Autonomous Truck-Trailer Parking – Path Planning and Path Tracking Control

by

A. Canberk Manav

A Dissertation Submitted to the
Graduate School of Engineering in Partial
Fulfillment of the Requirements for the Degree
of

Doctor of Philosophy

in

Mechanical Engineering

KOÇ ÜNİVERSİTESİ

Aug 02, 2022

Autonomous Truck-Trailer Parking – Path Planning and Path Tracking Control

Koc University

Graduate School of Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

A. Canberk Manav

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Prof. Dr. İsmail Lazoğlu (Advisor) _____

Prof. Dr. Metin Türkay _____

Assoc. Prof. Arif Karabeyoğlu _____

Prof. Dr. Hakan Temeltaş _____

Prof. Dr. Erdinç Altuğ _____

Date: _____

ABSTRACT

Maneuvering a truck-trailer system while docking is extremely challenging. This study aims to alleviate this problem by presenting a novel cascade path planning framework and an enhanced path-following control framework for autonomous semi-trailer docking. In the proposed system, the cascade path planning framework generates kinematically feasible and drivable maneuvers for trailer parking and the path-following control framework introduces adaptive controllers that utilize gain scheduling for forward and reverse path-following tasks in docking maneuvers to increase the robustness and path-following performance.

In the cascade path planning approach, a realistic and deterministic parking behavior model, iterative analytical method (IAM), is proposed and combined with an enhanced Closed-Loop Rapidly Exploring Random Tree (CL-RRT) approach. Cascade path planning approach combining CL-RRT with IAM mimicking real-world parking practice enables generation of both kinematically feasible and deterministic parking maneuvers with obstacle avoidance. For evaluation, different parking scenarios are generated and selected through a developed case generation tool. The proposed path planning approach is evaluated through MATLAB simulations for performance evaluation. The results achieved a noticeable success with a high rate of generated feasible maneuvers for truck-trailer parking.

In the proposed path following control framework, the system includes an improved pure pursuit controller with adaptive look-ahead distance for forward path following; a cascade controller of reverse pure pursuit, and a gain-scheduled linear-quadratic (LQ) control for reverse path-following. In the evaluation of the path-following performance of forward and reverse controllers, the closed-loop system of path-following controllers with the truck-trailer kinematic model is simulated in MATLAB/Simulink for various test cases, and the results are compared with those of other studies. Furthermore, different docking scenarios are generated via the cascade path planning algorithm for autonomous semitrailer docking. These are tested with a high-degree semi-trailer model within the IPG TruckMaker simulation environment, and with a full truck-trailer vehicle in the test field. The results of both simulations and physical testing clearly demonstrate improvements in terms of the control problem formulation, i.e. the stabilized path-following is obtained with acceptable path-following errors.

ÖZETÇE

Bir tır - treyler sistemini yerleştirme alanına yanaşırtırırken manevra yapmak son derece zordur. Bu çalışma, otonom yarı-treyler yanaştırma problemi için yeni bir kademeli yol planlama ve geliştirilmiş bir yol takip kontrol çerçevesi sunmaktadır. Önerilen sistemde, kademeli yol planlama çerçevesi, treyler park etme için kinematik olarak uygulanabilir ve sürülebilir manevralar üretir ve yol izleme kontrol çerçevesi, yanaşma manevralarındaki ileri ve geri yol takibi görevlerinde sağlamlığı ve yol takibi performansini artırmak için kazanç planlaması kullanan uyarlanabilir kontrolörler sunar. Kademeli yol planlama yaklaşımında, gerçekçi ve deterministik bir park davranışı modeli olarak yinelemeli analitik yöntem (IAM) önerilmiş, ve geliştirilmiş bir Kapalı Döngü Hızla Keşfeden Rastgele Ağaç (CL-RRT) yaklaşımı ile birleştirilmiştir. CL-RRT'yi gerçek dünyadaki park etme uygulamasını taklit eden IAM ile birleştiren kademeli yol planlama yaklaşımı, engellerden kaçınma ile hem kinematik olarak uygulanabilir hem de deterministik olan park manevralarının oluşturulmasını sağlar. Değerlendirme için, geliştirilmiş bir vaka oluşturma aracı vasıtasıyla farklı park senaryoları oluşturulur ve seçilir. Önerilen yol planlama yaklaşımı, performans değerlendirmesi için MATLAB simülasyonları aracılığıyla değerlendirilir. Sonuçlar, tır-treyler park etme için yüksek oranda üretilen uygulanabilir manevralarla gözle görülür bir başarı elde etmiştir.

Önerilen yol izleme kontrol çerçevesinde, sistem, ileri yol izleme için uyarlanabilir ileriye bakma mesafesine sahip gelişmiş bir saf takip (Pure Pursuit) kontrolörü içerir; ters yol takibi içinse kademeli bir ters saf takip ve kazanç programlı doğrusal-kuadratik (LQ) kontrolörü içerir. İleri ve geri kontrolörlerin yol takip performansının değerlendirilmesinde, yol takip eden kontrolörlerin kapalı çevrim sistemi tır -treyler kinematik modeli ile MATLAB/Simulink ortamında çeşitli test durumları için simüle edilmiş ve sonuçlar diğer çalışmalarla karşılaştırılmıştır. Ayrıca, otonom semi-treyler yanaşması için kademeli yol planlama algoritması aracılığıyla farklı yanaşma senaryoları oluşturulmuş, ve bunlar, IPG TruckMaker simülasyon ortamında yüksek dereceli semi-treyler modeli ile, ve test sahasında full tır-treyler araç ile test edilmiştir. Hem simülasyonların hem de fiziksel testlerin sonuçları, kontrol problemi formülasyonu açısından iyileştirmeleri açıkça göstermektedir, yani stabilize yol takibi, kabul edilebilir yol takibi hatalarıyla elde edilmiştir.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. İsmail Lazoğlu for his expert guidance and continuous support of my Ph.D. study and related research, for his patience and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would also like to thank Prof. Dr. Metin Türkay, Assoc. Prof. Arif Karabeyoğlu, Prof. Dr. Hakan Temeltaş and Prof. Dr. Erdinç Altuğ for reading this thesis and involving in my Ph.D. thesis defense committee.

I acknowledge Eren Aydemir from ADAS team of Ford Otosan A.S. for his support and contributions to this study.

Last but not least, I would like to thank my family: my wife, my parents and my brother for supporting me spiritually throughout writing this thesis, and for their absolute confidence in me. This thesis as well as my all previous success is dedicated to my family, especially to the newer members of my family; my daughter “Su” and my son “Can”.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

P_i, P_f: Initial and Final Generalized Coordinates, 15

P_m: Position of Interim Point, 28

P_{mid}: Position of Workspace Midpoint, 28

P_s: Position of Tree Expansion Start Point, 28

$rng_{\beta max}$: Maximum Standard Deviation of β in each Branch of Tree, 70

$rng_{\gamma max}$: Maximum.Standard Deviation of γ in each Branch of Tree, 70

ROS: Robot Operating System, 12

RRT: Rapidly Exploring Random Tree, 8

RTK-GPS: Real-Time Kinematic Global Positioning System, 3

Sol_{max}: Number of Enough Solutions, 26

S_{rnd}: Generated Random Sample, 25

T_{max}: Maximum Duration for Tree Expansion, 26

Ts_{max}: Maximum Duration for Solution Existed Loop, 69

v: Longitudinal Speed of the Truck, 13

w_{park} and h_{park}: width and height of the parking slot, 21

α: Steering Angle, 13

α_{max}: Maximum Steer Wheel Angle, 70

β: Heading Angle of the Truck, 12

δ: Heading Angle of the Trailer, 12

ε_{db}: CL-RRT Error Tolerance for Distance at Goal Point in Backward Direction, 71

ε_{df}: CL-RRT Error Tolerance for Distance at Goal Point in Forward Direction, 71

$\varepsilon_{\beta b}$: CL-RRT Error Tolerance for Truck Angle at Goal Point in Backward Direction, 71

$\varepsilon_{\beta f}$: CL-RRT Error Tolerance for Truck Angle at Goal Point in Forward Direction, 71

$\varepsilon_{\delta b}$: CL-RRT Error Tolerance for Trailer Angle at Goal Point in Backward Direction, 71

$\varepsilon_{\delta f}$: CL-RRT Error Tolerance for Trailer Angle at Goal Point in forward direction, 71

θ_h: Look-ahead Heading Error, 49

μ_b: Probability for L_b Cost Heuristic, 70

μ_L: Likelihood Parameter, 16

μ_p: Probability for L_p Cost Heuristic, 70

ρ_{obs}: Obstacle Density, 16

Υ Hitch Angle, 52

α(t): Steering Angle, 48

Chapter 1

INTRODUCTION

This thesis concerns the development of a realistic and deterministic path planning and an efficient and robust path tracking control framework for a truck-trailer system in parking scenarios, and the implementation of the planners and controllers both in a simulation environment and in a real full-scale truck-trailer system. In this chapter, the motivation, problem statement and outline will be presented.

1.1 Background

Advanced Driver Assistance Systems (ADAS) are widely used nowadays in passenger cars and heavy-duty vehicles, including Level 1 systems such as Adaptive Cruise Control (ACC), Automatic Emergency Braking (AEB) and Lane Keep Assist (LKA), as well as Level 2 systems such as Lane Change Control, Auto Park Assist, and Highway Assist. However, while Park Assist systems are usually provided for passenger cars, they are generally not an option for truck-trailer systems. A system that is provided, but only for some pickup trucks and SUVs, is a trailer backup assist where the driver uses a knob to operate a system in which the trailer is stabilized around a reference point. Reversing trailers is commonly performed when loading and unloading the trailer in docking stations and construction sites. However, due to the complexity of the task, truck drivers need to be experienced in backing a trailer into a narrow parking location with minimal maneuvering while avoiding collisions. As the requirement for parking assist in truck-trailer systems has increased in recent years, several studies currently exist in relation to optimal truck-trailer parking.

1.2 Problem Statement

The main purpose of this thesis is to develop, implement and validate an autonomous driving system for a full-scale truck and trailer system to fulfill parking tasks in use case scenarios at docking stations and construction sites with the existence of static obstacles and dynamic objects. The proposed system could also be implemented for forward parking, but such implementation is not within the scope of this study. Instead, this

study mainly focuses on trailer backing, a common procedure for truck drivers when loading and unloading trailers in docking stations and construction sites. Example use case locations of docking stations and construction sites are shown in Figure 1.1.

Figure 1.1: Use Cases – Parking in Docking Station and Construction Site

An autonomous driving system can be divided into three main parts; environmental information, motion planning, and motion control. Environmental information includes perception (sensor processing and fusion, localization and mapping) and environmental interpretation and prediction. It provides necessary information for both path planning and path tracking control such as position and orientation of truck-trailer system and static obstacles and predicted trajectories of dynamic objects. For the simulations, this

information is provided by the simulation environment; and for the field tests, localization is achieved by processing RTK-GPS sensors and mapping by using an offline occupancy map of the parking lot. In fact, only static obstacles are considered in TruckMaker simulations and field tests of Ford FMAX truck-trailer system; dynamic obstacles are only introduced in MATLAB simulations. It is worth also noting that for the temporarily absence of the GPS information, the environmental information part could be extended with the localization and tracking functions to provide tracked position and orientation information of the truck-trailer system. This part is out of the scope of this study, and recommended to be studied as a future work.

Motion planning determines possible trajectories, each of which consists of a planned path and corresponding velocity profile along the path. Decisions for safety including an emergency brake for a possible collision prediction with a dynamic object are also considered in the planning part. An off-axle one-trailer system is selected for this study where the hitching point is placed a certain distance away from the rear wheel axle of the truck. In this study, the path planning part to reach the vehicle to the parking goal pose is handled with a novel cascade path planning framework where kinematically feasible and drivable maneuvers are generated for trailer parking. A realistic and deterministic parking planner, Iterative Analytical Method (IAM), is proposed and combined with Closed-Loop Rapidly Exploring Random Tree (CL-RRT) approach for a cascade path planning. For faster convergence, the CL-RRT approach is enhanced by the introduction of an extended probabilistic Random Sample Generation method. The Cascade path planning approach selects the appropriate algorithm for the use case and enables the generation of kinematically feasible parking maneuvers with obstacle avoidance. Velocity profile generation is performed with the implementation of the jerk optimal velocity introduced in [1]. Truck-trailer parking in docking stations and construction sites mostly refers to path planning in narrow environments; therefore, multiple maneuvers switching between forward and reverse motion could only achieve the expected parking task in most cases. Hence the algorithms proposed in the cascade path planning framework in this study consider the multiple maneuver phenomenon.

Motion control, consisting of the lateral and longitudinal control, executes the planned driving trajectories set by the motion planning system. Based on the use case scenario, an appropriate vehicle model should be selected to design the controllers. For a parking

scenario, dynamic effects including slip and load weight of the trailer could be neglected with the assumption of low-speed maneuvering and negligible inclination in the parking lot. It is also worth noting that inclination, weather conditions such as rain and snow, and road conditions such as muddy and wet roads would significantly affect the slip condition, therefore ideal road conditions are accepted with no inclination and rain or mud conditions. Hence, a type of pure pursuit controller at the kinematic level is preferred for implementation in this study. However, even at low speeds, there may be a significant slip, especially with trailer axles during tight turns. Accordingly, adequate limitation of the curvature for the planned path is critical to limit the hitch angle for the path-following performance of the controller. Doing so recognizes that curvature limitation results in more conservative paths. Hence, the proposed cascade path planning framework in this study generates kinematically feasible and drivable maneuvers that consider maneuver limitations and nonholonomic constraints as well. Subsequently, an adaptive approach that utilizes gain scheduling is presented for forward and reverse path-following tasks in parking maneuvers. Such an approach increases robustness and path-following performance. An adaptive pure pursuit controller with look-ahead distance scheduling is introduced for forward path-following, and a cascade controller of adaptive reverse pure pursuit with look-ahead distance scheduling and a gain-scheduled LQ controller is introduced for reverse path-following.

For validation of the path planning framework, different parking scenarios are generated and selected through a developed case generation tool. The proposed path planning approach is evaluated through MATLAB simulations for performance evaluation. For the validation of the path-following control framework, the stability and robustness of the controllers are verified through extensive MATLAB and TruckMaker simulations that implement various initial conditions and scenarios. The controller performance is first compared as a benchmark with the results in other studies through MATLAB simulations of the closed-loop system, which is comprised of the path-following controllers and truck-trailer kinematic model presented in the next section. Using the cascade path planning framework for autonomous truck trailer parking creates different parking scenarios, and these are tested both in TruckMaker simulations and in field tests

of the Ford FMAX truck-trailer system. Control parameters are also estimated via MATLAB simulations on the kinematic truck-trailer model.

Longitudinal control is handled by a PD controller to follow the generated velocity profile and implemented successfully in MATLAB simulations. Due to the technical limitations in implementation on both TruckMaker simulations and on the real full-scale truck-trailer, longitudinal control is handled by the cruise control of the system itself. Hence, an allowed constant max longitudinal velocity is determined and set as the setpoint to the cruise control of the vehicle system for each maneuver, and brake control is activated under a certain threshold of the remaining distance to the maneuver endpoint.

In summary, the main contributions of this study are as follows:

1) Introducing a novel cascade path planning framework where Iterative Analytical Method (IAM) and Closed Loop Rapidly Exploring Random Tree Algorithm (CL-RRT) are combined and selected using generated likelihood and obstacle density parameters based on the real-world parking practices of truck drivers for reverse parking. Hence, the framework allows the user to combine the advantages of both algorithms with the introduction of the algorithm selector to determine which algorithm fits better for the specified parking scenario.
2) Introducing an extended probabilistic Random Sample Generation method to enhance the CL-RRT approach for faster convergence, where the tree expansion is further biased around certain interest points such as interim point and goal point to expand the tree most likely around the operation space of a certain maneuver set proposed in IAM.
3) Introducing an enhanced path tracking control framework including pure pursuit controllers with an adaptive version for semitrailer docking. In this implementation, path-following controllers, both in forward and reverse motion, are enhanced via gain scheduling to increase the robustness and path-following performance.
4) Demonstrating the improved robustness and tracking performance of the adaptive LQ control, compared to classical LQ control, through simulations in the verification of robustness.

5) Validating the cascade path planning algorithm by generating kinematically feasible maneuvers for trailer parking. TruckMaker simulations and real-world testing is then performed with a full-scale truck-semitrailer system. This achieves successful parking and stabilizes high-performance path-following while ensuring acceptable path-following errors regarding the control problem formulation.
6) Exhibiting, in the Path Tracking Benchmark Simulations section, the superior path convergence and following performance of the proposed control framework. Benchmarks of the proposed path-following controllers are introduced with the comparison of several significant studies in the literature.

1.3 Outline

This thesis is organized in the following chapters:
Chapter 2: Literature Review, presents related work on path planners and motion controllers.
Chapter 3: Modeling of Truck-Trailer System, presents a mathematical model of the tractor-trailer and discusses the avoidance of the jackknife.
Chapter 4: Motion Planning, describes the trajectory planning including the planned path and corresponding velocity profile.
Chapter 5: Motion Control, describes the applied lateral and longitudinal motion controllers.
Chapter 6: Results, presents the results from the validation experiments on the implementation including simulations and field tests.
Chapter 7: Conclusion, discusses and draws conclusions from the results and suggests future work.

1.4 System Architecture

System architecture mainly consists of two parts: Motion planning and Motion Control. Environmental information is provided as an occupancy map and obstacle data to the system. Motion planning includes Path Planning and Trajectory Planning. In the first step of Path Planning, the proposed cascaded path planning algorithm operates first and

generates the path within a predefined error threshold at the starting point. The closed-loop steering connection function provides a smooth transition from the vehicle position to the generated path. Finally, the β-Spline Generator outputs the smoothed path where slope and curvature continuity are assured. In trajectory planning, the jerk optimal velocity profile is generated and merged with the path, and a collision check is performed considering vehicle trajectory along the path in a dynamic environment. Motion control consists of lateral and longitudinal controllers. Lateral control is done with the proposed forward and reverse path tracking controllers for the truck-trailer system. Longitudinal control is provided with the introduction of the gas pedal and brake pedal controllers. The block diagram of the system architecture is shown in Figure 1.2.

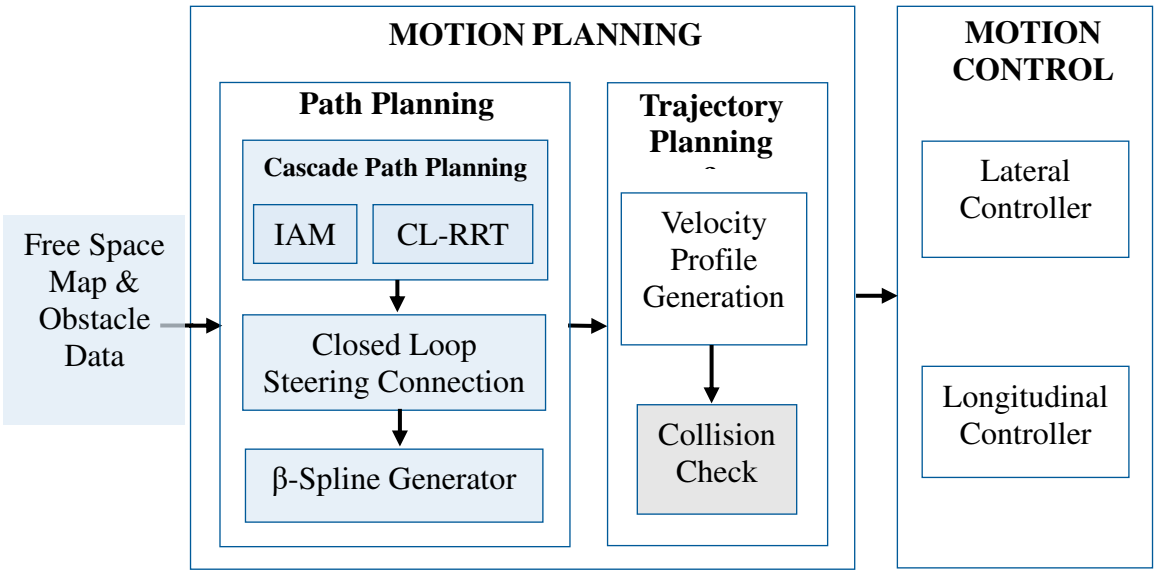


Figure 1.2: Block Diagram of System Architecture

Chapter 2

LITERATURE REVIEW

2.1 Path Planning

Truck-trailer system has nonholonomic constraints, i.e. nonintegrable differential expressions, including velocity and state vectors. An established method for nonholonomic path planning is to first generate a path for a holonomic vehicle and then modify the path to satisfy nonholonomic constraints [2]. However, injecting nonholonomic motion constraints into the motion planning beforehand and designing motion primitives based on them becomes more popular since it increases both the speed and robustness of the algorithm. Following this procedure, Divelbiss et. al. [3] introduced a nonholonomic iterative motion planning approach where path planning without obstacles and obstacle avoidance are formulated and solved separately as a nonlinear least square problem using equality constraints and inequality constraints respectively. In order to satisfy both equality and inequality constraints, the algorithm warps the entire path at each iteration. Zöbel et. al. [4] proposed a rule-based trajectory segmentation method for safe truck-trailer path planning. Gomez-Bravo et. al. [5] developed a parallel parking planning algorithm for truck-trailer based on changing trailer orientation. In this method, a set of analytically derivable maneuvers is introduced to change trailer orientation during the first step of the parking. Once the closed-loop system is designed by introducing simulations of the vehicle model and a proper collision check algorithm, various graph-based [6], interpolating curve-based or sample-based [7] path planning algorithms for single-body vehicle systems could be extended for the truck-trailer system. Another innovation was achieved by Rimmer et. al. [8], who generated a library of precomputed feasible path segments based on an empirical relation of curvature properties, and introduced a Dijkstra grid search-based path planning algorithm to determine a collision-free path through the pre-computed path segments. Elhassan et. al. [8] proposed a combination of the RRT algorithm with a Dubins path calculation in order to obtain kinematically feasible parking maneuvers. Evestedt et. al. introduced motion planning algorithms for backing 2-trailer systems using Closed-Loop RRT [9] and lattice planner [10], both providing kinematically

feasible paths by using sample-based search algorithms like RRT with closed-loop stabilized truck-trailer model and A* search in state lattice respectively. For improvement of the search speed, a pre-calculated look-up table for fast cost estimation [9], generation of motion primitives, and motion primitive reduction for state lattice formation [10] are introduced. Stahn et. al. [11] also used an A* search algorithm for a grid-based multi-dimensional path planner with up to five degrees of freedom. Due to the ease of implementation, probabilistic sampling-based approaches are increasingly preferred in motion planning algorithms; however, they cannot guarantee completeness and optimality. Furthermore, due to their non-deterministic nature, divergent path solutions could be achieved for the same configuration or very close initial configurations which would be mostly not welcomed by the driver.

2.2 Path Following Control

Different control techniques have been used in the literature to solve the path-following problem for autonomous vehicles in forward motion. The nonlinearity of the truck-trailer system, and its instability in reverse motion, are challenging aspects of the path-following element of trailer parking. However, low-speed maneuvering of the truck-trailer system in a forward motion for parking scenarios is a stable problem, hence the majority of the geometric path-following algorithms used for passenger cars, such as Pure Pursuit [12, 13], Modified Pure Pursuit [14], Stanley [15], and Alice [16], could also be used for forward path-following of truck-trailers, and a suitable algorithm could be selected based on such cases. Compared to other geometric methods, while the pure pursuit algorithm is more resistant to disturbances, it does result in cutting corners during turning based on the selected look-ahead distance. In response to this issue, Lombard et al. [17] extended the pure pursuit algorithm with an additional empirical coefficient: $1-kS$, in order to reduce the cutting of corners without reducing look-ahead distance. Other revisions have been made by Alonso et al. [18], who introduced an adaptive pure pursuit algorithm by measuring the look-ahead distance from the projected location of the vehicle onto the path; Campbell et al. [19], who compared classical and dynamic pure pursuit approaches, and Park et al. [20], who implemented an adaptive pure pursuit algorithm with dynamic look-ahead distance adjusted by vehicle speed, lateral error, and path curvature.

For robust control of the truck-trailer system, specifically in high-speed applications such as driving on the highway or platooning, different control techniques have been introduced that consider the states of both the truck and the trailer. Ji et al. [21] presented a robust lateral controller based on a backstepping variable structure control. The controller minimizes the lateral error and stabilizes the yaw angle in the presence of unknown disturbances. Chu et al. [22] designed a lateral controller based on an active disturbance rejection to assure the vehicle remains in the correct lane in the presence of uncertainties and external disturbances. Felez et al. [23] and Kassaeiyan et al. [24] applied linear (MPC) and nonlinear (NMPC) model predictive control respectively, while Nayl et. al [25] designed a novel sliding mode control.

These studies have mostly addressed the kinematic control problem of a truck trailer system, however, neglected the dynamic model of the vehicle system. To consider the inertial effects, centripetal forces, and torques, several studies have introduced the dynamic modeling and control of the truck-trailer system within the presence of model uncertainties. Barbosa et al. [26] applied a Robust Recursive Linear Quadratic Regulator for following the route of a loaded truck with varying payloads. Kati et al. proposed a robust static output feedback synthesis that utilizes active steering of the dolly to overcome parametric uncertainty regarding the payload variations in the vehicle dynamic model. This system assures stability and performance for a range of parameter values.

For reverse path-following of truck-trailers with off-axle trailers, Astolfi et al. [27] proposed a Lyapunov-based control algorithm that is able to track both rectilinear and constant curvature paths. The reverse path-following of an arbitrary path is challenging due to the nonlinearity and instability of the system. To handle these issues, Bolzern [28] introduced an exact linearization-based approach, while Pradalier et al. [29] designed a cascade controller for path stabilization and hitch angle control which achieves successful results in tests on a real truck-trailer system. Elhassan et al. [8] enhanced the approach by introducing a PI and Lyapunov controller with a linear control law based on lateral, orientation, and curvature errors. Rimmer et al. [30] introduced a similar control cost definition in which corresponding tuned gains are calculated analytically using the LQR approach. Altafini et al. [5] proposed a hybrid feedback control scheme using an LQ controller to stabilize internal angles of the truck-

trailer system while reversing. A similar approach to model-based state feedback optimal control is proposed by Evestedt et al. [31]. The authors designed an LQ controller and feedforward control by combining the LQ controller with pure-pursuit path-following control. This achieves improved results by targeting a piecewise linear reference path, and the design was tested on a small-scale truck with Ackerman steering and off-axle hitching. Furthermore, Wu et al. [32] proposed a Linear Quadratic Regulator (LQR) using steering rate instead of steering angle as the control input variable to reduce errors during the path-following of the route.

Following this literature review on the path planning and path-following control for a truck trailer system, several considerations will now be made regarding the design of path-following controllers for semitrailer docking tasks. The presence of nonholonomic constraints, as well as considerable nonlinearities and uncertainties in both kinematic and dynamic models, increase the complexity of the controller design. Proposed controllers presented in the literature for both forward and reverse path-following mainly focus on the kinematic level, whereas only a few studies address the control problem by considering both kinematic and dynamic models. Existing dynamic controllers produce feedback from sensor data regarding the need for exact linear and angular velocities of the truck trailer system in real-time, and such data is not normally available from real-world applications.

High-performance path following is needed in low-speed applications such as when reversing in docking stations and construction sites. Hence, a kinematic controller as a suitable low-speed control approach is both simple and real-time efficient. However, further improvements on such a controller need to be made both in path-following performance and robustness for it to become a reliable path-following controller for semitrailer docking tasks. Unfortunately, controllers at the kinematic level are either mostly tested only in simulations with the kinematic model, or with a small-scale multi-robot for real-world experiments. Furthermore, only a few tests include real-world applications with a full-scale truck-trailer system to observe the effect of dynamics on the stability and path-following performance.

Chapter 3

MODELING OF TRUCK-TRAILER SYSTEM

For the motion model, a general one-trailer system is introduced as illustrated in Figure 3.1. The generalized coordinates of the system are defined as $P = [\beta, \delta, x, y]$, where β and δ are the heading angles of the truck and the trailer respectively, and x and y are the positions of the rear axle center of the trailer. Alternatively, $\bar{P} = [\beta, \delta, \bar{x}, \bar{y}]$ could be used, where $\bar{x}$ and $\bar{y}$ are the positions of the rear axle of the truck. Furthermore, the term "pose" of the truck/trailer, which is commonly used in Robot Operating System (ROS), ADAS, and Computer Vision, refers to the total configuration of the corresponding body as a combination of position and orientation.

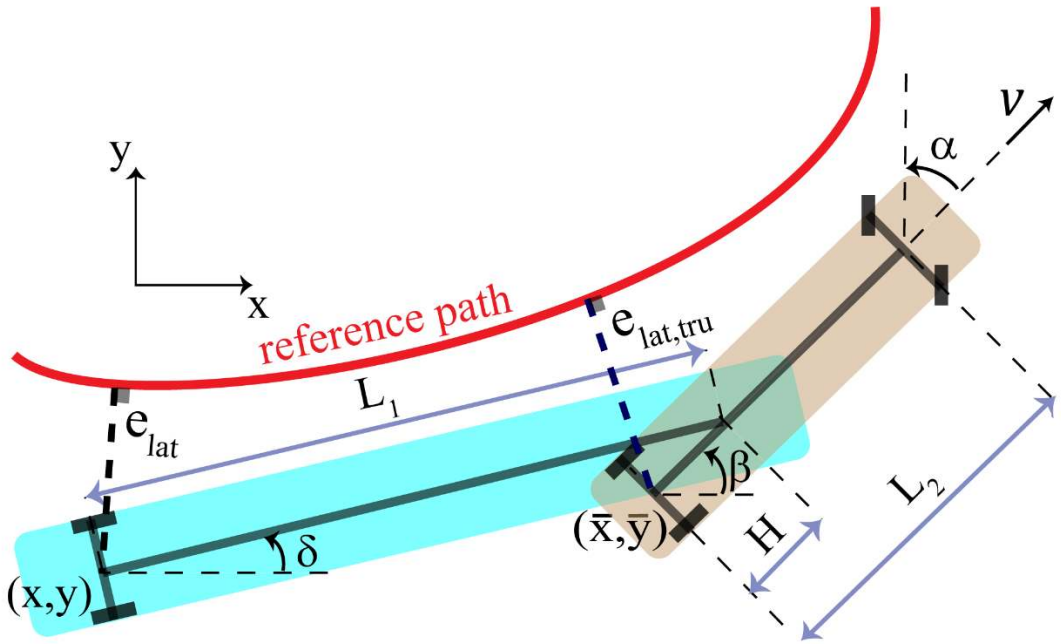


Figure 3.1: Geometry of truck-trailer system

The kinematic model of a general n-trailer system is derived using nonholonomic kinematic constraints. This is necessary due to the assumption of rolling occurring without slipping of the wheels. This assumption could be formulated in terms of nonholonomic kinematic constraints by deciding the instantaneous direction of the velocity vector of each axle [9]. Since this work focuses on path-following in parking scenarios where the intended operational vehicle speed is quite low, no-slip is a sustainable assumption. The equations of the system for a special one-trailer case are presented in a matrix form in (3.1).

$$\begin{bmatrix} \dot{\beta} \\ \dot{\delta} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{\sin(\beta(t)-\delta(t))}{L_1} & \frac{H\cos(\beta(t)-\delta(t))}{L_1} \\ \cos(\beta(t)-\delta(t))\cos\delta(t) & -H\sin(\beta(t)-\delta(t))\cos\delta(t) \\ \cos(\beta(t)-\delta(t))\sin\delta(t) & -H\sin(\beta(t)-\delta(t))\sin\delta(t) \end{bmatrix} \begin{bmatrix} v(t) \\ v(t)K(t) \end{bmatrix} \tag{3.1}$$

$$, \quad where\ K(t) = \frac{\tan\alpha(t)}{L_2}$$

L_1, L_2 represent the distances between the front and rear axles of the trailer and the truck respectively. H is the off-axle hitch length, α is the steering angle, K is the instantaneous curvature of the truck steering and v is the longitudinal speed of the truck. All of the aforementioned dimensions and terms are represented in Figure 3.1. The kinematic model shown is used to design an LQ controller for the stabilization of reverse path-following. The represented kinematic model for P is used for forward and reverse simulations in the closed-loop part of the CL-RRT algorithm and to design an LQ controller for stabilization of reverse path following. $\bar{P}$ alternative of the model is introduced in (3.2) and used for the partial solution of the Iterative Analytical Method (IAM).

$$\begin{bmatrix} \dot{\beta} \\ \dot{\delta} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{\sin(\beta(t)-\delta(t))}{L_1} & \frac{H\cos(\beta(t)-\delta(t))}{L_1} \\ \cos(\beta(t)) & 0 \\ \sin(\beta(t)) & 0 \end{bmatrix} \begin{bmatrix} v(t) \\ v(t)K(t) \end{bmatrix} \tag{3.2}$$

$$, \quad where\ K(t) = \frac{\tan\alpha(t)}{L_2}$$

Chapter 4

MOTION PLANNING

4.1 Cascade Path Planning Algorithm

In this study, IAM and CL-RRT approaches are proposed as components of a cascade path planning algorithm, where both approaches have different advantages compared to each other. IAM performs faster in less complicated workspaces, provides a deterministic and more realistic path, and computes a solution with much lower initial state tolerances at the starting point. On the other hand, CL-RRT has a higher solution rate in an obstacle crowded environment, is less dependent on the initial states and configurations of the truck-trailer system and parking environment, is still fast enough to solve in complicated environments, and can generate lower-cost paths due to the flexibility to select minimum cost alternative in multiple successful solutions through tree expansion. Each maneuver in IAM reflects different characteristics of truck-trailer reverse parking which are determined by examining real-world parking practices of truck drivers in docking stations. Since IAM is mimicking the real-world parking approach, its performance depends on the likelihood of the initial states and configurations to the proposed real-world parking maneuver set, however, CL-RRT ensures a much more generic path planning solution. One main advantage of IAM lies in its deterministic nature since a driver would expect the same motion planning suggestion for the same initial conditions and very close motion planning suggestions for slightly different initial conditions. In this respect, IAM provides a deterministic solution where the output is always the same for the same input set, whereas CL-RRT could output different solutions for the same input set due to its probabilistic nature. The motivation for proposing the cascade path planning framework is to combine the advantages of both algorithms with the introduction of an algorithm selector to determine which algorithm fits better for the specified parking scenario.

The cascade path planning algorithm shown in Figure 4.1 first checks the initial and desired final generalized coordinates P_i and P_f of the truck-trailer and generalized coordinates of obstacles $[P_{obs}]$ in the actual scenario; then derives the likelihood parameter μ_L for the initial configuration and constraints of the proposed parking

maneuver set $\bar{M}$ shown in Figure 4.2 and an obstacle density ρ_{obs} at the interested workspace of $\bar{M}$. Before derivation of μ_L, first an offline maximum likelihood estimation of a multivariate Gaussian distribution is done by deriving maximum likelihood estimators of the mean and covariance matrix $\hat{\mu}_n$ and $\hat{\Sigma}_n$ on the dataset of the localized initial truck-trailer generalized coordinates P_s^1 to P_s^n based on the real-world parking practices of truck drivers for reverse parking. Truck-trailer generalized coordinates are localized with respect to the frame aligned with the final configuration in the parking slot to derive an estimation for the initial relative configuration of the truck-trailer with respect to the desired parking configuration.

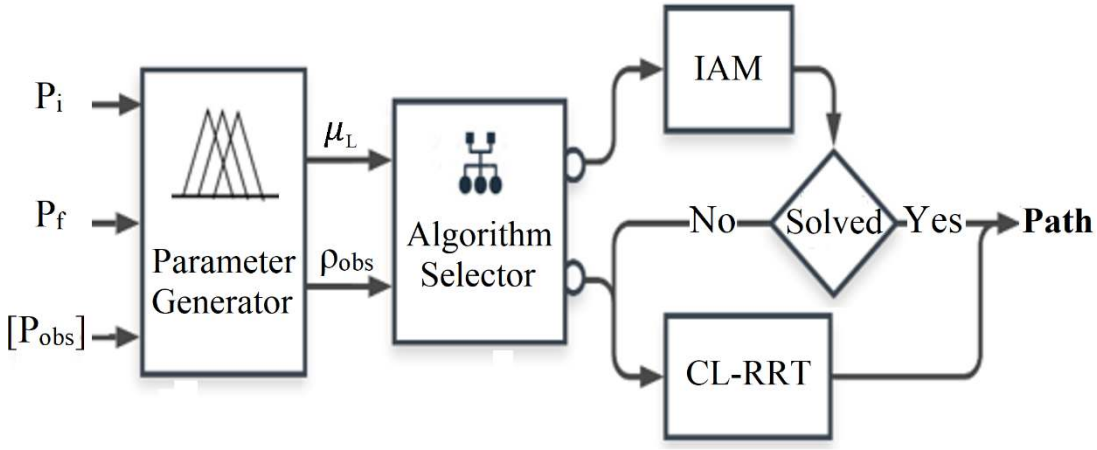


Figure 4.1. Flow diagram: Cascade path planning algorithm

$$L(P_s \mid \mu, \Sigma) = (2\pi)^{-2n}|\Sigma|^{-\frac{n}{2}}\exp\left(-\frac{1}{2}\sum_{k=1}^{n}(\mathrm{P}_s^k - \mu)^{\mathrm{T}}\Sigma^{-1}(\mathrm{P}_s^k - \mu)\right)$$
$$\hat{\mu}_n = \frac{1}{n}\sum_{k=1}^{n}\mathrm{P}_s^k \;\; ; \;\; \hat{\Sigma}_n = \frac{1}{n}\sum_{k=1}^{n}(\mathrm{P}_s^k - \hat{\mu}_n)(\mathrm{P}_s^k - \hat{\mu}_n)^{\mathrm{T}} \tag{4.1}$$

Subsequently, a new relative configuration candidate, P_s^*, is generated using P_i and P_f, and the likelihood of P_s^* is calculated as follows:

$$L\left(\mathrm{P}_s^* \mid \hat{\mu}_n, \hat{\Sigma}_n\right) = (2\pi)^{-2}\left|\hat{\Sigma}_n\right|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathrm{P}_s^* - \hat{\mu}_n)^{\mathrm{T}}\hat{\Sigma}_n{}^{-1}(\mathrm{P}_s^* - \hat{\mu}_n)\right) \tag{4.2}$$

Moreover, to determine the probability for the feasibility of the workspace boundaries, p_f, the probability density function of the interested workspace of $\bar{M}$ needs to be integrated between the workspace bounding box intervals $W_i\{x_{min}, y_{min}\}$ and

$W_f\{x_{max}, y_{max}\}$ of the actual scenario. Hence, a similar offline maximum likelihood estimation of a multivariate Gaussian distribution, $L\left(BB_w^* \mid \hat{\mu}_m, \hat{\Sigma}_m\right)$, is done for the workspace bounding box $BB_w\{x_{1-4}, y_{1-4}\}$ based on real-world parking practices. Finally, μ_L is designed as the weighted sum of the likelihood P_s^* and the probability for feasibility of the workspace boundaries p_f:

$$\begin{aligned} p_f &= p\left(W_i < BB_w^* < W_f \mid \hat{\mu}_m, \hat{\Sigma}_m\right) \\ \mu_L &= \tau_s\, L\left(\mathrm{P}_s^* \mid \hat{\mu}_n, \hat{\Sigma}_n\right) + \tau_f\, p_f \end{aligned} \tag{4.3}$$

The obstacle density ρ_{obs} at the interested workspace of $\bar{M}$ is derived by calculating the weighted total occupancy of obstacles inside the interested workspace represented by the maximum likelihood estimator of the mean $\hat{\mu}_m$ for the workspace bounding box. The occupancy of obstacles is calculated by dividing the total occupied area of obstacles in $\hat{\mu}_m$ by the total workspace area of $\hat{\mu}_m$, and weighting is provided with the gain k_{obs}.

$$\rho_{\mathrm{obs}} = \frac{k_{\mathrm{obs}} \sum_{k=1}^{\#\mathrm{obs}} \mathrm{A}_{occ}^k}{A_{\hat{\mu}_m}} \tag{4.4}$$

Checking the parameters μ_L and ρ_{obs} for certain thresholds, the cascade algorithm decides the selection of the feasible path planning algorithm between the Iterative Analytical Method and CL-RRT for the parking case. The thresholds in the algorithm selector and the weighting gains in the parameters are estimated through various test case simulations spanning the workspace. Subsequently, the collision check approach used in both IAM and CL-RRT algorithms, and the path smoothing approach used as post-processing of generated paths are explained in sections 4.4 and 4.5 respectively.

4.2 Iterative Analytical Method (IAM)

IAM is introduced with a general parking behavior consisting of four main maneuvers as shown in Figure 4.2. Each maneuver reflects different characteristics of truck-trailer reverse parking which are determined by examining real-world parking practices of truck drivers in docking stations. The first two maneuvers in a forward direction help the vehicle to locate at a desirable position to provide available space for the trailer to move toward the parking slot in a reverse motion. The third maneuver in a reverse

direction enables to come close to the parking slot, and the last maneuver arrives parking slot successfully while avoiding collision with neighbor parking slots.

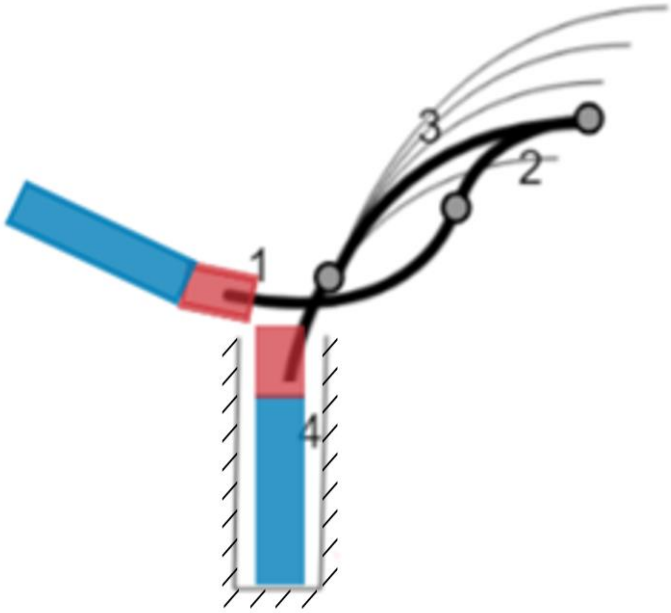


Figure 4.2. General real-world parking behavior as maneuver set $\bar{\boldsymbol{M}}$ for IAM

In this approach, the pure pursuit steering method is implemented to plan maneuvers. Namely, each maneuver is planned as a pure pursuit maneuver where a constant curvature maneuver to be tracked is defined, and then this maneuver is tracked via a pure pursuit controller in forward or reverse simulation to obtain simulated paths of truck and trailer which are used for collision check. Hence, maneuvers M_{1-4} are defined with constant curvatures and maneuver durations as $[K_{1-4}, t_{1-4}]$ with a constant speed of V, which enables to solve kinematic model differential equations for generalized coordinates $\bar{P}$ given the initial and final conditions of the system. The proposed method solves the system of error equations at the final state with an optimization method to minimize the error below a certain allowed threshold, hence the pose error is not zero at the final state. Therefore, the problem is solved reversely from the parking slot to the vehicle position to assure zero-configuration error at the final configuration, where the algorithm starts with the fourth maneuver (M_4) and completes with the first maneuver (M_1). As shown in the flow diagram of the algorithm in Figure 4.3, the algorithm first determines the collision-free escape from the parking slot with the iterative solution of the pure pursuit escape maneuver (M_4) through collision checked forward pure pursuit

simulations with decreasing curvature K_4 starting from $K_{4,max}$, maximum allowed curvature for M_4.

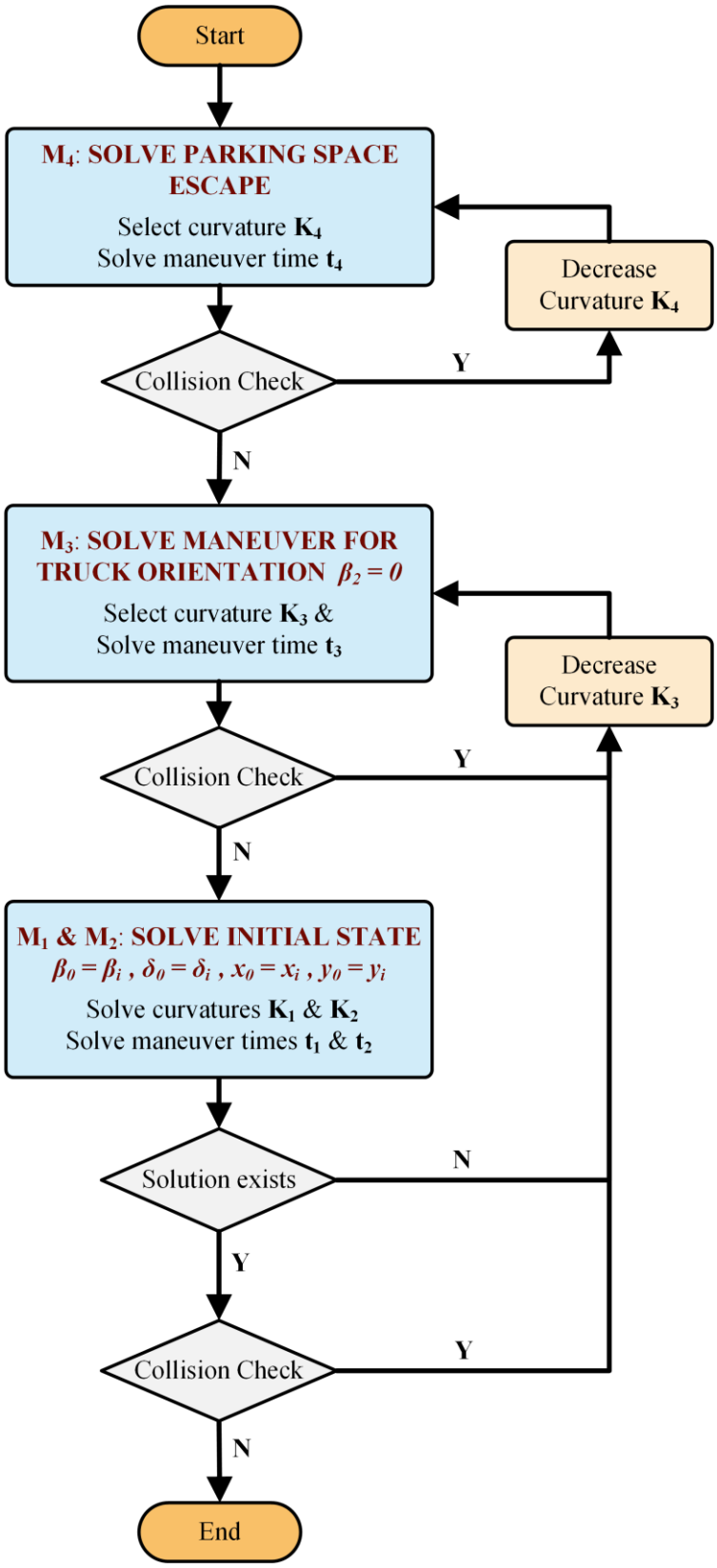


Figure 4.3. Flow diagram: Iterative Analytical Method (IAM)

By using basic trigonometry, $K_{4,max}$ is calculated analytically as in (4.5) based on the collision-free escape of the truck depicted in Figure 4.4, where w_{truck} is the width of the truck, w_{park} and h_{park} are width and height of the parking slot respectively. Since the analytical solution in (4.5) only assures collision avoidance of the truck from neighbor parking slots, a collision check is done for the simulated maneuver M_4, where simulated paths of the truck and trailer are checked within the occupancy of the workspace including obstacles.

$$K_{4,max} = \left| \frac{w_{park} - w_{truck}}{0.25\left(w_{park}^2 - w_{truck}^2\right) + h_{park}^2} \right| \tag{4.5}$$

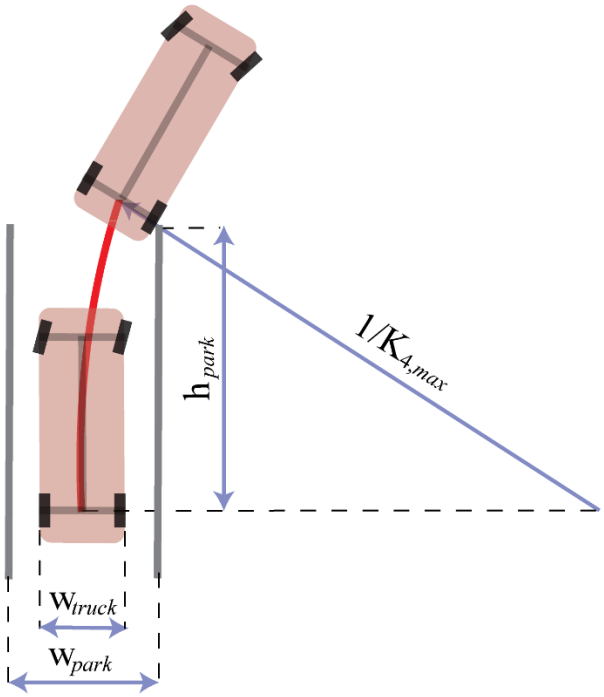


Figure 4.4. Collision-free escape maneuver from the parking slot (only truck)

The simulations run for a specific duration t_{sim} approximated as in (4.6) to assure that the truck trailer system is completely out of the parking slot. Once, a collision-free M_4 is obtained, the maneuver is shrunk to the first point of the escape from the parking slot which is determined by checking the side of the trailer bumper center point with respect to the upper line of the parking slot. Accordingly, t_4 is obtained based on the duration

of the shrunk maneuver M_4.

$$t_{sim} = \frac{1}{KV} sin^{-1}\big(K(L_1 + L_2 - H)\big) \tag{4.6}$$

Subsequently, the next maneuver (M_3) is planned and simulated with a higher curvature compared to K_4 in order to provide lateral space for further maneuvers (M_1 & M_2) and to aim at a solution path in a narrower space. Curvature is initially selected as the maximum curvature $K_{3,max}$ derived through the maximum allowed steering angle α_{max}; and maneuver time t_3 is solved as in (4.7) for a final truck orientation perpendicular to the parking slot ($\beta_2 = 0$ in local frame shown in Figure 4.2).

$$K_{3,max} = \frac{tan(\alpha_{max})}{L_2}\ , \quad t_3 = \frac{\beta_2 - \beta_3}{K_3\, V} \tag{4.7}$$

The simulated truck-trailer paths are checked for collision with the occupancy of the workspace; and if there is a collision, curvature K_3 is lowered iteratively in each step. Once, a collision-free M_3 is obtained, a minimization problem for the analytical solution of M_1 and M_2 is performed. If the solution exists, allowable positional and angular boundaries of the truck-trailer system are checked for the solution. If the solution fails, the algorithm steps back to plan M_3 with a lower curvature to enable more space for M_1 and M_2 where the maximum number of iterations for M_3 is determined based on the allowable boundaries of the workspace. When the solution is within the allowable boundaries, the algorithm completes path planning with the resulting four maneuvers as the successful parking path as shown in Figure 4.2.

For the analytical solution of maneuvers M_1 & M_2, maneuver parameters and generalized coordinates $\bar{P}$ given the initial and final conditions with regard to M_1 & M_2 are defined as illustrated in Figure 4.5. Using **MATLAB Symbolic Toolbox** and **Mathematica**, solutions for β, $\bar{x}, \bar{y}$ and δ at initial pose of M_1 given the initial conditions $\bar{P}_2$ are derived as in equations below where the total heading difference between initial and final poses of M_1 and M_2 are defined as $\theta_1 = K_1 t_1 V$, $\theta_2 = K_2 t_2 V$ respectively. System of error equations defined in matrix form $\bar{e} = [\,\beta_{\mathrm{e}}, \delta_{\mathrm{e}}\,, \bar{x}_{\mathrm{e}}, \bar{y}_{\mathrm{e}}] = [\beta_{\mathrm{i}} - \beta_0\,, \delta_{\mathrm{i}} - \delta_0\,, \bar{x}_{\mathrm{i}} - \bar{x}_0\,, \bar{y}_{\mathrm{i}} - \bar{y}_0] = 0$ are needed to be solved for K_1, t_1, K_2 and t_2. Since an analytical solution could not be achieved due to the complexity and nonlinearity of the equations, an optimization problem is solved for K_1, t_1, K_2 and t_2

where the cost function defined as $C = \left[{\beta_e}^2 + {\delta_e}^2 + {\bar{x}_e}^2 + {\bar{y}_e}^2 \right]$ is minimized using the Nelder-Mead simplex algorithm [33] until the coordinate errors are within the allowable tolerances. For further improvement, β_i and $\bar{y}_{\mathrm{i}}$ equations are solved for K_2 and t_2 and substituted to $\bar{x}_{\mathrm{i}}$ and δ_i equations where the cost function derived as $C = \left[{\delta_e}^2 + {\bar{x}_e}^2 \right]$ and is used in the minimization problem. Accordingly, this cost function accelerates the convergence to the desired solution of $K_1$and t_1.

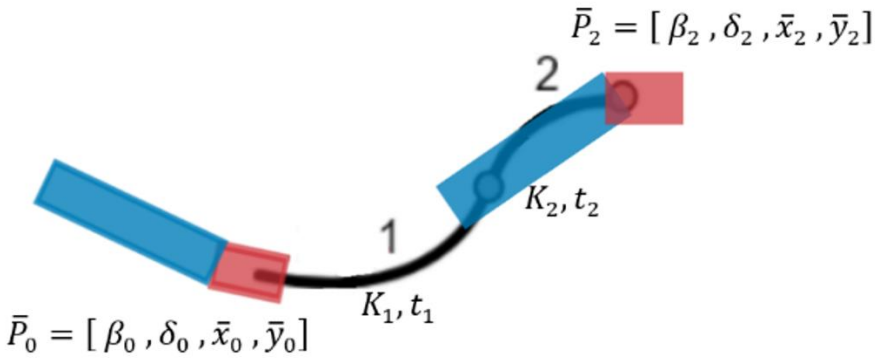


Figure 4.5. Illustration of maneuvers ***M1*** & ***M2***

$$\beta_i = \beta_2 + (\theta_1 + \theta_2) \tag{4.8}$$

$$\begin{aligned} \bar{x}_i = \bar{X}_2 + \frac{1}{K_2}\left[\sin(\beta_2 + \theta_2) - sin\ \beta_2\right] \\ + \frac{1}{K_1}\left[\sin(\beta_2 + (\theta_1 + \theta_2)) - \sin(\beta_2 + \theta_2)\right] \end{aligned} \tag{4.9}$$

$$\begin{aligned} \bar{y}_i = \bar{Y}_2 + \frac{1}{K_2}\left[(\cos(\beta_2 + \theta_2) - \cos\ \beta_2)\right] \\ - \frac{1}{K_1}\left[\cos(\beta_2 + (\theta_1 + \theta_2)) - \cos(\beta_2 + \theta_2)\right] \end{aligned} \tag{4.10}$$

$$\delta_i = \beta_2 + (\theta_1 + \theta_2) - 2\tan^{-1}(D_M) \tag{4.11}$$

where,

$$D_M = \frac{1}{K_1(L_1 - H)}(1 + D_a)\ ,\ D_a = \tan\left(\frac{D_b t_1 V}{2L_1} - \tan^{-1}\left(\frac{D_c}{K_2 D_b}\right)\right),$$

$$D_b = \sqrt{-1 + {K_2}^2\left({L_1}^2 - H^2\right)}\ ,\ D_c = K_2 - K_1 - K_1 D_d \tan\left(\frac{D_d t_2 V}{2L_1} - \tan^{-1}\left(\frac{D_e}{D_d}\right)\right)$$

$$D_d = \sqrt{-1 + {K_1}^2\left({L_1}^2 - H^2\right)}\ ,\ \ D_e = 1 - K_2\left(L_1 + H\tan\left(\frac{\beta_2 - \delta_2}{2}\right)\right)$$

4.3 Closed Loop Rapidly Exploring Random Tree (CL-RRT)

Since the RRT framework samples randomly in the state space and the truck-trailer model refers to nonholonomic constraints, the tracking feasibility of the branches in the expanded tree is not guaranteed unless the system is transformed to a closed loop with a controller that stabilizes the system model. Kuwata et. al. [14] introduced Closed Loop RRT (CL-RRT), where nonlinear closed-loop simulations are performed to compute dynamically feasible trajectories and draw samples in the controller input space. Since forward path tracking for the truck-trailer system is a stable problem like in a kinematic car model, a pure pursuit controller is feasible to use for reverse closed-loop simulations. However, since path following in reverse motion is an unstable problem for the truck-trailer system, a stabilizing controller is necessary as an inner loop to a reverse pure pursuit controller [9, 34, 35].

Entering a narrow parking area with a small error tolerance is hard to achieve with only random sampling and forward simulations, which is similar to the known bug trap problem RRTs face. Hence in this study, a CL-RRT algorithm for a truck-trailer system in the reverse direction is proposed where the tree expansion begins from the parking slot and a solution is searched for reaching the initial state of the vehicle. Starting for path planning reversely from the parking slot assures that the error at the parking slot will be zero and allowed error tolerance is checked between the goal pose and initial vehicle pose, which could be easily handled by the controller at the start of the maneuvering or by an additional connection curve using Dubins path or closed-loop steering [36] from the vehicle pose to the closest feasible path pose. Random sample generation is designed with a probabilistic multi-layer approach where the initial states of the vehicle, the position of the selected parking slot, and the boundaries of the workspace are taken into consideration. Collision avoidance is provided with the same collision check algorithm used in the IAM approach. A further path optimization feature is applied as in the implementation of Elhassan et. al. [8] in order to eliminate

unnecessary branches and to decrease total curvature change in the solution path in order to achieve a smoother solution. The flow diagram of the implemented CL-RRT algorithm is presented in Figure 4.6.

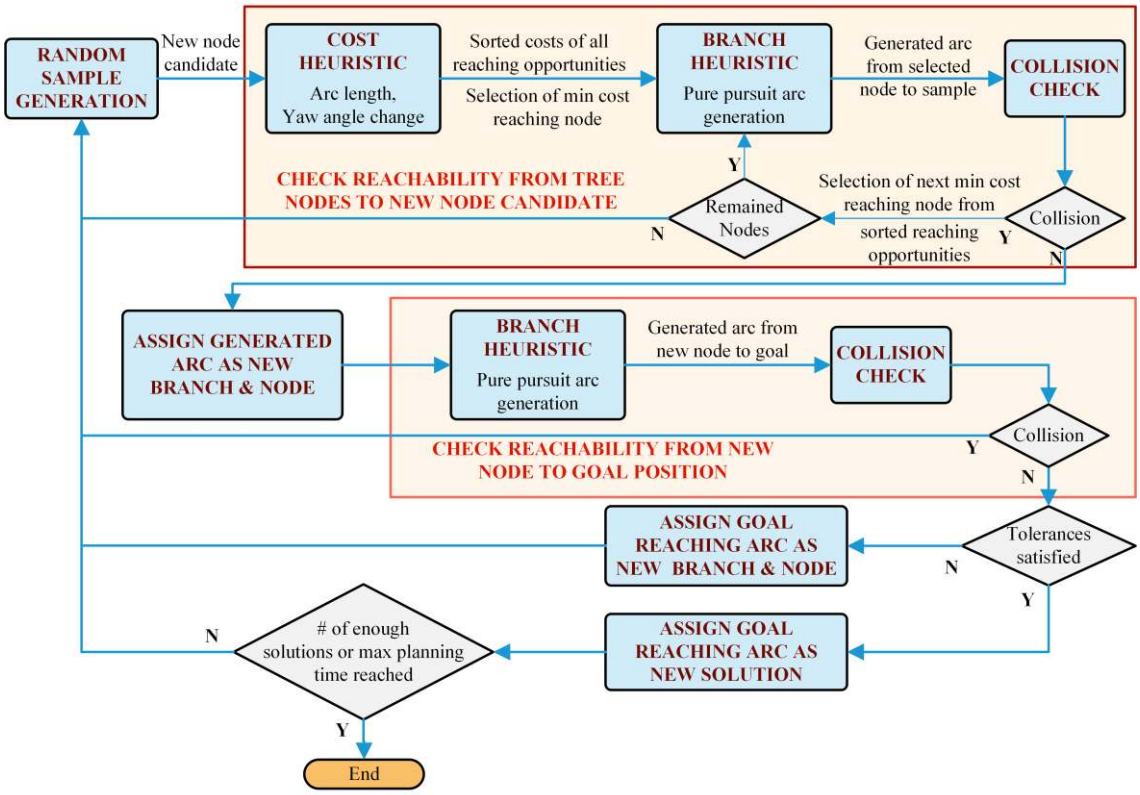


Figure 4.6. Flow diagram: CL-RRT Algorithm: Two levels of planning in each iteration – Assign new node if new node candidate is reachable from tree nodes, assign new solution path if goal position is reachable from the new node

The CL-RRT algorithm performs a tree expansion in each step through randomly generated samples $S_{rnd} = [S_x, S_y]$ in the controller input space defining feasible paths using closed-loop model simulations. The attempts to reach sample points are determined either in forward or reverse direction with a predefined probability in order to provide path planning consisting of both reverse and forward maneuvers.

A cost heuristic is defined for determining the best node to reach the sample node in each iteration where tree nodes are sorted according to the cost heuristic and the lowest cost node N_{min} is selected for reaching attempt. Evestedt et al. [9] defined the cost heuristic for determining the most feasible reaching tree node, N_{min}, to randomly generated point S_{rnd} with a cost function as a sum of path length and angular changes of truck-trailer orientations during the path. Since angular changes are included in the

cost function, a simulation for each node attempt is needed to be performed; hence they introduce an offline generated cost heuristic map to cover all possible simulation outputs by interpolating through the map. In this article, a cost heuristic is defined only considering the length of pure pursuit arcs of branch candidates, L_b, and therefore the cost could be calculated easily without performing any simulation before the selection of the most feasible reaching tree node.

The attempt from the lowest cost tree node to the sample node is designed as a pure pursuit arc and only this arc path is checked first for the collision disregarding the footprint of the truck-trailer system. If the arc path has a collision, the next lowest cost node is selected as N_{min} for the reaching attempt. If the arc path is collision-free, forward or reverse simulation is performed where the simulation time is selected randomly in an allowable interval lower than the expected simulation time to reach S_{rnd} in order to achieve shorter branches to increase the motility of the tree expansion. The truck-trailer paths obtained from the simulation are checked again for the collision considering the footprint of the truck-trailer system. If there is no collision, the final state of the simulation is added as a new node to the tree where the paths of the truck and trailer are also stored as a new branch of the tree. After a successful connection to the new node, a goal-reaching attempt is tested with a probabilistic choice of reverse or forward maneuver direction. For this step, a new simulation is performed, and truck-trailer paths are checked again for collision considering the truck-trailer footprints. If there is no collision and the goal is reached, goal evaluation is performed where the final state of the truck-trailer is checked for the predefined goal requirements. The goal connection is stored as a solution if the goal requirements are fulfilled. Otherwise, a shorter first part of the goal connection trajectory is added to the tree as a new node and branch again in order to increase the motility of the tree. The algorithm stops if a number of enough solutions, Sol_{max}, or maximum allowed path planning time, T_{max}, is reached. The minimum cost solution is selected as the best solution for the path planning based on the solution cost function definition. It is introduced as a function of total trajectory length, angular errors compared to final configuration, and number of maneuver direction change n_c as follows;

$$C = \sum_{i=2}^{N} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} + k_{ang}\left(\left|\beta_N - \beta_f\right| + \left|\delta_N - \delta_f\right|\right) + k_c\, n_c \tag{4.12}$$

where N is the total number of nodes in the trajectory, k_{ang} and k_c are weighting coefficients for angular errors and maneuver direction changes respectively. Penalty for maneuver direction changes pushes the planner to choose a solution with a lower number of direction changes during the trajectory. An example of algorithm implementation is illustrated in Figure 4.7, where blue and green lines depict forward and reverse branches in the tree, and yellow and red curves depict forward and reverse parking maneuvers respectively. Since tree branches are generated through truck-trailer simulations, only G_1 continuity is assured where yaw angle is continuous; however, the curvature has discontinuities at node points [37]. In order to assure G_2 continuity, where the curvature is continuous, and satisfy vehicle dynamics constraints (maximum steering rate, maximum vehicle speed); β-Spline generation in study [38] is implemented as explained in Section 4.6. Further implementation details including random sample generation and parameters for simulation and algorithm are explained in the following section and in Path Planning Simulations.

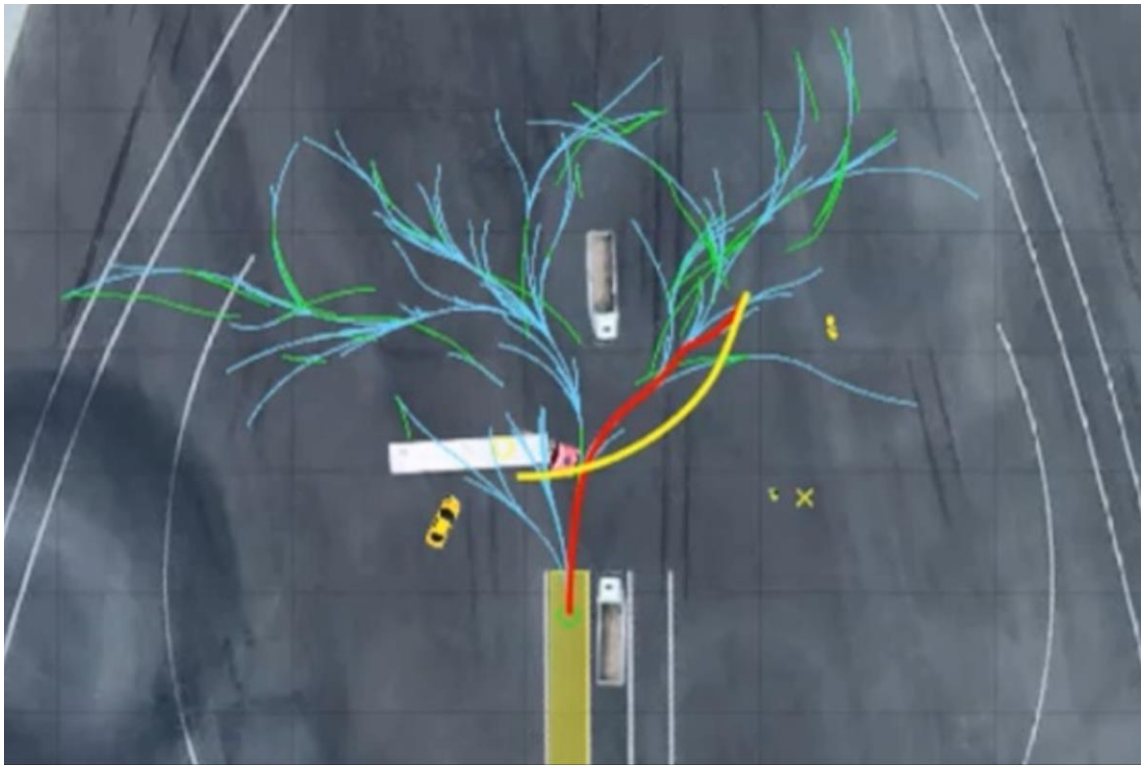


Figure 4.7. Implementation of CL-RRT Algorithm — Tree expansion: forward branches (blue), reverse branches (green), forward solution maneuver (yellow), reverse solution maneuver (red)

4.3.1 *Random Sample Generation*

A uniform sampling is used for exploring the bounded search area where a probabilistic selection of maneuver direction, bias $P_b = [x_b, y_b]$ and standard deviation $[\sigma_x, \sigma_y]$ is proposed. For fast convergence to a feasible solution, a selection of bias approach shown in Table 4.1 is used where tree expansion is partially directed to regions around the workspace midpoint P_{mid}, goal point P_f and an interim point P_m which lies L_m distance far across the line between goal point P_f and start point P_s as illustrated in in Figure 4.8. It is worth noting that L_m is selected randomly between a specified lower and upper bound in each iteration.

Table 4.1:
RANDOM SAMPLING STRATEGY

Direction	Bias	Standard deviation	Probability
Forward	P_{mid}	$\sigma_x = 67.4\ m\ , \sigma_y = 14.7\ m$	0.28
Forward	$\frac{(P_{mid} + P_m)}{2}$	$\sigma_x = (33.7 - 0.8L_m)\ m\ ,$ $\sigma_y = (14.7 - 0.2\ L_m)\ m$	0.52
Reverse	P_{mid}	$\sigma_x = 44.5\ m\ , \sigma_y = 14.7\ m$	0.15
Reverse	$\frac{(P_{mid} + P_f)}{2}$	$\sigma_x = 35.6\ m\ , \sigma_y = 11.8\ m$	0.05

Random sample bias and standard deviation sets are defined with probabilities

Using P_{mid} for bias enables uniform expansion in the bounded search area; on the other hand, the addition of P_m enhances the tree expansion to provide enough space for parking maneuvering where the likelihood of the tree expansion mimicking maneuver M_3 of the previously proposed maneuver set increases. Further addition of P_f for bias helps to direct the tree expansion to the goal region. Values selected in Table 4.1 are obtained through simulations performed for the test field case shown in Figure 4.8.

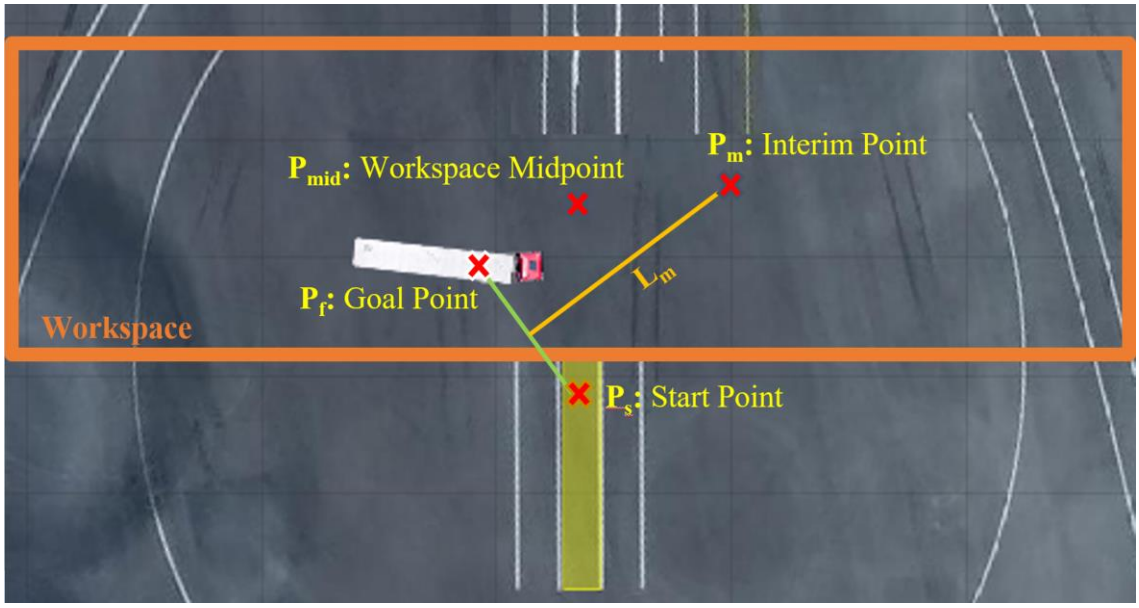


Figure 4.8. Random Sample Generation Approach: Probabilistic bias selection between workspace midpoint P_{mid}, interim point Pm and goal point P_f. P_m is derived from the perpendicular bisector of start point P_s and goal point P_f with a probabilistic distance of L_m

4.3.2 *Path optimization*

Path produced with CL-RRT may be suboptimal due to the introduction of some unnecessary nodes in the path which do not contribute anything but increase the number of turns and length of the path. This also makes the control task more difficult due to the increasing curvature change along the path. El-Hassan et. al. [8] implemented a path optimization algorithm where he uses the Dubins path to check for a possible connection between non-adjacent nodes and try to eliminate unnecessary intermediate nodes as illustrated in Figure 4.9. Dubins path is a one-directional shortest path, consisting of constant curvature arc segments and/or straight path segments, which connects two points in the two-dimensional Euclidean plane with a constraint on the curvature of the path and with the prescribed initial and terminal tangent to the path [39]. It consists of constant curvature arc segments and/or straight path segments. The same approach in [8] is implemented in this study for path optimization since it is computationally cheap.

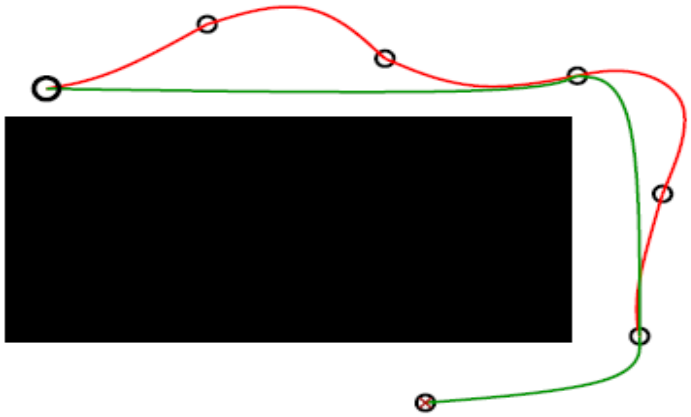


Figure 4.9: Illustration of Path Optimization Algorithm [8]

The algorithm forms a stack of the nodes in the suboptimal path and tries to connect the current node ($n_{current}$) starting from the first node to the target node (n_{target}) starting from the second last node using the Dubins path. Generated Dubins path is checked for collision with Fast Collision Check algorithm used for environmental check (including static obstacles) in CL-RRT expansion as well. If the Dubins path is feasible, intermediate nodes are deleted from the stack, otherwise n_{target} is moved one node back in the inner loop and $n_{current}$ one node forward in the outer loop, and the same procedure is repeated until $n_{current}$ reaches n_{target}.

The pseudocode of the path optimization is shown below:

```
stack := (n1; n2; …; nend);
ncurrent = n1;
ntarget = nend-1;
while ncurrent ~= nend-1 do
    while index(ncurrent) + 1 < index(ntarget) do
        connectionPath ← connectNodes(ncurrent, ntarget);
        if connectionPath is feasible then
            stack ← removeIntermediateNodes(stack, ncurrent, ntarget);
            break;
        else
            ntarget = ntarget-1;
        end
    end
    ncurrent = ncurrent+1;
    ntarget = nend-1;
end
optimizedPath ← connectNodes(stack);
```

Algorithm 3 - Pseudocode of Path Optimization Algorithm [8]

4.3.3 *Online CL-RRT Planning*

An extension of the CL-RRT algorithm for online planning in an environment with dynamic obstacles is introduced by modifying the algorithm with the introduction of several features into the algorithm. These features are explained in the following sections. Also, it is worth noting that, online CL-RRT planning is only implemented in MATLAB simulation environment since environmental information with the existence of dynamic objects could not be introduced in this work either for TruckMaker simulations or field tests. The implementation is shown in the Results section.

4.3.3.1 *Tree Update based on Environment*

The occupancy map is transformed into a hash table, where each cell in the map contains the information of existed nodes in it. Subsequently, the information of the extension nodes from each node is also stored in a tree structure with hash table format. Hence, due to the occupancy map update in changing time frames, nodes of the occupied cells in the map are determined and deleted together with the corresponding branches from the expansion tree with less computational effort thanks to the hash tables.

The mechanism of the tree update function is shown in Figure 4.10. Red squares depict colliding nodes to be deleted where the truck-trailer posed in these nodes collides with a dynamic object. White circles with red crosses indicate the extension of colliding nodes which are also infeasible due to the deletion of colliding nodes. Green crosses refer to the remaining feasible nodes after the deletion of infeasible ones. Blue and green branches are in forward and reverse directions respectively. Effective boxes of dynamic objects are depicted with white big squares. Further, the trajectory of the truck-trailer system is shown with the thick green curve.

4.3.3.2 *Tree Sampling & Feasibility Check*

The algorithm starts with offline CL-RRT for initialization and the vehicle starts to move once the first trajectory is generated. During the maneuvering, a tree update performs in the first step, then tree sampling is performed where a certain number of nodes are sampled from the remaining nodes of the tree based on a normal distribution

in order to keep the size of extension at a certain level. Also, in each time frame, the feasibility of the trajectory is checked using a two-level collision check algorithm.

Figure 4.10: Updating Tree by deleting unfeasible nodes

4.3.3.3 *Replanning with CL-RRT*

Tree expansion is performed in each time frame using offline CL-RRT with the inheritance of the sampled nodes and branches, and new trajectory candidates are generated and stored. The trajectory is changed if and only if the current trajectory is marked as infeasible. Also, note that stored trajectory candidates (with a maximum number of paths limited in the storage) are also checked for collision in each time frame and infeasible ones are deleted. If there is no feasible trajectory including the current one at a certain time frame, the vehicle system starts to make an emergency brake.

4.4 Collision Avoidance

Collision avoidance is achieved by a two-level collision check approach. The first level is for environmental collision check with maneuverable space, and the second is collision check with obstacles as shown in Figure 4.11.

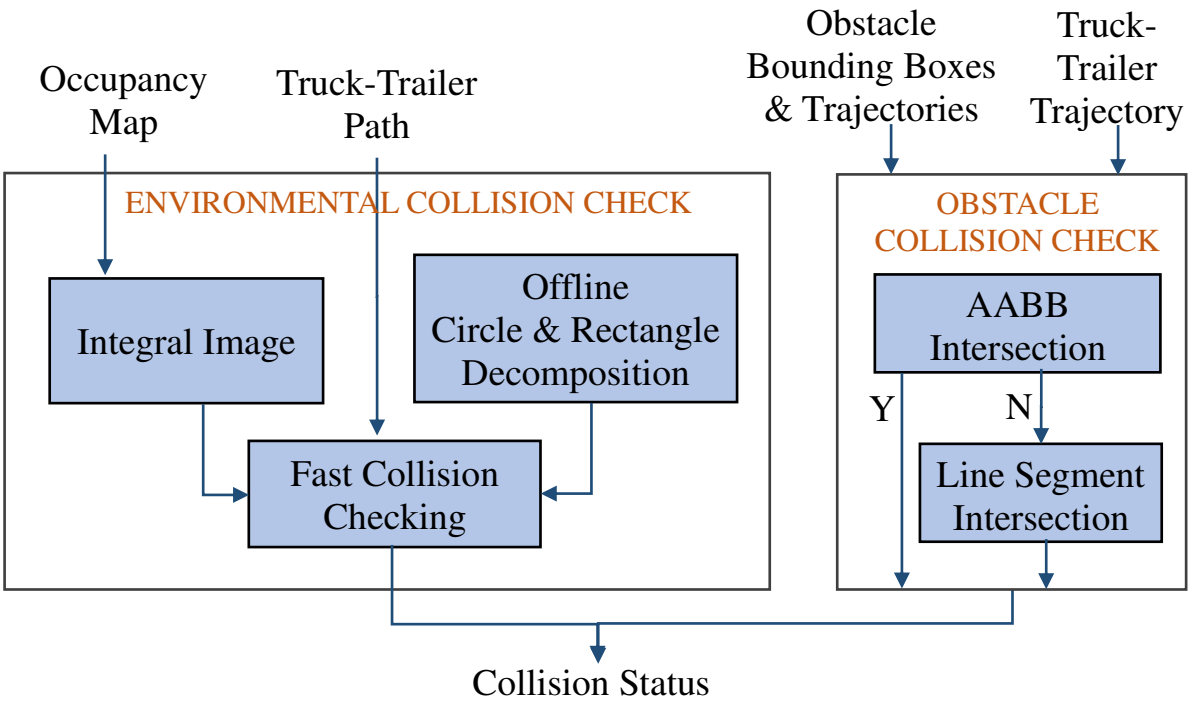


Figure 4.11. Block Diagram of Two-Level Collision Check Approach

For environmental collision check, a circle-rectangle decomposition-based method [40] is used where the shape of the vehicle is first decomposed by circles, and then circles are decomposed into rectangles. Due to the rotational invariance of circles, only the circle center points are needed to be transformed based on the updated vehicle pose, and all decomposed rectangles of circles are located based on the updated circle center positions with the fixed orientation aligned with the free space map of the environment. Examples of circle and rectangle decomposition are illustrated in Figure 4.12.

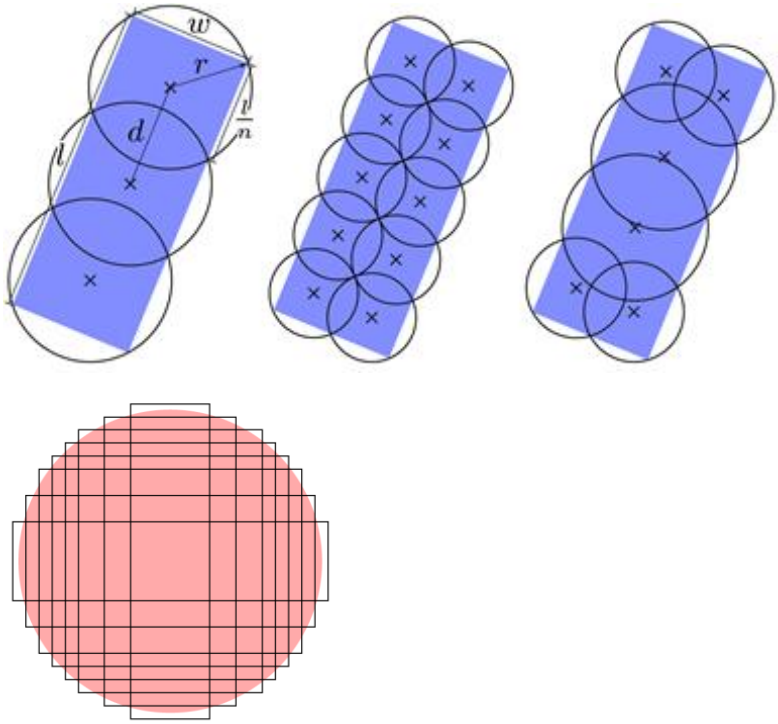


Figure 4.12. Illustration of Circle and Rectangle Decomposition [40]

To check the occupancy of rectangles much faster, an integral image of the occupancy map is calculated as below at each time frame due to the update of the occupancy:

$$I_o(X,Y) = \sum_{x \le X, y \le Y} o(x,y) \tag{4.13}$$

Having the integral image I_o, the occupancy of a rectangular region in the occupancy map could be easily determined through the following arithmetic equation.

$$\sum_{x,y \in A} o(x,y) \; = \; I_o(x_0,y_0) + \; I_o(x_1,y_1) - \; I_o(x_0,y_1) - \; I_o(x_1,y_0) \tag{4.14}$$

If $\sum_{x,y \in A} o(x,y)$ is equal to zero, the rectangle would be collision-free; otherwise, it indicates collision. Hence, this check is done for each of the decomposed rectangles at each pose of the vehicle along a trajectory.

For collision check with obstacles, a cascade algorithm is preferred by combining Axis Aligned Bounding Box (AABB) Intersection and Line Segment Intersection methods. Since AABB intersection is very fast to compute [41], it is first checked between truck-trailer and obstacles as a necessary condition. If there is no intersection via AABB algorithm, the collision check is completed with a collision-free state. Otherwise, line segment intersection is checked between each edge of the oriented rectangular bounding box of truck-trailer and obstacles, where the algorithm stops if an intersection is found [42].

The path planning framework ensures adaptiveness in the case of moving obstacles by providing online collision prediction via a two-level collision check approach and triggering a smooth or emergency braking based on the use case. Depending on the existence of obstacle trajectory predictions through the environmental perception of the system, different approaches could be followed. In the case of predicted trajectories of the dynamic obstacles, the bounding boxes of truck and trailer at each pose along their trajectories could be checked for a possible collision with the obstacle bounding boxes at the corresponding pose along their trajectories within the same time frame. Thus, an online collision prediction could be done at each time step during maneuvering considering the obstacle predictions as well. Conversely, if the obstacle predictions are missing, an additional safety margin is added to the bounding boxes of the obstacles which are updated via environmental perception in each time step without trajectory prediction. In this study, the dynamic obstacles with predicted trajectories are introduced only for MATLAB simulations of Online CL-RRT Planning in Section 6.1.2.4. In future work, it could be extended for TruckMaker simulations and field tests as well.

AABB intersection and Line Segment Intersection methods used for the obstacle check are briefly explained below.

AABB intersection:

One of the common representations for AABBs is the minimum and maximum coordinate values along each axis:

```
// region R = {
(x, y, z) | min.x<=x<=max.x, min.y<=y<=max.y, min.z<=z<=max.z
}
```

```
struct AABB {
Point min; Point max;
};
```

Overlap test between AABBs with min-max representation is straightforward as shown in the code below [41]:

```
int TestAABBAABB(AABB a, AABB b)
{
// Exit with no intersection if separated along an axis
if (a.max[0] < b.min[0] || a.min[0] > b.max[0]) return 0;
if (a.max[1] < b.min[1] || a.min[1] > b.max[1]) return 0;
if (a.max[2] < b.min[2] || a.min[2] > b.max[2]) return 0;
// Overlapping on all axes means AABBs are intersecting
return 1;
}
```

Algorithm 1 - AABB Intersection Algorithm [42]

Line Segment Intersection:

Determining whether two-line segments intersect, the following conditions are checked where one of them must hold for an intersection:

1. Each segment straddles the line containing the other.
2. An endpoint of one segment lies on the other segment.

To implement this, the SEGMENTS-INTERSECT algorithm is proposed in [42] which returns TRUE if segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect. The algorithm calls subroutines DIRECTION and ON-SEGMENT, which computes relative orientation using the cross-product and checks whether a colinear point with a segment lies on it, respectively. The pseudocode of the proposed algorithm is as shown below:

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
d1 = DIRECTION(p3, p4, p1)
d2 = DIRECTION(p3, p4, p2)
d3 = DIRECTION(p1, p2, p3)
d4 = DIRECTION(p1, p2, p4)

if ((d1 > 0 and d2 < 0) || (d1 < 0 and d2 > 0)) &
   ((d3 > 0 and d4 < 0) || (d3 < 0 and d4 > 0))
   return TRUE
elseif d1 == 0 and ON-SEGMENT(p3, p4, p1)
```

```
      return TRUE
elseif d2 == 0 and ON-SEGMENT(p3, p4, p2)
      return TRUE
elseif d3 == 0 and ON-SEGMENT(p1, p2, p3)
      return TRUE
elseif d4 == 0 and ON-SEGMENT(p1, p2, p4)
      return TRUE
else  return FALSE

DIRECTION(pi, pj, pk)
return (pk – pi)x(pj – pi)

ON-SEGMENT(pi, pj, pk)
if min(xi, xj)<= xk <=max(xi, xj) & min(yi, yj)<= yk <=max(yi, yj)
   return TRUE
else return FALSE
```

Algorithm 2 - Line Segment Intersection Algorithm [42]

4.5 Closed Loop Steering Connection

Both IAM and CL-RRT algorithms plan the path from the parking space to the vehicle position. IAM solves a minimization problem within a determined error threshold; on the other hand, CL-RRT reaches a solution path within the predefined yaw and position error tolerances at the start pose of the vehicle. Hence, in either case, the ego vehicle is initially not on track with the generated path. Therefore, a connection curve from starting pose of the ego vehicle to the generated path is needed to provide a smooth transition. The closed-loop steering connection function provides this smooth transition from the vehicle position to the generated path. It simply uses either forward or reverse path following controllers in this work depending on the use case. These methods are explained in the Lateral Control section in detail. The algorithm first checks if the initial yaw and position errors are below a defined threshold. If so, it does not operate since the error is assumed as small so that the controller can handle the transition where the transition would not cause a possible collision. Otherwise, the algorithm simulates the truck-trailer system starting from the initial vehicle pose referencing the generated path. Once the system is under a certain threshold of pose error, the simulation stops, and the simulated path of the truck-trailer system is merged with the untracked part of the generated path. The output of the algorithm is illustrated in Figure 4.13.

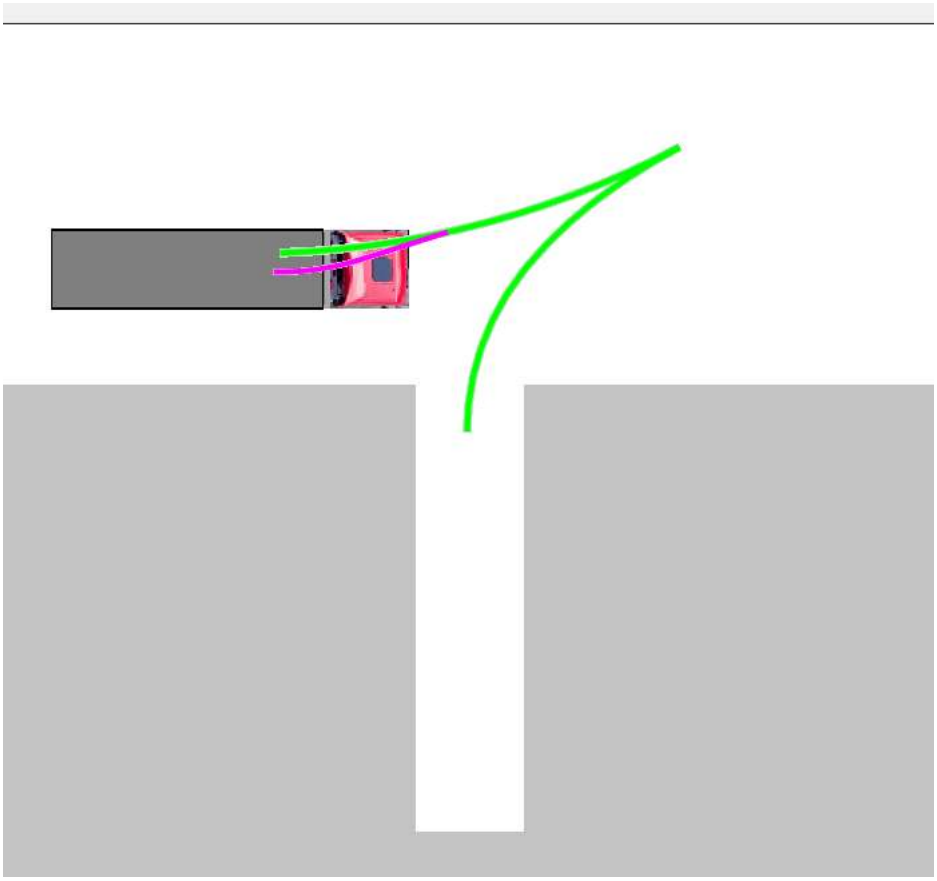


Figure 4.13: Illustration of Closed Loop Steering Connection

Since the simulated path is based on the vehicle model, it assures both slope and curvature continuity in the simulated part, but still not at the merging point and in the remaining untracked part of the generated path. Hence, the newly merged path with zero initial pose error is fed to the β-Spline Generator to transform the path so that slope and curvature continuity is assured along the whole path.

4.6 β-Spline Generator

There is two different notions, parametric (C^k) and geometric (G^k) continuity referring to the smoothness of a curve. Parametric continuity refers to the smoothness of both the curve and its parameterization. On the other hand, geometric continuity refers only to the smoothness of the curve itself. An example of different types of continuity is illustrated in Figure 4.9. In the first case (left), both parametric and geometric continuity is assured, however in the second case (right) only geometric continuity is satisfied since the parameter of the path in terms of speed is discontinuous [43].

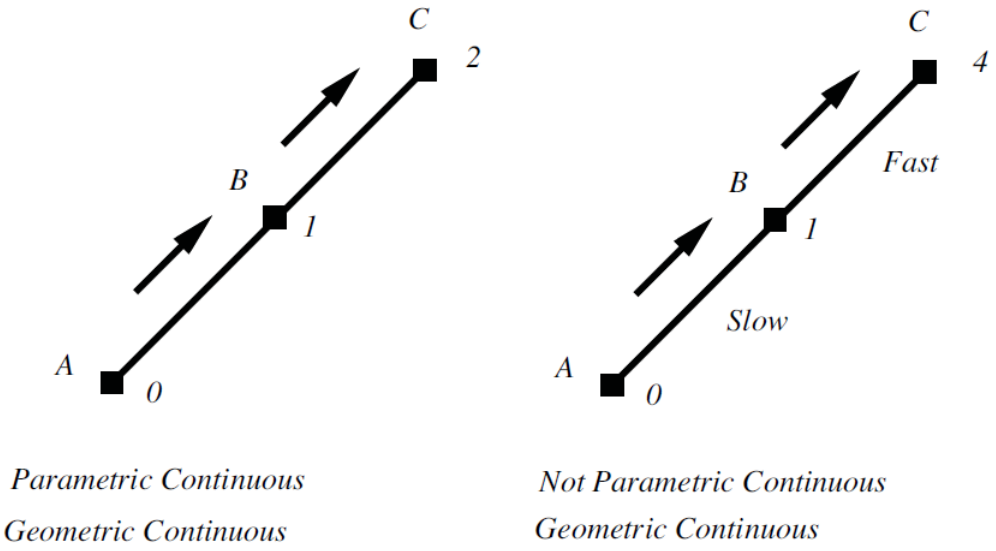


Figure 4.14: Illustration of Parametric and Geometric Continuity [43]

C^1 continuity refers to the continuity of the tangent vector (magnitude refers to the speed) of the path, whereas G^1 continuity refers to the continuity of the slope of the path. Further, C^2 refers to the continuity of the acceleration vector, whereas G^2 continuity refers to the continuity of the curvature of the path. Since parametric continuity deals with more degrees of freedom, it is more expensive to arrange. Also, satisfying parametric continuity at the path planning level is not critical for parking scenarios, since it deals additionally with the continuity and boundaries of vehicle speed and acceleration along the path, and a velocity profile generation part for low-speed maneuvering considering dynamic constraints handles these issues. Hence, dealing with geometric continuity is enough for path planning to check the smoothness of the path. A comparison of different geometric continuity definitions is shown in Figure 4.15.

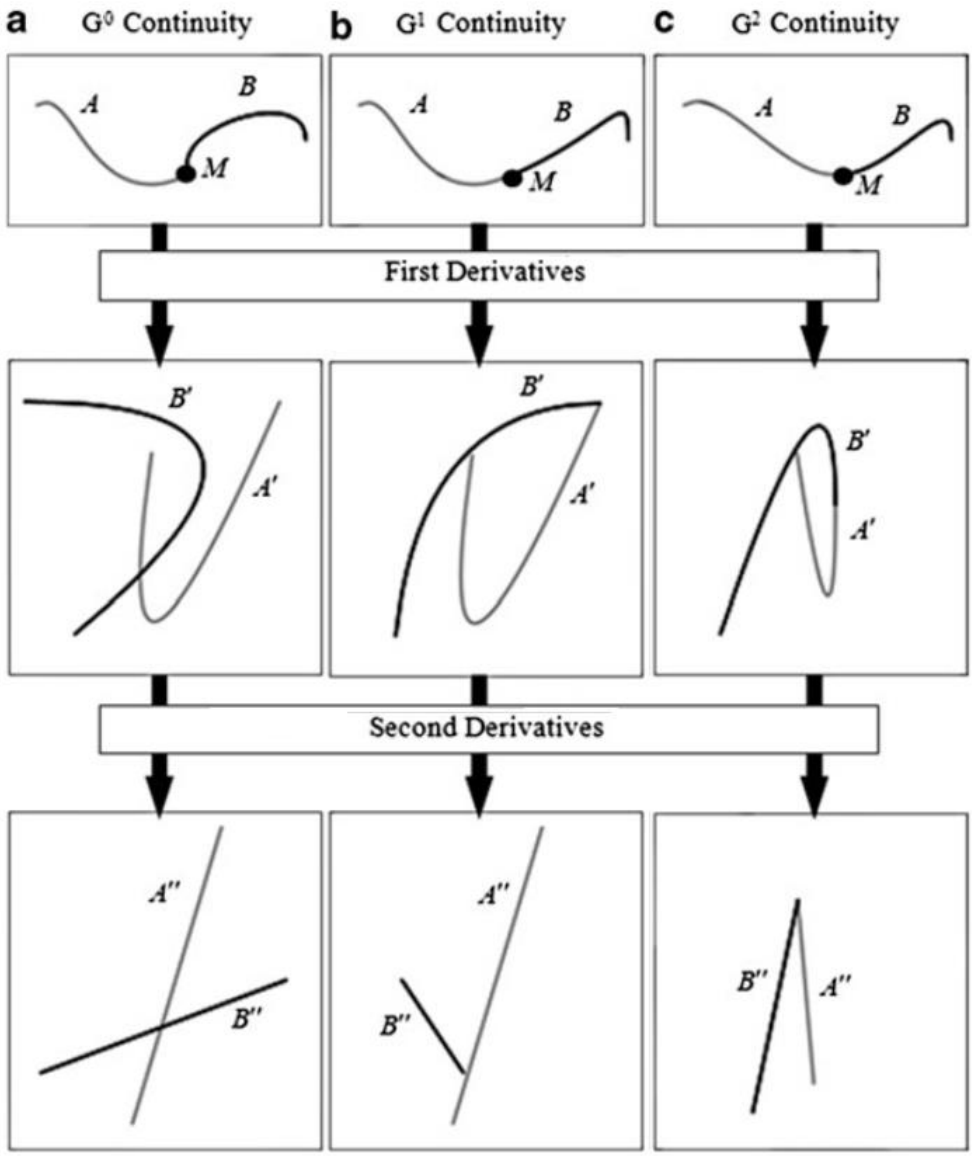


Figure 4.15: Comparison of Geometric Continuity Types

In the case of IAM and CL-RRT based path generation, G^1 continuity, only the slope (yaw angle) continuity of the path is satisfied since the path consists of constant curvature arc segments and there occurs discontinuity in curvature at the nodes of the path which connects different constant curvature arc segments. This would cause an abrupt change in the steering demand at points of discontinuity which would worsen the tracking performance of the lateral controllers. Hence in our implementation, satisfying curvature continuity along the path with a maximum allowed curvature boundary calculated based on the dynamic constraints of the truck-trailer system is needed for path planning to assure feasibility and trackability of the path after the generation of an

appropriate velocity profile for this path. Therefore, to assure G^2 continuity, where the curvature is continuous and satisfies vehicle dynamics constraints (maximum steering rate, maximum vehicle speed); β-Spline generation in the study [38] is implemented. The algorithm takes a set of constant radius arc segments with curvature discontinuity as input (G^1 continuity) and gives continuous curvature path (G^2 continuity) as output satisfying dynamic constraints and boundary conditions of the truck-trailer system. β-Spline curves are defined through a set of points called control vertices, and each set of adjacent four control vertices forms a control polygon, where the boundness of the spline curve within the polygon is assured. Control vertices, control polygon, and β-Spline curve are illustrated in Figure 4.16.

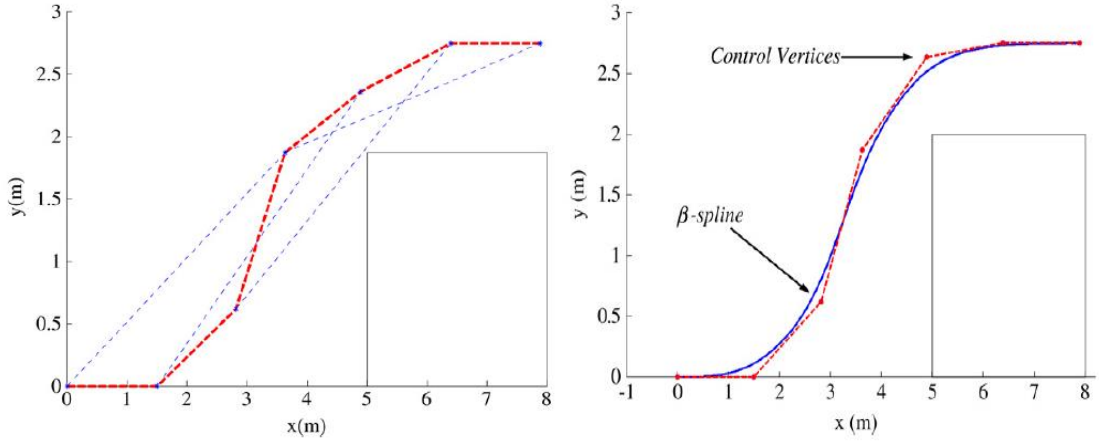


Figure 4.16: Illustration of control vertices, control polygon, and $\boldsymbol{\beta}$-spline curve **[38]**

β -Spline generation is implemented for each set of adjacent four control vertices as in the following equations:

$$Q_i(u) = \sum_{r=-2}^{1} b_r(\beta_1, \beta_2, u) V_{i+r}$$

$$where\ \beta_1 = 0, \beta_2 = 0$$

$$\begin{bmatrix} b_{-2}(u) \\ b_{-1}(u) \\ b_0(u) \\ b_1(u) \end{bmatrix} = \begin{bmatrix} 1/6 & -1/2 & 1/2 & -1/6 \\ 2/3 & 0 & -1 & 1/2 \\ 1/6 & 1/2 & 1/2 & -1/2 \\ 0 & 0 & 0 & 1/6 \end{bmatrix} \text{x} \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \tag{4.15}$$

First, constraints of vertex distance δ (distance between equally spaced vertices) are defined in terms of vehicle dynamics constraints such as maximum vehicle speed, maximum allowed steering angle, and maximum steering rate as shown below.

$$\delta > \delta_{1,\dots,k} \ , where$$

$$\delta_1 = \frac{2R_1}{3}\cos^{-1}\left(\frac{R_{min}}{R_1}\right) , \quad \delta_2 = \frac{2R_2}{3}\cos^{-1}\left(\frac{R_{min}}{R_2}\right)$$

$$\delta_3 = \frac{k_1.|\rho_1|.v.l}{3\dot{\phi}_{max}} \ , \quad \delta_4 = \frac{k_2.(|\rho_1| + |\rho_2|).v.l}{3\dot{\phi}_{max}} \tag{4.16}$$

$$\delta_5 = \frac{k_3.|\rho_2|.v.l}{3\dot{\phi}_{max}}$$

After vertex distance is determined, equally spaced vertices are generated using the start and endpoints of the arc segments. Subsequently, phantom vertices before the start point and after the endpoint of each maneuver are generated in order to maintain the start and final positions of maneuvers after position β-Spline generation. Consequently, β-Spline generation is implemented using defined vertices to generate a continuous curvature path.

4.7 Velocity Profile Generation

Since parking scenarios refer to low-speed maneuvering, it is not critical to have a velocity profile changing along the path perpetually to react optimally to the path parameters such as slope and curvature. Hence, a velocity profile consisting of acceleration, constant speed, and deceleration part satisfying vehicle dynamic constraints is enough to assure feasibility and trackability of the trajectory. For the optimality of the acceleration and deceleration part, the metric of necessary optimality is needed to be investigated. In the study of Wang et. al. [44], different metrics for speed profile design are compared including the third-order, fourth-order, fifth-order, and sixth-order temporal derivatives of the position, which are called the jerk, snap, crackle, and pop, respectively.

The optimal velocity profile design is defined with the following minimization problem with the constraints.

$$\min_{arg} J\big(x(t)\big) = \frac{1}{2}\int_0^{t_f}\left(\frac{d^n x}{dt^n}\right)^2 dt \tag{4.17}$$

$subjected\ to$

$$x(0) = x_0,\ \ x(T) = L$$

$$\frac{dx}{dt}(0) = 0,\ \ \frac{dx}{dt}(T) = 0$$

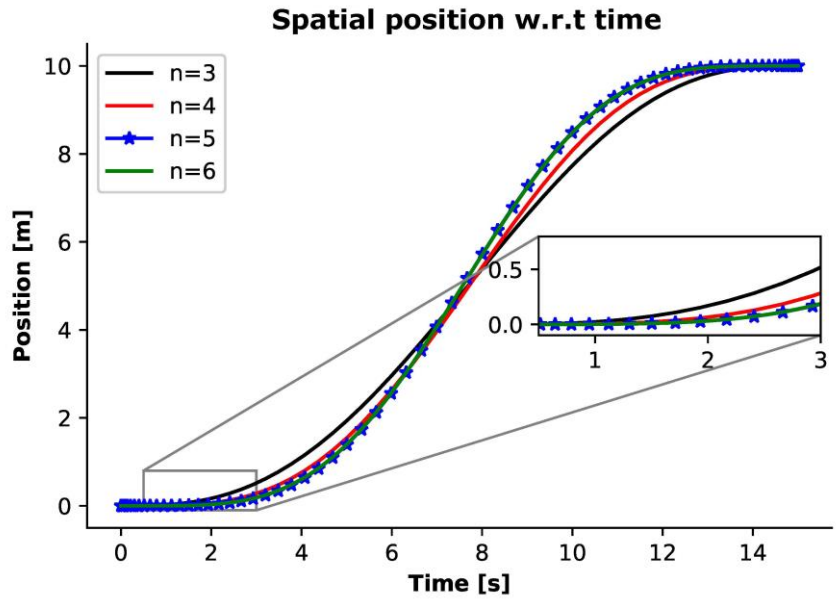
...

$$\frac{d^n x}{dt^n}(0) = 0,\ \ \frac{d^n x}{dt^n}(T) = 0$$

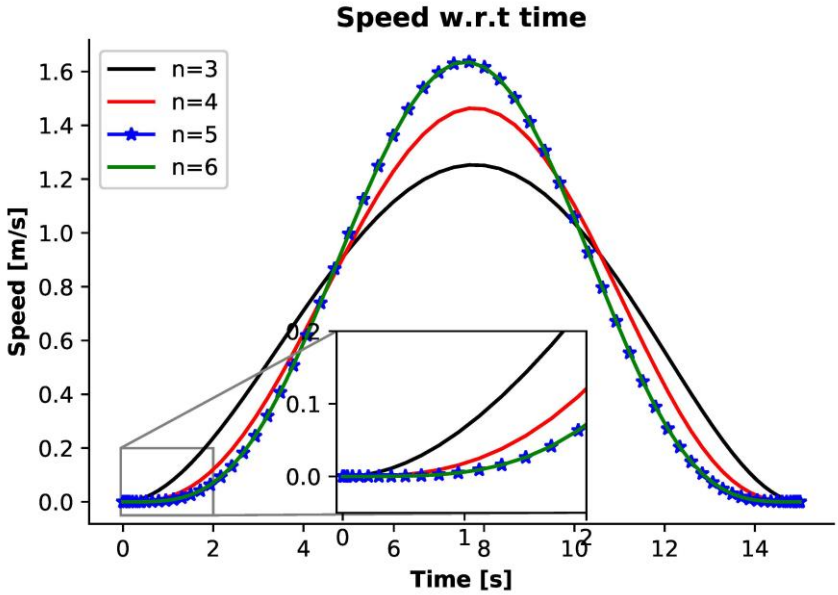
Results for orders from $n = 3\ to\ 6$ are shown in Figure 4.17, where the positional profile is similar for each order, but the speed profiles differ from each other. It is seen from Figure 4.17 that the ratio r between maximum speed V_{max} and average speed V_{avg} is increasing with increasing derivative order n. Ratio values for $n = 2\ to\ 4$ are calculated and shown in Table 4.2, where it is argued that the optimal value for r derived from psychophysical experiments is 1.75 which resembles mostly to minimum jerk profile with $n = 3$. Hence, a jerk optimal velocity profile is an adequate selection for optimality.

Table 4.2:
RELATION BETWEEN DERIVATIVE ORDER AND SPEED RATIO **[44]**

Methods	**Derivative order *n***	**Ratio *r***
Minimum acceleration	2	1.5
Minimum jerk	3	1.875
Minimum snap	4	2.186



(a)



(b)

Figure 4.17: Results of speed profile design with different metrics [44]

For jerk optimal velocity profile generation, the approach of Holmer [1] is implemented in this study, which considers several constraints to generate an optimum velocity profile, which consists of three parts: acceleration, constant speed, and declaration part.

The longitudinal movement of the vehicle is modelled as below;

$$\dot{s} = v \ , \ \dot{v} = a \ , \ \dot{a} = u$$
$$x^T = [v, a] \rightarrow \dot{x} = Ax + Bu$$
$$where$$
$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4.18}$$

where s is the distance moved along the path, v is the velocity, a is the longitudinal acceleration and u is the longitudinal jerk. The optimal velocity profile design is defined with the following minimization problem with the constraints:

$$min \int_0^T \frac{1}{2} u^2 dt$$

subjected to:

$$\dot{x} = Ax + Bu$$
$$v(0) = v_i \ , \ v(T) = v_f \tag{4.19}$$
$$a(0) = 0 \ , \ a(T) = 0$$

Solving the minimization problem, states of the longitudinal velocity profile (s, v, a, u) are derived as below:

$$u(t) = \frac{12}{T^3}(v_i - v_f)t - \frac{6}{T^2}(v_i - v_f) \tag{4.20}$$

$$a(t) = \frac{6}{T^3}(v_i - v_f)t^2 - \frac{6}{T^2}(v_i - v_f)t \tag{4.21}$$

$$v(t) = \frac{2}{T^3}(v_i - v_f)t^3 - \frac{3}{T^2}(v_i - v_f)t^2 + v_i \tag{4.22}$$

$$s(t) = \frac{1}{2T^3}(v_i - v_f)t^4 - \frac{1}{T^2}(v_i - v_f)t^3 + v_i t \tag{4.23}$$

An illustration of the states of the velocity profile specified in the above equations is shown in Figure 4.18.

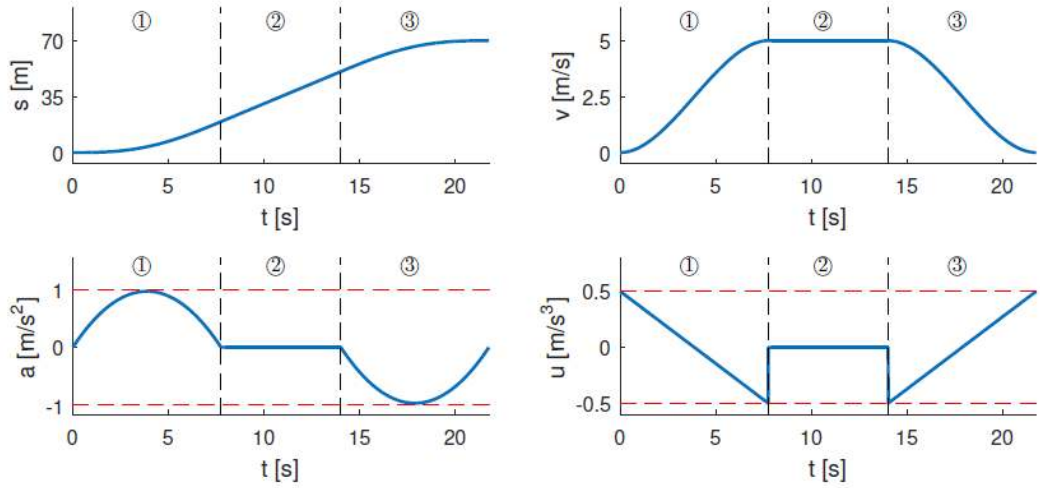


Figure 4.18: Illustration of states of jerk optimal velocity profile [1]

Dynamic constraints are needed to be determined based on the comfortable ride limitations such as $|a(t)| \leq a_{max}$, $|u(t)| \leq u_{max}$, $|a_{lat}(t)| \leq a_{lat_{max}}$ and $|u_{lat}(t)| \leq u_{lat_{max}}$. For a comfortable ride, it is concluded in [6] that maximum longitudinal acceleration a_{max} and maximum longitudinal jerk u_{max} specified as 1.5 m/s^2 and 3 m/s^3 respectively. Subsequently, maximum lateral acceleration $a_{lat_{max}}$ and maximum lateral jerk $u_{lat_{max}}$ for a comfortable ride is specified based on the study [45] as 2 m/s^2 and 0.9 m/s^3 respectively. Further, it is shown in the study [46] that maximum longitudinal acceleration could be represented alternatively as below using maximum lateral jerk constraint:

$$a_{max,1} = \frac{u_{lateral\,max} \cdot R_{min}}{3 \cdot V_f}, \quad where \quad R_{min} = \sqrt{\frac{L_2}{\tan(\alpha_{max})}} \tag{4.24}$$

It is worth noting that the maximum allowed steer angle α_{max} is a physical limitation of the vehicle and is already specified in Table 6.2. Hence, overall maximum longitudinal acceleration $a_{max,2}$ is determined in this work by selecting the maximum of the available constraints as below:

$$a_{max,2} = max\left(\frac{u_{lateral\,max} \cdot R_{min}}{3 \cdot V_f}, a_{max}\right) \tag{4.25}$$

Considering boundary conditions of the maneuver at $t = 0$ and $t = T$ and dynamic constraints such as maximum longitudinal jerk constraint, maximum longitudinal acceleration, and maximum lateral jerk; equations of longitudinal velocity profile yield the following inequalities:

$$T \geq \sqrt{6\frac{|v_i - v_f|}{u_{max}}} \tag{4.26}$$

$$T \geq \sqrt{\frac{3|v_i - v_f|}{2a_{max}}} \tag{4.27}$$

Substituting α_{max} with $\alpha_{max,1}$ in (4.27) yields to the inequality below:

$$T \geq \sqrt{\frac{9(v_i - v_f)^2}{2 \cdot u_{lateral\ max} \cdot R_{min}}} \tag{4.28}$$

Using inequalities in (4.26), (4.27) and (4.28), a T and v_f value should be selected such that the sum of the distance of the path s for acceleration and deceleration parts should not exceed the actual maneuver length.

Chapter 5

MOTION CONTROL

5.1 Control Problem Formulation

The integral of the absolute lateral error, IAE_{lat}, the mean absolute lateral error, MAE_{lat}, and the maximum absolute overshoot lateral error, MO_{lat},, are used as the performance metrics in this paper. IAE_{lat} denotes the accumulated error from the tracked path over time, and this is usually applied for developing a performance measure that targets small settling time. MAE_{lat} evaluates the overall path-following performance and MO_{lat} is a useful judge of the suitability for narrow environments [47]. Oliveira et al. [48, 49] propose an optimization objective, based on both truck and trailer errors, to achieve collision-free path following with a low swept area and a balanced trade-off between maximum cut-in and cut-out. On the other hand, the motivation for the proper selection of the performance metric within the control parameter estimation in this study is to allow the selection of the control parameters that are most likely to increase the stability of the controller. Hence, the performance metrics for control parameter estimation are selected in this study based on the IAE_{lat} calculated as the total accumulated lateral error with respect to the guidance point in each maneuver of the path: road-centering of the truck for forward maneuvers ($e_{lat,tru}$ with respect to the rear axle of the truck), and road-centering of the trailer for reverse maneuvers (e_{lat} with respect to the rear axle of the trailer). Both lateral errors, $e_{lat,tru}$ and e_{lat}, are shown in Figure 3.1. The performance metrics are calculated as follows:

$$
\begin{aligned}
IAE_{lat} &= \sum_{k=1}^{n} \int_{t_{i_k}}^{t_{f_k}} \begin{cases} \left|e_{lat,tru}(t)\right| dt & if\ forward \\ |e_{lat}(t)| dt & if\ reverse \end{cases} \\
MAE_{lat} &= {IAE_{lat}}/{T} \\
MO_{lat_k} &= \max_{t \in [t_{r_k}, t_{f_k}]} \begin{cases} \left|e_{lat,tru}(t)\right| & if\ forward \\ |e_{lat}(t)| & if\ reverse \end{cases} \\
MO_{lat} &= max\left(\left\{MO_{lat_k} : k = 1, \dots, n\right\}\right)
\end{aligned}
\tag{5.1}
$$

where n is the number of maneuvers in the path, t_{i_k}, t_{r_k} and t_{f_k} are the initial, rise, and final time for the k[th] maneuver, respectively, with MO_{lat_k} being the maximum lateral overshoot along with the kth maneuver and T the total operational time.

The control problem of the docking task of a truck-semitrailer system is formulated with the combination of a terminal acceptably/sufficiently small error for the whole configuration of the truck-trailer system while bounding maximum overshoot and mean absolute error in the lateral direction throughout the docking path. The acceptable final lateral error is defined as being $ef_{lat} < 0.10\ m$, the final truck and trailer orientation errors as being $ef_{\beta} < 3\ deg$ and $ef_{\delta} < 3\ deg$, the maximum lateral overshoot as being $MO_{lat} < 0.4\ m$ and the mean absolute lateral error as being $MAE_{lat} < 0.10\ m$. Successful parking is considered to be a final configuration within the limits of acceptable errors, with an off-path track error within acceptable overshoot. To achieve defined error conditions, the feasible collision-free reference path to a target parking location is first planned, and then the following subtasks are defined as follows for the control problem formulation:

1) Design an adaptive feedback controller that targets the rear axle of the truck as the guidance point to follow the maneuvers of the planned path in a forward direction.
2) Design an adaptive feedback controller that targets the rear axle of the semitrailer as the guidance point to follow the maneuvers of the planned path in a reverse direction.

The error definitions for forward and reverse path-following controllers are done as being the lateral errors $e_{lat,tru}$ and e_{lat}, respectively.

5.2 Lateral Control

5.2.1 *Forward Path Following*

Amidi et. al. [12] proposed a pure pursuit controller, where a goal point (g_x, g_y) is determined via a look-ahead distance L_d, referenced from the vehicle's rear axle $(\bar{x}, \bar{y})$, to intersect the desired path as illustrated in Figure 5.1. The steering angle $\alpha(t)$ is then calculated using the look-ahead heading error θ_{h}, which is obtained via goal point location (g_x, g_y):

$$\alpha(t) = tan^{-1}\left(\frac{L_2}{R(t)}\right), \quad where \quad R = \frac{L_d}{2\sin\theta_h(t)} \tag{5.2}$$

Since the following of the path is referenced through the rear axle of the truck for a truck-trailer system in forward motion, it is possible to directly transfer the kinematic model of a car to the truck-trailer system. Hence, the vehicle rear axle in the bicycle vehicle model in Figure 5.1 can be directly conveyed to the rear axle of the truck, as shown in Figure 3.1.

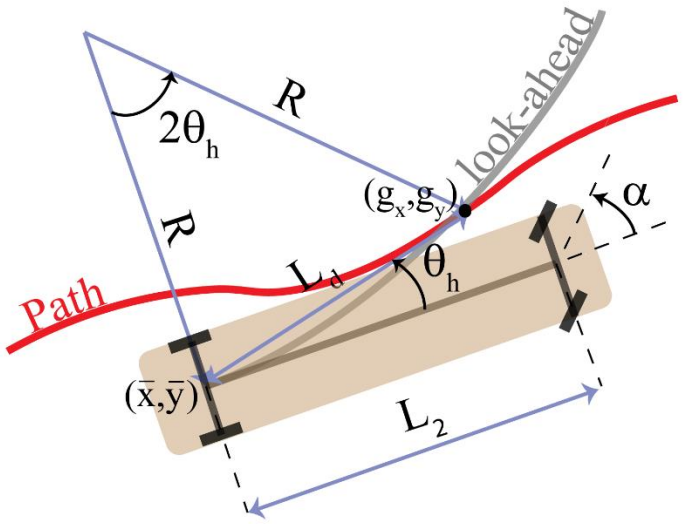


Figure 5.1. Pure pursuit geometry for a bicycle vehicle model

Lombard et al. [17] extended the pure pursuit algorithm with an additional empirical coefficient $(1 - k_s S)$ to reduce the "cutting corners" effect without reducing the look-ahead distance. It is worth noting that increasing the look-ahead distance increases the "cutting corner" effect while reducing it increases oscillations on the driven path [20]. S is defined as the surface area between the reference path and the pure pursuit arc, as illustrated in Figure 5.2, and k_s is an empirical coefficient determined through experiments as being 0.03 in the implementation performed in this study. The steering angle is calculated as follows:

$$\alpha(t) = \tan^{-1}\left(\left(1 - k_s S(t)\right)\frac{L_2}{\mathrm{R}(t)}\right), \tag{5.3}$$

where the empirical coefficient pushes the pure pursuit algorithm to decrease the surface area S, while improving path-following performance. To normalize the effect of

the area surface in this study, an extended version of the algorithm is proposed in which the area is divided into the square of the look-ahead distance L_d as below:

$$\alpha(t) = \tan^{-1}\left(\left(1 - \frac{k_s S(t)}{{L_d}^2}\right)\frac{L_2}{\mathrm{R}(t)}\right) \tag{5.4}$$

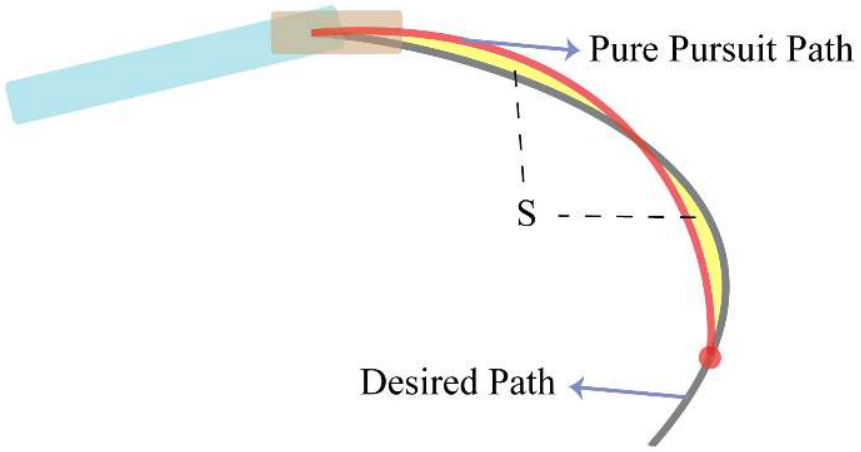


Figure 5.2. Lombard Algorithm Illustration [17]: truck (red), trailer (blue)

Since greater look-ahead distances increase the damping of the system and improve stability [19], a variable look-ahead distance is introduced as in (5.5), where L_d is linearly changed for the lateral path-following error e_{lat}, and a vehicle velocity v exists in an allowable interval between L_l and L_u. This means that L_d is increased for increased velocity and path-following error in order to decrease aggressive control and maintain stability by avoiding jackknifing and overshooting path-following errors.

$$L_d(t) = \begin{cases} L_l & L_e(t) < 0 \\ L_l + L_e\,(t) & otherwise \\ L_u & L_e(t) > L_u - L_l \end{cases} \tag{5.5}$$

$$where\ L_e(t) = k_v\, v(t) + k_e\, e_{lat}(t)$$

A closed-loop system of the proposed forward path-following controller with the truck-trailer kinematic model (3.1) is simulated in MATLAB/Simulink by first varying parameters k_s and L_d. Simulations are then performed by varying k_γ and k_P to visualize

the effect of Q gain scheduling over path-following performance. The simulations are performed by referencing a set of paths that are generated via the cascade path planning framework. To visualize the behavior of k_s to ensure minimum path-following error along L_d axis, the effect of L_d on path-following error is excluded by normalizing the path-following performance metric over L_d. Hence, for each $[k_s, L_d]$, the normalized integral of the absolute lateral error, $\overline{IAE}_{lat}$, is calculated along each path to form a subset of $\overline{IAE}_{lat}$ values for that particular $[k_s, L_d]$ pair. Next, the mean of the $\overline{IAE}_{lat}$ subset values for all $[k_s, L_d]$ pairs is plotted over k_s and L_d, as shown in Figure 5.3a. The red line indicates the minimum curve of the surface interpolated through the minimum tuple of the grid using a natural cubic spline.

$$\overline{IAE}_{lat} = \frac{\int_{t_i}^{t_f} \left| e_{lat,tru}(t) \right| dt}{L_d} \tag{5.6}$$

Lower and upper bounds for look-ahead distance, L_l and L_u, are also estimated, based on these simulation results. As soon as a feasible k_s gain for low lateral error is determined that covers all possible L_d selections, simulations are performed by varying k_v and k_e to visualize the effect of L_d gain scheduling over the path-following performance. Hence, the mean of the IAE_{lat} subset values for all $[k_v\ ,\ k_e]$ pairs are plotted over k_v and k_e, and shown in Figure 5.3b. The red line indicates the minimum curve of the surface interpolated through the minimum tuple of the grid using natural cubic spline, and this is used for estimating parameters k_v and k_e.

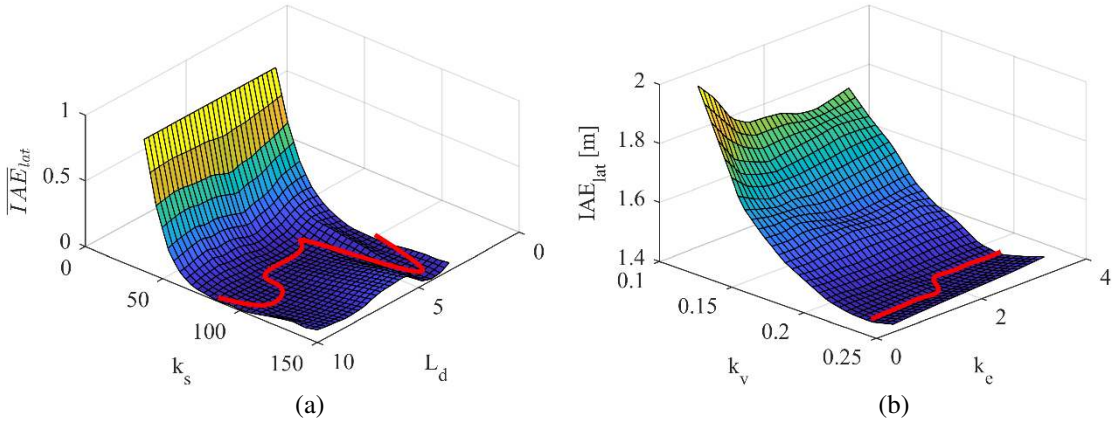


Figure 5.3. a) The effect of k_s gain on $\overline{IAE}_{lat}$ for changing look-ahead b) Effect of k_v and k_e gains on IAE_{lat}

5.2.2 *Reverse Path Following*

A cascade controller of reverse pure pursuit and an LQ controller is introduced in [31] for a two-trailer system in which the feed-forward term is achieved by a reverse pure pursuit controller, using the trailer rear axle as the guidance point for reference. The LQ control provides the closed-loop feedback term for the stabilization of internal angles. The proposed controller in this work is an enhanced version of the one-trailer system implementation of the cascade controller which utilizes Q gain scheduling for improved robustness and path-following performance. The robustness and path-following performance are compared in the robustness verification.

For the one-trailer implementation of the cascade controller for reverse pure pursuit and the LQ controller, a reverse pure pursuit algorithm is primarily defined with the adaptive look-ahead distance described in (5.5). Hence, the steering angle α_e needs to be represented in terms of γ_e. This algorithm considers the trailer rear axle to the desired path in a reverse motion and uses the same formulation in (5.2), although outputs from the hitch angle γ takes the control signal as shown in (5.7), rather than the steering angle α in forward motion. Hence, the representation of α in terms of γ is necessary and is implemented through the geometric transformations at the equilibrium point explained in the LQ design below.

$$\gamma(t) = tan^{-1}\left(\frac{2L_2 \sin\theta_h(t)}{L_d}\right) \tag{5.7}$$

The linearization of the one-trailer system around an equilibrium point, and the design of the LQ controller with gain scheduling, will now be explained in detail. The general steps for the linearization of a two-trailer system are introduced in [50, 31]. The implementation for the one-trailer system is presented in (5.8), (5.9), and (5.10), in which $\gamma = \beta - \delta$ is the measured hitch angle, γ_e and α_e are respectively the constant hitch angle and steering angle at the linearization point.

$$\dot{\gamma} = v(t)(\overline{A}\big(\gamma(t) - \gamma_e\big) + \overline{B}(\alpha(t) - \alpha_e)) \tag{5.8}$$

$$\overline{A} = -\frac{1}{L_1}(\cos\gamma_e + \frac{H}{L_2}\sin\gamma_e \tan\alpha_e) \tag{5.9}$$

$$\bar{B} = \frac{1}{L_2}(1 + \frac{H}{L_1}\cos\gamma_e)(1 + \tan^2\alpha_e) \tag{5.10}$$

LQ controller defines the steering angle as a function of α_e, γ_e and the optimal gain $L(\alpha_e, \gamma_e)$ which minimizes the quadratic cost function F of the linearized model, using Q as the design parameter, as shown in equations (5.11), (5.12).

$$\alpha(t) = \alpha_e - L(\alpha_e, \gamma_e)(\gamma(t) - \gamma_e)) \tag{5.11}$$

$$F = \int_0^\infty (Q(\gamma(t) - \gamma_e)^2 + (\alpha(t) - \alpha_e)^2)\, dt \tag{5.12}$$

Solving $L(\alpha_e, \gamma_e)$ for minimizing the cost function F, using the Riccati equation, yields:

$$\mathrm{L}(\alpha_e) = \bar{A} - \frac{\sqrt{\bar{A}^2 + \bar{B}^2 Q}}{\bar{B}} \tag{5.13}$$

$L(\alpha_e, \gamma_e)$ is solved at each step for a different linearization point, where LQ control operates as a stabilizing controller for the higher-level reverse pure pursuit path-following algorithm. The reverse path-following control generates only γ_e as the control signal by using (5.7) at each linearization point. Hence, the steering angle α_e needs to be represented in terms of γ_e, which is achieved in (5.14) through transformations using the geometry as shown in Figure 5.4.

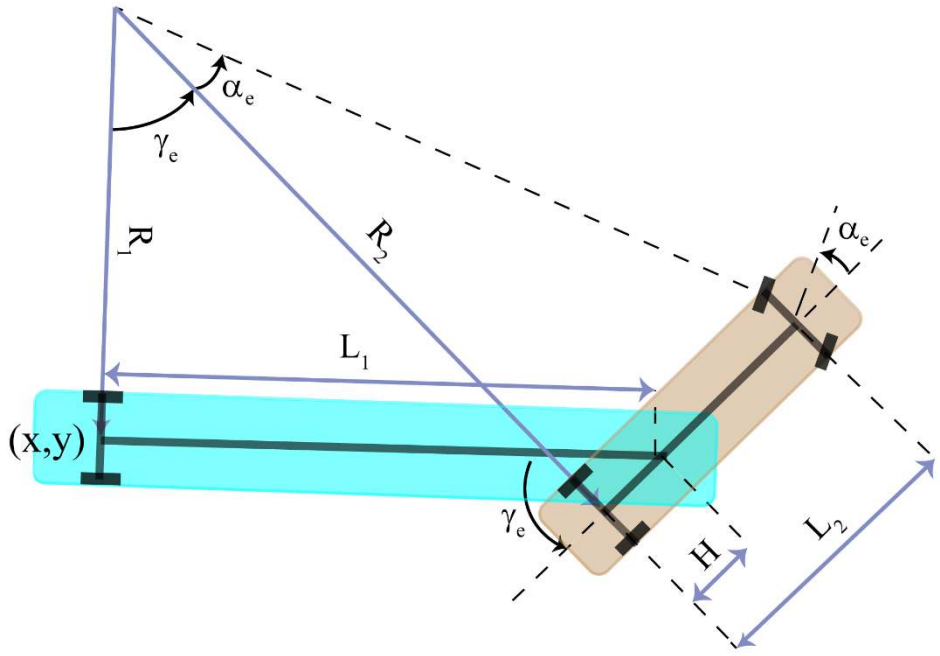


Figure 5.4. Truck-trailer geometry at the linearization point

$$\alpha_e = \tan^{-1}\left(\frac{L_2}{\sqrt{{R_1}^2 - H^2 + {L_1}^2}}\right) = \tan^{-1}\left(\frac{L_2 \sin\gamma_e}{L_1 - H\cos\gamma_e}\right) \tag{5.14}$$

Q as a design parameter determines the stability and persistence of the path-following process. Maintaining scheduling control by varying Q results in an interval, based on truck-trailer initial configuration and path curvature, which improves the reverse path-following performance as opposed to using a constant value. For an adequate look-ahead distance L_d, and low path-following errors, a more persistent path-following is preferred for improved path-following performance in which Q is selected with a greater value. On the other hand, for a large off-track from the desired path e_{lat}, or a large hitch angle γ, a lower Q value is needed to maintain stability by avoiding jackknifing or divergence from the path. The empirical parametrization of Q for gain scheduling is presented as follows:

$$\mathrm{Q}(t) = \begin{cases} Q_l & Q_e(t) < 0 \\ Q_l + Q_e(t) & otherwise, \\ Q_u & Q_e(t) > Q_u - Q_l \end{cases} \tag{5.15}$$

where

$$Q_e(t) = k_L\, L_d - k_P \left(|e_{lat}(t)| + k_\gamma\, |\sin \gamma(t)| \right)$$

Q gain is formulated linearly to the look-ahead distance L_d and lateral error parameter k_P, where the effect of hitch angle is also transferred only in the lateral direction by multiplying the hitch angle parameter k_γ with $|\sin \gamma(t)|$. Q is subsequently decreased for a lower L_d, a larger lateral error e_{lat}, and a larger hitch angle γ to maintain stability. Conversely, it is increased to improve persistence for larger L_d, lower lateral error e_{lat}, and lower hitch angle γ to increase convergence and path-following performance.

The closed-loop system of the proposed reverse path-following controller with the truck-trailer kinematic model (3.1) is simulated in MATLAB/Simulink, which depicts a set of paths generated via the cascade path planning framework. k_L is determined through simulations for optimal constant Q gain, where L_d is selected as a fixed value, and constant Q gain is varied through simulations. Simulations are subsequently performed by varying k_γ and k_P to visualize the effect of Q gain scheduling over path-following performance. This means that for each $[k_\gamma, k_P]$, the IAE_{lat} is calculated along each path to form a subset of IAE_{lat} values for that particular $[k_\gamma, k_P]$ pair. The mean of the IAE_{lat} subset values for all $[k_\gamma, k_P]$ pairs are then plotted over k_γ and k_P, as shown in Figure 5.5. The red line indicates the minimum curve of the surface interpolated through the minimum tuple of the grid, using natural cubic spline, and is used for the estimation of the parameters k_γ and k_P. L_d is then also defined as in (5.5). Hence, the steering angle α_e needs to be represented in terms of γ_e for reverse path-following, while the fixed value of L_d used in the parameter estimation simulations is set to L_u. Lower and upper bounds for Q gain, Q_l and Q_u, are also estimated, based on these simulation results.

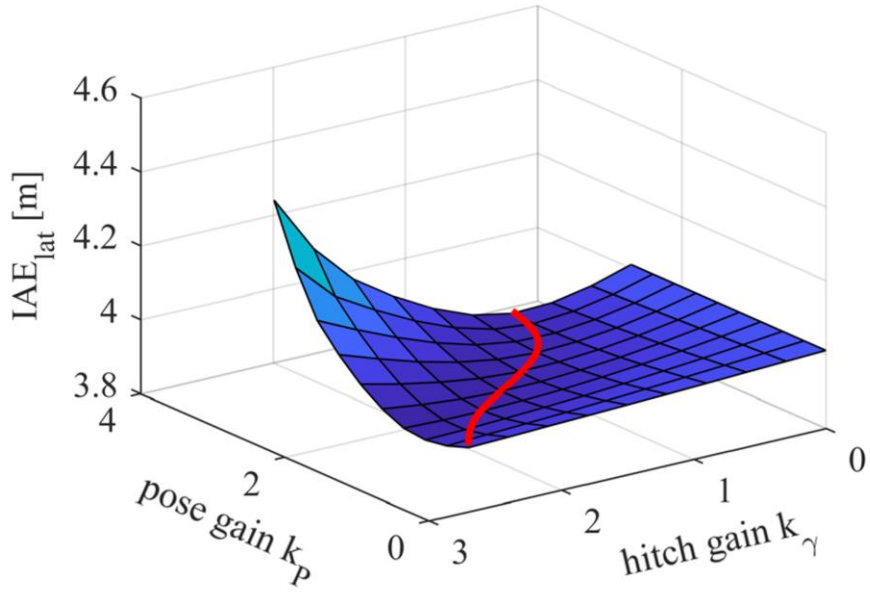


Figure 5.5. The effect of varying Q, based on k_P and k_γ gains, on IAE_{lat}

It is worth noting that the parameter estimations for both forward and reverse path-following controllers are conducted offline, however, the simulations for the parameter estimations are performed by referencing a set of 100 different paths consisting of constant curvature and straight segments with a fixed initial error state, in terms of truck trailer orientations, and with a lateral error to cover a significant portion of cases encountered. Subsequently, considering the critical hitch angle [51] for reverse path-following, and the negative effect of the slip on the kinematic model assumptions for forward and reverse path-following, the hitch angle is constrained with an allowed boundary of $[-45, 45]\ deg$ for both forward and reverse simulations. furthermore, the absolute steering angle rate is constrained with a maximum value of $34.37\ \mathrm{deg/s}$ to reflect the limitations of the actual steering assist system. the simulations for control parameter estimation are performed based on the vehicle parameters of the semi-trailer system used in the field tests and TruckMaker parking simulations as shown in Table 5.1, with vehicle speed v being set to $2.78\ \mathrm{m/s}$ and $-1.39\ \mathrm{m/s}$ for forward and reverse path-following simulations, respectively. the estimated controller parameters for TruckMaker parking simulations and field tests are shown in Table 5.2.

Table 5.1:
VEHICLE PARAMETERS FOR PARKING SIMULATIONS AND FIELD TESTS

Parameter	Symbol	Value
the wheelbase of the trailer	L_1	9 m
the wheelbase of the truck	L_2	3.6 m
off-axle hitch length	H	0.6 m

Table 5.2:
CONTROLLER PARAMETERS

Forward Path-Following Control	Value	Reverse Path-Following Control	Value
$\boldsymbol{k_s}$	97.5	k_L	0.55
$\boldsymbol{k_v}$	0.22	k_γ	1.12
$\boldsymbol{k_e}$	1.80	k_P	1.87
$\boldsymbol{L_l}$	3 m	Q_l	1.0
$\boldsymbol{L_u}$	10 m	Q_u	11.0

5.3 Longitudinal Control

Longitudinal control is handled with acceleration and deceleration requests using PD and PI controllers respectively to follow the generated velocity profile. Hence, the proposed longitudinal control consists of two parts; Gas and Brake Pedal Setpoint Control. Proposed gas pedal setpoint control is performed via PD control of the truck speed error as shown in Figure 5.6. Proposed brake setpoint control is activated when the vehicle reaches the deceleration part of the velocity profile in the trajectory. Once it is activated, it calculates the desired deceleration based on the equation:

$$a = \frac{v^2}{2s_e} \tag{5.16}$$

Then, desired truck speed based on the remaining distance is calculated using the desired deceleration value, and the truck speed error is obtained and fed to the PI

controller as shown in Figure 5.7. Since the setpoint determined in pedal setpoint control is negative in this case, the minimum of the two setpoints (setpoints from the gas pedal controller and brake pedal controller) is selected as the gas pedal setpoint.

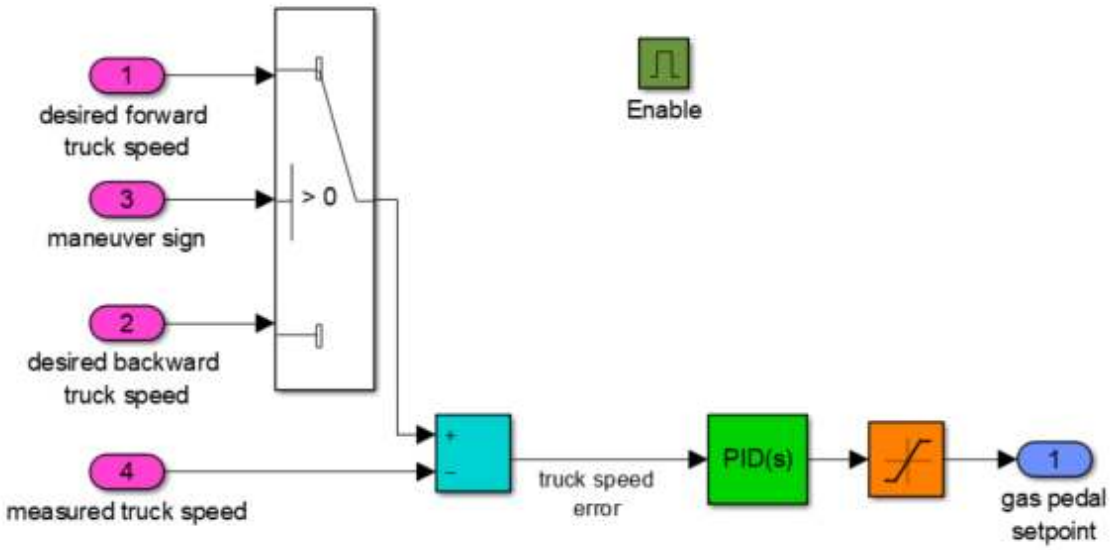


Figure 5.6. Gas Pedal Setpoint Control

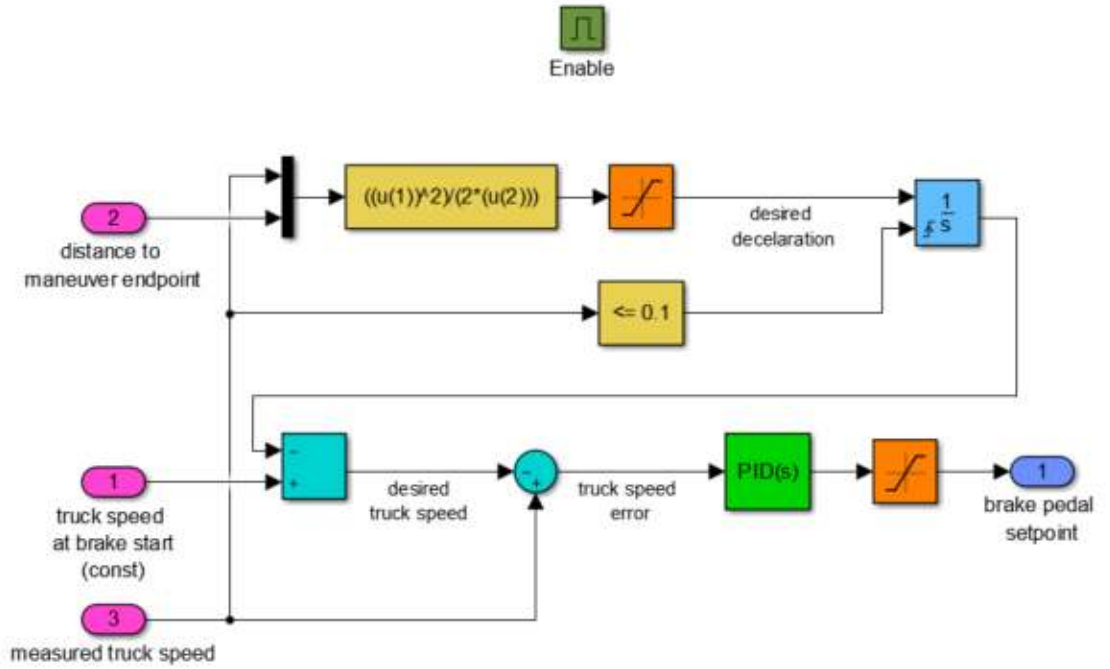


Figure 5.7. Brake Pedal Setpoint Control

Gas and brake setpoint control were implemented together successfully in MATLAB simulations to follow the trajectory with generated velocity profiles for each maneuver. Due to the technical limitations of the gas pedal control on the real full-scale truck-trailer and TruckMaker simulations, generated velocity profile could not be implemented in these cases. Hence, longitudinal control is handled by determining an allowed constant maximum longitudinal velocity and setting it as the setpoint to the cruise control of the vehicle for each maneuver and using the proposed brake pedal control only. Since the generated velocity profile is not presented, brake pedal setpoint control is activated when the remaining distance to the maneuver endpoint s_e is under a certain threshold value.

5.4 Stability & Robustness Verification

5.4.1 *Forward Path Following*

During forward motion, it is possible for the stability analysis of the controller in a kinematic car model to be directly conveyed to the truck-trailer system. This is because the path-following procedure is referenced through a truck's rear axle. During classic pure pursuit controller referencing of a straight or constant curvature path in forward motion, a stability analysis of the linearized system around the equilibrium state has been performed in [52] by applying the Routh-Hurwitz criterion. Stability is guaranteed when constant look-ahead distance is bounded by a function of time delay and path curvature.

Ollero et al. [53] presented straight path referencing for a modified pure-pursuit algorithm by introducing a different anchor point to the rear axle for referencing. They achieved a stability criterion of bounded look-ahead distance as a function of vehicle speed, actuator time constant and anchor point distance. As the actuator time constant and the anchor location are fixed for a vehicle system, it is possible for the look-ahead distance to be defined as a function of vehicle speed above the linear stability parameters. It is shown in [19] that classical and dynamic pure pursuit methods with adaptive look-ahead distance as a function of vehicle speed can be represented as linearized PD controllers during the path-following error process, and that, in such methods, the stability of the closed-loop system is shown via pole-zero plot.

Further analysis has been performed for nonlinear modifications to the pure pursuit algorithm by adjusting look-ahead distance with path-following error. It is concluded from this analysis that this nonlinearity could not be easily approximated since the look-ahead distance lies in the denominators of the linearized gains. It is therefore argued that while this type of approach is not well suited for analytical analysis, it is still a powerful tool to improve stability since a greater look-ahead distance increases the damping of the system. Increased path-following errors or vehicle speed demands a greater look-ahead distance to stabilize the system which is provided by this type of approach. The proposed algorithm in this work also uses a modified pure pursuit controller with adaptive look-ahead distance adjusted by vehicle speed and path-following error. In this study, stability verification is thus introduced by relying on numerical simulations in which an estimation of the stability region is targeted by varying initial states over a grid, and by checking the convergence to a straight path and S-shape path.

Extensive MATLAB and TruckMaker simulations with various initial conditions have been performed. MATLAB/Simulink simulations are based on the closed-loop system of the proposed path-following controller with the truck-trailer kinematic model (3.1). Since the kinematic model used in MATLAB simulations is insufficient by itself to hold the real system, TruckMaker simulations are also introduced. These are based on the TruckMaker model, which consisted of a high degree of freedom in the dynamics of suspension, steering, rigid chassis, truck cabin, and powertrain systems. Furthermore, TruckMaker allows the users to customize such a model with table data, meaning that it becomes a virtual twin of a considered test vehicle. Such an application, regarded as being a known real-time simulation environment for heavy-duty vehicles in which proposed lateral controllers are adapted for forward/reverse maneuvering and longitudinal control, is handled by cruise control of the simulation tool.

5000 MATLAB/Simulink and 800 TruckMaker simulations are performed for both straight and S-shape paths referencing with the same variation of initial conditions as shown in Table 5.3. For the simulations, the vehicle parameters are defined as in Table 5.1, with vehicle speed v being set to $1.39\ m/s$. Controller gains are defined as in Table 5.2 for forward path-following control. As expected, the truck-trailer system converges successfully to both the straight path and to the S shape path for all MATLAB simulations and all TruckMaker simulations in which the region of attraction covers the whole state space for both cases. Stability verification is summarized in

Table 5.3 in terms of the objective, simulation platform, reference path, initial conditions, number of simulations, and results.

Table 5.3:
STABILITY AND ROBUSTNESS VERIFICATION

Objective	Platform	Path	Initial Conditions	Number of Simulations	Result
Forward path-following stability	MATLAB/ Simulink	straight, S-shape	$\bar{x}_i = 0$ $\bar{y}_i = [-10, 10]\ m$ $\beta_i = [-90, 90]\ deg$	5000	Region of attraction covers the whole state space
Forward path-following stability	TruckMaker	straight, S-shape	$\bar{x}_i = 0$ $\bar{y}_i = [-10, 10]\ m$ $\beta_i = [-90, 90]\ deg$	800	Region of attraction covers the whole state space
Reverse path-following stability	MATLAB/ Simulink	straight	$x_i = 0$ $\gamma_{max,all} = 45\ deg$ $y_i = [-4, 4]\ m$ $\gamma_i = [-45, 45]\ deg$ $\delta_i = [-60, 60]\ deg$	8000	Region of attraction covers a large fraction of state space (Fig. 8)
Reverse path-following stability	MATLAB/ Simulink	straight, S-shape	$x_i = 0$ $y_i = [-4, 4]\ m$ $\delta_i = [-60, 60]\ deg$ $\gamma_{max,all} = [45, 54]\ deg$	1600	Region of attraction covers a large fraction of state space (Fig. 9)
Reverse path-following stability	TruckMaker	straight, S-shape	$x_i = 0$ $y_i = [-4, 4]\ m$ $\delta_i = [-\pi/3, \pi/3]\ rad$ $\gamma_{max,all} = [45, 54]\ deg$	1600	More conservative region of attraction compared to MATLAB/Simulink results, still covers a large fraction of state space, comparable performance within certain initial condition threshold (Fig. 9)
Reverse path-following performance	MATLAB/ Simulink	straight, S-shape	$ei_{lat} = [1.0, 4.0]\ m$ $\delta_i = [15, 60]\ deg$	16	Convergence via smooth steering, $e_{ss_{lat}}$= [0.01,0.09] m and [0.01,0.21] m for straight and S-shape paths, respectively (Figs. 10 and 11)
Reverse path-following robustness	MATLAB/ Simulink	straight	$ei_{lat} = [1.5, 4.0]\ m$ $Q = [0.1, 12.0]$	726	Adaptive Q gain performs more robustly to lateral error changes compared to fixed Q gain (Figs. 12 and 13)

5.4.2 *Reverse Path Following*

It can be seen from [50, 31, 35] that stability in reversing a truck-trailer system could not be achieved with a linear state feedback controller. This is because it is impossible to eliminate the possibility of a jack-knife by merely reversing due to unstable joint angle kinematics. However, a region of attraction can be derived through extensive numerical stability verification. This can be achieved by simulating the system in the reverse direction with various initial states. Furthermore, a switching control scheme is proposed to change maneuver direction between forward and reverse, due to the state location with respect to the region of attraction [50, 35].

In relation to this, Ljungqvist et al. [54] introduced a state-feedback controller with feedforward action for a two-trailer system that guarantees local asymptotical stability around a set of paths during reverse motion. It can be shown that, for a pre-computed motion primitive set of an eight-shaped path, LQ-controller is able to locally stabilize the vehicle around the path with disturbance handling. This can be done by verifying a decreasing Lyapunov function where the path is bounded by the parameters within the Jacobian matrix. The work is extended in [55, 56] to guarantee further stability for combined motion planning of forward and reverse maneuvers in which a lattice planner is introduced with pre-computed motion primitives, and the closed-loop system, consisting of the control system and lattice planner, is modeled as a nonlinear hybrid system. The presented framework begins to prove the stability of the closed-loop system by introducing a low-level controller for each motion primitive, which is bound by the path-following error and exhibits a guarantee to decay towards zero. This framework is completed via discrete-time Lyapunov techniques which show that the path-following error between switching points on the lattice planner is also bounded and decays towards zero.

Since path planning in this work is not based on pre-computed motion primitives, and the nonlinearity of the LQ controller is further extended with the proposed Q gain scheduling, a stability verification is introduced which relies on numerical simulations in both MATLAB/Simulink and TruckMaker. In this procedure, a stability region is achieved by varying initial states over a grid, and by checking convergence with the

selected path. MATLAB/Simulink simulations are performed based on the closed-loop system of the proposed path-following controller with the truck-trailer kinematic model (3.1). Hence, as stated for the forward path-following in the previous section, the stability verification through the MATLAB/Simulink simulations is insufficient by itself, as is the reverse path-following, in holding the real system. Therefore, TruckMaker simulations are also introduced for reverse path following, in which the more conservative region of attraction for TruckMaker stability verification is able to hold the real system.

First, a set of 8000 MATLAB/Simulink simulations are performed for straight path referencing with the variation of initial conditions as shown in Table 5.3. The region of attraction formed via simulations covers a large fraction of the state space as shown in Figure 5.8a, which could be further used to bound path segment generation in path planning with input constraints. As expected, the region of attraction for angles as a horizontal cross-section of the stability region grows with the increasing initial path-following error. This is due to the vehicle system having more space to converge to the path. The narrowest stability region of angles is at $ei_{lat} = 0$, as shown in Figure 5.8b, which still covers a significant portion of the initial configuration space. It is worth noting that y_i refers to the initial lateral error, ei_{lat}, since $x_i = 0$.

Subsequently, 1600 MATLAB/Simulink and 1600 TruckMaker simulations are performed for both straight and S-shape paths referencing with the same variation of initial conditions as shown in Table 5.3. It is worth noting that initial hitch angle, γ_i, could not be varied through the simulations due to the limitation of the TruckMaker simulation environment, while the initial hitch angle could not be set to a different value other than zero.

The regions of attraction formed via TruckMaker simulations cover a large fraction of the state space, as shown in Figure 5.9, where they are also compared with the regions of attraction obtained from MATLAB simulations. As expected, the regions of attraction from TruckMaker simulations shrink in general for the same scenarios when compared to the ones from MATLAB simulations, reflecting the negative impact of the dynamic effect of truck trailer system dynamics on stability. On the other hand, at high initial lateral error, and as initial orientation error grows, possible instability is caused by the vehicle's inability to provide the required persistent control in the light of the dynamic constraints allowed. However, in low initial lateral errors, and as the initial

orientation error grows, there is likely to be instability caused by jack-knifing, since more aggressive control is applied. For this reason, the dynamic effect and slip in TruckMaker simulations may have a positive effect on stability in such cases through the inclusion of an extra damping effect on the system when compared to the MATLAB kinematic model.

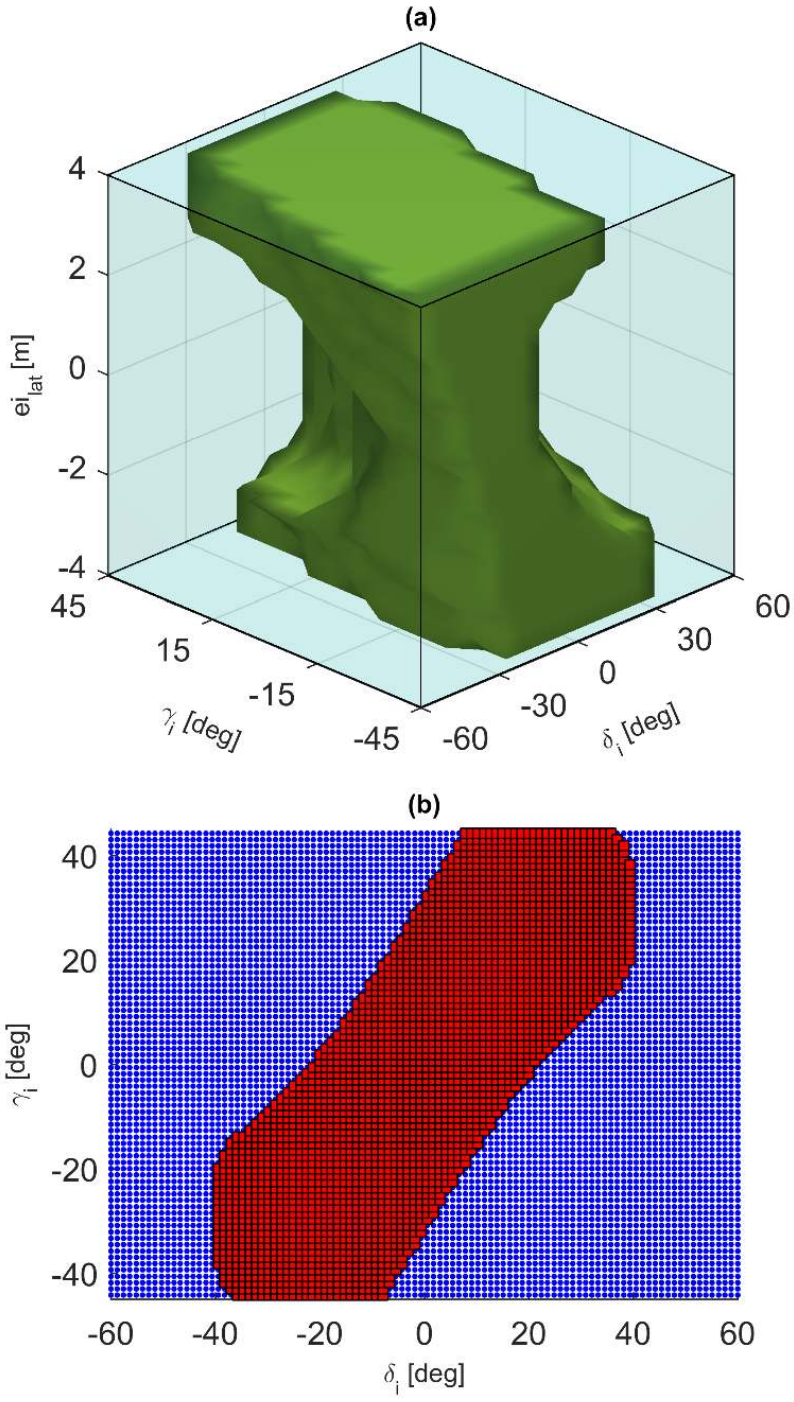


Figure 5.8. a) The simulated region of attraction for different initial states of ($ei_{lat} = y_i$ since $x_i = 0$) ei_{lat}, y_i and δ_i for straight path reversing. The light blue cubic region

corresponds to the whole state space of initial states, while the green region corresponds to stable initial states b) The cross-section of the region at $ei_{lat} = 0$ is the region of attraction of angles.

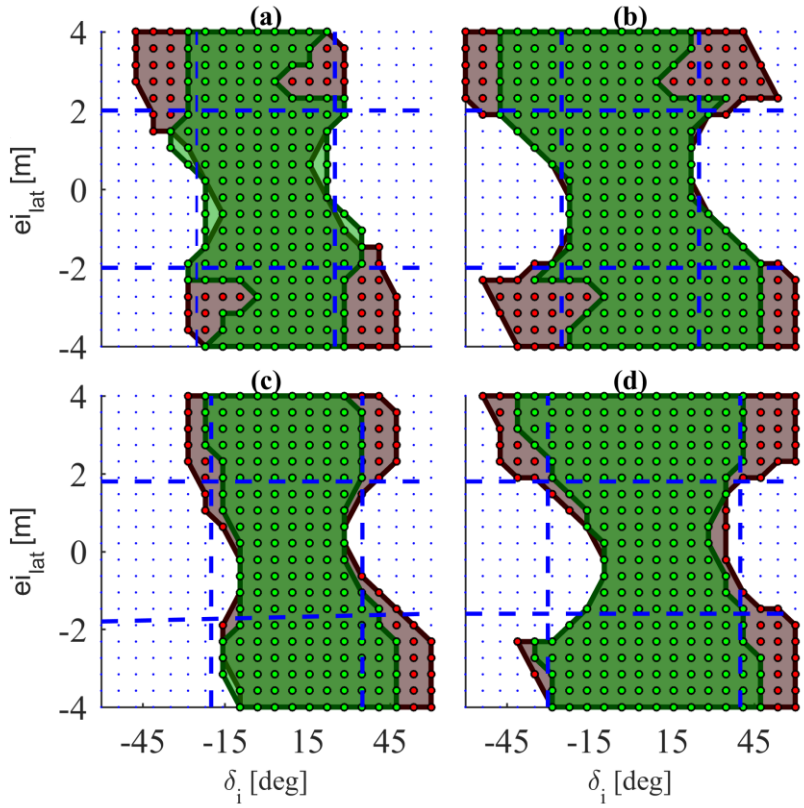


Figure 5.9. The simulated region of attraction for different initial states of ($ei_{lat} = y_i$ since $x_i = 0$) and δ_i: TruckMaker (green), MATLAB (red), are thresholds for comparable performance (dotted blue). a) Straight path reversing with $\gamma_{max,all} = 45\ deg$, b) Straight path reversing with $\gamma_{max,all} = 54\ deg$, c) S shape path reversing with $\gamma_{max,all} = 45\ deg$, b) S shape path reversing with $\gamma_{max,all} = 54\ deg$

Therefore, although negative dynamic effects still exist, the significance of this positive effect demonstrates that shrinkage is less in these 'low lateral error' regions than in other regions. This is despite there being evidence of expansion in some examples e.g. a straight path for γ_{max} of 45 deg in Figure 5.9a. Consequently, it is worth noting that the shrinkages mostly take place when the initial error states are above certain thresholds, as shown by blue dotted lines in Figure 5.9, while comparable performance is achieved below these values in terms of stability. For all simulations regarding the stability verification of reverse path-following controller, the look-ahead distance L_d is set to 10 m.

Path-following performance analysis is subsequently performed by plotting the truck trailer path convergence, along with steer and hitch angles. This can be seen in Figure

5.10 and Figure 5.11 for a subset from the TruckMaker simulations which has been performed for stability verification. The subset consists of 8 simulations for each of the straight and S shape paths with initial states as shown in Table 5.3 separately.

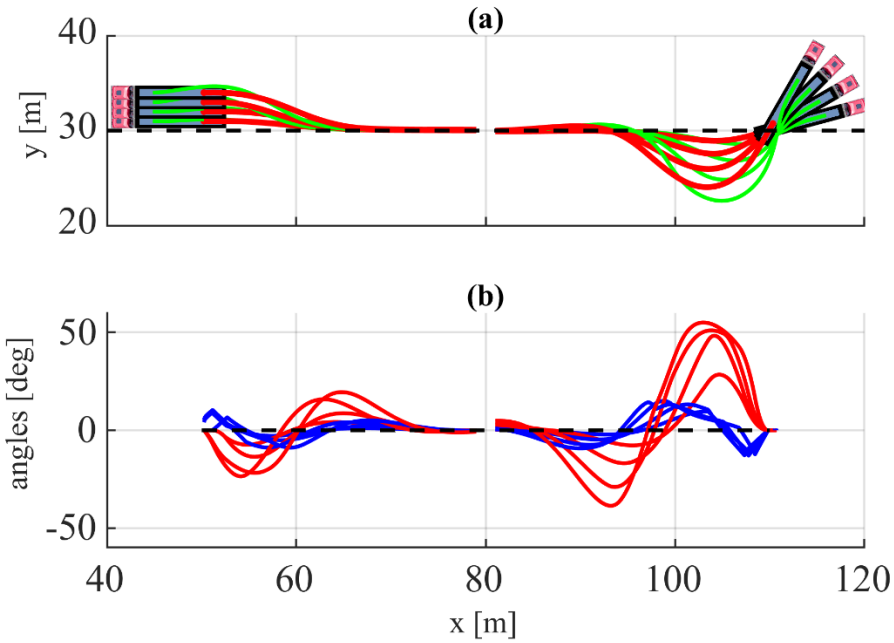


Figure 5.10. TruckMaker simulation for straight path referencing in reverse motion with initial states of ei_{lat} **= [1.0, 2.0, 3.0, 4.0]** ***m*** (left) and δ_i **= [15, 30, 45, 60]** ***deg*** (right). a) xy plot: simulated truck path (green), simulated trailer path (red), reference path (dotted black), b) Steering angle (blue), hitch angle, **γ** (red), reference hitch angle (dotted black)

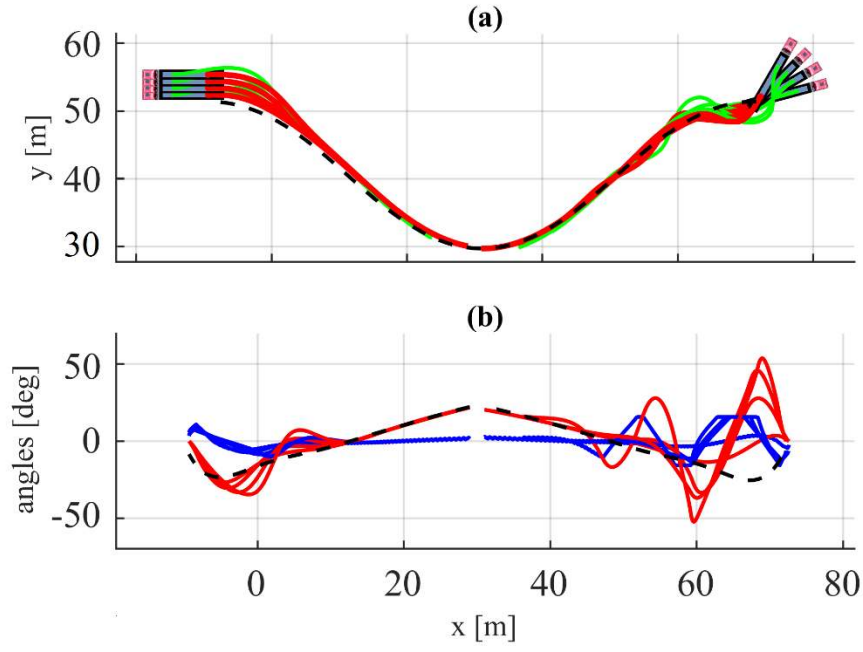


Figure 5.11. TruckMaker simulation for S shape path referencing in reverse motion

with initial states of $\boldsymbol{ei_{lat}}$ **= [1.0, 2.0, 3.0, 4.0]** $\boldsymbol{m}$ (left) and $\boldsymbol{\delta_i}$ **= [15, 30, 45, 60]** $\boldsymbol{deg}$ (right). a) xy plot: simulated truck path (green), simulated trailer path (red), reference path (dotted black), b) steering angle (blue), hitch angle, $\boldsymbol{\gamma}$ (red), reference hitch angle (dotted black).

The convergences to both straight and S shape paths are achieved via smooth steering, where the hitch angle converges to the desired hitch angle profile along the path calculated from the path curvature. Straight referencing scenarios achieve steady-state path-following errors, $e_{ss_{lat}}$, within a range of $0.01\ m$ to $0.09\ m$, whereas the range for S shape referencing scenarios is between $0.01\ m$ and $0.21\ m$.

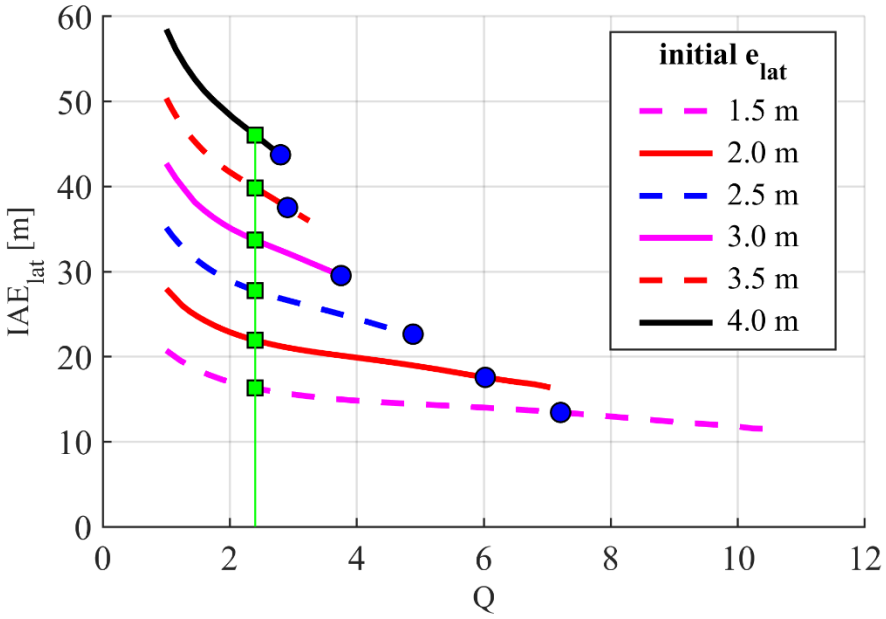


Figure 5.12. Simulations with constant $\boldsymbol{Q}$: constant $\boldsymbol{Q}$ varied through simulations for each initial following error state $\boldsymbol{ei_{lat}}$, stable intervals of $\boldsymbol{Q}$ is shown for each $\boldsymbol{ei_{lat}}$. Blue dots refer to the $\boldsymbol{IAE_{lat}}$ magnitudes achieved by adaptive $\boldsymbol{Q}$, and corresponding constant $\boldsymbol{Q}$ candidates to obtain the same performance. Green squares refer to a selection of a single constant $\boldsymbol{Q}$ gain lying in the stable intervals of all initial following error states.

For robustness verification, straight path-referencing simulations for different initial path-following errors are primarily performed by using the cascade control of reverse pure pursuit and LQ controller with the constant Q. The look-ahead distance is selected as $L_d = 8\ m$, and constant Q gain is varied through the simulations as shown in Table 5.3 to determine the boundaries of Q, based on the stability for different initial path-following errors, ei_{lat}. The results, including stable Q intervals and corresponding

IAE_{lat}, are shown in Figure 5.12. The proposed path-following controller with adaptive Q gain in the stabilizer is implemented in simulations for the same initial path-following errors. As shown in Figure 5.13, the controller varies Q along the maneuver smoothly for tests with different initial lateral error states $ei_{lat} = [1.5, 2.0, 2.5, 3.0, 3.5, 4.0]\, m$, while IAE_{lat} values for these tests are marked with blue dots in Fig. 12 to show constant Q conjugates that achieve the same magnitudes of the path-following performance. The path-following error performance achieved with adaptive Q for each test is very close to the best performance of the constant Q implementation, which is achieved in each test with a different Q value close to its stability boundary, but which lies out of the stability region of the others. A selection of a single constant Q gain within the stable intervals of all initial path-following error states is shown by the green squares in Figure 5.12. It is seen that the performance of adaptive Q gain implementations, as shown in blue dots, could not be achieved for a selected single Q value.

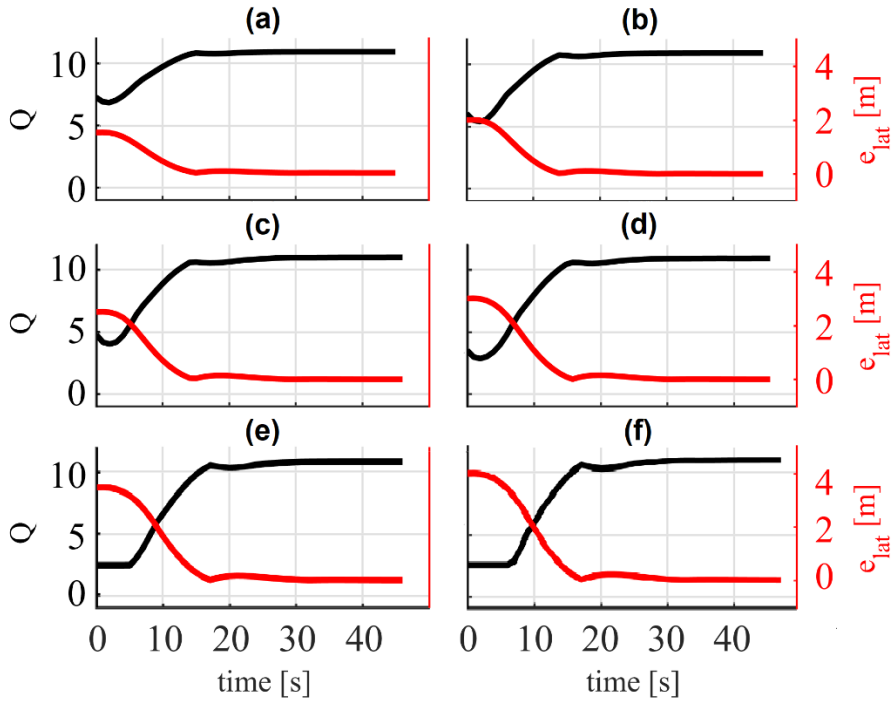


Figure 5.13. Robustness Simulation with adaptive $\boldsymbol{Q}$: Change of $\boldsymbol{Q}$ (black) and $\boldsymbol{e_{lat}}$ (red) in time for different initial following error states $\boldsymbol{ei_{lat}}$

For all simulations regarding the stability and robustness verification of reverse path-following controller, the vehicle parameters are defined as in Table 5.1 and controller

gains as in Table 5.2 for reverse path-following control, with vehicle speed v being set to $-1.39\ m/s$. Stability and robustness verifications are summarized in Table 5.3 in terms of the objective, simulation platform, reference path, initial conditions, number of simulations, and results.

Chapter 6

RESULTS

6.1 Path Planning Simulations

6.1.1 *Simulation and Algorithm Parameters for CL-RRT*

For the CL-RRT algorithm outer loop, five different simulation parameters are defined for solution convergence performance as shown in Table II. CL-RRT algorithm loop operates for a maximum duration of T_{max}, and restarts tree expansion for max iteration number of $Iter_{max}$ if no solution is found where the total algorithm duration with no solution reaches up to T_{max} x $Iter_{max}$. If at least one solution is found in a CL-RRT loop, the maximum duration T_{max} is lowered to Ts_{max} in order to have a lower run time, where the algorithm stops either if number of enough solutions Sol_{max} is achieved or computation time reached to Ts_{max}, maximum duration for solution existed loop, reached.

Table 6.1:
CL-RRT SIMULATION PARAMETERS

Parameters	Value
T_{max}	12 s
$Iter_{max}$	5
Sol_{max}	6
Ts_{max}	6 s

Further, a probabilistic choice for cost heuristic definition of full path length L_p from the tree start point P_s to S_{rnd} is introduced in order to increase branching variation from Ps as well. Note that, simulation to reach from N_{min} to S_{rnd} is performed in each iteration and the feasibility of the branch is checked both for collision and allowed angular tolerances defined for changes in orientation of truck-trailer as listed in Table 6.2.

Table 6.2:
CL-RRT ALGORITHM PARAMETERS

Algorithm Parameters	symbol	Value
Probability for $\boldsymbol{L_b}$ cost heuristic	μ_b	0.96
Probability for $\boldsymbol{L_p}$ cost heuristic	μ_p	0.04
max steer wheel angle	α_{max}	0.68 rad
max γ range in each branch	$rng_{\gamma_{max}}$	0.79 rad
max β range in each branch	$rng_{\beta_{max}}$	0.68 rad
max abs sum of β change in each branch	$\sum\lvert\beta_i - \beta_{i-1}\rvert$	7.54 rad
max abs sum of δ change in each branch	$\sum\lvert\delta_i - \delta_{i-1}\rvert$	7.54 rad

A goal simulation is checked if it refers to a feasible solution where goal evaluation parameters are introduced in Table 6.3. Parameters consist of error tolerances for Euclidian distance and truck-trailer yaw angles and differentiate for forward and reverse reaching to the goal pose.

Table 6.3:
CL-RRT GOAL PARAMETERS

Direction	Goal Requirements	Symbol	Value
Forward	error tolerance for distance	ε_{df}	0.4 m
Forward	error tolerance for truck angle β	$\varepsilon_{\beta f}$	0.15 rad
Forward	error tolerance for trailer angle δ	$\varepsilon_{\delta f}$	0.20 rad
Reverse	error tolerance for distance	ε_{db}	0.6 m
Reverse	error tolerance for truck angle β	$\varepsilon_{\beta b}$	0.30 rad
Reverse	error tolerance for trailer angle δ	$\varepsilon_{\delta b}$	0.40 rad

Values selected in Table 6.1, Table 6.2, and Table 6.3 are obtained through simulations performed for the test field case shown in Figure 4.8.

6.1.2 *CL-RRT Validation in different Scenarios*

The performance of the CL-RRT algorithm is tested in different challenging scenarios and use cases to understand the capability of the algorithm. Scenarios including possible static obstacles in the docking station, construction site, and parking in a maze environment are generated and tested. An HMI (Human Machine Interface) is developed to generate different use cases with different environmental configurations including various types of obstacles. In this section, only the capability of the algorithm for various scenarios is tested. The performance of the cascade path planning algorithm as the whole path planning framework is tested in Section 6.2.2 where an extensive number of scenarios are generated and simulated and the results are compared in terms of computation time, tree growth, and path cost.

6.1.2.1 *Parking in Docking Station*

Four scenarios are tested with the presence of static objects such as parked vehicles, where both the occupancy of the docking area and obstacle list information are used. The aim is to introduce a very narrow parking area occupied by several static objects such as trucks, passenger cars, bicycles, and pedestrians. In the first two scenarios, three obstacles are inserted into the simulation environment, and the resulting paths for feasible parking are two-maneuver paths with one forward and one reverse as shown in Figure 6.1 and Figure 6.2.

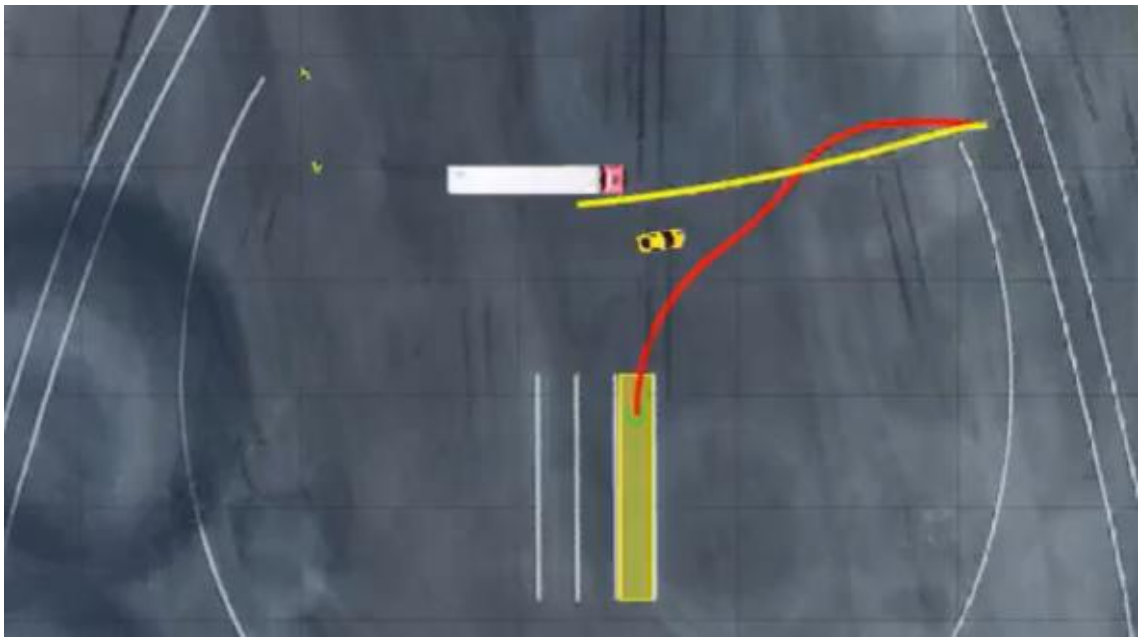


Figure 6.1. Resulting parking path for scenario 1

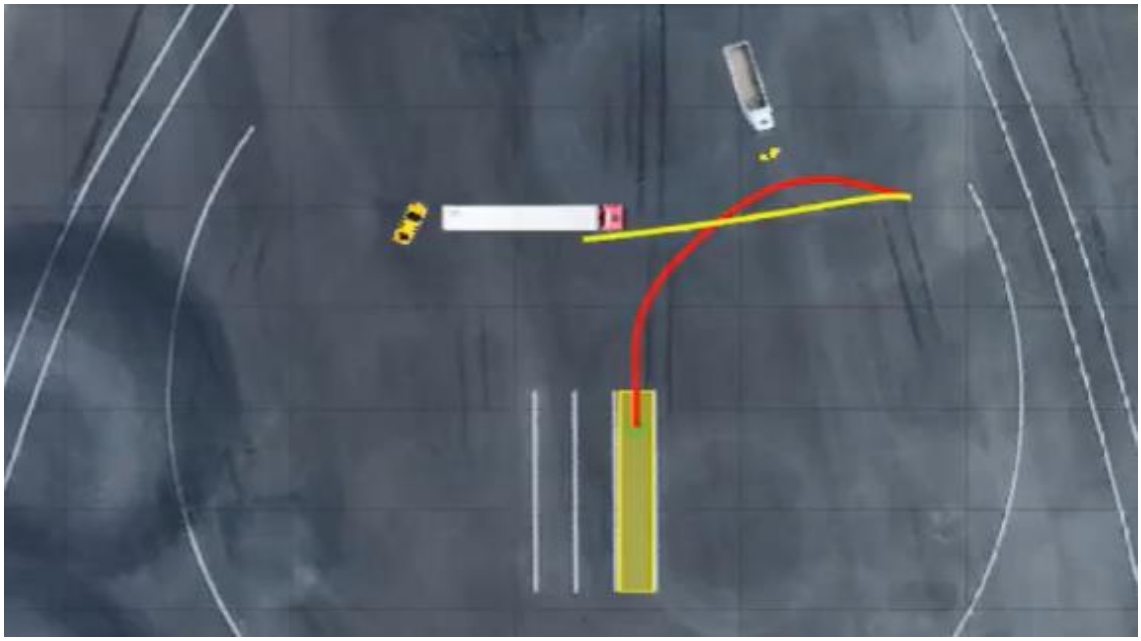


Figure 6.2. Resulting parking path for scenario 2

For scenarios 3 and 4, a more challenging environment is introduced with five obstacles in the vicinity of the parking space. The resulting paths for feasible parking consist of four maneuvers; two forward and two reverse, as shown in Figure 6.3 and Figure 6.4.

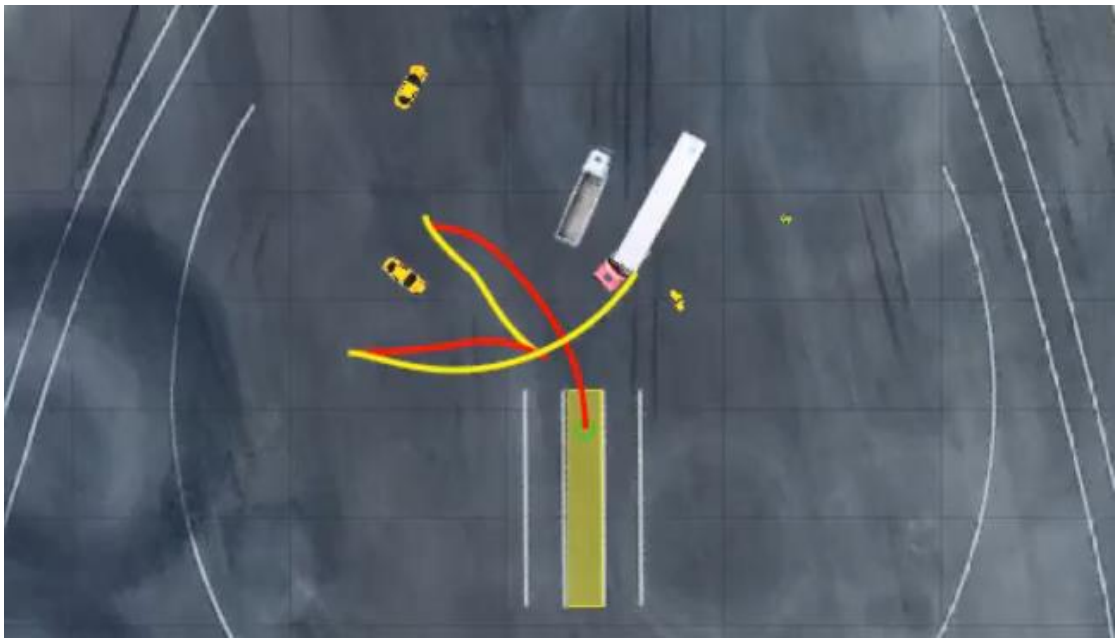


Figure 6.3. Resulting parking path for scenario 3

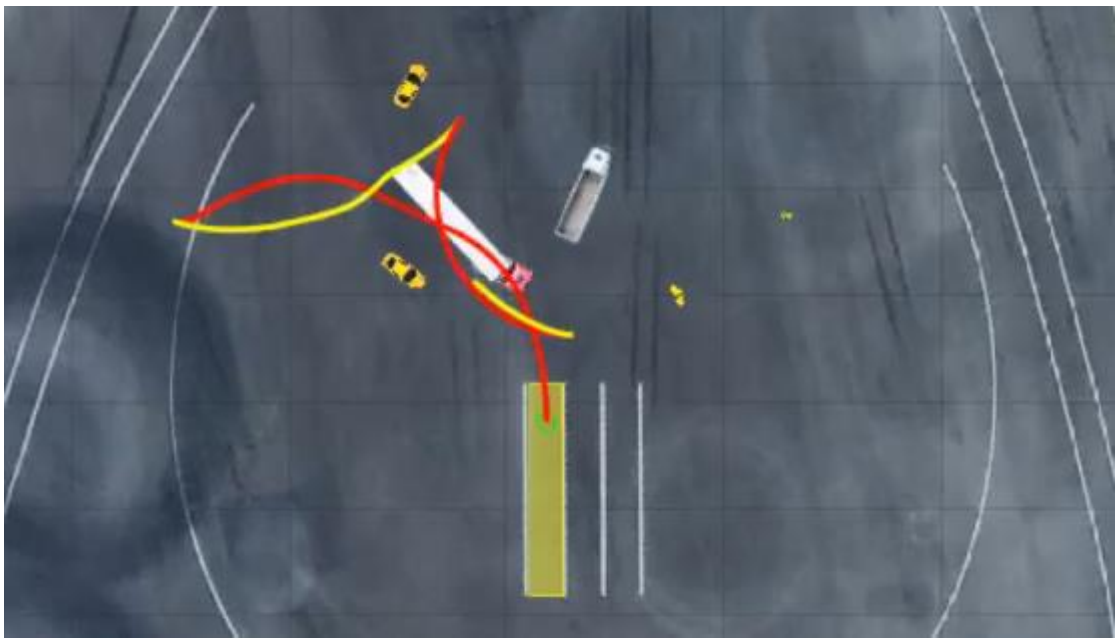


Figure 6.4. Resulting parking path for scenario 4

Simulation parameters and performance metrics of the CL-RRT algorithm in terms of solution time, tree growth, and path cost are shown as in below table for these 4 scenarios.

Table 6.4:
CL-RRT SIMULATION PARAMETERS & PERFORMANCE METRICS FOR DOCKING STATION SCENARIOS

		Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Parameters	$\boldsymbol{T_{max}}$	12 s	12 s	12 s	12 s
	$\boldsymbol{Iter_{max}}$	5	5	5	5
	$\boldsymbol{Sol_{max}}$	6	6	6	6
	$\boldsymbol{Ts_{max}}$	6 s	6 s	6 s	6 s
Performance Metrics	$\boldsymbol{T_{sol}}$	1.04 s	1.86 s	3.52 s	4.78 s
	$\#_{Tree\ Nodes}$	174	283	506	871
	C_{path}	42.40	53.48	75.80	96.52

6.1.2.2 *Parking on Construction Site*

The workspace shown with a big green rectangle, and the occupancy map shown with small yellow squares in Figure 6.5 are pre-computed offline. Since the whole occupancy map shown on the left is not necessary to use and also increases the computational effort, a window of occupancy based on the workspace as shown on the right is taken as the occupancy map for this parking scenario. Based on the random sample generation approach, a random square cell is selected in the workspace and checked for feasibility based on the occupancy map. Once the square cell is feasible, a random sample is selected inside the square region with a standard normal distribution.

Figure 6.5. Visualization of occupancy map and workspace of the construction site

Three scenarios are tested without the presence of static objects such as parked vehicles, but only the occupancy of the vicinity of the construction site is used. Two of these scenarios locate the truck trailer system in the lane of the construction site, one locates on the road across the construction site as a more challenging problem. The first two have solution paths with two maneuvers. The third one finds a solution path with four maneuvers to provide a feasible path for parking. The last scenario is implemented where four static vehicles are introduced near the construction site. The algorithm finds again a four-maneuver solution path to provide feasible parking. The resulting paths are shown in Figure 6.6. Note that the tree expansion is also shown in the fourth scenario where the tree avoids expanding onto the static objects.

Simulation parameters and performance metrics of the CL-RRT algorithm in terms of solution time, tree growth, and path cost are shown as in below table for these 4 scenarios.

Figure 6.6. Resulting parking paths for different scenarios on construction site

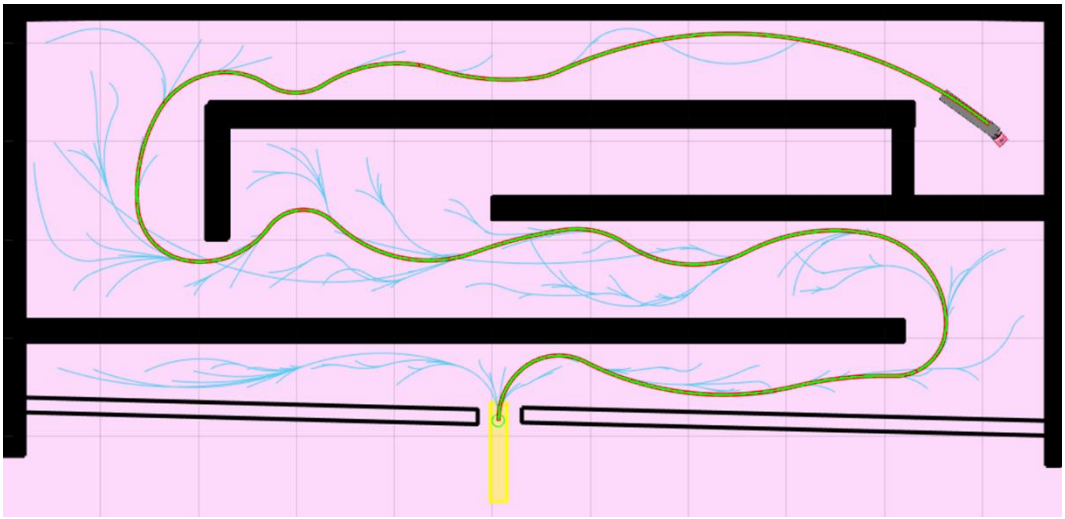
Table 6.5:
CL-RRT SIMULATION PARAMETERS & PERFORMANCE METRICS
FOR CONSTRUCTION SITE SCENARIOS

		Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Parameters	T_{max}	12 s	12 s	12 s	12 s
	$Iter_{max}$	5	5	5	5
	Sol_{max}	4	4	4	4
	Ts_{max}	8 s	8 s	8 s	8 s

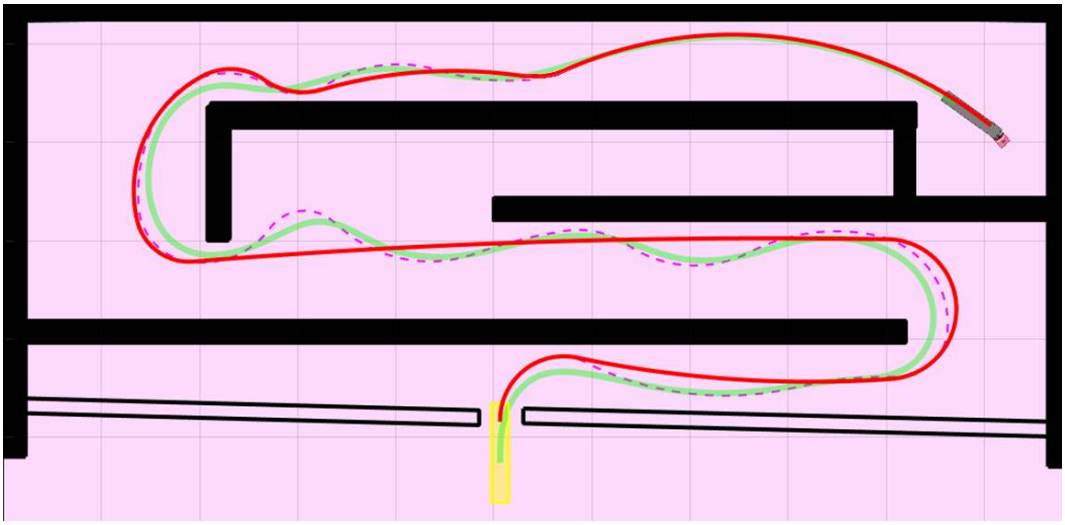
Performance Metrics	T_{sol}	1.91 s	2.25 s	3.78 s	4.46 s
	$\#_{Tree\ Nodes}$	295	332	527	764
	C_{path}	48.74	57.40	78.27	86.03

6.1.2.3 *Parking in Maze Environment*

CL-RRT algorithm is tested in maze environment where interim point bias probability of random sample generation is selected as zero since the parking scenario is not a usual one of truck trailer system located near the parking space. Only workspace center and goal point biases are considered, and the algorithm is tested for two test cases with different initial positions of the truck-trailer system. The generated path and the tree expansion for the first scenario are shown in Figure 6.7a, and the transformation of the generated path to the optimized path via the path optimization function is shown in Figure 6.7b.



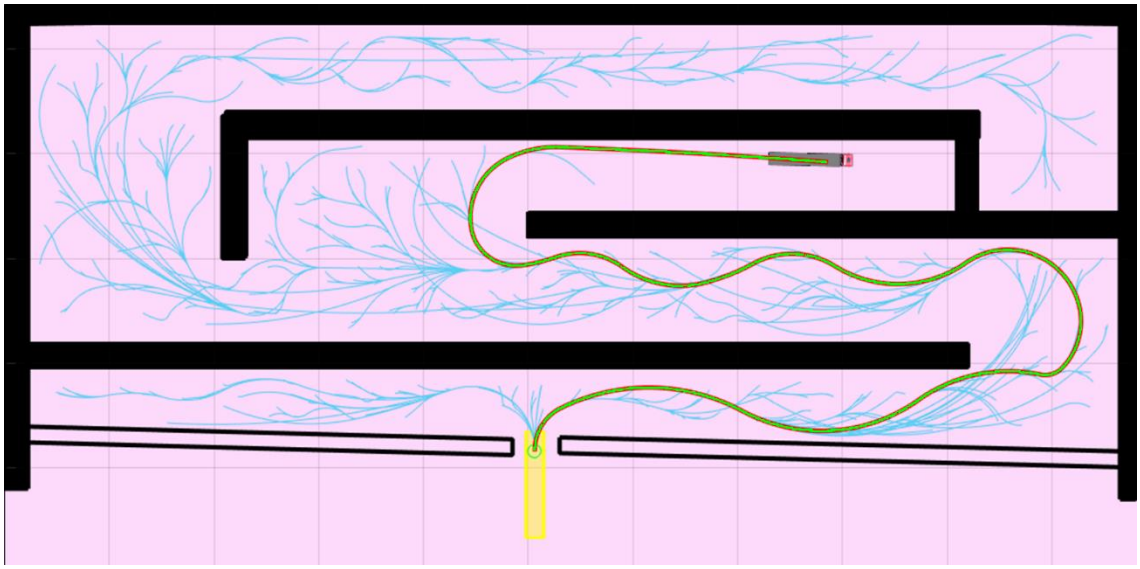
(a)



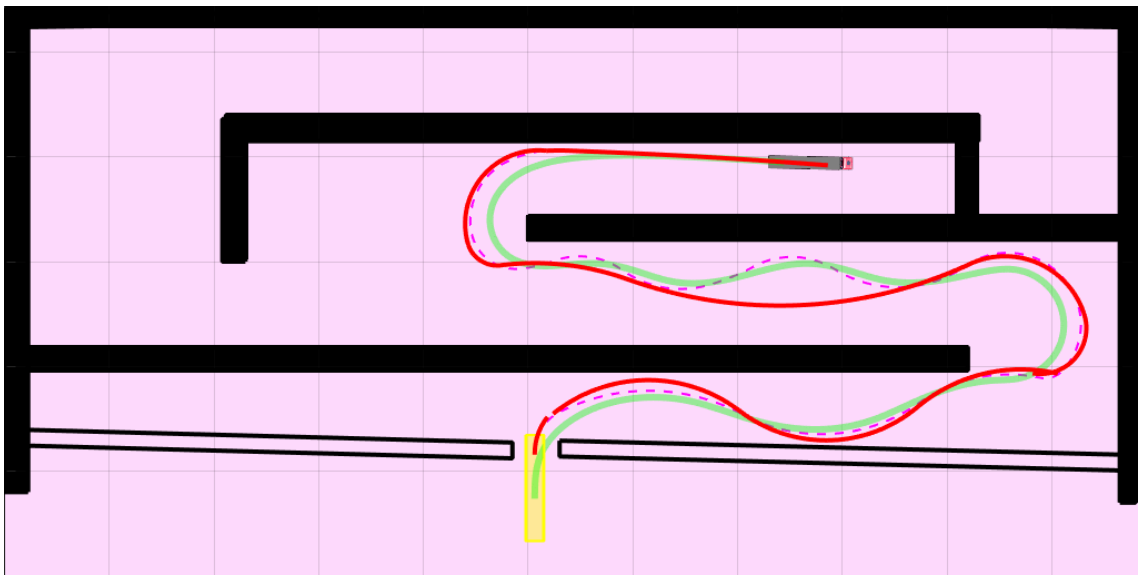
(b)

Figure 6.7. Resulting parking path in maze environment for scenario 1: (a) tree expansion and generated path, (b) visualization of the generated path in terms of truck and trailer path, and optimized path via path optimization function

The generated path and the tree expansion for the second scenario are shown in Figure 6.8a, and the transformation of the generated path to the optimized path via the path optimization function is shown in Figure 6.8Figure 6.7b. It is clearly seen in both scenarios, the tree successfully expands inside the maze, and prioritizes the expansion based on the goal pose (initial pose of ego vehicle), where the tree first expands along the way to the goal point in both scenarios. Moreover, path optimization outputs successfully a more efficient and still feasible path in terms of collision avoidance.



(a)



(b)

Figure 6.8. Resulting parking path in maze environment for scenario 2: (a) tree expansion and generated path, (b) visualization of the generated path in terms of truck and trailer path, and optimized path via path optimization function

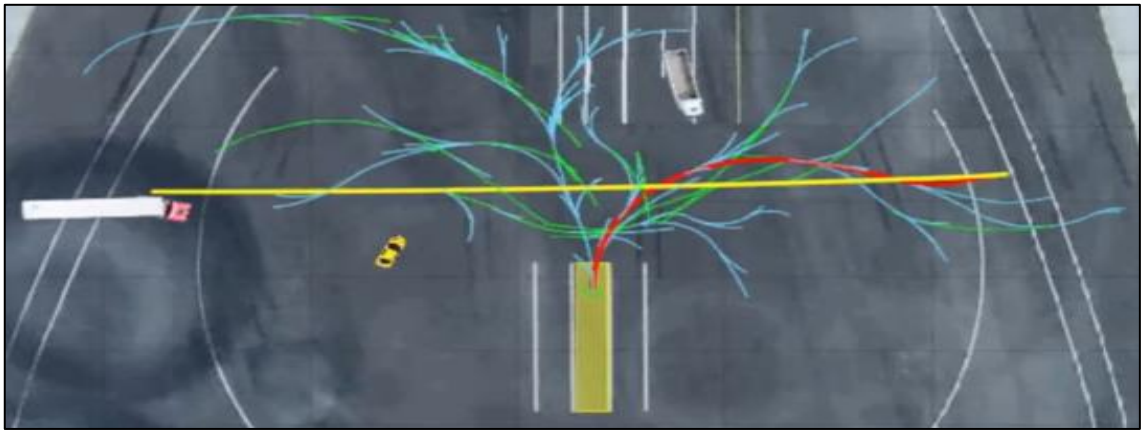
Simulation parameters and performance metrics of the CL-RRT algorithm in terms of solution time, tree growth, and path cost are shown as in below for these 4 scenarios.

Table 6.6:
CL-RRT SIMULATION PARAMETERS & PERFORMANCE METRICS
FOR MAZE ENVIRONMENT SCENARIOS

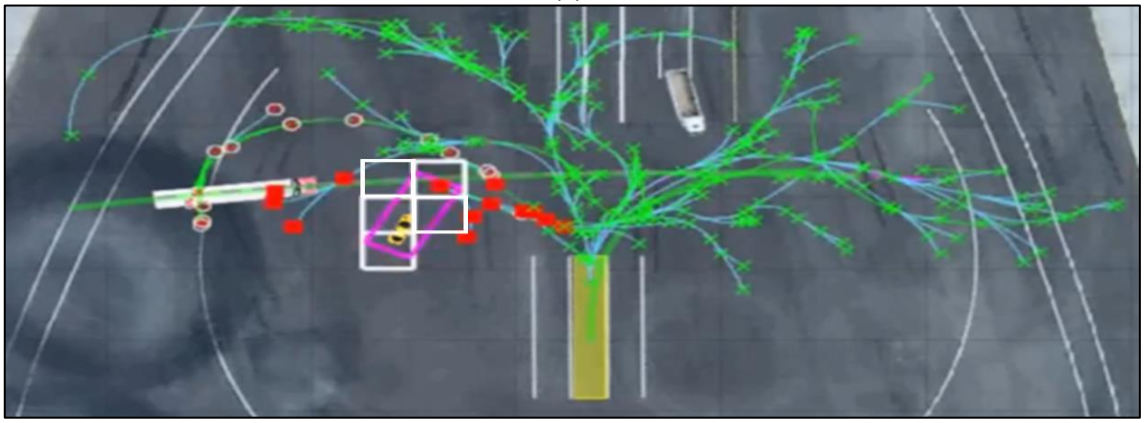
		Scenario 1 without PO	**Scenario 1 with PO**	**Scenario 2 without PO**	**Scenario 2 with PO**
Parameters	$\boldsymbol{T_{max}}$	30 s	30 s	30 s	30 s
	$\boldsymbol{Iter_{max}}$	3	3	3	3
	$\boldsymbol{Sol_{max}}$	4	4	4	4
	$\boldsymbol{Ts_{max}}$	15 s	15 s	15 s	15 s
Performance Metrics	$\boldsymbol{T_{sol}}$	8.72 s	8.76 s	12.61 s	12.64 s
	$\#_{Tree\ Nodes}$	1277	1277	1684	1684
	C_{path}	206.48	153.06	117.51	102.40

6.1.2.4 *Parking in Dynamic Environment with Online CL-RRT*

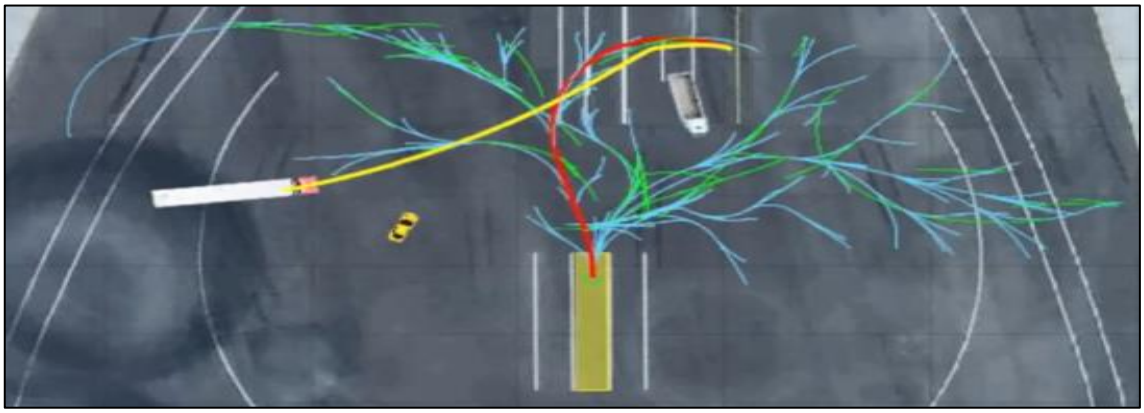
Online CL-RRT operation process is shown in Figure 6.9, where the screenshots of the online CL-RRT simulation are shown by depicting the tree expansion and update, generated and updated paths, and the effect of the dynamic objects. The algorithm starts with offline CL-RRT for initialization in Figure 6.9a where the initial trajectory is generated and the vehicle started to move. In Figure 6.9b, the interference of the effective zone of the dynamic object (passenger car) into the generated path and expanded tree is shown, where the affected nodes of the tree are shown in red squares, and the tree is trimmed. Hence, the vehicle starts to slow down to avoid possible collision and reinitiates path planning with the trimmed tree expansion. In Figure 6.9c, a new trajectory is generated and the vehicle performs tracking control to follow that trajectory. Finally, in Figure 6.9d, the effective zones of the dynamic objects (passenger car, truck) interfere again with the generated path and expanded tree. The vehicle breaks down for an emergency stop to avoid a collision. The tree is trimmed again by removing the affected nodes shown in red squares.



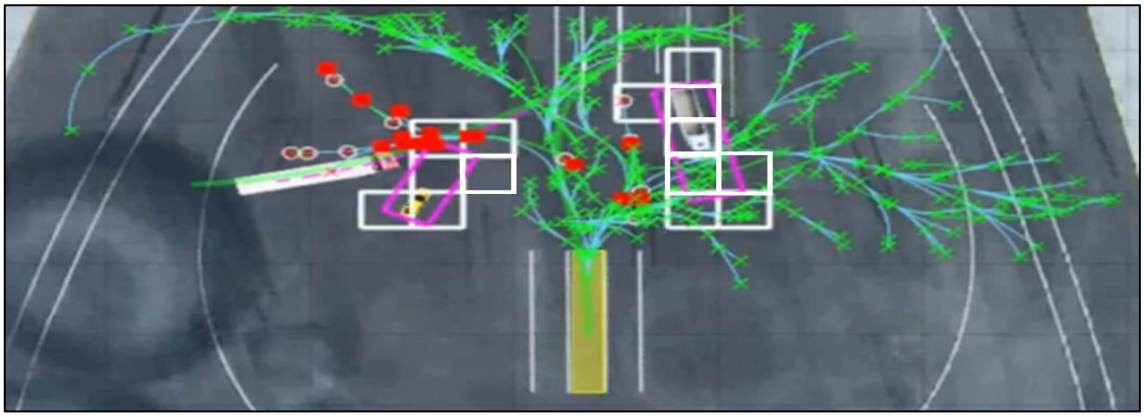
(a)



(b)



(c)



(d)

Figure 6.9. Online CL-RRT operation in a dynamic environment

In Figure 6.10, successful parking with online CL-RRT in a dynamic environment is shown. The algorithm starts with offline CL-RRT for initialization in Figure 6.10a where the initial trajectory is generated and the vehicle started to move. In Figure 6.10b, the vehicle continues to follow its trajectory. In Figure 6.10c, the effective zone of the dynamic object (passenger car) interferes with the generated path. Hence, the vehicle starts to brake to avoid possible collision and reinitiates path planning. In Figure 6.10c, the vehicle waits at a standstill after the braking, since the path planning did not find a new path yet. In Figure 6.10d, the dynamic object passes the ego vehicle and the last trajectory becomes feasible again. In Figure 6.10e, the vehicle starts to move again the ego vehicle and the last trajectory becomes feasible again and another dynamic object (the truck) interferes with the trajectory of the ego vehicle. Hence, the vehicle slows down and reinitiates path planning. In Figure 6.10f, a new trajectory is generated and the ego vehicle starts to follow that trajectory. Finally, in Figure 6.10g and Figure 6.10h, the ego vehicle continues to follow its trajectory and parks in the selected parking space.

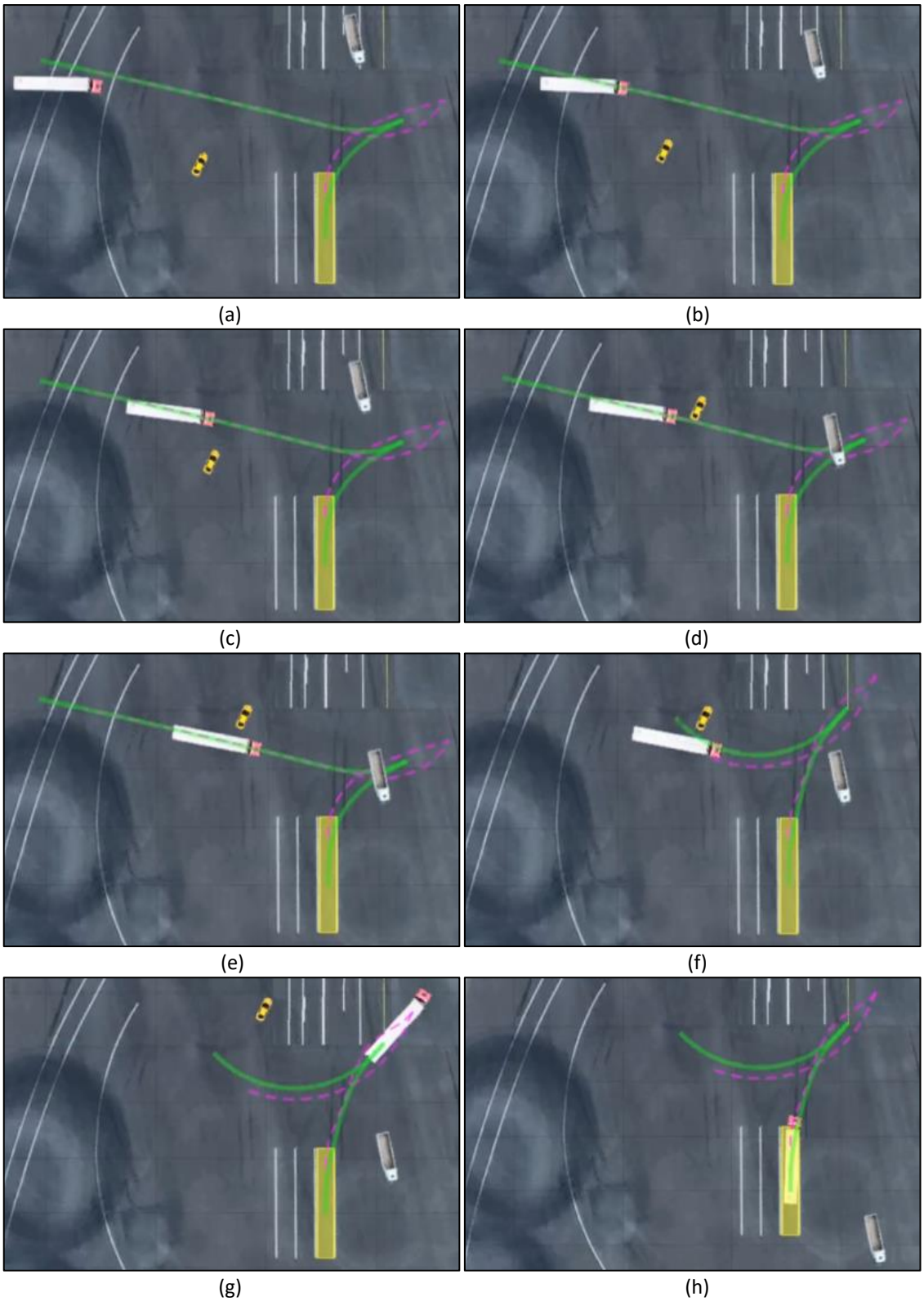


Figure 6.10. Successful parking with online CL-RRT in a dynamic environment

6.1.3 *Cascade Path Planning Simulations*

The performance of the path planning algorithm is evaluated through a series of parking test case simulations run on a standard laptop computer with an Intel Core i7-6600U@2.6GHz CPU. A parking test case generation tool is developed where the initial configuration of the truck trailer system and a random number of static obstacles with different types (pedestrian, bicycle, passenger car, and truck) are randomly determined and generated as possible test cases to park the vehicle in a narrow parking slot. In fact, the effective occupancy of each obstacle is determined based on its critical safety level. For instance, for a passenger, a larger effective safety zone is added around its original occupancy compared to a vehicle. The user interface for case generation and simulations for successful parking maneuvers are shown in Figure 6.11.

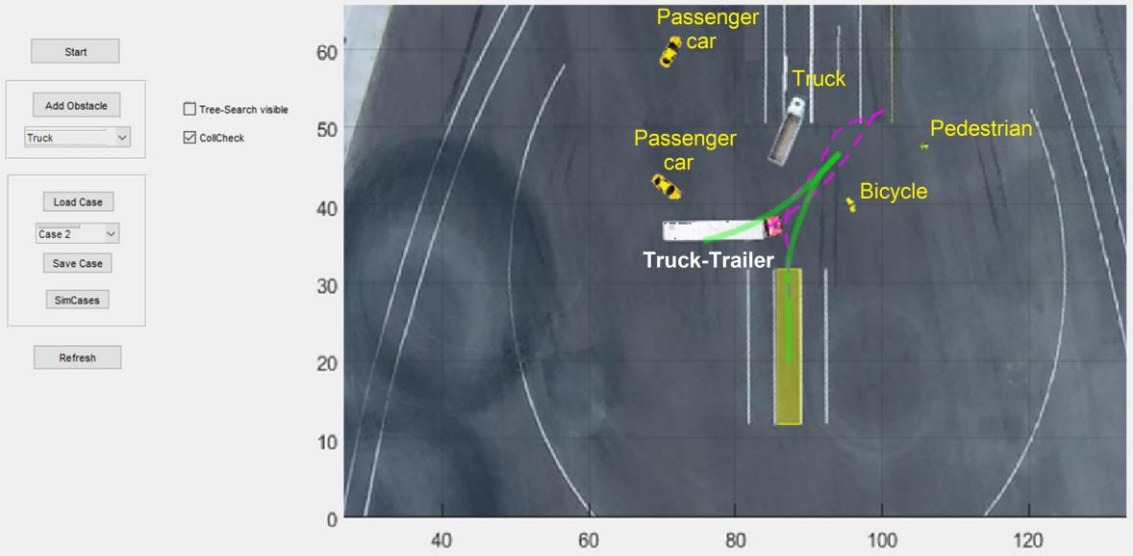


Figure 6.11. GUI for generating parking scenarios and simulating for successive parking maneuvers: Test cases with different types of obstacles (truck, passenger car, bicycle, and pedestrian) could easily be generated, saved, and loaded. Path planning for the test cases could be performed and depicted on the screen

After further elimination of unrealistic test scenarios manually, 47 different test scenarios with 3 different parking slot selections (total 141 test cases) are generated as a combination of 5 different truck-trailer initial poses and 8-12 different environment configurations with a set of obstacles for each truck-trailer pose. The cascade path planning algorithm runs 10 times for each test case and a mean of 10 runs is stored as the solution for the corresponding test case. In other words, 1410 solution attempts are tested with the algorithm. Successful parking solutions for 4 test cases with the selection

of the IAM method and 2 test cases with the CL-RRT method are depicted in Figure 6.12.

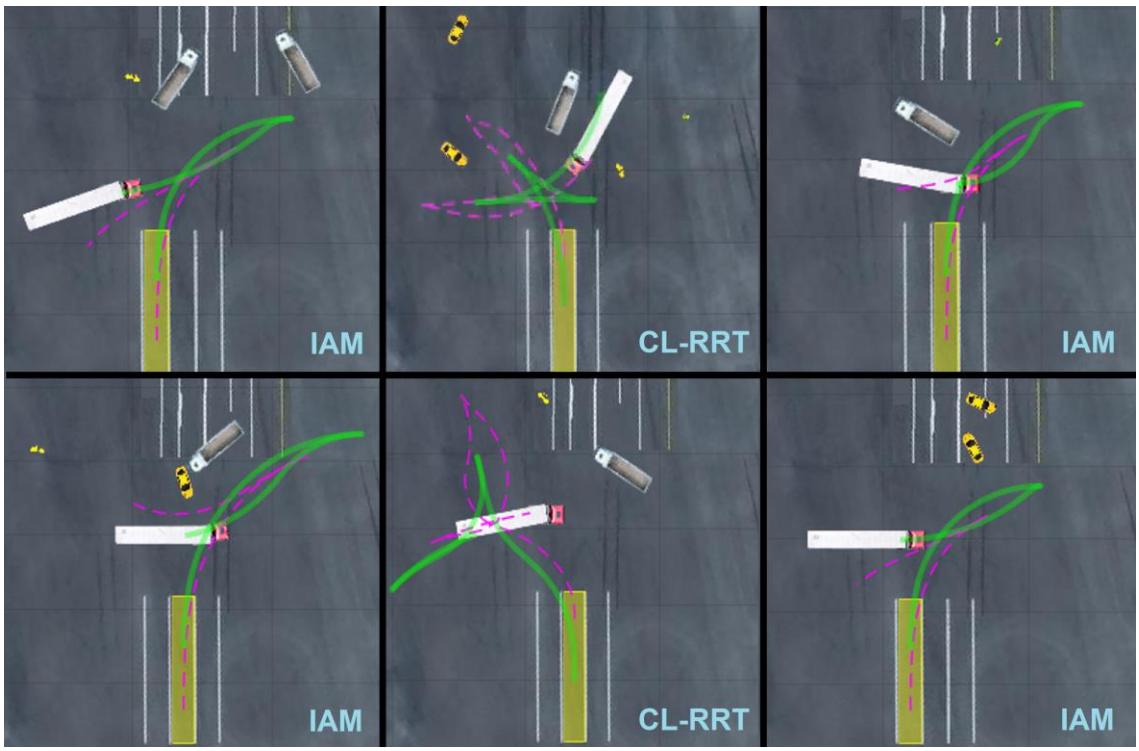


Figure 6.12. Successive Path Planning Examples: 6 different path generation simulations are depicted. Four of them are solved with IAM and two of them are solved with CL-RRT. The truck path is shown with magenta dashed lines and the trailer path with green lines.

For 1410 solution attempts, the algorithm achieves a solution success rate of 99.7%, where 67.38% of solutions are achieved with the selection of IAM and 32.62% with CL-RRT. The algorithm selector achieves a successful estimation rate of 85.1% for the selection of IAM, where IAM finds a solution path for the test case. Hence, only 14.9% of the IAM selections perform a further CL-RRT run to find a successful solution path. 89.2% of the solutions achieved with the cascade algorithm were found within 2 s with a total mean solution time of 0.82 s, whereas IAM and CL-RRT have mean solution times of 0.31 s and 1.87 s respectively. It is shown in the solution time histogram in Figure 6.13a, that IAM achieves solutions much faster and in a very narrow time interval, whereas on the other hand CL- RRT solutions have a larger distribution in solution time due to their probabilistic nature. CL-RRT solutions have lower path costs due to their flexibility to succeed in multiple solutions for a test case through tree expansion and to select the one with minimum cost. Most of the CL-RRT solutions

expanded less than 500 nodes where an average of 1293 nodes was expanded when maximum planning time has elapsed.

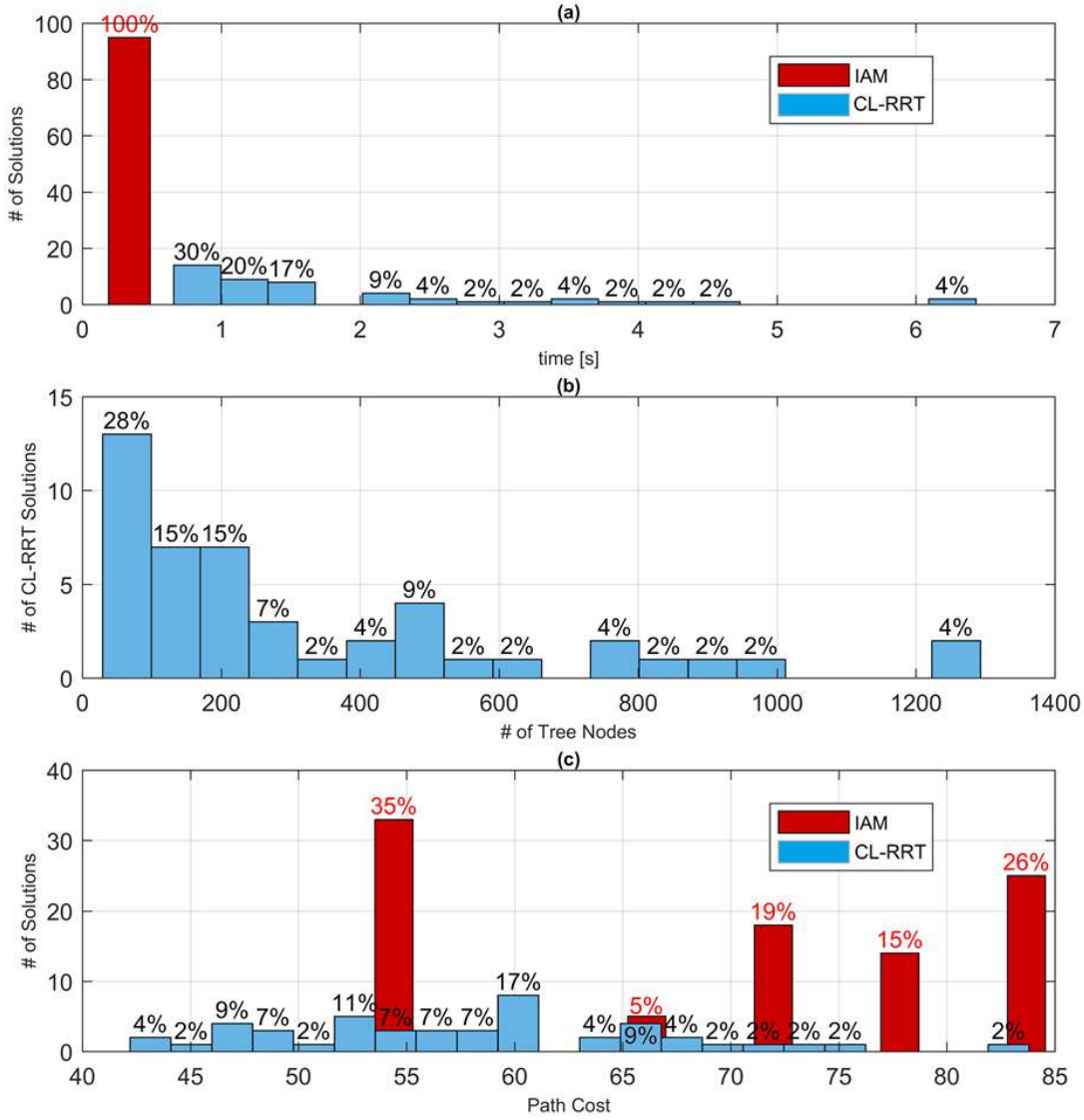


Figure 6.13. Histogram over the solutions for the path planning. Time, total path cost, and the number of tree nodes are shown from top to bottom respectively.

To evaluate the performance of the cascade path planning algorithm, further simulations with the same test cases are performed by only introducing CL-RRT as the path planner, where the results for 1410 simulations are shown in Figure 6.14. The mean of the total simulation time is increased to 1.49 s since the mean solution time is 1.33 s for the test cases selected for IAM in cascade algorithm simulations. It is noted that the mean solution time for the remaining test cases is 1.89 s which is very close to the value for CL-RRT selections in cascade algorithm simulations, as expected. The solution

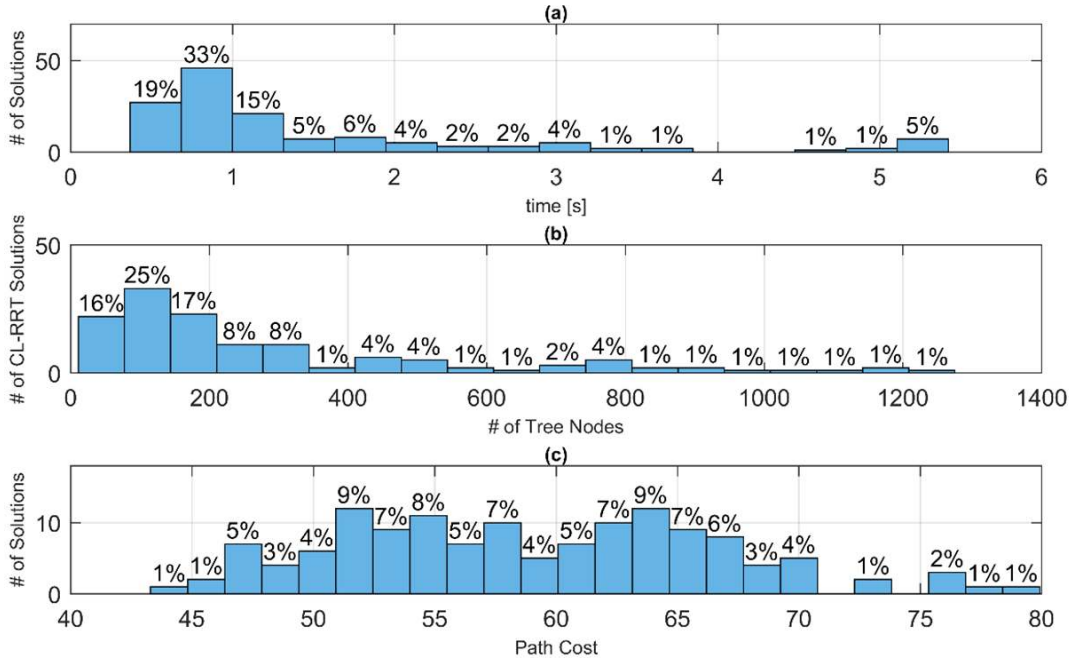


Figure 6.14. Histogram over the solutions for the path planning of enhanced CL-RRT with random sample generation. Computation time, number of nodes in tree expansion and total path cost are shown from top to bottom respectively.

success rate is also very close to the cascade path planning algorithm with a value of 99.4%. Further, the distribution of the path cost, and the number of nodes in tree expansion are close to the results in cascade algorithm simulations. It is noted that the percentage of the test cases with less than 500 nodes in tree expansion is increased slightly. Since tree expansion for test cases selected for IAM are less complicated compared to the ones selected for CL-RRT in cascade algorithm simulations, the solution to these test cases requires less expansion to converge in a solution path. It is seen that introduction of IAM with a cascade algorithm decreases computational effort by around 45%. Moreover, it provides deterministic and realistic paths for a significant part of the test cases.

Next, simulations for the same test cases are performed to evaluate the performance of the random sample generation approach, where the proposed random sampling approach is excluded from the algorithm and a random normal sampling only around the workspace midpoint is included. The results for 1410 simulations are shown in Figure 6.15, where the total mean solution time increases further to 1.76 s and the solution success rate decreases to 94.9%. It is clearly seen that introduction of a random sampling approach decreases the computational effort by around 15.3% and increases the solution success rate by around 4.5%.

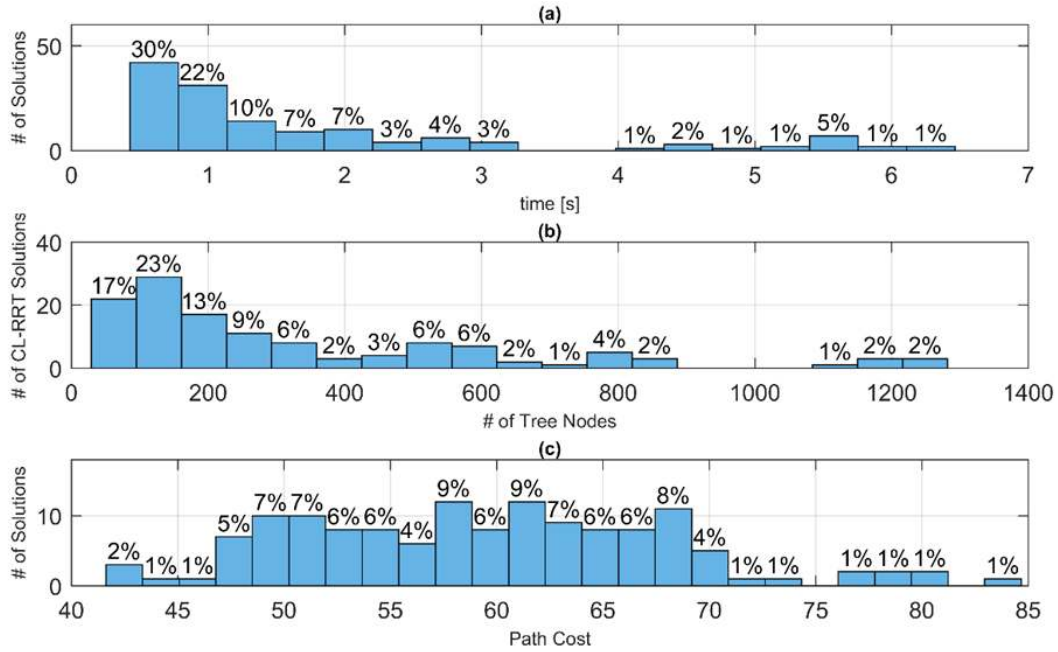


Figure 6.15. Histogram over the solutions for the path planning of CL-RRT with random normal sampling around workspace midpoint. Computation time, number of nodes in tree expansion and total path cost are shown from top to bottom respectively.

Finally, the predefined 141 test cases are tested with a lattice planner and A* search as in [10], where the lattice is planned with a position grid with discretization accuracy $r = 0.5\,m$ for both $x^d$and y^d, an equilibrium trailer angle discretization, δ^d, of 16 angles and an equilibrium steering angle discretization of 3 angles, $\alpha_e^d = [-0.2117, 0, 0.2117]$. The results of comparison for lattice-planner and cascade planner are shown in Fig, 16, where the mean path cost is increased by 5.09% to the value of 69.22 and the maximum path cost is increased by 33.5% to the value of 115.6 for lattice-planner simulations compared to the cascade planner. Further, the solution success rate for 141 test cases is decreased by 22.47% to the value of 77.3 in lattice-planner simulations compared to the cascade planner.

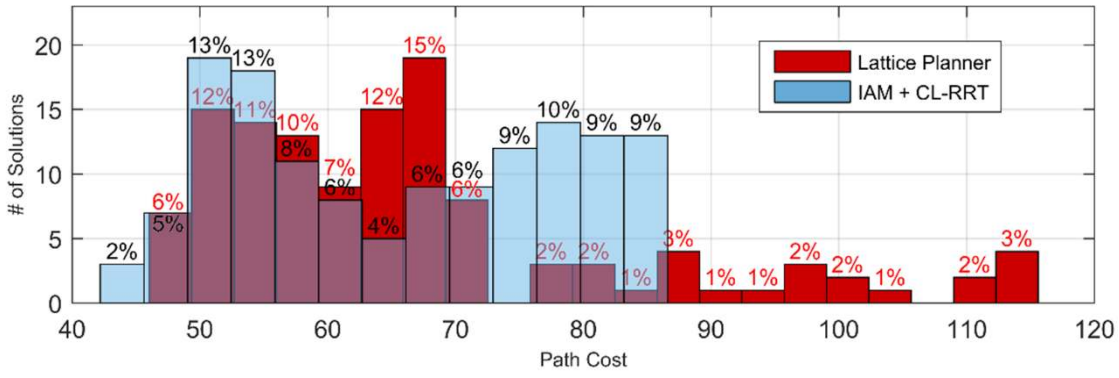


Figure 6.16. Total path cost histogram over the solutions for the path planning of Lattice Planner and Cascade Planner

6.1.4 *Related Work and Discussion*

Palmieri et. al. [57] developed a Theta*-RRT algorithm for nonholonomic motion planning and tested it with a truck-trailer system. It first generates an any-angle path from the vehicle position to the target position. Then it performs a tree expansion around the generated path by using a steering function to satisfy nonholonomic constraints. Subsequently, Li et. al. [34] introduced a progressively constrained optimal control algorithm (PCOC) for truck-trailer path planning in narrow environments where it starts with a coarse path generated by hybrid A* incorporated with Reed-Shepp curves, where the boundary conditions and collision avoidance constraints are fully ignored. Next, the generated coarse path is utilized as the initial guess for further step-by-step optimization based on the boundary conditions, collision avoidance, and kinematic constraints. The algorithm outputs optimized paths with the remarkable success of maneuverability in narrow environments. The main drawbacks of both algorithms lie in the discretization of search space in Theta* and hybrid A*, initialization with single maneuver solutions using Theta* and hybrid A*, and generating single maneuver solution paths in the end. Hence, both algorithms lack bidirectional maneuverability unlike CL-RRT-based approaches [9, 36], and are not well suited in use cases of trailer parking in narrow environments such as docking stations and construction sites, where the common solution path mostly consists of multiple maneuvers in forward and reverse directions. Especially for a use case where a multiple maneuver solution is necessary for the beginning due to the environment with high obstacle density as in Figure 6.12, the initial guess with hybrid A* + Reed-Shepp in [34] without considering the collision constraints may generate a single maneuver solution and could not be a suitable coarse path for further optimization including the collision avoidance constraints. Hence, the algorithm could not guarantee the optimal solution in such a use case. Moreover, first maneuvers in the solution path need to provide the necessary space for the vehicle to perform final reverse maneuvering successfully into the parking slot, hence the tree expansion around the any-angle path would not be a realistic approach for truck-trailer parking in the aforementioned use cases since it may not provide necessary tree expansion to find the solution path.

Furthermore, Hybrid A* is implemented for passenger cars with bidirectional maneuverability including U-Turn and parking capabilities [58], where the motion

primitives are generated in both forward and backward directions considering the collision constraints. Therefore, this approach could be extended to the truck-trailer system with an efficient steering method selection. Nevertheless, the algorithm will still suffer from the discretization of the search space where a possible solution maneuver in a narrow environment may not be achieved. Accordingly, hybrid A* is guaranteed to yield realizable paths, but it is not complete and may fail to find a path, where the possibility of success decreases with coarser discretization [59]. Consequently, the computational benefits of using the Reed–Shepp algorithm highly depend on the density of obstacles in the environment. Hence, Reed-Shepp expansions in obstacle-crowded environments such as in Figure 6.12 will mostly result in collisions, and therefore hybrid A* + Reed-Shepp would not work in such use cases [58].

TABLE 6.7:
Motion Planning Techniques for Truck-Trailer Systems

Algorithm	**Completeness & Optimality**	**Optimality**	**Application**
Theta*-RRT **[57]**	probabilistically complete	suboptimal	implemented for truck-trailer
Hybrid A*+PCOC **[34]**	incomplete	suboptimal	implemented for truck-trailer
CL-RRT **[9]**	probabilistically complete	suboptimal	implemented for truck-trailer
IAM+CL-RRT	probabilistically complete	suboptimal	implemented for truck-trailer
Lattice-A*/Dijkstra **[10]**	resolution complete	resolution optimal	implemented for truck-trailer
RRT* **[60]**	probabilistically complete	asymptotically optimal	extendable for truck-trailer
Bidirectional Hybrid A* **[58]**	incomplete	suboptimal	extendable for truck-trailer

Ljungqvist et. al. [10], Stahn et. al. [11], and Rimmer et. al. [61] developed lattice-based motion planning frameworks for two-trailer systems, where a library of motion primitives is generated and used to form an offline state lattice, and a collision-free path

is searched in the state lattice using A* search for [10, 11] and Dijkstra grid search algorithm for [61]. Such lattice or grid-based algorithms with the generation of offline motion primitives provide faster solutions compared to RRT-based solutions since motion primitives in RRTs are generated during the tree expansion. Furthermore, lattice-based approaches are not randomized and resolution optimal. However, lattice-based algorithms have the drawback of not using the full maneuvering capabilities of the truck-trailer system and possible initial and final configuration errors due to the discretization in the lattice, where the truck trailer position and orientation may not perfectly fit on a grid point. Increasing the grid resolution will decrease the configuration errors, however, it increases the computational effort as well. Accordingly, CL-RRT provides more flexible sampling, which tends to complete the search with fewer iterations and lower-cost solutions. The results of comparison for lattice-planner and cascade planner are shown in the previous section, where it is seen that the solution rate is lower and solution path costs are higher for the lattice planner compared to the cascade planner. It is also worth noting that the initial configuration error could be overcome with the introduction of a simple transition curve approach such as Dubins path or closed-loop steering [36] to connect starting ego vehicle pose to the first maneuver of the generated path since there is possibly more allowed free space around the initial maneuver to allow such a connection. However, a more sophisticated approach with higher computational effort such as solving an optimization problem [62] as post-processing is needed as post-processing to eliminate final configuration error since the final configuration lies in the parking slot and collision avoidance constraints are needed to be injected into the path optimization. Otherwise, extensive sampling is needed to achieve an acceptable goal pose due to the inability of the CL-RRT method to reach an exact goal configuration [9]. On the other hand, both IAM and CL-RRT starting from the goal pose in the parking slot assure zero final configuration error, and initial configuration error could be eliminated easily with one of the simple transition curve approaches.

Under mild technical conditions, the solution cost of sampling-based algorithms converges almost surely to a non-optimal value. Accordingly, the CL-RRT algorithm does not give any optimality guarantees on the solution and tends to give clearly suboptimal solutions, where a feasible path is found quickly and the remaining available computation time is used to improve the solution [14]. On the other hand, the extended

algorithm RRT* guarantees asymptotical optimality which is further implemented for nonholonomic systems in [60] through a locally optimal steering method, Dubins vehicle (not for truck-trailer systems), in order to satisfy the local controllability condition. However, the performance of the Dubins path is also highly dependent on the density of the obstacles as in the Reed-Shepp case. Hence, a feasible locally optimal steering method for truck-trailer systems with remarkable performance in environments with high obstacle density is firstly needed to extend the CL-RRT approach to RRT* in narrow parking spaces and guarantee asymptotical optimality. Consequently, discussed algorithms are summarized in Table V for completeness and optimality.

This study proposes IAM as the faster and more deterministic approach covering most of the scenarios for truck-trailer parking in narrow environments and CL-RRT as the more generic method with its probabilistic completeness [63] to cover remaining scenarios mostly interpreted in an environment crowded by obstacles. Both algorithms provide bidirectional maneuverability in the solution path consisting of maneuvers in forward and reverse directions, which is the most common characteristic in use cases of trailer parking in narrow environments such as docking stations and construction sites. A cascade path planning framework is proposed to combine the advantages of both algorithms with the introduction of an algorithm selector to determine which algorithm fits better for the specified parking scenario. Since the cascade framework directs the problem to CL-RRT after an unsuccessful IAM solution attempt, the estimation performance of the algorithm selector plays an important role in the computation performance of the cascade framework. The algorithm selector achieves a successful estimation rate of IAM selection in various test cases, which decreases the number of unnecessary IAM solution attempts.

6.2 Path Following Benchmark Simulations

The closed-loop system of path-following controllers with the truck-trailer kinematic model (3.1) is simulated in MATLAB/Simulink for various test cases and compared with the results in other studies. For simulations in both forward and reverse motions, the hitch angle is constrained by a permitted boundary of $[-45, 45]\ deg$. Benchmark simulations are summarized in Table 6.8 in terms of the objective, vehicle parameters, initial error state, controller gains and bounds, and results.

TABLE 6.8:
Benchmark Simulations

Objective	Vehicle Parameters	Initial Error State	Controller Gains and Bounds	Result
Straight *path in reverse motion*	$L_1 = 1.2\ m$ $L_2 = 1.2\ m$ $H = -0.45\ m$	$P_e = [\tilde{\beta}, \tilde{\delta}, e_{lat}] = [0, 0, 1]$	$k_L = 3.5$, $k_P = 0.48$ $k_\gamma = 0.52$ $k_v = 0.22$, $k_e = 2.98$ $L_u = 2.8\ m$, $L_l = 6\ m$ $Q_l = 2.0$, $Q_u = 10.5$	Converges in cases where [22] has marginally stable or unstable results, MO_{lat} = [0.04,0.09] m $e_{ss_{lat}}$ = [0.0021,0.0037] m (Fig. 14)
Circular path in reverse motion	$L_1 = 5\ m$ $L_2 = 5\ m$ $H = -2.50\ m$	$P_e = [\tilde{\beta}, \tilde{\delta}, e_{lat}] = [0, 0, 0]$	$k_L = 1.20$, $k_P = 0.60$ $k_\gamma = 0.30$ $k_v = 0.22$, $k_e = 2.98$ $L_u = 2.8\ m$, $L_l = 6\ m$ $Q_l = 0.50$, $Q_u = 3.60$	Lower maximum overshoot and steady-state error $MO_{lat} = 9.9$ m < 14.2 m [20] $e_{ss_{lat}} = 0.12$ m < 0.40 m [20] (Fig. 15)
A figure eight-shaped path in reverse motion	$L_1 = 9\ m$ $L_2 = 3.60\ m$ $H = 0.60\ m$	$P_e = [\tilde{\beta}, \tilde{\delta}, e_{lat}] = [0.1, 0.3, 0.42]$, $P_e = [\tilde{\beta}, \tilde{\delta}, e_{lat}] = [0.1, 0.1, 1]$	$k_L = 3.67$, $k_P = 2.48$ $k_v = 0.22$, $k_e = 1.0$ $k_\gamma = 0.52$ $L_u = 2.8\ m$, $L_l = 8\ m$ $Q_l = 0.50$, $Q_u = 11$	Faster convergence, $T_s = 11.8\ s$, $8.2\ s < 60\ s$ [36, 37] $e_{ss_{lat}} = 0.04\ m$ (Fig. 16)
A figure-eight shaped path in forward motion	$L_1 = 9\ m$ $L_2 = 3.60\ m$ $H = 0.60\ m$	$P_e = [\tilde{\beta}, \tilde{\delta}, e_{lat,tr}] = [\pi/6, 0, 3]$	$k_s = 16$, $k_v = 0.22$ $k_e = 1.80$ $L_u = 1\ m$, $L_l = 10\ m$	Faster convergence, $T_s = 8.6\ s < 45\ s$ [37] (Fig. 17)

6.2.1 *Straight Path in Reverse Motion*

The simulation platform is adjusted concerning the test parameters in the study of Pradalier [29] to compare with controller performance based on the simulation results. The vehicle parameters are defined, and controller gains are selected as shown in Table 6.8. Three simulations are performed with different maximum steering rates $\dot{\alpha}_{max}$ of $20\ deg/s$, $15\ deg/s$ and $10\ deg/s$, and the results are shown as in Figure 6.17. As compared to the simulation results in [29], where a marginally stable system at the rate limit of $15\ deg/s$ is achieved, but the trailer could not be stabilized at the rate limit of $10\ deg/s$; the results in this work converge for all three cases in a very close path-

following performance. This is shown in Figure 6.17b, where MO_{lat} is seen to vary from $0.04\ m$ to $0.09\ m$, and the steady state error ranges from $0.0021\ m$ to $0.0037\ m$.

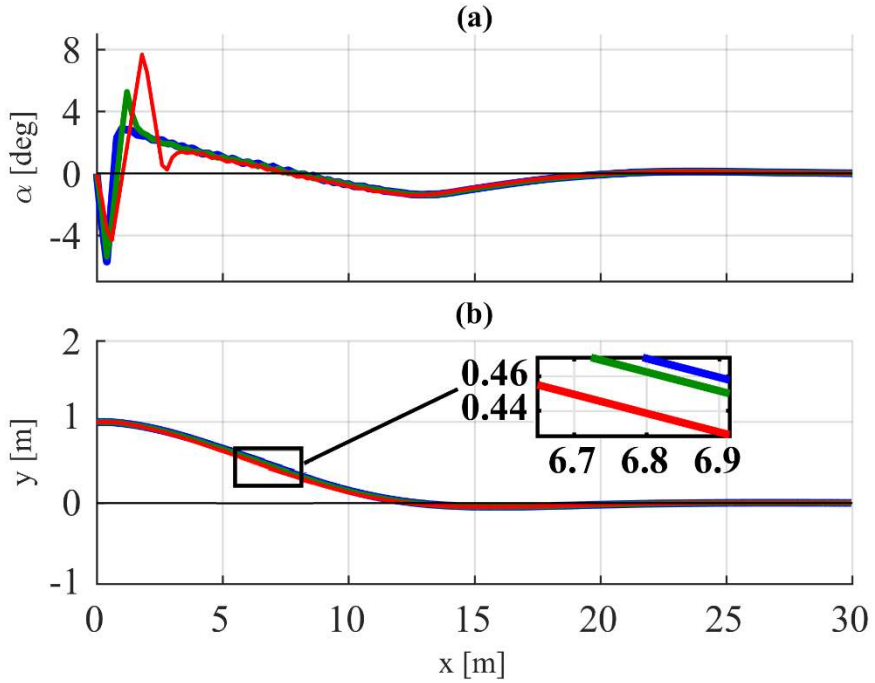


Figure 6.17. A straight path referencing simulation in reverse motion with different maximum steering rates $\dot{\alpha}_{max}$ $[deg/s]$: 20 (blue), 15 (green), 10 (red). a) steer change along x, b) xy plot with zoomed section

6.2.2 *Circular Path in Reverse Motion*

The simulation platform is adjusted according to the test parameters of a study by Astolfi [27], which compares controller performance based on the simulation results. The vehicle parameters and initial error states are defined, and controller gains are selected as shown in Table 6.8. The absolute steering angle rate is constrained by a maximum value of $34.37\ deg/s$. The simulation runs for 500 seconds to track a circular path, and the results are shown in Figure 6.18. It is worth noting that the aim of the setup is to achieve path-following in a counterclockwise direction. In this setup, the vehicle is placed in the opposite direction to observe the performance of the controller in first turning the vehicle to the correct direction before converging with the path. As compared to the simulation results shown in [27], where the truck-trailer converges with the path after $180\ s$ with a MO_{lat} of $14.2\ m$, and there is a steady state path-following

error of 0.40 m; the truck-trailer system simulated in this work converges in 80 s with a MO_{lat} of 9.9 m, and there is a steady state path-following error of 0.12 m.

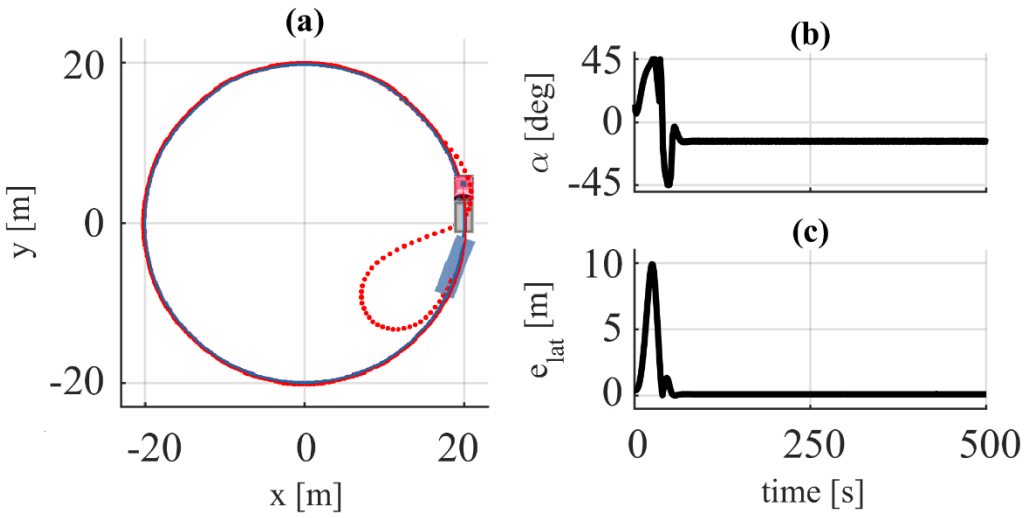


Figure 6.18. Circular path referencing simulation in reverse motion: a) xy plot b) Steering change in time c) following error change e_{lat} in time

As compared to the results in [27], where the truck-trailer converges to the path in 180 s with a maximum following overshoot of 14.2 m and steady state following error of 0.40 m; the truck-trailer system simulated in this work converges in 80 s with a maximum following overshoot of 9.9 m and steady state following error of 0.12 m.

6.2.3 *A Figure Eight-shaped Path in Reverse Motion*

The simulation platform is adjusted for a figure eight-shaped path, which is determined based on a study by Ljungqvist [55], to compare controller performance in terms of the results in studies [55, 54]. Since the simulations in these studies are performed on a two-trailer system, it has not been possible to adjust the vehicle parameters for the one-trailer system presented in this work. Therefore, the vehicle parameters of our test truck have been implemented for the simulations. Two simulations are performed in reverse motion with a vehicle speed of $v = -1\ m/s$, with $\dot{\alpha}_{max}$ being set to as determined by studies [54], [55]. The absolute steering angle rate is constrained with a maximum value of 34.37 deg/s. The vehicle parameters and initial error states are defined, and

controller gains are selected as shown in Table 6.8. As compared to the results in [55, 54], where the two-trailer systems converge around $60\,s$, the one-trailer system converges at $11.8\,s$ and $8.2\,s$ for the two simulations, respectively, as shown in Figure 6.19. Faster convergence achieved in one-trailer system simulations is not a surprising result, since stabilizing a two-trailer system is a more complex task to achieve. Subsequently, the steady-state error for both simulations is $0.04\,m$.

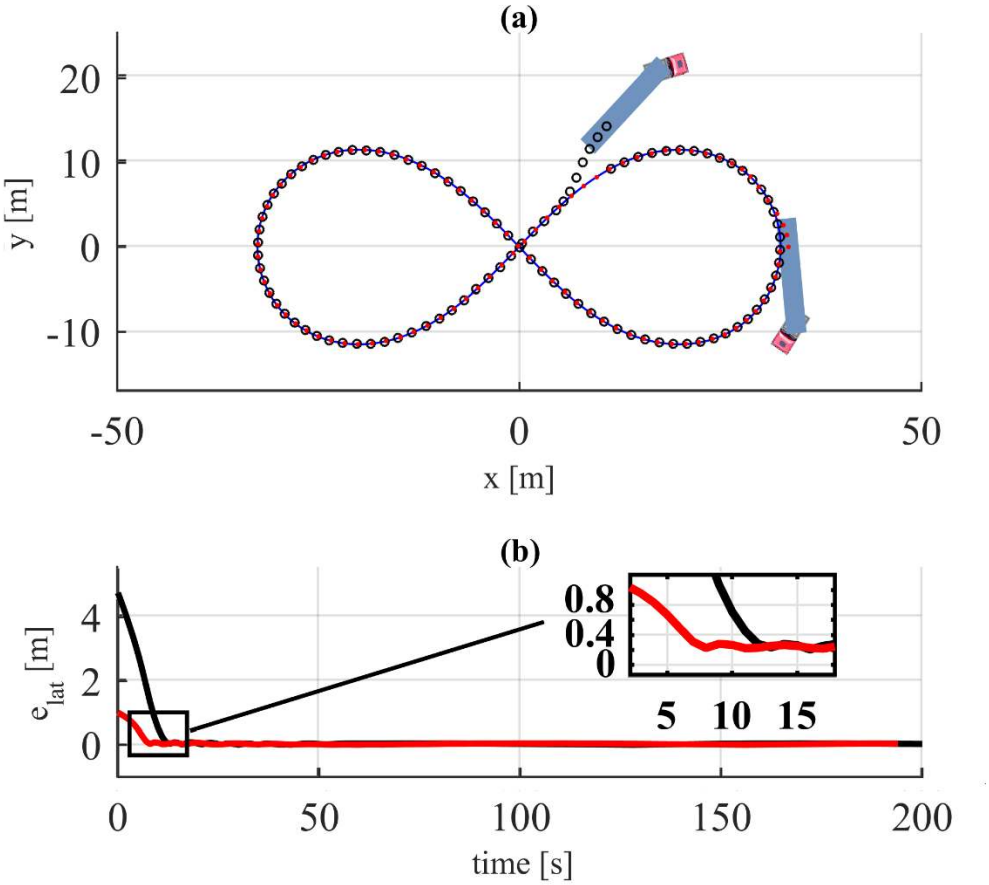


Figure 6.19. Figure eight-shaped path referencing simulations in reverse motion: a) xy plots b) Path-following error e_{lat} changes in time for simulations with an initial error state of P_e = [0.1, 0.3, 4.2] (black) and $[0.1, 0.1, 1]$ (red)

6.2.4 *A Figure-eight Shaped Path in Forward Motion*

The simulation platform in the previous section, which includes vehicle parameters, is tested in forward motion to compare controller performance with the results in the study [55]. The initial error state is defined, and controller gains are selected as shown in Table 6.8. Simulation is performed in a forward motion with a vehicle speed of $v = -1\ m/s$. The absolute steering angle rate is constrained by a maximum value of

40 deg/s. Results are shown in Figure 6.20. As compared with the results in [55] where the two-trailer system converges to the path around $45s$; the one-trailer system simulated in this work converges at $8.6s$ with negligible MO_{lat}. It could be argued that the performance of the forward controller simulated in kinematic models is not affected by the number of trailers. This is because the path-following is referenced through the rear axle of the truck, and so the kinematic model of a car could be directly transferred to the truck-trailer system in forward motion.

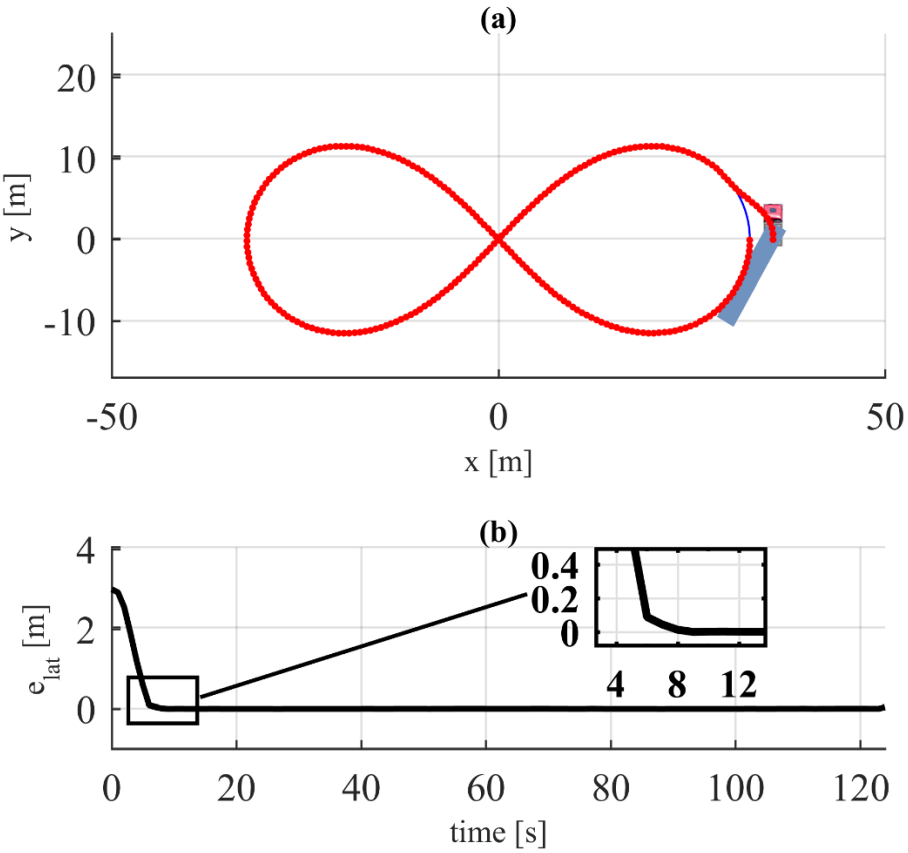


Figure 6.20. A figure eight-shaped path referencing simulation in forward motion: a) xy plot b) Path-following error e_{lat} changes in time within the enlarged section

6.3 TruckMaker Parking Simulation Results

The performance of the proposed controllers for forward and reverse path-following is evaluated in parking scenarios, regarding the control formulation defined in section 5.1, in the TruckMaker simulations and real-world experiments which were performed during the field testing. The path-following performance of the truck-trailer system in parking scenarios is evaluated in TruckMaker simulations for different parking scenarios, which were generated using the cascade path planning framework for

autonomous truck trailer parking. The maximum permitted steering angle $\alpha_{max,all}$, as an algorithm parameter, is determined in (5.14) by assigning $\gamma_{max,all}$ as $45\ deg$ to limit the path curvature. Longitudinal control is handled by TruckMaker's cruise control, which limits the vehicle speed to $2.78\ m/s$ while forward path-following and $-1.39\ m/s$ while reverse path-following. Precision braking control is also implemented with a PI controller which brings the vehicle to a halt at an acceptable lateral error threshold at the end of each maneuver.

6.3.1 *Adaptation of the TruckMaker Model for Multiple Maneuver Following*

The adaptation of the Simulink model for multiple maneuvers following is done as in Figure 6.21, where a state machine is introduced which switches between states of gas pedaling (cruise control), braking, and steering only (steering at stationary condition between two maneuvers).

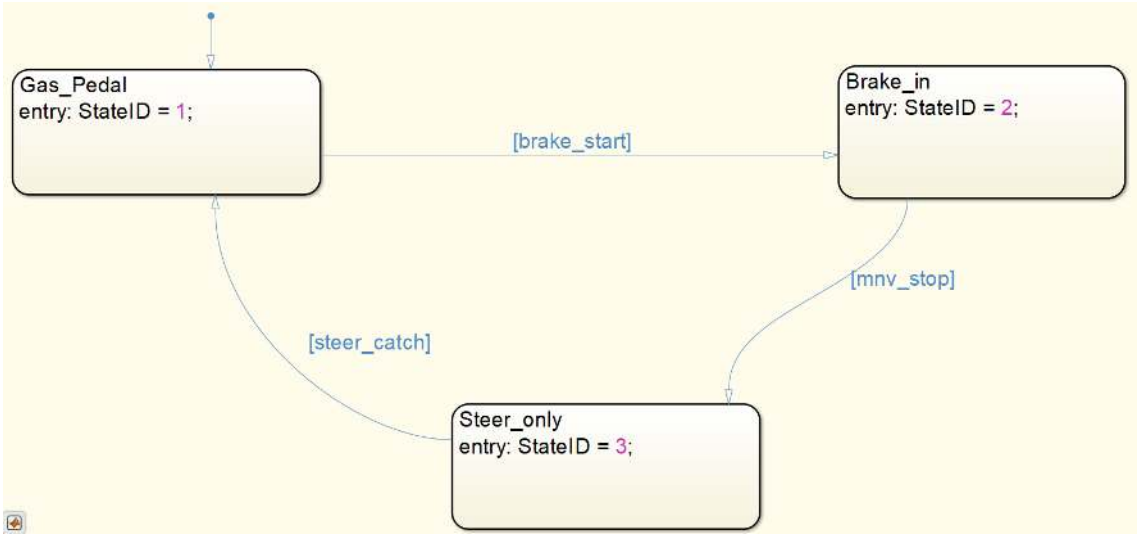


Figure 6.21. State flow for switching between different states

Implementation of switching conditions between gas pedaling, braking, and steering only states, their corresponding flags, and distance to the maneuver endpoint calculation is shown in Figure 6.22.

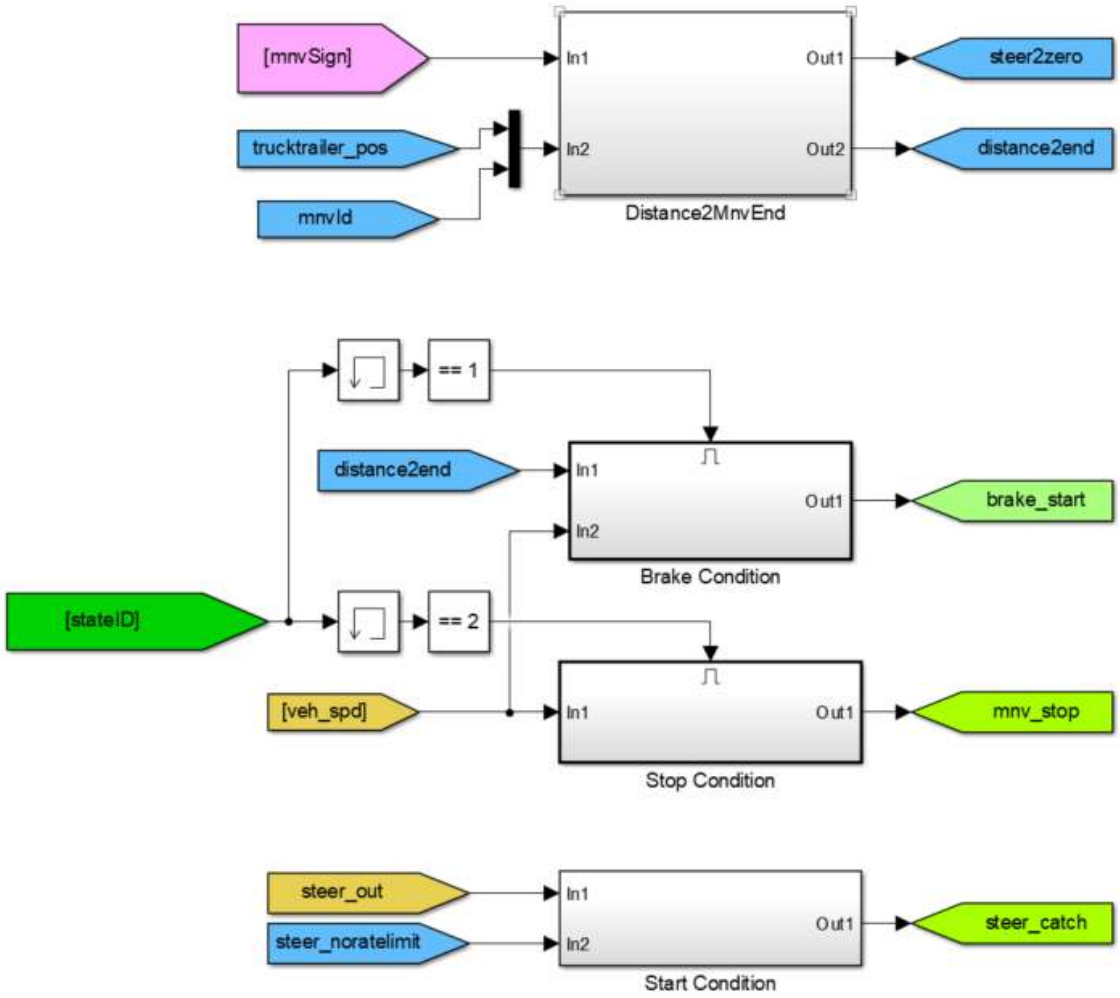


Figure 6.22. Switching conditions of states and distance to maneuver end calculation

Maneuver numbers are counted with a counter at each stop of the vehicle and assigned to the maneuver index for each maneuver, then the maneuver sign and corresponding maneuver curve are determined using the maneuver index and the total path data as shown in Figure 6.23. Finally, maneuver sign, state ID, maneuver curve, distance to maneuver endpoint, and vehicle speed are fed to the lateral and longitudinal controllers, and steering, gas, and braking setpoint control signal are generated as shown in Figure 6.24.

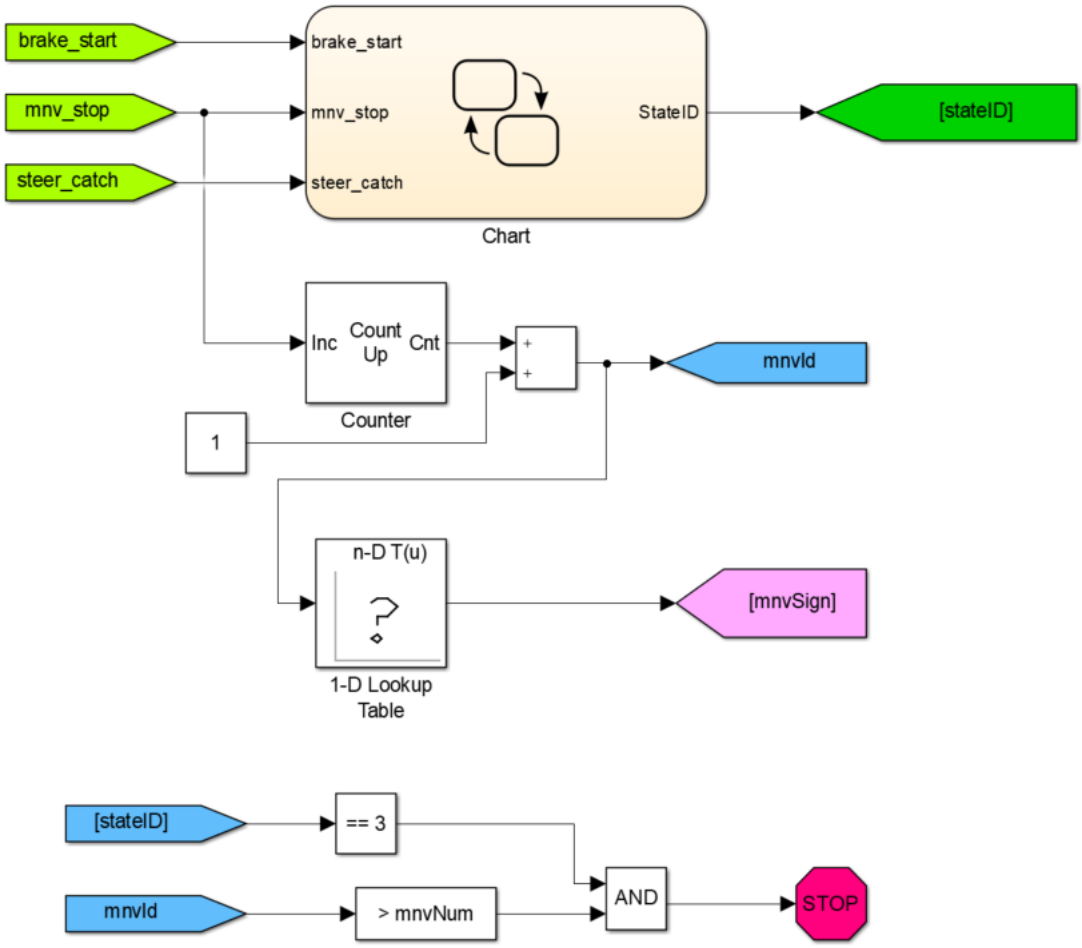


Figure 6.23. Counting and indexing maneuvers, and determination of state ID

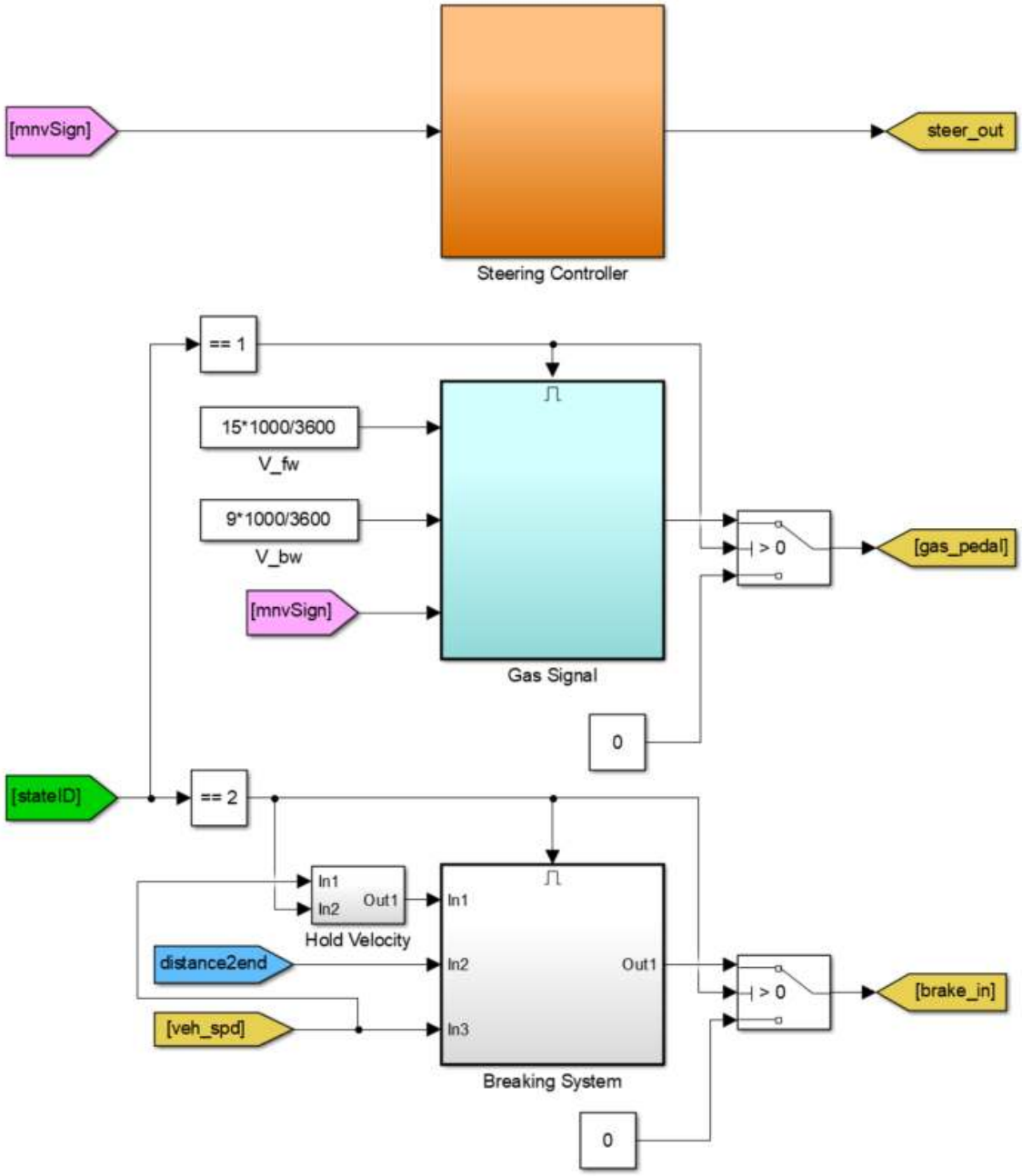


Figure 6.24. Introduction of lateral and longitudinal controllers in TruckMaker model

The visualization of parking maneuvers in the TruckMaker simulation is depicted in Figure 6.25. Pictures a) to f) refer to discrete-time interval vehicle positions during maneuvering. The forward maneuver is depicted from a) to c), where the gear number reaches up to 2 during throttle and stops after braking. The reverse maneuver is depicted from d) to f), where gear is selected as R during the maneuver and the vehicle stops after braking.

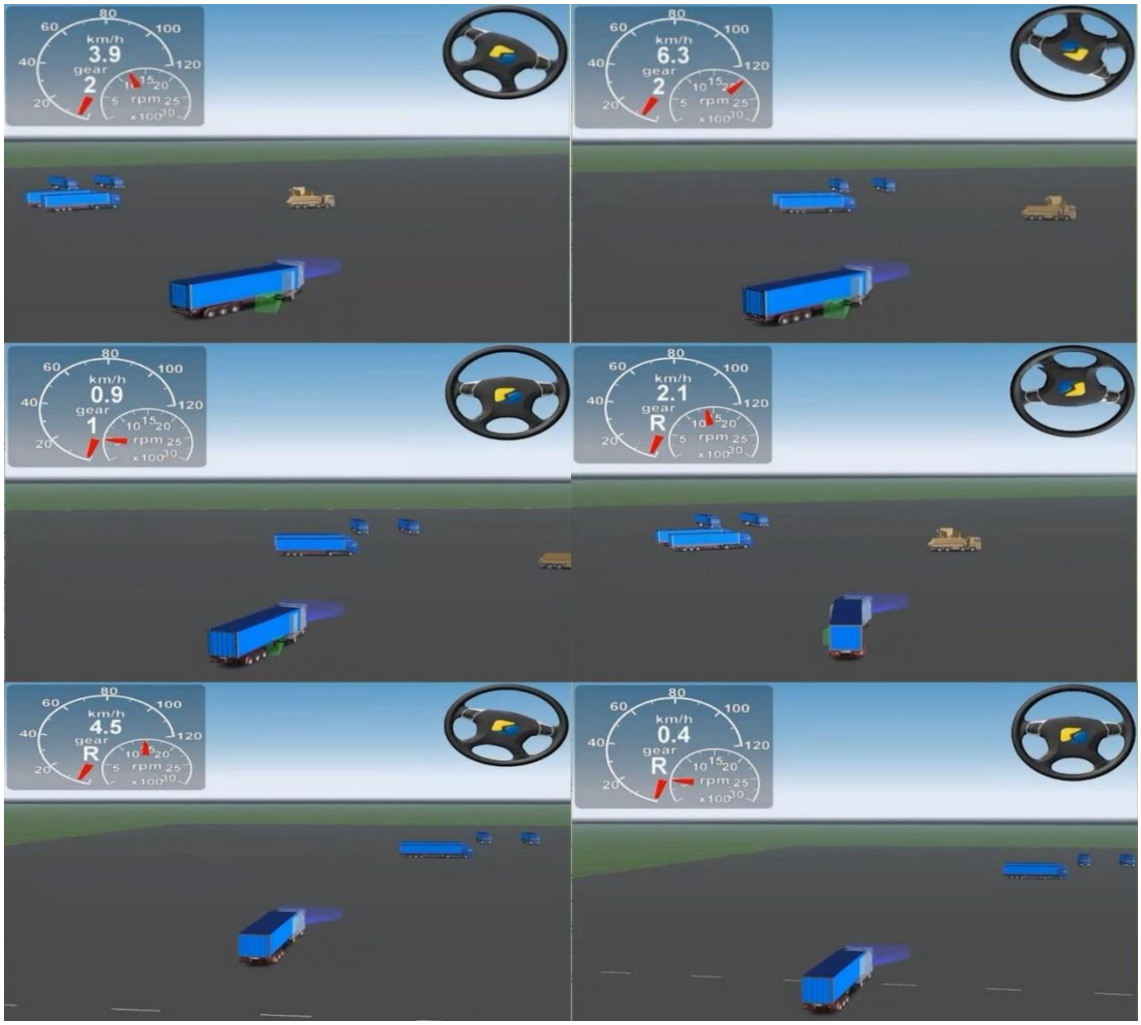


Figure 6.25. TruckMaker simulation visualization for autonomous trailer parking: Pictures a) to f) refer to discrete-time interval vehicle positions during maneuvering. The forward maneuver is depicted from a) to c), where the gear number reaches up to 2 during throttle and stops after braking. The backward maneuver is depicted from d) to f), where gear is selected as R during the maneuver and the vehicle stops after braking.

6.3.2 *TruckMaker Simulation Results*

Three parking scenarios with two-maneuver paths are generated using the cascade path planning framework for autonomous truck trailer parking, and these are shown in Figure 6.26a, Figure 6.26d, and Figure 6.26g. Successful parking is achieved via the TruckMaker simulations where the ef_{lat} ranges between $0.004\,m$ and $0.159\,m$. It is seen that during the path-following of sections with larger curvature values, MO_{lat} goes as high as $0.35\,m$, where MAE_{lat} along the paths remains within a range of $0.074\,m$ and $0.081\,m$. The maximum hitch angle γ_{max} reached along the path is $37.04\,deg$, which is, as expected, lower than $\gamma_{max,all}$. IAE_{lat} as a suitable path-following performance index varies between 4.19 and 6.02.

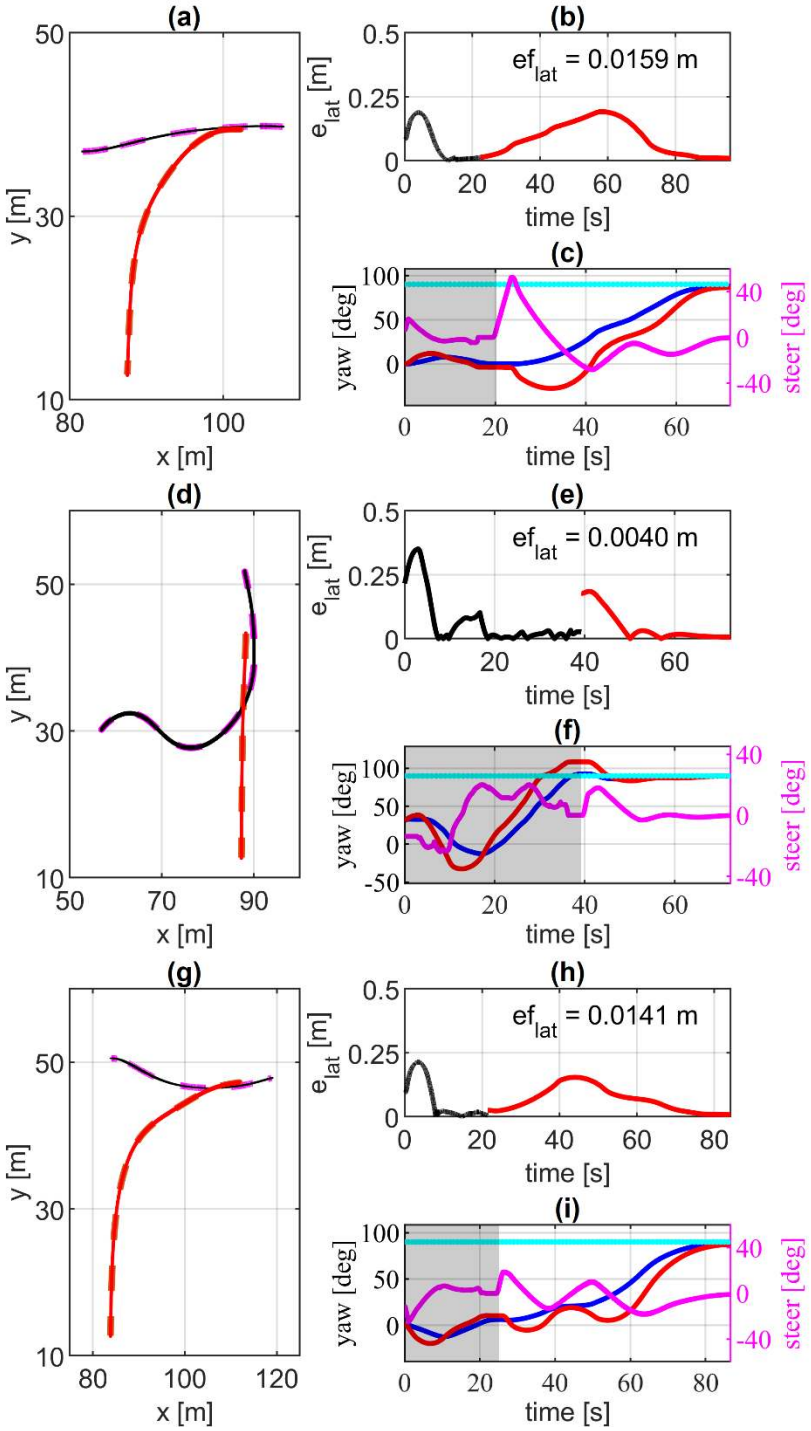


Figure 6.26. Results of the TruckMaker simulation in three scenarios: a), d), g) planned path – forward (black), reverse (red) vs tracked path – forward (magenta), reverse (dark green) b), e), h) lateral error, e_{lat}: forward (black), reverse (red) c), f), i) steering angle, α (magenta), truck-trailer yaw angles, β (blue), δ (red), final desired heading (cyan). Maneuver direction is indicated with background colors: forward (gray), reverse (white).

It would be possible to improve the path-following performance of the lateral controllers by introducing a speed profile generation algorithm that adapts vehicle speed based on the road curvature, rather than using cruise control. This would avoid steering requests that were above the steering rate limit of the physical system. The initial value of lateral error e_{lat} refers to the allowed initial error tolerance of the path planning algorithm. Furthermore, the discontinuity in e_{lat} during the maneuver change refers to the separate calculation of e_{lat} for forward and reverse maneuvers, in which e_{lat} is calculated via the truck rear axle in forward maneuvers, and via the trailer rear axle in reverse maneuvers. As shown in Figure 6.26c, Figure 6.26f, and Figure 6.26i; the lateral controller produces a smooth steering angle with no saturation due to the limits of the max steering angle. Convergences of truck-trailer heading angles to the desired final configuration ef_{β} and ef_{δ} are observed within an error interval lower than $1.24\ deg$.

6.4 Field Test Path-following Results

For real-world testing, a Ford FMAX truck and trailer are instrumented with two OxTS RT3003 RTK-GPS for localization, an electrohydraulic steering assist system, TeDrive iHSA, for steering control and dSpace MicroAutobox II for communication between vehicle CAN, motion planning user interface, and embedded controllers. Forward and reverse path-following controllers are operated via MicroAutobox II with an update frequency of 100 Hz. Truck RTK-GPS is located behind the driver seat or onto the rigid ground of the bed. Trailer RTK-GPS is located in the right corner of the trailer and fixed onto a flat, rigid surface. Localization is achieved by the transformation of latitude and longitude measurement through the RTK-GPS device into cartesian coordinates. These sensors are used so that path-following performance can be observed without additional error resources. The second processing unit, a Windows PC, is responsible for generating the paths that will be tracked. It consists of an HMI that allows the user to select or create a custom path. This planned path for parking is published as UDP packets to MicroAutobox II, through which lateral control input signals are calculated and lateral controllers are utilized. Longitudinal control is handled by the cruise control of the vehicle, in which the vehicle speed is limited to $2.78\ m/s$ for forward path-following and $-1.39\ m/s$ for reverse path-following.

Three parking scenarios with two-maneuver paths are introduced using the cascade path planning framework for autonomous truck trailer parking. This is shown in Figure

6.27a, Figure 6.27d, and Figure 6.27g; in which the $\gamma_{max,all}$ is selected as $45\,deg$. Successful parking is achieved for each scenario with a range of final lateral error, ef_{lat}, between $0.022\,m$ and $0.057\,m$. MO_{lat} during the path-following of sections with larger curvature values reaches $0.34\,m$, while MAE_{lat} along the paths remain within a range of $0.070\,m$ to $0.079\,m$. The maximum hitch angle γ_{max} achieved along the path is 0.48, which is, as expected, lower than $\gamma_{max,all}$. IAE_{lat} as a path-following performance index varies between 4.46 and 6.10, a similar interval to that observed in TruckMaker simulations. As shown in Figure 6.27c, Figure 6.27f, and Figure 6.27i; the same smooth steering angle behavior is observed as achieved in simulation tests, and no saturation occurred due to the limits of max steering angle. The truck-trailer final heading angle configurations of ef_{β} and ef_{δ} converge within an angle error reaching 1.76 deg.

The real-world visualization of parking maneuvers is depicted in Figure 6.28. The path planning algorithm used for both TruckMaker simulations and field tests plans the path from the parking space to the vehicle position and is based on an allowed tolerance at the vehicle's starting position. Hence, the ego vehicle is initially not on track and excessive forward motions occur in the first part of the forward maneuvers due to the explained initial configuration error. It is suggested that these excessive forward motions could be reduced by lowering allowed tolerances in the start ego pose of the path planning.

It was concluded that sudden lateral errors during the switching point between maneuvers for both TruckMaker and field tests are the result of the brake controller stopping the vehicle earlier than expected in some tests. More specifically, the vehicle stops on the first maneuver just before reaching the endpoint, which is connected to the first point of the second maneuver. Stopping in this way causes an initial configuration error at the start of the second maneuver, even though the last lateral error in the first maneuver is close to zero. It was therefore concluded that initial configuration error at switching points of maneuvers could be lowered in future work by improving the brake controller. For the forward path-following task, the controller could be extended by incorporating feedback control in future work. This would admit the guidance point on the trailer axle, hence the off-track error caused by the switching of guidance points between forward and reverse maneuvers could be further decreased.

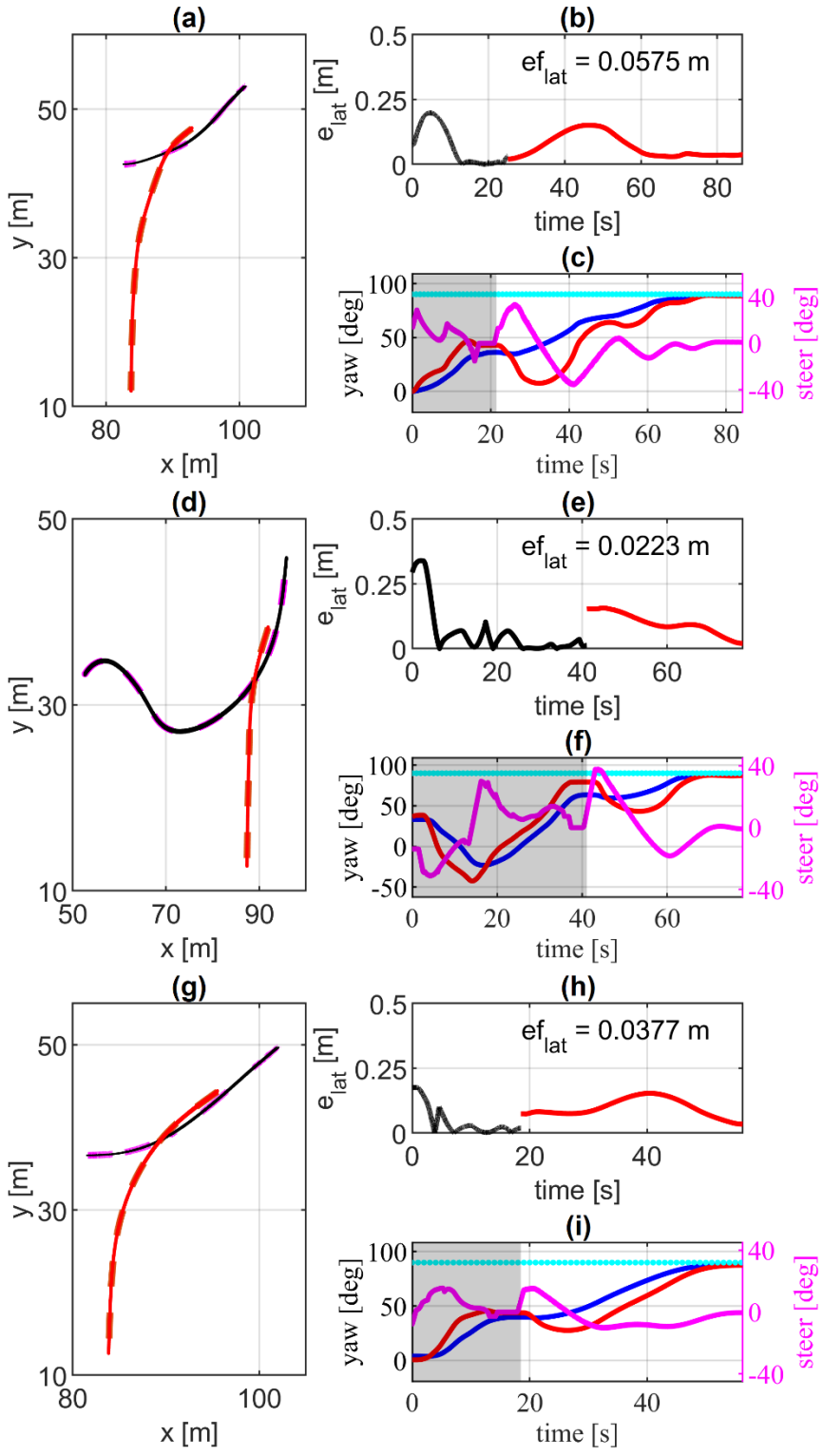


Figure 6.27. Real-world truck trailer test results for three scenarios: a), d), g) planned path – forward (black), reverse (red) vs tracked path – forward (magenta), reverse (dark green) b), e), h) lateral error, e_{lat}: forward (black), reverse (red) c), f), i) steering angle, α (magenta), truck-trailer yaw angles, $\boldsymbol{\beta}$ (blue), $\boldsymbol{\delta}$ (red), final desired heading (cyan). Maneuver direction is indicated with background colors: forward (gray), reverse

(white).

Figure 6.28. Real-world visualization of autonomous trailer parking maneuvers with fully equipped Ford F-MAX truck at Inonu test facility

Chapter 7

CONCLUSION

In this study, a novel cascade path planning framework for autonomous truck-trailer parking is presented. In the framework, Iterative Analytical Method (IAM) and Closed Loop Rapidly Exploring Random Tree Algorithm (CL-RRT) are combined and selected using generated likelihood and obstacle density parameters based on the real-world parking practices of truck drivers for reverse parking. Hence, the framework allows the user to combine the advantages of both algorithms with the introduction of the algorithm selector to determine which algorithm fits better for the specified parking scenario.

An enhanced path-following framework for autonomous semi-trailer docking is also presented in this study. The framework consists of pure pursuit controllers with an adaptive version for semitrailer docking to increase the robustness and path-following performance. In this chapter, the contributed results are summarized, and the future work is discussed.

7.1 Results and Discussions

In the cascade path planning framework, if there is a possible solution for IAM, it precedes CL-RRT due to its deterministic nature and faster solution time. The performance of the cascade path planning algorithm is tested through simulations of various parking cases, and it is shown that the cascade algorithm achieves a successful estimation rate of 85.1% for the selection of IAM. For the same set of simulations, the total success rate of the cascade algorithm reaches 99.7% with a mean solution time of 0.82 s for obtaining a feasible solution after the selection of either method. The introduction of IAM with cascade planner decreases computational effort by around 45% compared to the CL-RRT algorithm. Subsequently, the CL-RRT algorithm is enhanced with a random sample generation approach for faster convergence around 15.3% compared to random normal sampling around the workspace midpoint. CL-RRT solutions achieve lower path costs compared to IAM due to their flexibility to select minimum cost alternatives in multiple successful solutions through tree expansion. It is worth noting that, enhanced CL-RRT algorithm is tested also for different scenarios

such as parking in a docking station, construction site and maze environment. Further, the online CL-RRT algorithm is introduced and tested through MATLAB simulations for dynamic environment use cases.

For low-speed maneuvering in docking scenarios, forward and reverse controllers at the kinematic level are simple and real-time efficient, however, improved robustness and following performance is required for acceptable parking. To increase the robustness and path-following performance, an adaptive pure pursuit controller with look-ahead distance scheduling is introduced for forward path-following; a cascade controller of adaptive reverse pure pursuit with look-ahead distance scheduling and a gain-scheduled LQ controller is proposed for reverse path-following. Robustness verification clearly shows that the path-following error performance achieved with adaptive Q gain could not be achieved with constant Q gain in a classic LQ controller. Stability verification and performance evaluation are performed via extensive MATLAB and TruckMaker simulations to observe the effect of the dynamics on the stability region in different conditions. For benchmark, various test cases are performed and compared with the results of other studies. It was observed that path-following performance is more robust and promising with superior convergence times. This was validated by initially using the cascade path planning algorithm to generate kinematically feasible and collision-free maneuvers for trailer parking. TruckMaker simulations and real-world testing with a full-scale truck-semitrailer system are subsequently performed which achieve successful parking regarding the problem formulation in section II.A within a final position error range of up to $0.057\,m$, with final orientational errors up to $1.76\,deg$, and with MAE_{lat} below $0.081\,m$ and during path-following.

7.2 Future Work

For the ease of estimation, a cost heuristic is defined only through the length of pure pursuit arcs of branch candidates in each random sample to tree nodes attempt to avoid performing any simulation before selection of the most feasible reaching tree node. The cost definition could be improved in future work with the additional term of angle changes and by interpolating the cost through a look-up table which would be generated through offline simulations on a grid. Further, a feasible locally optimal steering method for truck-trailer systems could be introduced to extend the CL-RRT approach to RRT* in narrow parking spaces to guarantee asymptotical optimality.

MO_{lat} as an indicator for obstacle avoidance performance reaches a distance of up to $0.35\,m$ during maneuvering due to the initial configuration errors. This could be reduced in future work by lowering allowed tolerances at the initial ego pose in the path planning framework, and by improving the brake controller performance. As the off-track error is defined with respect to the guidance point of the vehicle system, the switching of this point between forward and reverse controllers requires high-performance path following, and so any moderate off-track error in the final configuration of a maneuver may escalate the initial off-track error of the next maneuver in the opposite direction. Hence, the forward path following task, the controller could be extended with feedback control in future work which admits the guidance-point on the trailer axle. Accordingly, the off-track error caused by the switching of guidance points between forward and backward maneuvers could be further decreased.

In the Online CL-RRT algorithm, only a simplified decision-making algorithm is introduced, where the vehicle only performs emergency braking whenever a collision is imminent. It could be extended in future work with the introduction of a more sophisticated state machine to define more detailed dynamic states of the vehicle and improve the dynamic behavior of the vehicle in a dynamic environment. Next, it could be further validated in TruckMaker simulations and real-world testing with the full-scale truck-semitrailer system.

Furthermore, to overcome use cases of temporarily absent GPS information, the environmental information part could be extended as a future work with the localization and tracking functions to provide tracked position and orientation information of the truck-trailer system.

References

[1] O. Holmer, *Motion Planning for a Reversing Full-Scale Truck and Trailer System,* MS thesis. Linköping University, 2016.

[2] F. Lamiraux, S. Sekhavat and J.-P. Laumond, "Motion planning and control for hilare pulling a trailer," *IEEE Transactions on Robotics,* vol. 15, no. 4, p. 640–652, 1999.

[3] A. Divelbiss and J. Wen, "A path space approach to nonholonomic motion planning in the presence of obstacles," *IEEE Transactions on Robotics and Automation,* vol. 13, no. 3, pp. 443 - 451, 1997.

[4] D. Zöbel, "Trajectory segmentation for the autonomous control of backward motion for truck and trailer," *IEEE Transactions on Intelligent Transportation Systems,* vol. 4, no. 2, pp. 59 - 66, 2003.

[5] F. Gomez-Bravo, F. Cuesta and A. Ollero, "Autonomous tractor-trailer back-up manoeuvering based on changing trailer orientation," *IFAC Proceedings Volumes,* vol. 38, no. 1, p. 301–306, 2005.

[6] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 4, pp. 1135 - 1145, 2016.

[7] M. Liang, X. Jianru, K. Kuniaki, Z. Jihua, M. Chao and Z. Nanning, "Efficient sampling-based motion planning for on-road autonomous driving," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 4, pp. 1961 - 1976, 2015.

[8] A. Elhassan, *Autonomous driving system for reversing an articulated vehicle,* MS thesis. KTH Royal Institute of Technology, Sept. 2015.

[9] N. Evestedt, O. Ljungqvist and D. Axehill, "Motion planning for a reversing general 2-trailer configuration using closed-loop RRT," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[10] 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, pp. 2455-2461*, 2017.

[11] R. Stahn, T. Stark and A. Stopp, "Laser scanner-based navigation and motion planning for truck–trailer combinations," in *in Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics pp. 1–6*, Zurich, 2007.

[12] O. Amidi and C. Thorpe, "Integrated mobile robot control. Technical report," CMU-RI-TR-90-17, Robotics Institute, Carnegie Mellon University, May 1990.

[13] M. Samuel, M. Hussein and M. Binti, "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle," *International Journal of Computer Applications,* vol. 135, no. 1, pp. 35-38, 2016.

[14] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions,* vol. 17, no. 5, p. 1105–1118, 2009.

[15] S. Thrun, M. Montemerlo, D. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny and G. Hoffmann, "Stanley: the robot that won the DARPA grand challenge," *Journal of Field Robotics,* vol. 23, no. 9, p. 661–692, 2006.

[16] M. Linderoth, K. Soltesz and R. M. Murray, "Nonlinear lateral control strategy for nonholonomic vehicles," in *2008 American Control Conference. IEEE*, Piscataway, NJ, 2008.

[17] A. Lombard, X. Hao, A. Abbas-Turki, A. El Moudni, S. Galland and A.-U.-H. Yasar, "Lateral control of an unmaned car using GNSS positionning in the context of connected vehicles," in *The 7th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2016), Procedia Computer Science 98 (2016) 148 – 155*, 2016.

[18] A. Kelly and A. Stentz, "An Approach to Rough Terrain Autonomous Mobility," in *International Conference on Mobile Planetary Robots*, Santa Monica, 1997.

[19] S. F. Campbell, *Steering control of an autonomous ground vehicle with application to the DARPA urban challenge,* Master's thesis: Massachusetts Institute of Technology, 2007.

[20] M. Park, S. Lee and W. Han, "Development of steering control system for autonomous vehicle using geometry-based path tracking algorithm," *ETRI Journal,* vol. 37, no. 3, p. 617–625, 2015.

[21] X. Ji, X. He, C. Lv, Y. Liu and J. Wu, "Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits," *Control Engineering Practice,* vol. 76, p. 41–53, 2018.

[22] Z. Chu, Y. Sun, C. Wu and N. Sepehri, "Active disturbance rejection control applied to automated steering for lane keeping in autonomous vehicles," *Control Engineering Practice,* vol. 74, p. 13–21, 2018.

[23] J. Felez, C. García-Sanchez and J. A. Lozano, "Control design for an articulated truck with autonomous driving in an electrified highway," *IEEE Access,* vol. 6, pp. 60171 - 60186, 2018.

[24] P. Kassaeiyan, B. Tarvirdizadeh and K. Alipour, "Control of tractor-trailer wheeled robots considering self-collision effect and actuator saturation limitations," *Mechanical Systems and Signal Processing,* vol. 127, p. 388–411, 2019.

[25] T. Nayl, G. Nikolakopoulos, T. Gustafsson, D. Kominiak and R. Nyberg, "Design and experimental evaluation of a novel sliding mode," *Robotics and Autonomous Systems,* vol. 103, p. 213–221, 2018.

[26] F. M. Barbosa, L. B. Marcos, M. M. da Silva, M. H. Terra and V. G. Junior, "Robust path-following control for articulated heavy-duty vehicles," *Control Engineering Practice,* vol. 85, p. 246–256, 2019.

[27] A. Astolfi, P. Bolzern and A. Locatelli, "Path-tracking of a tractor-trailer vehicle along rectilinear and circular paths: A Lyapunov-based approach," *IEEE Trans. Robot. Autom,* vol. 20, no. 1, p. 154–160, 2004.

[28] P. Bolzern, R. DeSantis, A. Locatelli and D. Masciocchi, "Path-tracking for articulated vehicles with off-axle hitching," *IEEE Transactions on Control Systems Technology,* vol. 6, no. 4, pp. 515 - 523, 1998.

[29] C. Pradalier and K. Usher, "Robust trajectory tracking for a reversing tractor trailer," *Journal of Field Robotics,* vol. 25, no. 6/7, pp. 378-399, 2008.

[30] A. J. Rimmer and D. Cebon, "Theory and practice of reversing control on multiply-articulated vehicles," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* vol. 230, no. 7, p. 899–913, 2016.

[31] N. Evestedt, O. Ljungqvist and D. Axehill, "Path tracking and stabilization for a reversing general 2-trailer configuration using a cascaded control approach," in *IEEE Intelligent Vehicles Symposium (IV) pp. 1156–1161*, 2016.

[32] T. Wu and J. Y. Hung, "Lateral position control for a tractor-trailer system using steering rate input," in *IEEE 26th International Symposium on Industrial Electronics (ISIE)*, Edinburgh, 2017.

[33] J. C. Lagarias, J. A. Reeds, M. H. Wright and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *SIAM Journal of Optimization,* vol. 9, no. 1, p. 112–147, 1998.

[34] B. Li, T. Acarman, Y. Zhang, L. Zhang, C. Yaman and Q. Kong, "Tractor-trailer vehicle trajectory planning in narrow environments with a progressively constrained optimal control approach," *IEEE Transactions on Intelligent Vehicles,* vol. 5, no. 3, pp. 414-425, 2019.

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

[36] K. Kvarnfors, *Motion planning for parking a truck and trailer system,* MA thesis. KTH Royal Institute of Technology, Sept. 2019.

[37] S. R. Chang, "G2 continuity smooth path planning using cubic polynomial interpolation with membership function," *Journal of Electrical Engineering and Technology,* vol. 10, no. 2, pp. 676-687, 2015.

[38] F. Gómez-Bravo, F. Cuesta, A. Ollero and A. Viguria, "Continuous curvature path generation based on β-spline curves for parking manoeuvres," *Robotics and Autonomous Systems,* vol. 56, no. 4, pp. 360-372, 2008.

[39] X. Bui, J. Boissonnat, P. Soueres and J. Laumond, "Shortest Path Synthesis for Dubins Non-

Holonomic Robot," in *IEEE Conference on Robotics and Automation*, 1994.

[40] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *IEEE Intelligent Vehicles Symposium IV pp. 518-522*, 2010.

[41] C. Ericson, Real-Time Collision Detection, Morgan Kaufmann Publishers Inc., Elsevier, 2004.

[42] C. Stein, T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms 3rd Edition, The MIT Press, 2009.

[43] J. L. Comba, *Computer Graphics Geometric Modeling,* Lecture Notes Handout, Stanford University, 1993.

[44] Y. Wang, J.-R. Chardonnet and F. Merienne, "Speed Profile Optimization for Enhanced Passenger Comfort: An Optimal Control Approach," in *21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

[45] J. Eriksson and L. Svensson, *Tuning for Ride Quality in Autonomous Vehicle Application to Linear Quadratic Path Planning Algorithm,* Master Thesis, Uppsala University, 2015.

[46] A. Baybura. and K. A., "Determination of minimum horizontal curve radius used in the design of transportation structures, depending on the limit value of comfort criterion lateral jerk," in *In Proceedings of Knowing to manage the territory, protect the environment, evaluate the cultural heritage*, 2012.

[47] M. Abdelwahab, V. Parque, A. M. R. Fath Elbab, A. A. Abouelsoud and S. Sugano, "Trajectory Tracking of Wheeled Mobile Robots Using Z-Number Based Fuzzy Logic," *IEEE Access,* vol. 8, pp. 18426-18441, 2020.

[48] R. L. O. Oliveira, P. F. Lima and B. Wahlberg, "A Geometric Approach to On-road Motion Planning for Long and Multi-Body Heavy-Duty Vehicles," *2020 IEEE Intelligent Vehicles Symposium,* pp. 999-1006, 2020.

[49] R. Oliveira, O. Ljungqvist, P. Lima and B. Wahlberg, "Optimization-Based On-Road Path Planning for Articulated Vehicles," *IFAC-PapersOnLine,* vol. 53, no. 2, pp. 15572-15579, 2020.

[50] C. Altafini, A. Speranzon and K. H. Johansson, "A feedback control scheme for reversing a truck and trailer vehicle," *IEEE Trans. Robot. Autom,* vol. 17, no. 6, pp. 915-922, 2001.

[51] J. Chiu and A. Goswami, "The critical hitch angle for jackknife avoidance during slow backing up of vehicle–trailer systems," *Vehicle System Dynamics,* vol. 52, pp. 1015 - 992, 2014.

[52] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli and J. P. Howk, "Motion Planning in Complex Environments using Closed-loop Prediction," in *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit.*, 2008.

[53] A. Ollero and G. Heredia, "Stability Analysis of Mobile Robot Path Tracking," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3, pp. 461-466*, 1995.

[54] 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, pp. 2455-2461*, 2016.

[55] O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo and H. Pettersson, "A path planning and path-following control framework for a general 2-trailer with a car-like tractor," *Journal of Field Robotics,* vol. 36, no. 8, pp. 1345-1377, 2019.

[56] O. Ljungqvist, D. Axehill and J. Löfberg, "On stability for state-lattice trajectory tracking control," in *Annual American Control Conference*, Milwaukee, WI, USA, 2018.

[57] L. Palmieri, S. Koenig and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[58] 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,* vol. 29, no. 5, pp. 485-501, 2010.

[59] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke and e. al., ""Junior: The Stanford entry in the Urban Challenge" in The DARPA urban challenge: autonomous vehicles in city traffic," *Berlin, Heidelberg, Germany: springer,* vol. 56, pp. 91-123, 2009.

[60] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, pp. 7681-7687, 2010.

[61] A. J. Rimmer and D. Cebon, "Planning collision-free trajectories for reversing multiply-articulated vehicles," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 7, p. 1998–2007, 2016.

[62] O. Holmer, *Motion Planning for a Reversing Full-Scale Truck and Trailer System,* MA thesis. Linköping University, June 2016.

[63] T. Kunz and M. Stilman, "Kinodynamic RRTs with Fixed Time Step and Best-Input Extension Are Not Probabilistically Complete," in *Algorithmic Foundations of Robotics XI*, 2015, pp. 233-244.

[64] B. S. Chen, C. S. Wu and H. J. Uang, "A minimax tracking design for wheeled vehicles with trailer based on adaptive fuzzy elimination scheme," *IEEE Transactions on Control Systems Technology,* vol. 8, no. 3, pp. 418 - 434, 2000.

[65] P. Ritzen, E. Roebroek, N. van de Wouw, Z. P. Jiang and H. Nijmeijer, "Trailer Steering Control of a Tractor–Trailer Robot," *IEEE Transactions on Control Systems Technology,* vol. 24, no. 4, pp. 1240 - 1252, 2016.

[66] M. Werling, P. Reinisch, M. Heidingsfeld and K. Gresser, "Reversing the general one-trailer system: Asymptotic curvature stabilization and path tracking," *Intelligent Transportation Systems, IEEE Transactions,* vol. 15, no. 2, p. 627–636, 2014.

[67] M. Michałek, "Cascade-Like Modular Tracking Controller for non-Standard N-Trailers," *IEEE Transactions on Control Systems Technology,* vol. 25, no. 2, pp. 619 - 627, 2017.

[68] M. Michałek and M. Kiełczewski, "Cascaded VFO Set-Point Control for N-Trailers With On-Axle Hitching," *IEEE Transactions on Control Systems Technology,* vol. 22, no. 4, pp. 1597 - 1606, 2014.

[69] N. Zimic and M. Mraz, "Decomposition of a complex fuzzy controller for the truck-and-trailer reverse parking problem," *Mathematical and Computer Modelling,* vol. 43, no. 5-6, pp. 632-645, 2006.

[70] C. Altafini, "Some properties of the general n-trailer," *International Journal of Control,* vol. 74, no. 4, pp. 409-424, 2001.

[71] O. Ljungqvist, *Motion Planning and Stabilization for a Reversing Truck and Trailer System,* MS thesis. Linköping University, 2015.

[72] C. Altafini, "Some properties of the general n-trailer," *International Journal of Control,* vol. 74, no. 4, pp. 409-424, 2001.

Appendix

Linearization of a nonlinear model

A time-invariant nonlinear system with no static input-output relation could be defined as below;

$$\begin{aligned} \dot{\mathbf{p}} &= F(\mathbf{p},\mathbf{u}) = A(\mathbf{p}) + B(\mathbf{p},\mathbf{u}) \\ \mathbf{y} &= H(\mathbf{p}) \end{aligned} \tag{A.1}$$

where **p, u** and **y** denote the states of the system, the control inputs and the measurements of the states respectively. In order to use linear control theory, the above equation shall be linearized around a stationary point $(\mathbf{p_e}, \mathbf{u_e})$, where $F(\mathbf{p_e}, \mathbf{u_e}) = 0$ with the necessary condition that F is one time continuously differentiable in a region of $(\mathbf{p_e}, \mathbf{u_e})$. Then, the first order Taylor expansion of (A.1) yields,

$$\begin{aligned} \dot{\mathbf{p}} &= A(\mathbf{p} - \mathbf{p_e}) + B(\mathbf{u} - \mathbf{u_e}) \\ \mathbf{y} &= C.\mathbf{p} \end{aligned} \tag{A.2}$$

where the matrices **A**, **B** and **C** are defined as below

$$\begin{aligned} \mathbf{A} &= \left.\frac{\partial A(\mathbf{p})}{\partial \mathbf{p}}\right|_{(\mathbf{p_e})} + \left.\frac{\partial B(\mathbf{p},\mathbf{u})}{\partial \mathbf{p}}\right|_{(\mathbf{p_e},\mathbf{u_e})} \\ \mathbf{B} &= \left.\frac{\partial B(\mathbf{p},\mathbf{u})}{\partial \mathbf{u}}\right|_{(\mathbf{p_e},\mathbf{u_e})} \\ \mathbf{C} &= \left.\frac{\partial H(\mathbf{p})}{\partial \mathbf{p}}\right|_{(\mathbf{p_e})} \end{aligned} \tag{A.3}$$

Then, the final linearized model with $\tilde{\mathbf{p}} = \mathbf{p} - \mathbf{p_e}$ and $\tilde{\mathbf{u}} = \mathbf{u} - \mathbf{u_e}$ yields to

$$\begin{aligned} \dot{\mathbf{p}} &= A\tilde{\mathbf{p}} + B\tilde{\boldsymbol{u}} \\ \mathbf{y} &= C.\mathbf{p} \end{aligned} \tag{A.4}$$

Linear Quadratic control

For a given linearized and controllable system as in (A.4), a Linear Quadratic controller (LQ) can locally stabilize the open-loop unstable system and control it to follow a specified reference signal [64]. The LQ controller can be defined as

$$\mathbf{u} = L_r(\mathbf{r}) + L(\mathbf{p} - \mathbf{p_e}) \tag{A.5}$$

where **r** is the reference signal. The feedback gain *L* is designed by solving the optimal control problem in (A.6) together with (A.4).

$$\min_{\mathbf{u}} J = \min_{\mathbf{u}} \int_0^\infty (\bar{\mathbf{p}}^{\mathrm{T}} Q_1 \bar{\mathbf{p}} + \bar{\mathbf{u}}^{\mathrm{T}} Q_2 \bar{\mathbf{u}})\, dt \tag{A.6}$$

Once the positive semi-definite weight matrices Q_1 and Q_2 are chosen as diagonal, the optimal feedback gain yields to

$$L = Q_2^{-1} B^{\mathrm{T}} P \tag{A.7}$$

where *P* is the solution to the algebraic Riccati equation (A.8), where *A* and *B* are defined by (A.4).

$$Q_1 + A^T P + PA - PBQ_2^{-1}B^{\mathrm{T}}P = 0 \tag{A.8}$$

Once L_r is written as $L_r(\mathbf{r}) = \mathbf{u}_d$, where $\mathbf{u}_d$ is the desired control signal, the LQ controller yields to

$$\mathbf{u} = \mathbf{u}_d - L\bar{\mathbf{p}} \tag{A.9}$$

In the special case with only one control signal **u**, Q_2 is a scalar and the cost function (A.6) can be manipulated with $Q \equiv Q_1/Q_2$ as in below:

$$\min_{\mathbf{u}} J = \min_{\mathbf{u}} Q_2 \int_0^{\infty} (\overline{\mathbf{p}}^{\mathrm{T}} Q \overline{\mathbf{p}} + \overline{\mathbf{u}}^2)\, dt = \min_{\mathbf{u}} \int_0^{\infty} (\overline{\mathbf{p}}^{\mathrm{T}} Q \overline{\mathbf{p}} + \overline{\mathbf{u}}^2)\, dt \qquad \text{(A.10)}$$

with the solution

$$0 = Q + A^T P + PA - PBB^{\mathrm{T}} P$$

$$L = B^{\mathrm{T}} P \qquad \text{(A.11)}$$

$$\mathbf{u} = \mathbf{u}_d - L\overline{\mathbf{p}}$$