x}_e[k]$, where S is given in (56). As can be seen, $V_d(\tilde{x}_e[k])$ is a valid discrete-time Lyapunov function for $D = 18$ m, since $V_d(\tilde{x}_e[k])$ (black triangles) is monotonically decreasing towards zero as a function of the number of executed motion primitives. When $D = 10$ m, $V_d(\tilde{x}_e[k])$ (blue diamonds) decays towards zero, but not monotonously. When $D = 1$ m, $V_d(\tilde{x}_e[k])$ (red star) does not converge towards zero. From our practical experience, allowing the short motion primitives have not caused any problems, since repeated switching between short straight forward and backward motion primitives is of limited practical relevance for the planning problems typically considered in this work.

8.2 Simulation results

Results from a quantitative analysis of the lattice planner is first presented, where its performance has been statistically evaluated in Monte Carlo simulations in two practically relevant scenarios. Then, simulation results for the path-following controller during ideal conditions where perfect state information is available is given to demonstrate its performance. The simulations have been performed on a standard laptop computer with an Intel Core i7-6820HQ@2.7GHz CPU.

8.2.1 Simulation results for the lattice planner

Two different path planning scenarios are used to evaluate the performance of the lattice planner. One thousand Monte Carlo simulations are performed for each scenario, where the goal state $z_G \in \mathbb{Z}_d$ and/or the initial state $z_I \in \mathbb{Z}_d$ are randomly selected from specified regions that are compliant with the specified state-space discretization $\mathbb{Z}_d$. For simplicity, it is assumed that the vehicle starts and ends in a straight vehicle configuration where all vehicle segments are lined up, *i.e.*, the steering angle and the joint angles are constrained to zero. A goal state and an initial state, respectively, is thus specified by a position (x_3, y_3) and an orientation θ_3 of the semitrailer. As

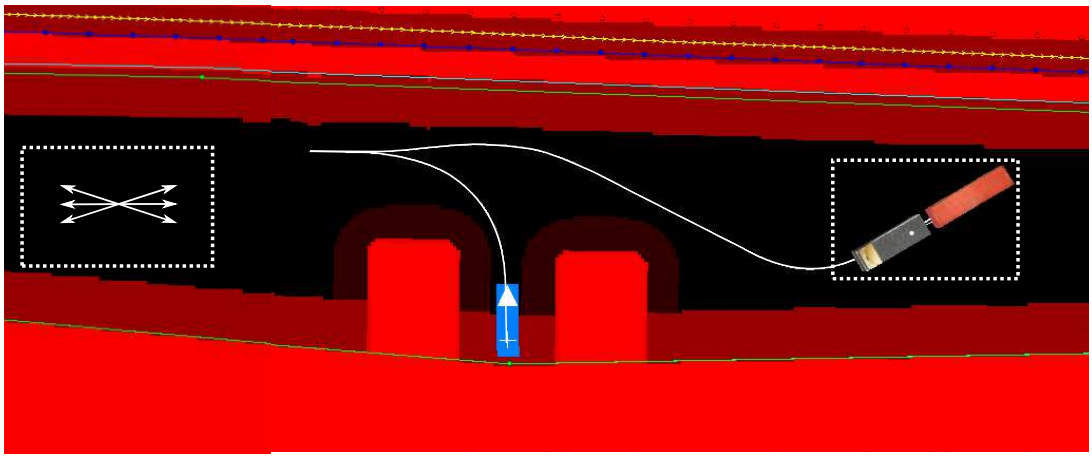


Figure 10: An overview of the parking planning problem. The goal position of the axle of the semitrailer $(x_{3,G}, y_{3,G})$ is marked by the white cross inside the blue rectangle, where the white arrow specifies its goal orientation $\theta_{3,G}$. The initial position $(x_{3,I}, y_{3,I})$ is uniformly sampled within the two white-dotted rectangles, and the initial orientation $\theta_{3,I} \in \Theta$ is sampled from six different initial orientations. The white path illustrates the planned path for the axle of the semitrailer for one out of 1000 Monte Carlo simulations. The area occupied by obstacles is colored in red and the black area is free-space.

explained in Section 7.1, the ARA* search is initialized with heuristic inflation factor $\gamma = 2$. This factor is then iteratively decreased by 0.1 every time a path to the goal for a specific γ has been found. To evaluate the computation time and the quality of the produced solution, the lattice planner was allowed to plan until an optimal solution with $\gamma = 1$ was found. In the Monte Carlo simulations, each time a solution for a specific γ is found, the accumulated planning time and the value of the cost function J_D are stored. During the simulations, a planning problem is marked unsolved if the planning time exceeds 60 s and a solution with $\gamma = 1$ has not yet been found.

The first planning scenario is illustrated in Figure 10, and the objective is to plan a parking maneuver from a randomly selected initial state $z_I \in \mathbb{Z}_d$ to a fixed goal state $z_G \in \mathbb{Z}_d$. In Figure 10, the goal position of the axle of the semitrailer $(x_{3,G}, y_{3,G})$ is illustrated by the white cross inside the blue rectangle, and the white arrow specifies its goal orientation $\theta_{3,G}$. The initial position of the axle of the semitrailer $(x_{3,I}, y_{3,I})$ is uniformly sampled from two different 20 m $\times$ 15 m rectangles on each side of the goal location and the initial orientation of the semitrailer $\theta_{3,I}$ is randomly selected from six different initial orientations, as depicted in Figure 10.

In all simulations, the lattice planner was able to find an optimal path ($\gamma = 1$) to the goal within the allowed planning time of 60 s (max: 40 s). A statistical evaluation of the simulation results from one thousand Monte Carlo simulations are provided in Figure 11, where the planning time (Figure 11a), and the level of suboptimality ΔJ_D (Figure 11b) between the cost J_D for a specific γ and the optimal cost J_D^* for each planning experiment are plotted. In the box plots, the red central mark of each bar is the median, the bottom and top edges of the boxes indicate the 25th and 75th percentiles, respectively, and the whiskers extends to the most extreme data points. As can be seen in Figure 11a, the planning time is drastically increasing with decreasing γ. For most of

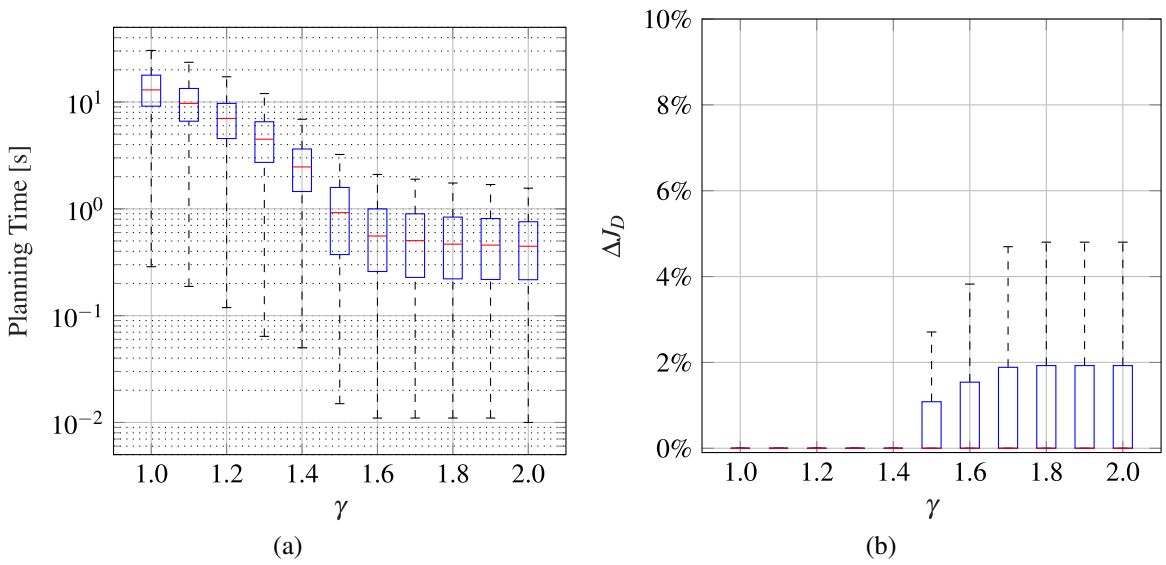


Figure 11: A statistical evaluation of the parking problem scenario (see Figure 10) over 1000 Monte Carlo simulations. In (a), a box plot of the planning time as a function of the heuristic inflation factor. In (b), a box plot of the level of suboptimality ΔJ_D as a function of the heuristic inflation factor.

the problems, a feasible solution to the goal with $\gamma = 2$ was found within 0.7 s, while a median planning time of 13 s was needed to find an optimal solution with $\gamma = 1$. In Figure 11b, the quality of the produced solution in terms of level of suboptimality ΔJ_D as a function of γ is displayed. For $\gamma \geq 1$, the provided theoretical guarantee is that the cost for a feasible solution J_D satisfies $J_D \leq \gamma J_D^*$, where J_D^* denotes the optimal cost. For all iterations of the ARA*, the median level of suboptimality is 0 % and the extreme values for large γ are about 5 %. For this scenario, it is clear that the guaranteed upper bound of γ-suboptimality is a conservative bound.

A loading/offloading site is used as the second planning scenario and the setup is illustrated in Figure 12. In this scenario, the lattice planner has to plan a path from a randomly selected initial state $z_I \in \mathbb{Z}_d$ to one of the six loading bays, or plan how to exit the site. In the Monte Carlo simulations, the initial position of the semitrailer $(x_{3,I}, y_{3,I})$ is uniformly sampled from a 25 m $\times$ 25 m square (see, Figure 12), and the initial orientation of the semitrailer $\theta_{3,I}$ is randomly selected from one of its sixteen discretized orientations, *i.e.*, $\theta_{3,I} \in \Theta$.

Also in this scenario, the lattice planner was always able to find an optimal path to the goal within the allowed planning time of 60 s (max: 27 s). A statistical evaluation of the simulation results from one thousand Monte Carlo simulations are presented in Figure 13. From the box plots in 13, it can be seen that the planning time is also in this scenario increasing with decreasing heuristic inflation factor γ. However, the median planning time to find an optimal solution with $\gamma = 1$ was only 0.84 s and most problems where solved within 3 s. The main reason for this improvement in terms of planning time compared to the parking scenario is because the precomputed HLUT yields a better estimation of the true cost-to-go in this less constrained environment. However, as can be seen in Figure 13b, the extreme values for the level of suboptimality ΔJ_D is about 43 % for large γ. Compared to the parking scenario, a heuristic inflation factor of $\gamma = 1.2$ is needed in this

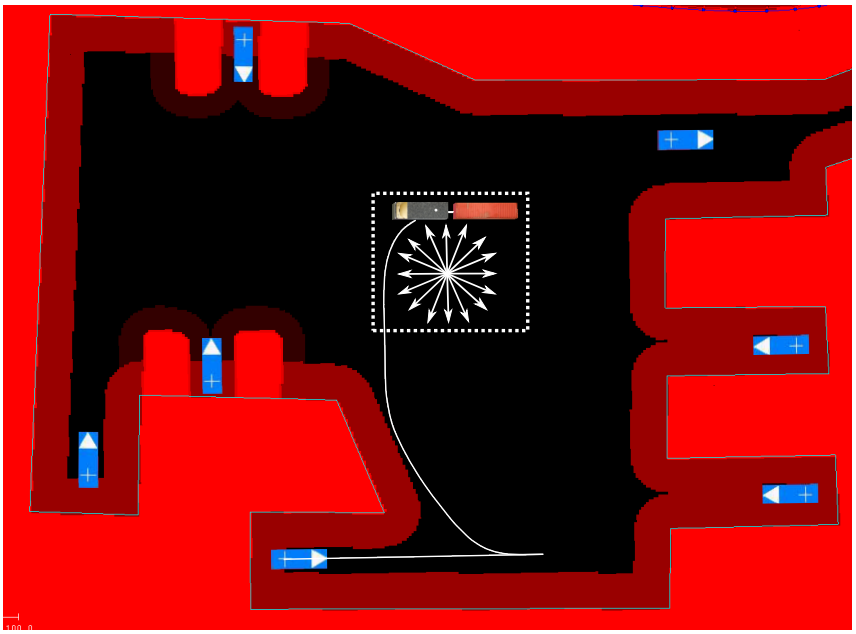


Figure 12: An overview of the loading/offloading site planning problem. The goal positions of the axle of the semitrailer $(x_{3,G}, y_{3,G})$ are illustrated by the white crosses inside the blue rectangles, where the white arrow specifies its goal orientation $\theta_{3,G}$. The initial position $(x_{3,I}, y_{3,I})$ is uniformly sampled within the two white-dotted rectangles, and the initial orientation $\theta_{3,I}$ are sampled from sixteen different initial orientations. The white path illustrates the planned path for the axle of the semitrailer for one case out of 1000 Monte Carlo simulations. The area occupied by obstacles is colored in red and the black area is free-space.

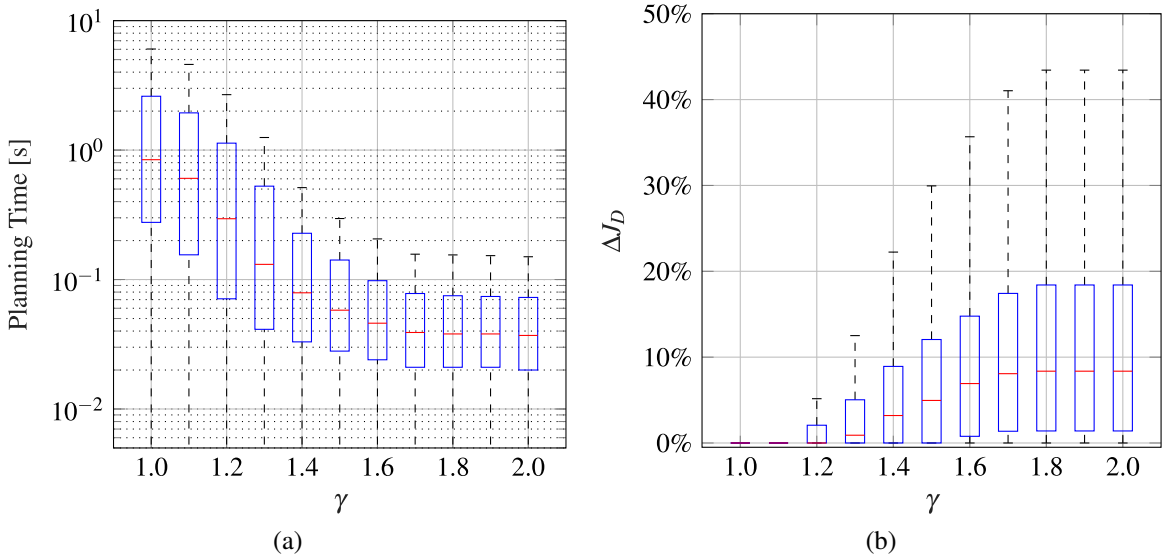


Figure 13: A statistical evaluation of the loading/offloading scenario (see Figure 12) over 1000 Monte Carlo simulations. In (a), a box plot of the planning time as a function of the heuristic inflation factor and. In (b), a box plot of the level of suboptimality ΔJ_D as a function of the heuristic inflation factor.

scenario to obtain a median level of suboptimality of 0 %. One reason for this greedy behavior in this scenario compared to parking scenario is that the there exist more alternative paths to the goal. This implies that the probability of finding a suboptimal path to the goal increases [36].

8.2.2 Path following of a figure-eight nominal path

Nominal paths of the shape of a figure-eight are used to evaluate the performance of the proposed path-following controller in backward and forward motion. These nominal paths are used as a benchmark since they expose the closed-loop system for a wide range of practically relevant maneuvers, *e.g.*, enter, exit and keep a narrow turn. To generate the figure-eight nominal path in forward motion, a list of waypoints of the same shape was first constructed manually. The nominal path was generated by simulating the model of the G2T with a car-like tractor (1) in forward motion with $v(t) = 1$ m/s, together with the pure pursuit controller in [25]. The path taken by the vehicle $(x_r(\tilde{s}), u_r(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$ was then stored and used as the nominal path in forward motion. The established symmetry result in Lemma 1 was then used to construct the figure-eight nominal path in backward motion.

In analogy to the design of the hybrid path-following controller (34), the OCP in (33) is solved with decay-rate $\varepsilon = 0.01$. In both cases, the optimal objective function to (33) is zero, which implies that the proposed hybrid path-following controller (33) is able the locally stabilize the path-following error model (25) around the origin while tracking the figure-eight nominal path in forward and backward motion, respectively. To confirm the theoretical analysis and to illustrate how the proposed path-following controller handles disturbance rejection, the closed-loop system is simulated with a perturbation in the initial path-following error states. For backward track-

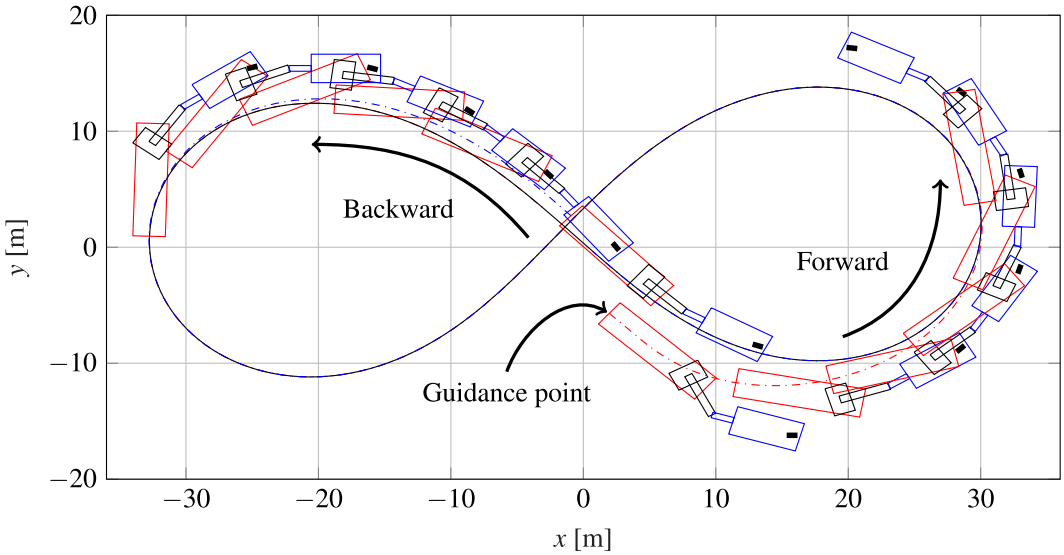
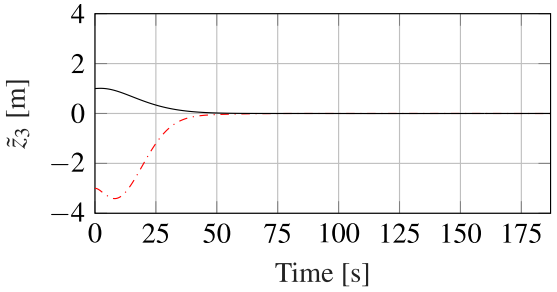
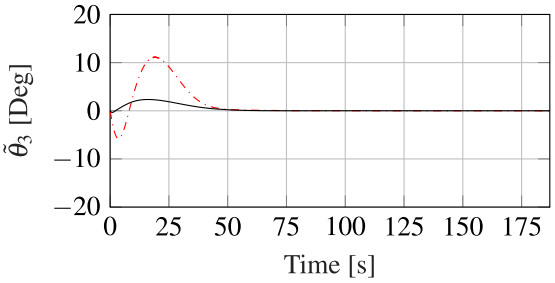


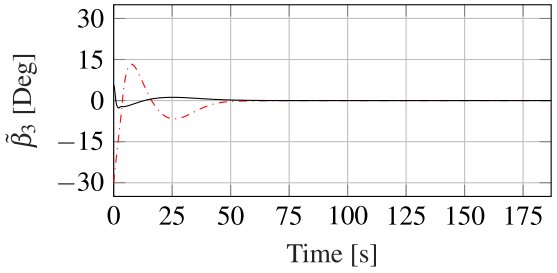
Figure 14: Simulation results of backward and forward tracking of the figure-eight nominal path, where the nominal path of the axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$ is the black solid line. The dashed-dotted blue (red) path is the path taken by the axle of the semitrailer $(x_3(\cdot), y_3(\cdot))$ during the backward (forward) path execution.



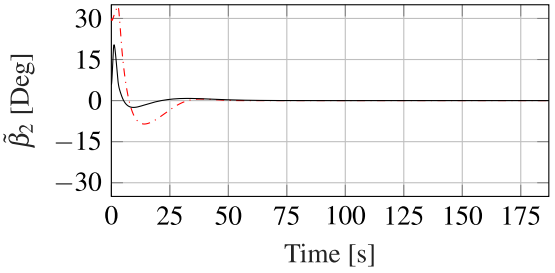
(a) Lateral error for the axle of the semitrailer.



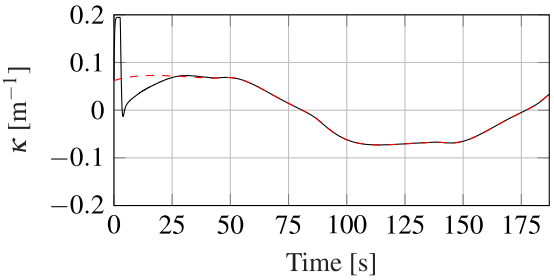
(b) Orientation error of the semitrailer.



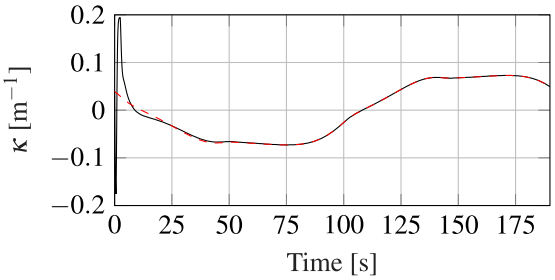
(c) The joint angle error between the semitrailer and the dolly.



(d) The joint angle error between the dolly and the tractor.



(e) Forward tracking: The controlled curvature $\kappa(t)$ (black line) and the nominal feedforward $\kappa_r(\tilde{s}(t))$ (red dashed line).



(f) Backward tracking: The controlled curvature $\kappa(t)$ (black line) and the nominal feedforward $\kappa_r(\tilde{s}(t))$ (red dashed line).

Figure 15: Simulation results of backward and forward tracking of the figure-eight nominal path in Figure 14. In (a)–(d), the forward (dashed-dotted red line) and backward (black line) path-following error states are plotted, and in (e)–(f), the controlled curvature of the tractor are displayed.

ing, the initial path-following error is chosen as $\tilde{x}_e(0) = [1\text{ m }\ 0\ \ 0.1\text{ rad }\ 0.1\text{ rad}]^T$ and for forward tracking it is chosen as $\tilde{x}_e(0) = [-3\text{ m }\ 0\ \ -\pi/6\text{ rad }\ \pi/6\text{ rad}]^T$. To perform realistic simulations, the steering angle of the car-like tractor is constrained according to the values in Table 1. The velocity of the car-like tractor is set to $v = v_r(s)$, *i.e.*, 1 m/s for forward tracking and $v = -1$ m/s for backward tracking.

The simulation results are provided in Figure 14–15. In Figure 14, the resulting paths taken by the axle of the semitrailer $(x_3(\cdot), y_3(\cdot))$ is plotted together with its nominal path $(x_{3,r}(\tilde{s}), y_{3,r}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$. The resulting trajectories for the path-following error states are presented in Figure 15a–15d. As theoretically verified, the path-following error states are converging towards the origin. The controlled curvature of the car-like tractor is plotted in Figure 15e and Figure 15f for the forward and backward tracking simulation, respectively. From these plots, it is clear that the feedback part in the path-following controller $\tilde{\kappa}(t)$ is responsible for disturbance rejection and the feedforward part $\kappa_r(\tilde{s})$ takes care of path-following.

8.3 Results from real-world experiments

The path planning and path-following control framework is finally evaluated in three different real-world experiments. First, the performance of the path-following controller and the nonlinear observer are evaluated by path-tracking of a precomputed figure-eight nominal path in backward motion. Then, two real-world experiments with the complete path planning and path-following control framework are presented. To validate the performance of the path-following controller and the nonlinear observer, a high-precision RTK-GPS[6] was mounted above the midpoint of the axle of the semitrailer. The authors recommend the supplemental video material[7] for real-world demonstration of the proposed framework.

8.3.1 Path following of a figure-eight nominal path

The figure-eight nominal path in backward motion that was used in the simulations, is also used here as the nominal path to evaluate the joint performance of the path-following controller and the nonlinear observer. The real-world path-following experiment is performed on an open gravel surface at Scania's test facility in Södertälje, Sweden. During the experiment, the longitudinal velocity of the rear axle of the car-like tractor was set to $v = -0.8$ m/s and results from one lap around the figure-eight nominal path are provided in Figure 16–18.

[6]The RTK-GPS is a Trimble SPS356 with a horizontal accuracy of about 0.1 m.
[7]youtu.be/IBA-8wom5zQ

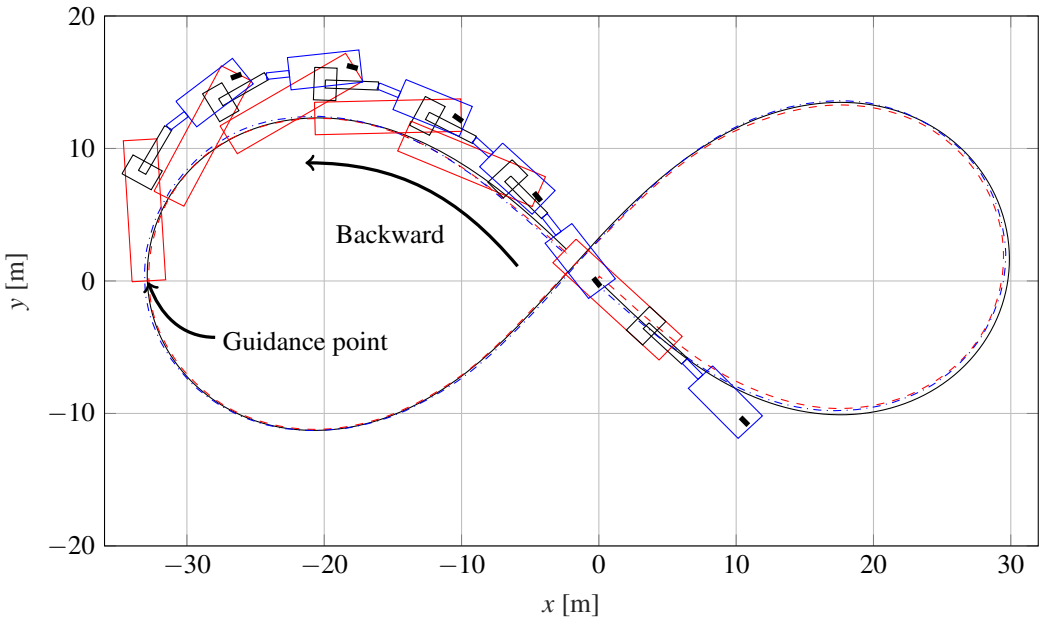
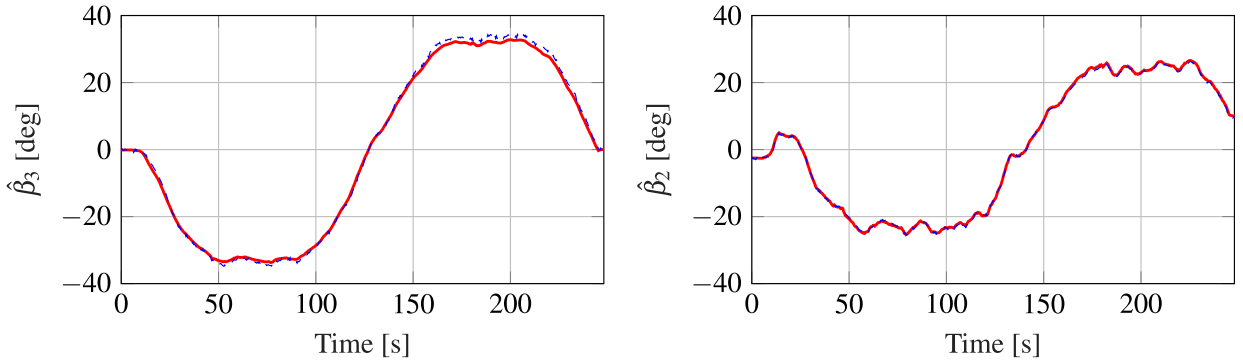


Figure 16: Results from real-world experiments of backward tracking of the figure-eight nominal path, where the nominal path of the axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$ is the black solid line. The dashed red line is the estimated path taken by the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ and the dashed-dotted blue line is the ground truth path $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ measured by the external RTK-GPS.

Figure 16 shows the nominal path for the position of the axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$, compared to its ground truth path $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ and its estimated path $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ around one lap of the figure-eight nominal path. A more detailed plot is provided in Figure 18, where all four estimated error states $\tilde{x}_e(t)$ are plotted. From these plots, we conclude that the path-following controller is able to keep its estimated lateral path-following error $\tilde{z}_3(\cdot)$ within ± 0.5 m (avg. 0.21 m), while at the same time keep the orientation and joint angle errors within acceptable error tolerances. As in the simulation trails, it can be seen from Figure 18b that the feedforward part $\kappa_r(s)$ of the path-following controller takes care of path-following and the feedback part $\tilde{\kappa} = K_{\text{rev}}\tilde{x}_e$ is responsible for disturbance rejection.

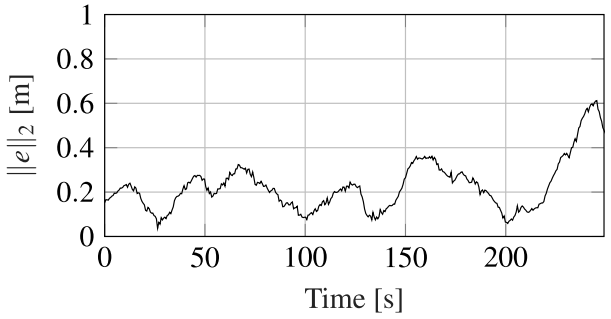
The performance of the nonlinear observer are presented in Figure 18a and Figure 17. In Figure 18a, the Euclidean norm of the difference between the estimated position of the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ and its ground truth $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ measured by the external RTK-GPS is presented. The maximum estimation error is 0.6 m and the average error is 0.23 m. It can be seen from Figure 18c that the estimated lateral path-following error for the axle of the semitrailer $\tilde{z}_3$ is increasing at the end of the maneuver. The reason for this is because the nonlinear observer is not able to track the absolute position of the axle of the semitrailer with high precision in this part of the maneuver. Likely causes to this estimation error are probably because of known asymmetries in the tractor's steering column [66] and the unavoidable lateral slip-effects of the wheels of the dolly and the semitrailer which are not captured by the kinematic model of the vehicle [70]. Note that the absolute position of the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ is estimated from GPS measurements of the car-like tractor's position and its orientation, propagated about 14 m through two hitch connections whose angles are estimated using only a LIDAR sensor on the car-like tractor. In Figure 17b and 17a, the estimated trajectories of the joint-angles, $\hat{\beta}_2$ and $\hat{\beta}_3$, are compared with their derived angles based on the outputs from the RANSAC algorithm, respectively. The maximum (avg.) errors in the residuals $\hat{\beta}_2 - \beta_2$ and $\hat{\beta}_3 - \beta_3$, are 0.83° (avg. 0.27°) and 2.18° (avg. 0.8°), respectively.



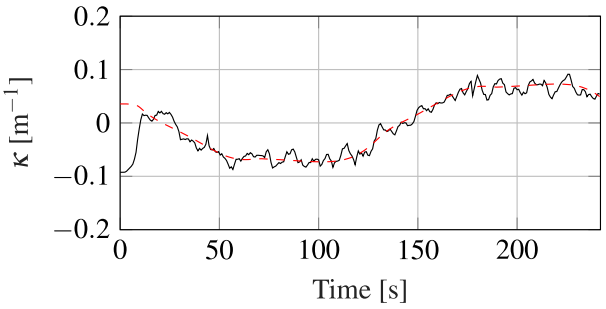
(a) Estimated joint angle between the semitrailer and the dolly.

(b) Estimated joint angle between the dolly and the tractor.

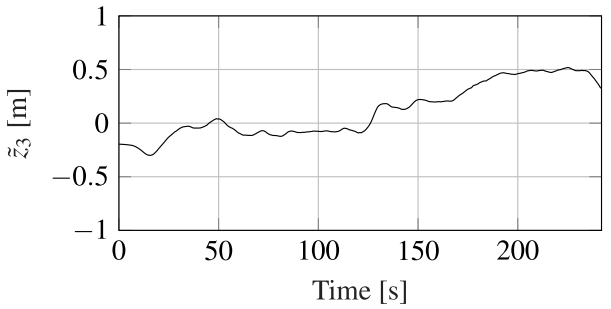
Figure 17: Results from real-world experiments of backward tracking of the figure-eight nominal path in Figure 16. The red solid lines are the estimated joint angles, $\hat{\beta}_2$ and $\hat{\beta}_3$, and the dashed blue lines are the computed joint angles, β_2 and β_3, from the RANSAC measurements.



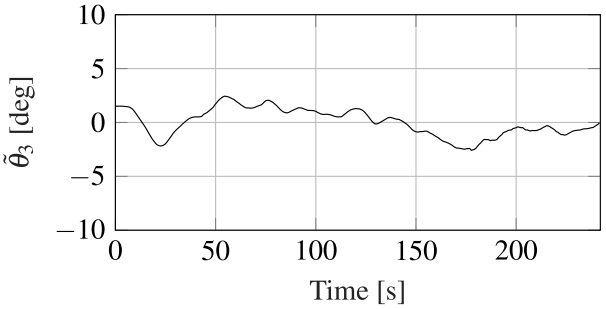
(a) The norm of the position estimation error $\|e(t)\|$ for the axle of the semitrailer.



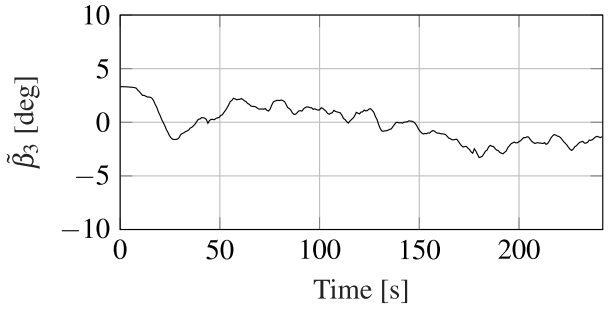
(b) The controlled curvature $\kappa(t)$ (black line) and the nominal feed-forward $\kappa_r(\tilde{s}(t))$ (red dashed line).



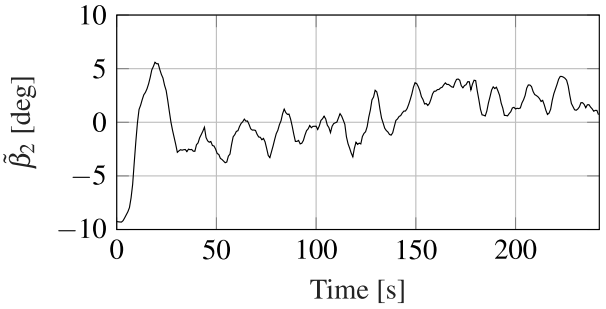
(c) Estimated lateral error for the axle of the semitrailer.



(d) Estimated orientation error of the semitrailer.



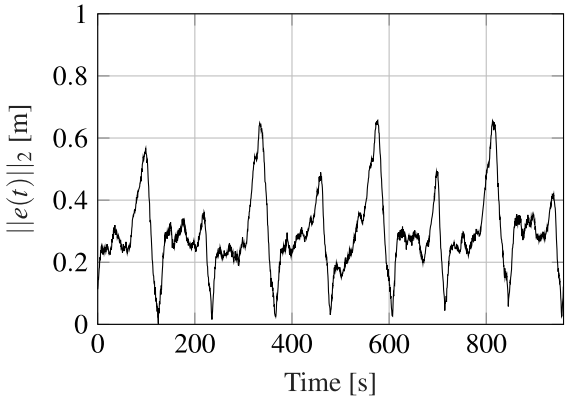
(e) Estimated joint angle error between the semitrailer and the dolly.



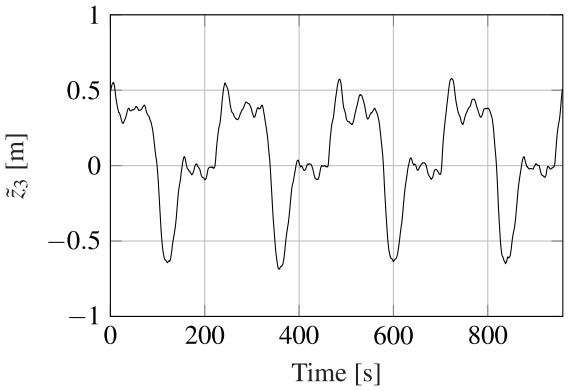
(f) Estimated joint angle error between the dolly and the tractor.

Figure 18: Results from real-world experiments of backward tracking of the figure-eight nominal path in Figure 16. In (a), the norm of the position estimation error is plotted and in (b), the controlled curvature of the tractor is plotted. In (c)–(f), the estimated path-following error states are plotted.

To illustrate the repeatability of the system, the same figure-eight nominal path was executed multiple times. This experiment was performed at another occasion on rougher ground surface conditions compared to the first experiment. The resulting estimated lateral control error $\tilde{z}_3$ and the Euclidean norm of the position error $||e(t)||_2$ over four consecutive laps are presented in Figure 19. As can be seen, both errors are bounded and have a periodic behavior of approximately 250 seconds, *i.e.*, one lap time around the figure-eight nominal path.



(a) The norm of the position estimation error of the axle of the semitrailer.



(b) Estimated lateral path-following error for the axle of the semitrailer.

Figure 19: Results from real-world experiments of backward tracking of the figure-eight nominal path in Figure 16 over four consecutive laps. In (a), the norm of the position estimation error and in (b), the estimated lateral path-following error z_3. This experiment was performed under rougher ground surface conditions compared to the first experiment.

8.3.2 Two-point turn

In this section, the complete path planning and path-following control framework is evaluated in a real-world experiment. The G2T with a car-like tractor is operating on dry asphalt on a relatively narrow road at Scania's test facility. The scenario setup is shown in Figure 20 and the objective is to change the orientation of the semitrailer with 180° while at the same time move the vehicle about 40 m longitudinally. Similarly to the parking planning problem in Figure 10, the precomputed HLUT may underestimate the cost-to-go due to the confined environment. Despite this, the lattice planner found an optimal solution ($\gamma = 1$) in 0.6 s and the ARA* search expanded 720 vertices. As a comparison, a planning time of only 29 ms was needed for this example to find a motion plan with $\gamma = 1.3$, *i.e.*, a solution that is guaranteed to be less than 30 % worse than the optimal one.

In Figure 20, the white path illustrates the planned path for position of the semitrailer's axle $(x_{3,r}(\tilde{s}), y_{3,r}(\tilde{s}))$, $\tilde{s} \in [0, \tilde{s}_G]$. As can be seen, the solution is mainly composed of a 90°-turn in backward motion followed by a 90°-turn in forward motion. The execution of the planned two-point turn maneuver is visualized in Figure 21, where the estimated path taken by the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ is plotted together with its ground truth path $(x_{3,GT}(\cdot), \hat{y}_{3,GT}(\cdot))$ measured by the external RTK-GPS.

More detailed plots are provided in Figure 22, where the vehicle is changing from backward to forward motion at $t = 60$ s. In Figure 22a, the norm of the position estimation error for the axle of the semitrailer is plotted. In this scenario, the maximum position estimation error was 0.35 m and the mean absolute error was 0.21 m. The path-following error states are plotted in Figure 22c–22f. From these plots, it can be seen that the estimated lateral control error $\tilde{z}_3(t)$, which is plotted in Figure 22c, has a maximum absolute error of 0.37 m and a mean absolute error of 0.12 m. Except from initial transients, the joint angle errors, $\tilde{\beta}_3$ and $\tilde{\beta}_2$, attain their peak values when the vehicle is changing from backward to forward motion at $t = 60$ s. There are multiple possible explanations to

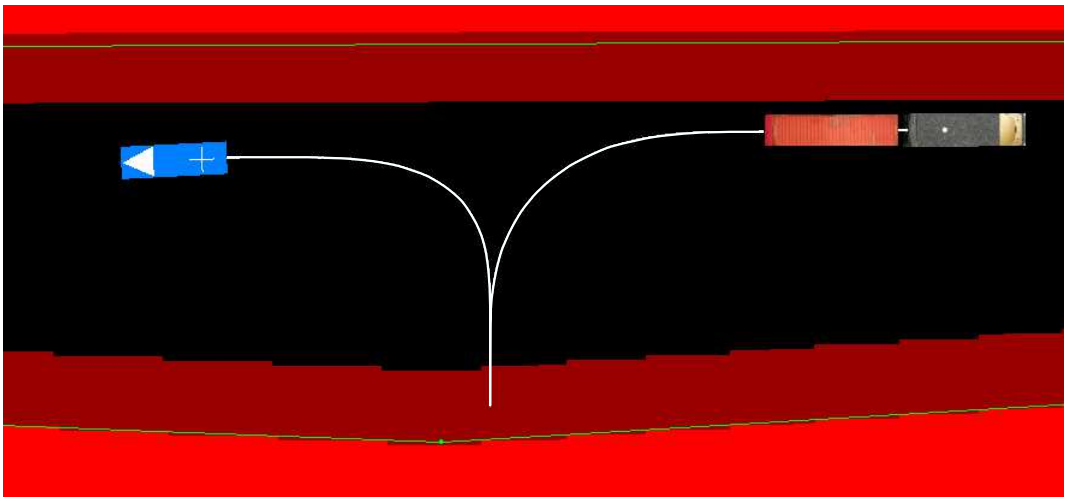


Figure 20: Illustration of the planned two-point turn maneuver. The goal position of the semitrailer is illustrated by the white cross inside the blue rectangle, where the white arrow specifies its goal orientation. The white path is the planned path for axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$.

this phenomenon. Except from possible estimation errors in the joint angles, one possibility is that lateral dynamical effects arise when the vehicle is exiting the tight 90°-turn in backward motion. However, the path-following controller is still able to compensate for these disturbances, as can be seen for $t \in [50, 80]$ s in Figure 22b.

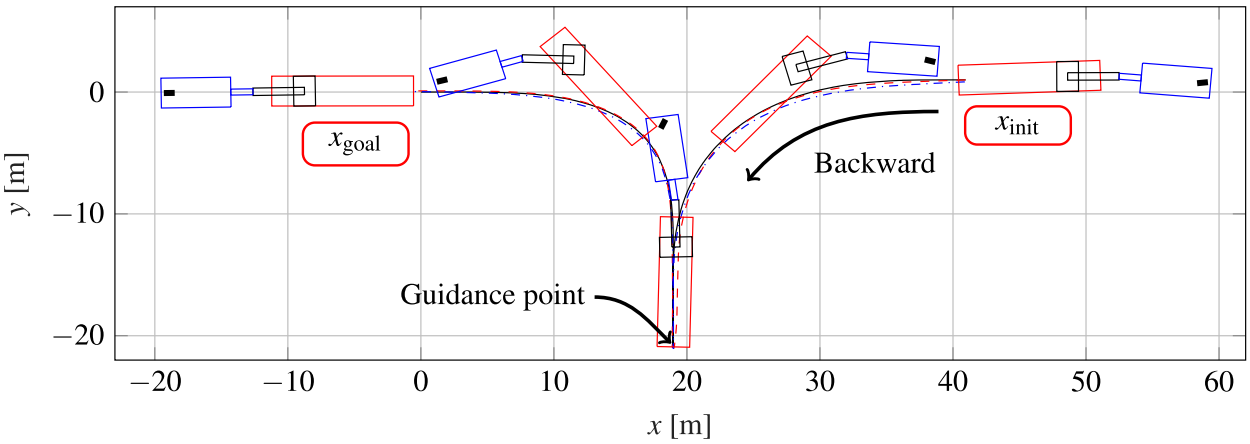
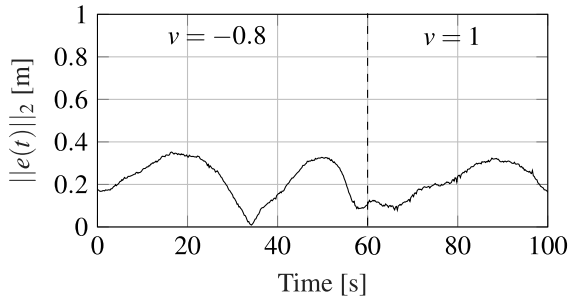
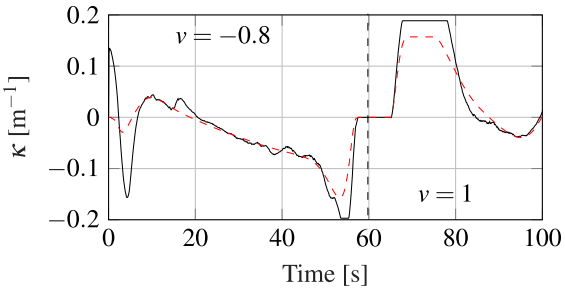


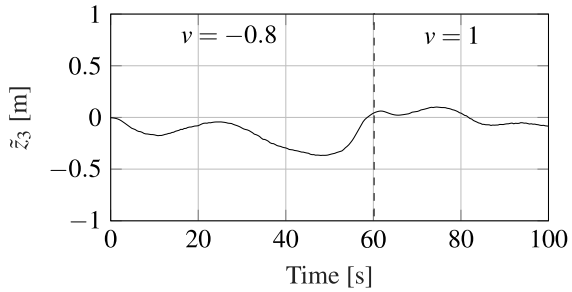
Figure 21: Results from real-world experiments while executing the planned two-point turn maneuver. The black line is the planned path for the axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$. The dashed red line is the estimated path taken by the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ and the dashed-dotted blue line is the ground truth path $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ measured by the external RTK-GPS.



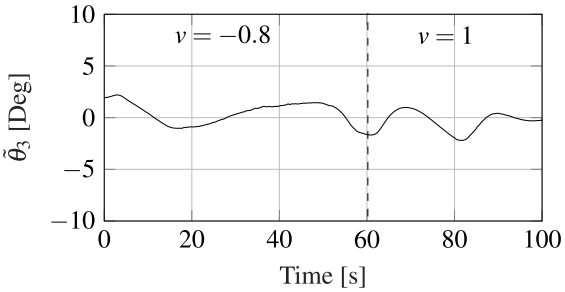
(a) The norm of the position estimation error of the axle of the semitrailer.



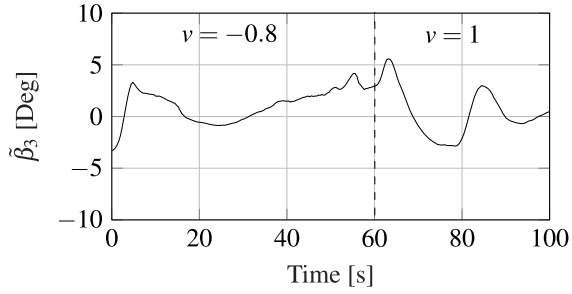
(b) The controlled curvature $\kappa(t)$ (black line) and the nominal feed-forward $\kappa_r(\tilde{s}(t))$ (red dashed line).



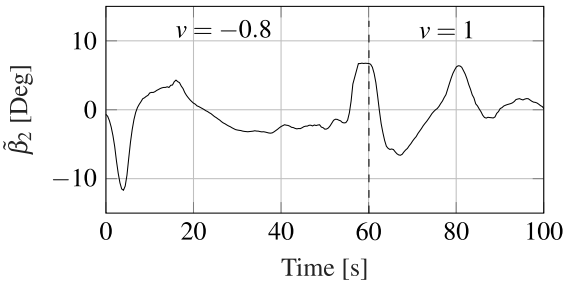
(c) Estimated lateral error for the axle of the semitrailer.



(d) Estimated orientation error of the semitrailer.



(e) Estimated joint angle error between the semitrailer and the dolly.



(f) Estimated joint angle error between the dolly and the tractor.

Figure 22: The absolute position estimation error (a), the controlled curvature of the tractor (b) and the estimated path-following error states (c)–(f) during the execution of the planned two-point turn maneuver in Figure 21.

8.3.3 T-turn

The final real-world experiment is an open area planning problem on the same gravel surface as the execution of the figure-eight nominal path was performed. The open area planning problem is shown in Figure 23, where the G2T with a car-like tractor is intended to change the orientation of the semitrailer with 180° together with a small lateral and longitudinal movement. In this scenario, the planning time for finding an optimal solution ($\gamma = 1$) was only 38 ms and the ARA* search expanded only 22 vertices. The reason why such a small amount of vertex expansions was needed is because the precomputed HLUT perfectly estimates the cost-to-go in free-space scenarios.

Figure 23 shows the nominal path for the axle of the semitrailer $(x_{3,r}(\cdot), y_{3,r}(\cdot))$, which essentially is composed by two 90°-turns in forward motion together with a parallel maneuver in backward motion. In this example, the impact of penalizing complex backward motions is clear,

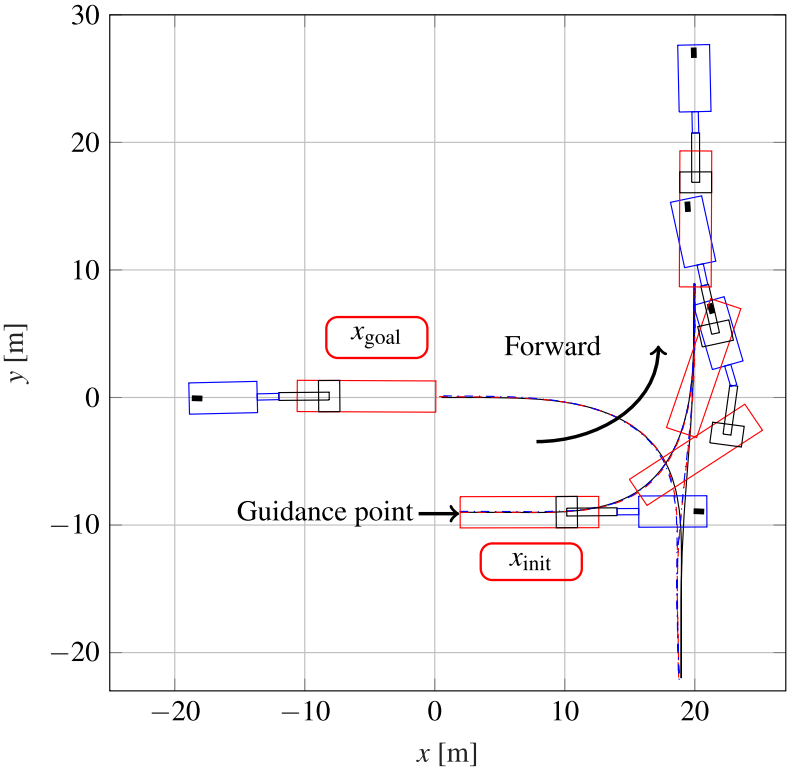


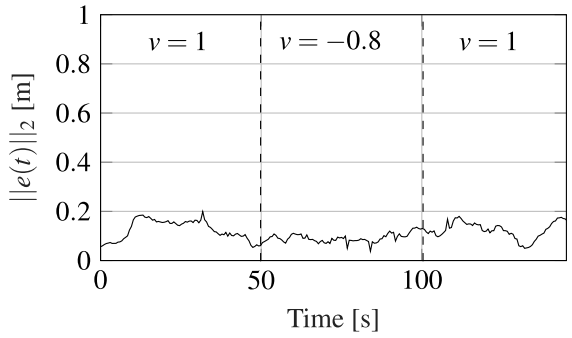
Figure 23: Results from real-world experiments while executing the planned T-turn maneuver $(x_{3,r}(\cdot), y_{3,r}(\cdot))$ (black line). The dashed red line is the estimated path taken by the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ and the dashed-dotted blue line is the ground truth path $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ measured by the external RTK-GPS.

the advanced maneuvers are performed while driving forwards if allowed by the surrounding environment. In the same plot, the estimated path taken by the axle of the semitrailer $(\hat{x}_3(\cdot), \hat{y}_3(\cdot))$ and its ground truth path $(x_{3,GT}(\cdot), y_{3,GT}(\cdot))$ obtained from the external RTK-GPS are presented.

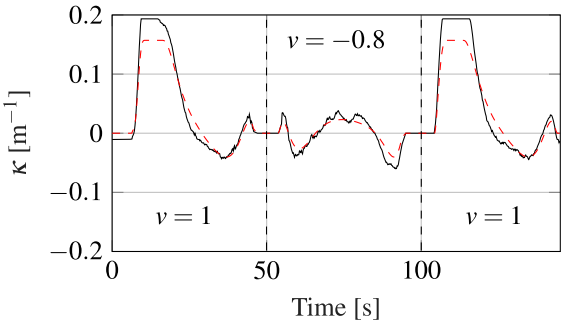
More detailed plots are provided in Figure 24, where the vehicle is changing from forward to backward motion at $t = 50$ s and from backward to forward motion at $t = 100$ s. In Figure 24a, the norm of the position estimation error for the axle of the semitrailer is plotted. In this experiment, the maximum estimation error was 0.20 m and the mean absolute error was 0.12 m. The path-following error states are presented in Figure 24c–24f. In Figure 24c, the estimated lateral control error $\tilde{z}_3$ is plotted, where the maximum absolute error was 0.31 m and the mean absolute error was 0.11 m. In this experiment, both joint angle errors, $\tilde{\beta}_2$ and $\tilde{\beta}_3$, as well as the orientation error of the semitrailer $\tilde{\theta}_3$, lie within $\pm 5^\circ$ for the majority of the path execution. The controlled curvature κ of the car-like tractor is plotted in Figure 24b. Similar to the two-point turn experiment, it can be seen that for large nominal curvature values κ_r, the feedback part in the path-following controller is compensating for lateral dynamical effects that are not captured by the kinematic vehicle model.

8.4 Discussion of lessons learned

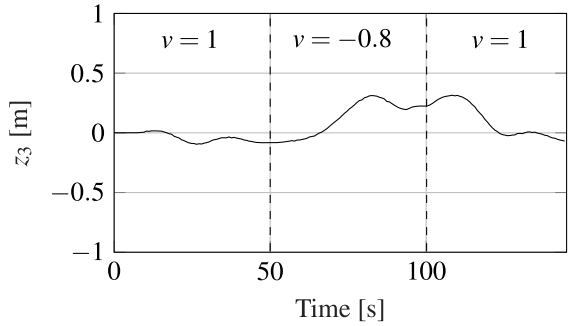
The proposed path planning and path-following control framework has been successfully deployed on a full-scale test platform. Since the full system is built upon several modules, an important key



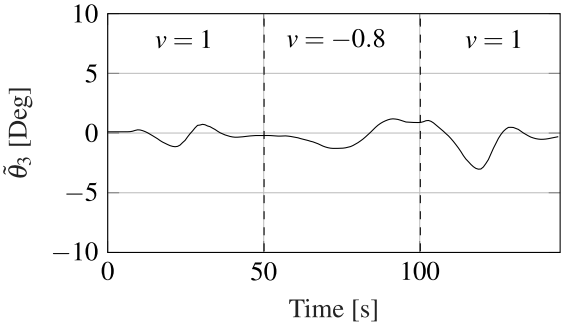
(a) The norm of the position estimation error of the axle of the semitrailer.



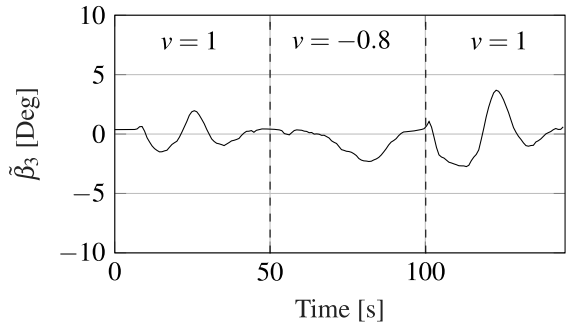
(b) The controlled curvature $\kappa(t)$ (black line) and the nominal feed-forward $\kappa_r(\tilde{s}(t))$ (red dashed line).



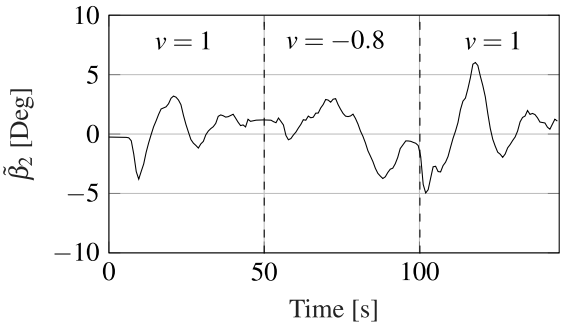
(c) Estimated lateral error for the axle of the semitrailer.



(d) Estimated orientation error of the semitrailer.



(e) Estimated joint angle error between the semitrailer and the dolly.



(f) Estimated joint angle error between the dolly and the tractor.

Figure 24: Absolute position error (a), controlled curvature (b) and estimated path-following error (c)–(f), during the execution of the T-point turn maneuver in Figure 23.

to fast deployment was to separately test and evaluate each module in simulations. By performing extensive simulations during realistic conditions, the functionality of each module as well as the communication between them could be verified before real-world experiments was performed.

As illustrated in the real-world experiments, the performance of the system in terms of path-following capability is highly dependent on accurate estimates of the vehicle's state. The tuning and calibration of the nonlinear observer was also the most time-consuming part of the process when the step from simulations to real-world experiments was taken. The main difficulty was to verify that the nonlinear observer was capable of tracking the true trajectory of the position and orientation of the axle of the semitrailer as well as the two joint angles, despite that their true state trajectories were partially or completely unknown. To resolve this, data was collected from manual

tests with the vehicle. This data was then used offline to tune the covariance matrices in the EKF and to calibrate the position and orientation of the rear view LIDAR sensor. For the calibration of the LIDAR sensor, an accurately calibrated yaw angle was found to be very important.

The deployment of the hybrid path-following controller and the lattice planner only required minor tuning compared to simulations. For the design of the path-following controller, the penalty for the lateral path-following error $\tilde{z}_3$ was found to be the most important tuning parameter which had the largest effects on the region of attraction for the closed-loop system. However, since the lattice planner is planning from the vehicle's current state, the initial error in $\tilde{z}_3$ will be small and a rather aggressive tuning of the path-following controller was possible.

9 Conclusions and future work

A path planning and path-following control framework for a G2T with a car-like tractor is presented. The framework is targeting low-speed maneuvers in unstructured environments and has been successfully deployed on a full-scale test vehicle. A lattice-based path planner is used to generate kinematically feasible and optimal nominal paths in all vehicle states and controls, and the ARA* graph-search algorithm is used during online planning. To follow the planned path, a hybrid path-following controller is developed to stabilize the path-following error states of the vehicle and a nonlinear observer is proposed that is only utilizing information from sensors that are mounted on the car-like tractor, which makes the solution compatible with basically all of today's commercially available semitrailers that have a rectangular body. The framework is first evaluated in simulations and then in three different real-world experiments and results in terms of closed-loop performance and real-time planning capabilities are presented. In the experiments, the system shows that it is able to consistently solve challenging planning problems and that it is able to execute the resulting motion plans, despite no sensors on the dolly or semitrailer.

A drawback with the lattice-based path planning framework is the need of manually selecting the connectivity in the state lattice. Even though this procedure is done offline, it is both nontrivial and time-consuming. Future work includes automating this procedure to make the algorithm more user-friendly and compatible with different vehicle parameters. Moreover, the discretization of the vehicle's state-space, restricts the set of possible initial states the lattice planner can plan from and desired goal states that can be reached. As mentioned in the text, this is a general problem with sampling-based motion planning algorithms, which could for example be alleviated by the use of numerical optimal control as a post-processing step [8, 19, 34, 51]. Thus, future work includes exploiting the structure of the path planning problem and develop an efficient and numerically stable online smoothing framework by, *e.g.*, the use of numerical optimal control as a backbone.

The main benefits of the proposed hybrid path-following controller include that it is computationally efficient and that it is tailored to follow nominal paths composed of full state and control information. A limitation of the proposed controller is that it does not explicitly handle constraints on states and controls, but since the lattice planner plans from the vehicle current location and provides a nominal path in all states and controls which satisfies these constraints, they can be

neglected locally around the nominal path. Regarding scalability, as long as a nominal path is provided, it is possible to generalize the proposed path-following controller to handle more complex paths and tractor-trailer system with additional trailers. The region of attraction for the closed-loop system will however become smaller with increasing path complexity as well as increasing number of trailers.

At some parts of the figure-eight path-following experiments, the magnitude of the estimation error for the position of the axle of the semitrailer had a size which potentially could cause problems in narrow environments. Hence, reasonable future work also includes exploring alternative onboard sensors as well as using external sensors that can be placed at strategic locations where high-accuracy path tracking is critical, *e.g.*, when reversing to a loading bay.

Appendix A

Lemma 1. *Denote $z(s)$, $s \in [0, s_G]$, as the solution to* (5) *that satisfies $|\alpha(\cdot)| \leq \alpha_{max} < \pi/2$, when the control signal $u_p(s) \in \mathbb{U}_p$, $s \in [0, s_G]$ is applied from the initial state $z(0)$ which ends at the final state $z(s_G)$. Moreover, denote $\bar{z}(\bar{s})$, $\bar{s} \in [0, s_G]$ as the distance-reversed solution to* (5) *when the distance-reversed control signal*

$$\bar{u}_p(\bar{s}) = \begin{bmatrix} -v(s_G - \bar{s}) & u_\omega(s_G - \bar{s}) \end{bmatrix}^T, \quad \bar{s} \in [0, s_G] \tag{57}$$

is applied from the initial state $\bar{z}(0) = [x(s_G)^T \;\; \alpha(s_G) \;\; -\omega(s_G)]^T$. Then, $z(s)$, $s \in [0, s_G]$ and $\bar{z}(\bar{s})$, $\bar{s} \in [0, s_G]$ are unique and they are related according to

$$\bar{z}(\bar{s}) = \begin{bmatrix} x(s_G - \bar{s})^T & \alpha(s_G - \bar{s}) & -\omega(s_G - \bar{s}) \end{bmatrix}^T, \quad \bar{s} \in [0, s_G]. \tag{58}$$

In particular, the final state is $\bar{z}(s_G) = [x(0)^T \;\; \alpha(0) \;\; -\omega(0)]^T$.

Proof. Given a piecewise continuous $u_p(s) = [v(s) \;\; u_\omega(s)]^T \in \mathbb{U}_p$, $s \in [0, s_G]$, define

$$\tilde{f}_z(s, z) \triangleq f_z(z, u_p(s)) = \begin{bmatrix} v(s) f(x, \tan\alpha / L_1) \\ \omega \\ u_\omega(s) \end{bmatrix}.$$

Direct calculations verify that $f(x, \tan\alpha/L_1)$ in (1) is continuous and continuously differentiable with respect to z for all $z \in \mathbb{Z}_o \in \{z \in \mathbb{R}^7 \mid |\alpha| < \pi/2\}$. This is true since $f(x, \tan\alpha/L_1)$ is composed of sums and products of trigonometric functions which are continuous and continuously differentiable with respect to z for all $z \in \mathbb{Z}_o$. Furthermore, $\tilde{f}_z(s, z)$ is piecewise continuous in s since $f_z(z, u_p(s))$ is continuous in u_p for all $z \in \mathbb{Z}_o$. Therefore, on any interval $[a, b] \subset [0, s_G]$ where $u_p(\cdot)$ is continuous, $\tilde{f}_z(s, z)$ and $[\partial \tilde{f}_z(s, z)/\partial z]$ are continuous on $[a, b] \times Z_o$. Then, from Lemma 3.2 in [31], the vector field $\tilde{f}_z(s, z)$ is piecewise continuous in s and locally Lipschitz in z, for all $s \in [0, s_G]$ and all $z \in \mathbb{Z}_o$. Define $\mathbb{Z}_c = \{z \in \mathbb{R}^7 \mid |\alpha| \leq \alpha_{\max}\}$ which is a compact subset of $\mathbb{Z}_o$. Then, from Theorem 3.3 in [31], every solution $z(s)$, $s \in [0, s_G]$ that lies entirely in $\mathbb{Z}_c$ is unique for all $s \in [0, s_G]$.

Now, let $z(s)$, $s \in [0, s_G]$, be the unique solution to (5) assumed to lie entirely in $\mathbb{Z}_c$, when the control signal $u_p(s) \in \mathbb{U}_p$, $s \in [0, s_G]$ is applied from the initial state $z(0)$ which ends at the final state $z(s_G)$. Introduce $(\bar{z}(\bar{s}), \bar{u}_p(\bar{s}))$, $\bar{s} \in [0, s_G]$ with

$$\bar{z}(\bar{s}) = \begin{bmatrix} x(s_G - \bar{s})^T & \alpha(s_G - \bar{s}) & -\omega(s_G - \bar{s}) \end{bmatrix}^T, \quad \bar{s} \in [0, s_G], \tag{59a}$$

$$\bar{u}_p(\bar{s}) = \begin{bmatrix} -v(s_G - \bar{s}) & u_\omega(s_G - \bar{s}) \end{bmatrix}^T, \qquad \bar{s} \in [0, s_G]. \tag{59b}$$

Since

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\bar{s}}\bar{z}(\bar{s}) &= \frac{\mathrm{d}}{\mathrm{d}\bar{s}} \begin{bmatrix} x(s_G - \bar{s}) \\ \alpha(s_G - \bar{s}) \\ -\omega(s_G - \bar{s}) \end{bmatrix} = \{s = s_G - \bar{s}\} = \frac{\mathrm{d}}{\mathrm{d}s} \begin{bmatrix} x(s) \\ \alpha(s) \\ -\omega(s) \end{bmatrix} \Bigg|_{s = s_G - \bar{s}} \underbrace{\frac{\mathrm{d}s}{\mathrm{d}\bar{s}}}_{=-1} \\ &= \begin{bmatrix} -v(s) f(x(s), \tan\alpha(s)/L_1) \\ -\omega(s) \\ u_\omega(s) \end{bmatrix} \Bigg|_{s = s_G - \bar{s}} = \begin{bmatrix} \bar{v}(\bar{s}) f(\bar{x}(\bar{s}), \tan\bar{\alpha}(\bar{s})/L_1) \\ \bar{\omega}(\bar{s}) \\ \bar{u}_\omega(\bar{s}) \end{bmatrix} \\ &= f_z(\bar{z}(\bar{s}), \bar{u}_p(\bar{s})), \ \bar{s} \in [0, s_G], \end{aligned}$$

(59) is also a solution to (5) from the initial state $\bar{z}(0) = [x(s_G)^T \ \ \alpha(s_G) \ \ -\omega(s_G)]^T$. Finally, since the solution $\bar{z}(\bar{s})$, $\bar{s} \in [0, s_G]$ also lies entirely in $\mathbb{Z}_c$, this solution is also unique.

Proof of Theorem 1

Let $(z(s), u_p(s))$, $s \in [0, s_G]$ denote a feasible solution to the optimal path planning problem (7) with objective functional value J. Now, consider the reverse optimal path planning problem

$$\underset{\bar{u}_p(\cdot),\, \bar{s}_G}{\text{minimize}} \qquad \bar{J} = \int_0^{\bar{s}_G} L(\bar{x}(\bar{s}), \bar{\alpha}(\bar{s}), \bar{\omega}(\bar{s}), \bar{u}_\omega(\bar{s}))\, \mathrm{d}\bar{s} \tag{60a}$$

$$\text{subject to} \qquad \frac{\mathrm{d}\bar{z}}{\mathrm{d}\bar{s}} = f_z(\bar{z}(\bar{s}), \bar{u}_p(\bar{s})), \tag{60b}$$

$$\bar{z}(0) = z_G, \quad \bar{z}(\bar{s}_G) = z_I, \tag{60c}$$

$$\bar{z}(\bar{s}) \in \mathbb{Z}_{\text{free}}, \quad \bar{u}_p(\bar{s}) \in \mathbb{U}_p. \tag{60d}$$

Then, using the invertible transformations (57)–(58):

$$\bar{z}(\bar{s}) = \begin{bmatrix} x(s_G - \bar{s})^T & \alpha(s_G - \bar{s}) & -\omega(s_G - \bar{s}) \end{bmatrix}^T, \quad \bar{s} \in [0, s_G], \tag{61a}$$

$$\bar{u}_p(\bar{s}) = \begin{bmatrix} -v(s_G - \bar{s}) & u_\omega(s_G - \bar{s}) \end{bmatrix}^T, \qquad \bar{s} \in [0, s_G] \tag{61b}$$

and $\bar{s}_G = s_G$, the reverse optimal path planning problem (60) becomes

$$\underset{u_p(\cdot),\, s_G}{\text{minimize}} \quad \bar{J} = \int_0^{s_G} L(x(s_G-\bar{s}), \alpha(s_G-\bar{s}), -\omega(s_G-\bar{s}), u_\omega(s_G-\bar{s}))\,\mathrm{d}\bar{s} \tag{62a}$$

$$\text{subject to} \quad \frac{\mathrm{d}}{\mathrm{d}\bar{s}} \begin{bmatrix} x(s_G-\bar{s}) \\ \alpha(s_G-\bar{s}) \\ -\omega(s_G-\bar{s}) \end{bmatrix} = \begin{bmatrix} -v(s_G-\bar{s}) f(x(s_G-\bar{s}), \tan\alpha(s_G-\bar{s})/L_1) \\ -\omega(s_G-\bar{s}) \\ u_\omega(s_G-\bar{s}) \end{bmatrix}, \tag{62b}$$

$$\begin{bmatrix} x(s_G)^T & \alpha(s_G) & -\omega(s_G) \end{bmatrix}^T = z_G, \tag{62c}$$

$$\begin{bmatrix} x(0)^T & \alpha(0) & -\omega(0) \end{bmatrix}^T = z_I, \tag{62d}$$

$$\begin{bmatrix} x(s_G-\bar{s})^T & \alpha(s_G-\bar{s}) & -\omega(s_G-\bar{s}) \end{bmatrix}^T \in \mathbb{Z}_{\text{free}}, \tag{62e}$$

$$\begin{bmatrix} -v(s_G-\bar{s}) & u_\omega(s_G-\bar{s}) \end{bmatrix}^T \in \mathbb{U}_p. \tag{62f}$$

Let $s = s_G - \bar{s}$, $s \in [0, s_G]$. It then follows from Lemma 1 that (62b) simplifies to $\frac{\mathrm{d}z}{\mathrm{d}s} = f_z(z(s), u_p(s))$. From Assumption 1 it follows that

$$\begin{aligned}
\bar{J} &= \int_0^{s_G} L(x(s_G-\bar{s}), \alpha(s_G-\bar{s}), -\omega(s_G-\bar{s}), u_\omega(s_G-\bar{s}))\,\mathrm{d}\bar{s} = \{s = s_G - \bar{s}\} \\
&= -\int_{s_G}^{0} L(x(s), \alpha(s), -\omega(s), u_\omega(s))\,\mathrm{d}s \\
&= \int_0^{s_G} L(x(s), \alpha(s), -\omega(s), u_\omega(s))\,\mathrm{d}s = \{L(x,\alpha,-\omega,u_\omega) = L(x,\alpha,\omega,u_\omega)\} \\
&= \int_0^{s_G} L(x(s), \alpha(s), \omega(s), u_\omega(s))\,\mathrm{d}s = J.
\end{aligned} \tag{63}$$

Hence, the OCP in (62) can equivalently be written as

$$\underset{u_p(\cdot),\, s_G}{\text{minimize}} \quad J = \int_0^{s_G} L(x(s), \alpha(s), \omega(s), u_\omega(s))\,\mathrm{d}s \tag{64a}$$

$$\text{subject to} \quad \frac{\mathrm{d}z}{\mathrm{d}s} = f_z(z(s), u_p(s)), \tag{64b}$$

$$\begin{bmatrix} x(s_G)^T & \alpha(s_G) & -\omega(s_G) \end{bmatrix}^T = z_G, \tag{64c}$$

$$\begin{bmatrix} x(0)^T & \alpha(0) & -\omega(0) \end{bmatrix}^T = z_I, \tag{64d}$$

$$\begin{bmatrix} x(s)^T & \alpha(s) & -\omega(s) \end{bmatrix}^T \in \mathbb{Z}_{\text{free}}, \tag{64e}$$

$$\begin{bmatrix} -v(s) & u_\omega(s) \end{bmatrix}^T \in \mathbb{U}_p. \tag{64f}$$

From the symmetry of the set $\mathbb{U}_p = \{-1, 1\} \times [-u_{\omega,\max}, u_{\omega,\max}]$, (64f) is equivalent to $u_p(s) = [v(s)\ \ u_\omega(s)]^T \in \mathbb{U}_p$. From Assumption 2, the constraint in (64e) is equivalent to the constraint $z(s) = [x(s)^T\ \ \alpha(s)\ \ \omega(s)]^T \in \mathbb{Z}_{\text{free}}$ holds. Finally, since $z_I = [x_I^T\ \ \alpha_I\ \ 0]^T$ and $z_G = [x_G^T\ \ \alpha_G\ \ 0]^T$, the

problem in (64) can equivalently be written as

$$\underset{u_p(\cdot),\, s_G}{\text{minimize}} \quad J = \int_0^{s_G} L(x(s), \alpha(s), \omega(s), u_\omega(s))\, \mathrm{d}s \tag{65a}$$

$$\text{subject to} \quad \frac{\mathrm{d}z}{\mathrm{d}s} = f_z(z(s), u_p(s)), \tag{65b}$$

$$z(0) = z_I, \quad z(s_G) = z_G, \tag{65c}$$

$$z(s) \in \mathbb{Z}_{\text{free}}, \tag{65d}$$

$$u_p(s) \in \mathbb{U}_p, \tag{65e}$$

which is identical to the optimal path planning problem in (7). Hence, the OCPs in (7) and (9) are equivalent [14] and the invertible transformation relating the solutions to the two equivalent problems is given by (61) and $\bar{s}_G = s_G$. Hence, if an optimal solution to one of the problems is known, an optimal solution to the other one can immediately be derived using (61) and $\bar{s}_G = s_G$. Or more practically, given an optimal solution in one direction, an optimal solution in the other direction can be trivially found. □

Derivation of the path-following error model

In this section, the details regarding the derivation of the model in (18) are presented. First, the nominal path in (14) gives:

$$\frac{\mathrm{d}\theta_{3,r}}{\mathrm{d}\tilde{s}} = v_r \kappa_{3,r}, \quad \tilde{s} \in [0, \tilde{s}_G], \tag{66a}$$

$$\frac{\mathrm{d}\beta_{3,r}}{\mathrm{d}\tilde{s}} = v_r \left(\frac{\sin\beta_{2,r} - M_1 \cos\beta_{2,r} \kappa_r}{L_2 \cos\beta_{3,r} C_1(\beta_{2,r}, \kappa_r)} - \kappa_{3,r} \right), \quad \tilde{s} \in [0, \tilde{s}_G], \tag{66b}$$

$$\frac{\mathrm{d}\beta_{2,r}}{\mathrm{d}\tilde{s}} = v_r \left(\frac{\kappa_r - \frac{\sin\beta_{2,r}}{L_2} + \frac{M_1}{L_2}\cos\beta_{2,r}\kappa_r}{\cos\beta_{3,r} C_1(\beta_{2,r}, \kappa_r)} \right), \quad \tilde{s} \in [0, \tilde{s}_G]. \tag{66c}$$

Moreover, the models of θ_3, β_3 and β_2 in (1) can equivalently be represented as

$$\dot{\theta}_3 = v_3 \frac{\tan\beta_3}{L_3}, \tag{67a}$$

$$\dot{\beta}_3 = v_3 \left(\frac{\sin\beta_2 - M_1 \cos\beta_2 \kappa}{L_2 \cos\beta_3 C_1(\beta_2, \kappa)} - \frac{\tan\beta_3}{L_3} \right), \tag{67b}$$

$$\dot{\beta}_2 = v_3 \left(\frac{\kappa - \frac{\sin\beta_2}{L_2} + \frac{M_1}{L_2}\cos\beta_2\kappa}{\cos\beta_3 C_1(\beta_2, \kappa)} \right), \tag{67c}$$

where v has been replaced with v_3 using (3). Now, since $\tilde{\theta}_3(t) = \theta_3(t) - \theta_{3,r}(\tilde{s}(t))$, the chain rule together with the equation for $\dot{\tilde{s}}$ in (17a) yields

$$\begin{aligned} \dot{\tilde{\theta}}_3(t) &= \dot{\theta}_3 - \dot{\tilde{s}} \frac{\mathrm{d}}{\mathrm{d}\tilde{s}} \theta_{3,r}(\tilde{s}) \\ &= v_3 \left(\frac{\tan(\tilde{\beta}_3 + \beta_{3,r})}{L_3} - \frac{\kappa_{3,r} \cos\tilde{\theta}_3}{1 - \kappa_{3,r}\tilde{z}_3} \right) = v_3 f_{\tilde{\theta}_3}(\tilde{s}, \tilde{x}_e, \tilde{\kappa}), \quad t \in \Pi(0, \tilde{s}_G). \end{aligned} \tag{68}$$

In analogy, taking the time-derivative of $\tilde{\beta}_3(t) = \beta_3(t) - \beta_{3,r}(\tilde{s}(t))$ and applying the chain rule gives

$$\begin{aligned}\dot{\tilde{\beta}}_3 &= \dot{\beta}_3 + \dot{\tilde{s}}\frac{\mathrm{d}}{\mathrm{d}\tilde{s}}\beta_{3,r}(\tilde{s}) \\ &= v_3\left(\frac{\sin(\tilde{\beta}_2+\beta_{2,r}) - M_1\cos(\tilde{\beta}_2+\beta_{2,r})(\tilde{\kappa}+\kappa_r)}{L_2\cos(\tilde{\beta}_3+\beta_{3,r})C_1(\tilde{\beta}_2+\beta_{2,r},\tilde{\kappa}+\kappa_r)} - \frac{\tan(\tilde{\beta}_3+\beta_{3,r})}{L_3}\right. \\ &\left. -\frac{\cos\tilde{\theta}_3}{1-\kappa_{3,r}\tilde{z}_3}\left(\frac{\sin\beta_{2,r} - M_1\cos\beta_{2,r}\kappa_r}{L_2\cos\beta_{3,r}C_1(\beta_{2,r},\kappa_r)} - \kappa_{3,r}\right)\right) = v_3 f_{\tilde{\beta}_3}(\tilde{s},\tilde{x}_e,\tilde{\kappa}), \quad t\in\Pi(0,\tilde{s}_G).\end{aligned} \tag{69}$$

Finally, taking the time-derivative of $\tilde{\beta}_2(t) = \beta_2(t) - \beta_{2,r}(\tilde{s}(t))$ and applying the chain rule again yields

$$\begin{aligned}\dot{\tilde{\beta}}_2 &= \dot{\beta}_2 + \dot{\tilde{s}}\frac{\mathrm{d}}{\mathrm{d}\tilde{s}}\beta_{2,r}(\tilde{s}) \\ &= v_3\left(\left(\frac{\tilde{\kappa}+\kappa_r - \frac{\sin(\tilde{\beta}_2+\beta_{2,r})}{L_2} + \frac{M_1}{L_2}\cos(\tilde{\beta}_2+\beta_{2,r})(\tilde{\kappa}+\kappa_r)}{\cos(\tilde{\beta}_3+\beta_{3,r})C_1(\tilde{\beta}_2+\beta_{2,r},\tilde{\kappa}+\kappa_r)}\right)\right. \\ &\left. -\frac{\cos\tilde{\theta}_3}{1-\kappa_{3,r}\tilde{z}_3}\left(\frac{\kappa_r - \frac{\sin\beta_{2,r}}{L_2} + \frac{M_1}{L_2}\cos\beta_{2,r}\kappa_r}{\cos\beta_{3,r}C_1(\beta_{2,r},\kappa_r)}\right)\right) = v_3 f_{\tilde{\beta}_2}(\tilde{s},\tilde{x}_e,\tilde{\kappa}), \quad t\in\Pi(0,\tilde{s}_G),\end{aligned} \tag{70}$$

which finalizes the derivation. Moreover, by inserting $(\tilde{x}_e,\tilde{\kappa}) = (0,0)$ in (68)–(70) yield $\dot{\tilde{\theta}}_3 = \dot{\tilde{\beta}}_3 = \dot{\tilde{\beta}}_2 = 0$, $\forall t\in\Pi(0,\tilde{s}_G)$, *i.e.*, the origin is an equilibrium point. Finally, from (3), we have that $v_3 = vg_v(\beta_2,\beta_3,\kappa)$ and the models in (68)–(70) can in a compact form also be represented as

$$\dot{\tilde{\theta}}_3 = vg_v(\tilde{\beta}_2+\beta_{2,r},\tilde{\beta}_3+\beta_{3,r},\tilde{\kappa}+\kappa_r)f_{\tilde{\theta}_3}(\tilde{s},\tilde{x}_e,\tilde{\kappa}), \quad t\in\Pi(0,\tilde{s}_G), \tag{71a}$$

$$\dot{\tilde{\beta}}_3 = vg_v(\tilde{\beta}_2+\beta_{2,r},\tilde{\beta}_3+\beta_{3,r},\tilde{\kappa}+\kappa_r)f_{\tilde{\beta}_3}(\tilde{s},\tilde{x}_e,\tilde{\kappa}), \quad t\in\Pi(0,\tilde{s}_G), \tag{71b}$$

$$\dot{\tilde{\beta}}_2 = vg_v(\tilde{\beta}_2+\beta_{2,r},\tilde{\beta}_3+\beta_{3,r},\tilde{\kappa}+\kappa_r)f_{\tilde{\beta}_2}(\tilde{s},\tilde{x}_e,\tilde{\kappa}), \quad t\in\Pi(0,\tilde{s}_G), \tag{71c}$$

where the origin is an equilibrium point since $f_{\tilde{\theta}_3}(\tilde{s},0,0) = f_{\tilde{\beta}_3}(\tilde{s},0,0) = f_{\tilde{\beta}_2}(\tilde{s},0,0) = 0$, $\tilde{s}\in[0,\tilde{s}_G]$.

Acknowledgments

The research leading to these results has been founded by Strategic vehicle research and innovation (FFI). The authors gratefully acknowledge the Royal Institute of Technology for providing the external RTK-GPS. The authors acknowledge the anonymous reviewers for their valuable suggestions and critical comments which have improved the contents of this paper. The authors would also like to express their gratitude to Scania CV for providing necessary hardware, as well as software and technical support.

References

[1] C. Altafini. Some properties of the general n-trailer. *International Journal of Control*, 74(4):409–424, 2001.

[2] C. Altafini. Path following with reduced off-tracking for multibody wheeled vehicles. *IEEE Transactions on Control Systems Technology*, 11(4):598–605, 2003.

[3] C. Altafini, A. Speranzon, and K.-H. Johansson. Hybrid control of a truck and trailer vehicle. In *Hybrid Systems: Computation and Control*, pages 21–34. Springer, 2002.

[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*, 17(6):915–922, Dec 2001.

[5] B. Anderson and J. B. Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[6] 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*, 2018.

[7] O. Andersson, O. Ljungqvist, M. Tiger, D. Axehill, and F. Heintz. Receding-horizon lattice-based motion planning with dynamic obstacle avoidance. In *Proceedings of the 57th IEEE Conference on Decision and Control*, pages 4467–4474, 2018.

[8] H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. J. Lilienthal. Fast, continuous state path smoothing to improve navigation accuracy. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation*, pages 662–669, 2015.

[9] D. Arnström. State estimation for truck and trailer systems using deep learning. Master's thesis, Linköping University, 2018.

[10] A. Astolfi, P. Bolzern, and A. Locatelli. Path-tracking of a tractor-trailer vehicle along rectilinear and circular paths: A lyapunov-based approach. *IEEE transactions on robotics and automation*, 20(1):154–160, 2004.

[11] K. Bergman and D. Axehill. Combining homotopy methods and numerical optimal control to solve motion planning problems. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, pages 347–354, 2018.

[12] S. Beyersdorfer and S. Wagner. Novel model based path planning for multi-axle steered heavy load vehicles. In *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems*, pages 424–429, Oct 2013.

[13] P. Bolzern, R. M. DeSantis, A. Locatelli, and D. Masciocchi. Path-tracking for articulated vehicles with off-axle hitching. *IEEE Transactions on Control Systems Technology*, 6(4):515–523, 1998.

[14] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[15] S. P. Boyd, L. Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.

[16] L. Caup, J. Salmen, I. Muharemovic, and S. Houben. Video-based trailer detection and articulation estimation. In *proceedings of the 2013 IEEE Intelligent Vehicles Symposium*, pages 1179–1184, 2013.

[17] W. Chung, M. Park, K. Yoo, J. I. Roh, and J. Choi. Backward-motion control of a mobile robot with n passive off-hooked trailers. *Journal of mechanical science and technology*, 25(11):2895–2905, 2011.

[18] M. Cirillo. From videogames to autonomous trucks: A new algorithm for lattice-based motion planning. In *proceedings of the 2017 IEEE Intelligent Vehicles Symposium*, pages 148–153, 2017.

[19] M. Cirillo, T. Uras, and S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 232–239, 2014.

[20] J. David and P. V. Manivannan. Control of truck-trailer mobile robots: a survey. *Intelligent Service Robotics*, 7(4):245–258, 2014.

[21] R. A. Decarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.

[22] A. W. Divelbiss and J. T. Wen. Trajectory tracking control of a car-trailer system. *IEEE Transactions on Control systems technology*, 5(3):269–278, 1997.

[23] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[24] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[25] N. Evestedt, O. Ljungqvist, and D. Axehill. Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3690–3697, 2016.

[26] N. Evestedt, O. Ljungqvist, and D. Axehill. Path tracking and stabilization for a reversing general 2-trailer configuration using a cascaded control approach. In *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium*, pages 1156–1161, June 2016.

[27] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[28] C. Fuchs, F. Neuhaus, and D. Paulus. Advanced 3-d trailer pose estimation for articulated vehicles. In *proceedings of the 2015 IEEE Intelligent Vehicles Symposium*, pages 211–216, June 2015.

[29] F. Gustafsson. *Statistical sensor fusion*. Studentlitteratur, 2010.

[30] M. Hafner and T. Pilutti. Control for automated trailer backup. Technical report, SAE Technical Paper, 2017.

[31] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall New Jersey, 1996.

[32] R. A. Knepper and A. Kelly. High performance state lattice planning using heuristic look-up tables. In *IROS*, pages 3375–3380, 2006.

[33] F. Lamiraux, S. Sekhavat, and J. P. Laumond. Motion planning and control for hilare pulling a trailer. *IEEE Transactions on Robotics and Automation*, 15(4):640–652, Aug 1999.

[34] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[35] J. Levinson et al. Towards fully autonomous driving: Systems and algorithms. In *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium*, pages 163–168, 2011.

[36] M. Likhachev, G. J. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in neural information processing systems*, pages 767–774, 2004.

[37] X. Liu and D. Cebon. A minimum swept path control strategy for reversing articulated vehicles. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, pages 1962–1967, June 2018.

[38] O. Ljungqvist, D. Axehill, and A. Helmersson. Path following control for a reversing general 2-trailer system. In *Proceedings of the 55th IEEE Conference on Decision and Control*, pages 2455–2461, 2016.

[39] O. Ljungqvist, D. Axehill, and J. Löfberg. On stability for state-lattice trajectory tracking control. In *Proceedings of the 2018 American Control Conference, Milwaukee*, June 2018.

[40] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer. Lattice-based motion planning for a general 2-trailer system. In *Proceedings of the 2017 IEEE Intelligent Vehicles Symposium, Los Angeles*, pages 2455–2461, June 2017.

[41] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289. IEEE, 2004.

[42] M. M. Michałek. A highly scalable path-following controller for n-trailers with off-axle hitching. *Control Engineering Practice*, 29:61–73, 2014.

[43] M. M. Michałek. Cascade-like modular tracking controller for non-standard n-trailers. *IEEE Transactions on Control Systems Technology*, 25(2):619–627, March 2017.

[44] M. M. Michalek. Cascaded approach to the path-following problem for n-trailer robots. In *9th International Workshop on Robot Motion and Control*, pages 161–166, July 2013.

[45] M. M. Michalek. Non-minimum-phase property of n-trailer kinematics resulting from off-axle interconnections. *International Journal of Control*, 86(4):740–758, 2013.

[46] M. M. Michałek and D. Pazderski. Forward tracking of complex trajectories with non-standard n-trailers of non-minimum-phase kinematics avoiding a jackknife effect. *International Journal of Control*, pages 1–14, 2018.

[47] M. Montemerlo et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[48] J. Morales, J. L. Martinez, A. Mandow, and A. J. Garcia-Cerezo. Steering the last trailer as a virtual tractor for reversing vehicles with passive on- and off-axle hitches. *IEEE Transactions on Industrial Electronics*, 60(12):5729–5736, Dec 2013.

[49] R. M. Murray and S. S. Sastry. Steering nonholonomic systems in chained form. In *proceedings of the 30th IEEE Conference on Decision and Control*, pages 1121–1126 vol.2, Dec 1991.

[50] P. Nyberg. Stabilization, sensor fusion and path following for autonomous reversing of a full-scale truck and trailer system. Master's thesis, Linköping University, 2016.

[51] R. Oliveira, M. Cirillo, and B. Wahlberg. Combining lattice-based planning and path optimization in autonomous heavy duty vehicle applications. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, pages 2090–2097, 2018.

[52] B. Paden, M. Cap, 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*, 1(1):33–55, March 2016.

[53] S. Pettersson and B. Lennartson. Stability and robustness for hybrid systems. In *Proceedings of 35th IEEE Conference on Decision and Control*, Dec 1996.

[54] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[55] A. J. Rimmer and D. Cebon. Implementation of reversing control on a doubly articulated vehicle. *Journal of Dynamic Systems, Measurement, and Control*, 139(6):061011, 2017.

[56] P. Rouchon, M. Fliess, J. Levine, and P. Martin. Flatness, motion planning and trailer systems. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2700–2705 vol.3, Dec 1993.

[57] W. J. Rugh and W. J. Rugh. *Linear system theory*, volume 2. prentice hall Upper Saddle River, NJ, 1996.

[58] M. Sampei and K. Furuta. On time scaling for nonlinear systems: Application to linearization. *IEEE Transactions on Automatic Control*, 31(5):459–462, 1986.

[59] M. Sampei, T. Tamura, T. Kobayashi, and N. Shibui. Arbitrary path tracking control of articulated vehicles using nonlinear control theory. *IEEE Transactions on Control Systems Technology*, 3(1):125–131, 1995.

[60] C. Samson. Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1):64–77, Jan 1995.

[61] C. D. Saxe and D. Cebon. A visual template-matching method for articulation angle measurement. In *proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 626–631, Sept 2015.

[62] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *The International Journal of Robotics Research*, 17(8):840–857, 1998.

[63] I. Skog and P. Händel. In-car positioning and navigation technologies—a survey. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21, March 2009.

[64] O. J. Sørdalen. Conversion of the kinematics of a car with n trailers into a chained form. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pages 382–387. IEEE, 1993.

[65] D. Tilbury, R. M. Murray, and S. Sastry. Trajectory generation for the n-trailer problem using goursat normal form. *IEEE Transactions on Automatic Control*, 40(5):802–819, 1995.

[66] O. Törnroth and T. Nyberg. Design and implementation of a strategy for path tracking on autonomous heavy-duty vehicles. Master's thesis, Linköping University, 2018.

[67] C. Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[68] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[69] M. Werling, P. Reinisch, M. Heidingsfeld, and K. Gresser. Reversing the general one-trailer system: Asymptotic curvature stabilization and path tracking. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):627–636, 2014.

[70] C. B. Winkler. Simplified analysis of the steady-state turning of complex vehicles. *Vehicle System Dynamics*, 29(3):141–180, 1998.

[71] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.

[72] K. Yoo and W. Chung. Pushing motion control of n passive off-hooked trailers by a car-like mobile robot. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 4928–4933, May 2010.

[73] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli. Autonomous parking using optimization-based collision avoidance. In *Proceedings of the 2018 IEEE Conference on Decision and Control*, pages 4327–4332, Dec 2018.