Linköping Studies in Science and Technology. Licentiate Thesis No. 1832
Licentiate's Thesis

On motion planning and control for truck and trailer systems

Oskar Ljungqvist

LINKÖPING UNIVERSITY

Linköping studies in science and technology. Thesis
No. 1832

On motion planning and control for truck and trailer systems

Oskar Ljungqvist

LINKÖPING UNIVERSITY

This is a Swedish Licentiate's Thesis.

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree.
A Doctor's Degree comprises 240 ECTS credits (4 years of full-time studies).
A Licentiate's degree comprises 120 ECTS credits,
of which at least 60 ECTS credits constitute a Licentiate's thesis.

Linköping studies in science and technology. Licentiate Thesis
No. 1832

On motion planning and control for truck and trailer systems

Oskar Ljungqvist

oskar.ljungqvist@liu.se
www.control.isy.liu.se
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-7685-130-2 ISSN 0280-7971

Printed by LiU-Tryck, Linköping, Sweden 2019

To my family!

Abstract

During the last decades, improved sensor and hardware technologies as well as new methods and algorithms have made self-driving vehicles a realistic possibility in the near future. Thanks to this technology enhancement, many leading automotive and technology companies have turned their attention towards developing advanced driver assistance systems (ADAS) and self-driving vehicles. Autonomous vehicles are expected to have their first big impact in closed areas, such as mines, harbors and loading/offloading sites. In such areas, the legal requirements are less restrictive and the surrounding environment is more controlled and predictable compared to urban areas. Expected positive outcomes include increased productivity and safety, reduced emissions and the possibility to relieve the human from performing complex or dangerous tasks. Within these sites, different truck and trailer systems are used to transport materials. These systems are composed of several interconnected modules, and are thus large and highly unstable while reversing. This thesis addresses the problem of designing efficient motion planning and feedback control frameworks for such systems.

First, a cascade controller for a reversing truck with a dolly-steered trailer is presented. The unstable modes of the system is stabilized around circular equilibrium configurations using a gain-scheduled linear quadratic (LQ) controller together with a higher-level pure pursuit controller to enable path following of piecewise linear reference paths. The cascade controller is then used within a rapidly-exploring random tree (RRT) framework and the complete motion planning and control framework is demonstrated on a small-scale test vehicle.

Second, a path following controller for a reversing truck with a dolly-steered trailer is proposed for the case when the obtained motion plan is kinematically feasible. The control errors of the system are modeled in terms of their deviation from the nominal path and a stabilizing LQ controller with feedforward action is designed based on the linearization of the control error model. Stability of the closed-loop system is proven by combining global optimization, theory from linear differential inclusions and linear matrix inequality techniques.

Third, a systematic framework is presented for analyzing stability of the closed-loop system consisting of a controlled vehicle and a feedback controller, executing a motion plan computed by a lattice planner. When this motion planner is considered, it is shown that the closed-loop system can be modeled as a nonlinear hybrid system. Based on this, a novel method is presented for analyzing the behavior of the tracking error, how to design the feedback controller and how to potentially impose constraints on the motion planner in order to guarantee that the tracking error is bounded and decays towards zero.

Fourth, a complete motion planning and control solution for a truck with a dolly-steered trailer is presented. A lattice-based motion planner is proposed, where a novel parametrization of the vehicle's state-space is proposed to improve online planning time. A time-symmetry result is established that enhance the numerical stability of the numerical optimal control solver used for generating the motion primitives. Moreover, a nonlinear observer for state estimation is developed which only utilizes information from sensors that are mounted on the truck, making the system independent of additional trailer sensors. The proposed framework is implemented on a full-scale truck with a dolly-steered trailer and results from a series of field experiments are presented.

Populärvetenskaplig sammanfattning

Under de senaste årtiondena har utvecklingen av sensor- och hårdvaruteknik gått i en snabb takt och nya metoder och algoritmer har introducerats. Tack vare denna snabba utveckling har många ledande fordonstillverkare och teknikföretag börjat satsat på att utveckla avancerade förarstödsystem (eng. *advanced driver assistance systems* (ADAS)) och självkörande fordon. Även forskningen inom autonoma fordon har under de senaste årtiondena kraftig ökat då en rad problem återstår att lösas. Förarlösa fordon förväntas få sitt första stora genombrott i slutna miljöer, såsom gruvor, hamnar, lastnings- och lossningsplatser. I sådana områden är lagstiftningen mindre hård jämfört med stadsområden och omgivningen är mer kontrollerad och förutsägbar. Några av de förväntade positiva effekterna är ökad produktivitet och säkerhet, minskade utsläpp och möjligheten att avlasta människor från att utföra svåra eller farliga uppgifter. Inom dessa platser används ofta lastbilar med olika släpvagnskombinationer för att transportera material. En sådan fordonskombination är uppbyggd av flera ihopkopplade moduler och är således utmanande att backa då systemet är instabilt. Detta gör det svårt att utforma ramverk för att styra sådana system vid exempelvis autonom backning.

Ett självkörande fordon är ett mycket komplext system bestående av en rad olika komponenter som är designade för att lösa separata delproblem. Två viktiga komponenter i ett självkörande fordon är dels rörelseplaneraren som har i uppgift att planera hur fordonet ska röra sig för att på ett säkert sätt nå ett överordnat mål, och dels den banföljande regulatorn som har i uppgift att se till att den planerade manövern faktiskt utförs i praktiken trots störningar och modellfel. I denna avhandling presenteras flera olika metoder för att planera och utföra komplexa manövrar för en lastbil med släpvagn. De presenterade metoderna är avsedda att användas som ADAS eller som komponenter i ett helt autonomt system.

När ett autonomt fordon designas är det viktigt att systemet beter sig som tänkt. För planerings- och reglernivån kan detta översättas till att en planerad manöver utförs på ett kontrollerat och tillförlitligt sätt, det vill säga att det slutna systemet är stabilt. I denna avhandling presenteras två olika ramverk för att på förhand verifiera stabilitet hos det slutna systemet bestående av ett styrt fordon och en banföljande regulator, som utför en manöver beräknad av en rörelseplanerare. Ramverken baseras på att ålägga begränsningar på rörelseplaneraren och designa den banföljande regulatorn så att dess reglerfel garanterat är begränsade och avtar mot noll.

Experimentell validering är viktigt för att motivera att en föreslagen metod är användbar i praktiken. I denna avhandling har flera av de föreslagna rörelseplanerings- och reglermetoderna implementerats på en småskalig testplattform och utvärderats i en kontrollerad labbmiljö. Till sist presenteras ett fullständigt rörelseplanerings- och reglerramverk för en lastbil med släpvagn. Här föreslås en olinjär observatör för tillståndsskattning som endast använder sig av information från sensorer som är monterade på lastbilen, vilket gör systemet oberoende av sensorer på släpvagnen. Det föreslagna planerings- och reglerramverket har implementerats på en fullskalig lastbil med släpvagn och resultat för en serie av fältexperiment presenteras.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor Daniel Axehill for his excellent guidance and encouragement throughout this work. Thank you for our many discussions, your never-ending enthusiasm and support over the last three years. I would also like to direct a special thanks to Niclas Evestedt for fantastic collaboration in several projects during the last years. Furthermore, I am very grateful for valuable collaboration and support from Anders Helmersson and my co-supervisor Johan Löfberg. Financial support from the Strategic vehicle research and innovation (FFI) (Contract number: 2017-01957) and Scania CV are hereby gratefully acknowledged.

Also, thank you Svante Gunnarsson for inviting me to be part of the Automatic control group and for your excellent work as the former Head of the Division of Automatic Control. Thank you, Martin Enqvist for keeping up a nice and friendly working climate as the new Head of the Division. Thank you Ninna Stensgård for helping me with administrative things and for making sure everything runs smoothly in the group.

I gratefully acknowledge the help from Gustaf Hendeby regarding LaTeX issues and for providing the LaTeX-class that has been used to write this thesis. I also appreciate the help from Per Boström-Rost, Kristoffer Bergman, Martin Lindfors and Anders Helmersson for proof reading parts of this thesis. Your constructive feedback have gratefully improved the final touch of this thesis.

During my time as a PhD student at the Division of Automatic Control I have gained a lot of new and wonderful friends. A special thanks to Johan Dahlin and Andreas Bergström for all our refreshing lunch runs. Johan, thank you for your guidance during the beginning of my PhD studies and for being a good friend. Thank you Kristoffer Bergman and Per Boström-Rost for all our productive discussions over the years and for making the best out of the international WASP-trips. Thank you Daniel Simon and Isak Nielsen for an unforgettable road trip in the USA, where visiting Grand Canyon and saving the life of an injured stranger from a sandstorm in Death Valley are some of the highlights from the trip. Thank you everyone else from the Automatic Control group for making this a stimulating environment!

I am also grateful to Scania CV and the Autonomous Transport Solutions for long and fruitful collaboration. A special thanks to the Autonomous Motion team including Henrik Pettersson, Lars Hjort, Marcello Cirillo, Assad Alam, Christoffer Norén and many more. Thank you Henrik for learning me about classic sports cars and for widening my music taste during our long days on Scania's test track.

Last but not the least, I would like to thank my family for their encouragement, love and limitless support. Marie, thank you for all your love, for all wonderful adventures we have done together and for those to come.

Linköping, January 2019
Oskar Ljungqvist

Contents

Notation

Acronyms

Acronyms	Meaning
OCP	Optimal control problem
DP	Dynamic programming
NLP	Nonlinear programming
BVP	Boundary-vaule problem
PMP	Pontryain's minimum (maximum) principle
SQP	Sequentially quadratic programming
RRT	Rapidly-exploring random tree
CL-RRT	Closed-loop RRT
HLUT	Heuristic look-up table
AABB	Axis-aligned bounding box
OBB	Oriented bounding box
PID	Proportional, integral, derivative
LQ	Linear quadratic
MPC	Model predictive control
LTV	Linear time-varying
LPV	Linear parameter-varying
ARE	Alebraic Riccati equation
LMI	Linear matrix inequality
LDI	Linear differential inclusion
EKF	Extended Kalman filter
ADAS	Advanced driver assistance systems
FFI	Fordonsstrategisk forskning och innovation
GPS	Global positioning system
RADAR	Radio detection and ranging
IMU	Internal measurement unit
LIDAR	Light detection and ranging

Notation

Notation	Meaning
$\mathbb{R}$	The set of real numbers
$\mathbb{R}_+$	The set of positive real numbers
$\mathbb{Z}$	The set of integers
$\mathbb{Z}_+$	The set of positive integers
$\mathbb{Z}_{++}$	The set of strictly positive integers
$\mathbb{R}^n$	The n-dimensional Euclidean space
$\mathbb{R}^{n\times m}$	The set of real matrices with n rows and m columns
$\mathbb{S}^n_+$	The set of positive semi-definite matrices with n columns
$\mathbb{S}^n_{++}$	The set of positive definite matrices with n columns
$A \succeq 0$	The matrix A is positive semi-definite
$A \succ 0$	The matrix A is positive definite
rank A	The rank of a matrix A
Co P	The convex hull of a set of points P
$\lvert\mathcal{S}\rvert$	The cardinality of a set $\mathcal{S}$

Part I

Background

1 Introduction

This chapter introduces the reader to the research field of self-driving vehicles and advanced driver assistance systems (ADAS) and provides a motivation why research in this field is important. At the end of this chapter, an overview of the contributions and the outline of this thesis will be presented.

1.1 Background and motivation

In the last decades, emerging sensor and hardware technologies have made the idea of self-driving vehicles a near future reality. Ever since the groundbreaking DARPA Grand Challenges (Buehler et al., 2007) and DARPA Urban Challenge (Buehler et al., 2009) were held, many leading automotive manufacturers and technology companies have turned their attention to developing self-driving vehicles. Removing the human from the steering wheel is predicted to have positive effects on road safety, reduced greenhouse gas emissions and enhanced utilization of the overall vehicle fleet (Burns, 2013). Car manufacturers see added value to their customers and the possibility to gain a competitive edge to their competitors by providing this technology. At the same time, the transportation industry sees growing demands for delivering goods and transporting people. However, since the transportation industry is one of the largest emitter of greenhouse gases, the European Road Transport Research Advisory Council (2013) and the European Commission (2011) have set up strict goals for improvement of the European transportation system with an overall efficiency improvement by 50% in 2030 compared to 2010 while reducing emissions by 60%. To fulfill these requirements, the transportation industry sees a lot of potential in autonomous and intelligent transportation systems and are therefore also shifting their attention towards this fast-moving field.

Apart from environmental and efficiency aspects, safety is another concern. Annually, over 40 million people are injured in road traffic related accidents (World Health Organization, 2015), where most accidents occur due to human errors (European Commission,

Figure 1.1: *A truck with a dolly-steered trailer. In Paper E, this vehicle is used for experimental evaluation of a proposed motion planning and control solution.*

2011). These statistics show that a car is by far the most dangerous transportation alternative (Ernst and Young, 2015). Systems to increase safety have been considered for many years in the automotive industry, where anti-lock braking and electronic stability program are examples of ADAS that have been developed for this purpose. Thanks to recent development in sensor technology, *e.g.*, radio detection and ranging (RADAR), light detection and ranging (LIDAR) and camera sensors, more advanced ADAS have been developed and are now standard in many of today's (modern) cars. These systems can detect and handle critical situations much faster than an average driver. Examples of such systems are lane keeping assist, trailer assist, adaptive cruise control and queue assist. These systems are taking more and more control over the vehicle to aid the driver in critical or mentally exhausting situations. With even more advanced systems, such as parallel parking assist and the Tesla autopilot, fully autonomous vehicles are getting closer and have the potential to revolutionize the transportation sector. Autonomous vehicles are not only predicted to reduce the traffic related accidents, but also to transform today's transportation towards a more service-based system. Sharing autonomous vehicles predicts to result in a better overall utilization of the available vehicle fleet and contribute to a more sustainable future (Burns, 2013; Thrun, 2010).

Today, autonomous driving in urban areas with pedestrians, cyclists and other moving vehicles is still a hard challenge with many unsolved problems. This together with the legislation changes needed to drive autonomously on public roads make closed areas, such as mines, harbors and loading/offloading sites, perfect areas for initial deployment of self-driving vehicles. Here, expected positive outcomes are increased productivity and safety, reduced emissions, lower wear on the equipment and the possibility to relieve the human from performing complex or dangerous tasks. Within these sites, different truck and trailer combinations can be used to efficiently transport goods and other material. One such example is a truck with a dolly-steered trailer that is depicted in Figure 1.1. This vehicle combination consists of a truck with front-wheel steering, a dolly and a semi-trailer. This system is called a general 2-trailer system, where the word general refers to the non-zero off-axle hitch connection between the truck and the dolly (Altafini et al., 2002).

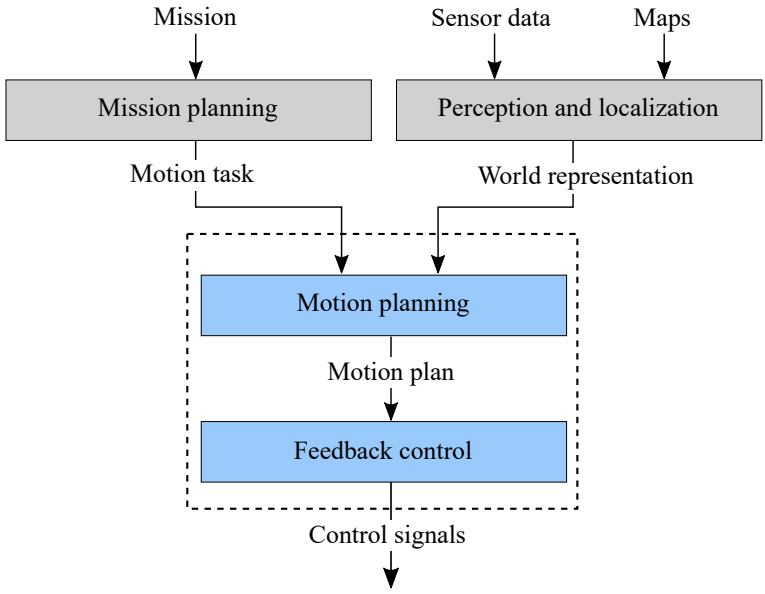


Figure 1.2: *An overview of a simplified system architecture for a self-driving vehicle where the blue subsystems are considered in this thesis. Inspired and adapted from (Evestedt, 2016; Lima, 2018).*

In comparison to cars and trucks without trailers, truck and trailer systems are larger and highly unstable while reversing. In particular, the dolly-steered trailer case is extra challenging. These properties increase the difficulty of designing efficient *motion planning* and *feedback control* frameworks for such systems, *e.g.*, for autonomous reversing. Moreover, to guarantee that a self-driving vehicle is behaving as intended, it is critical that stability of the closed-loop system is rigorously analyzed. Otherwise, the behavior of the system may be undesirable and in worst case even dangerous.

Before presenting the objective and the contributions of this thesis, the system architecture of a self-driving vehicle is first explained.

1.2 System architecture

A self-driving vehicle is a complex system that is composed of multiple subsystems, where each subsystem is designed to solve separate tasks. These subsystems communicate by transmitting and retrieving information between each other using a well-defined interface. An overview of a simplified system architecture of a self-driving vehicle is depicted in Figure 1.2. This work is focusing on the subsystems that are colored in blue, *i.e.*, the motion planner and feedback controller.

First, a self-driving vehicle needs a mission to perform. In industrial applications, a mission is in many cases defined in a complex way, where multiple vehicles need to cooperate to solve the task. A *mission planner* or *task planner* divides this problem into smaller subproblems and delegates them to suitable vehicles based on their capabilities. An example of a mission can be to load a truck and trailer with rocks. A solution would be to instruct the truck and trailer to go to a suitable loading area and the excavator to

pick up rocks and load the trailer until it is fully loaded. These subproblems can be further divided into even smaller subproblems. In this thesis, the focus is on solving one subproblem which is of great importance: To safely and robustly control a truck and trailer system such that the vehicle automatically can move to a desired location in absence of any clearly defined driving path. In such scenarios, the vehicle is said to operate in an *unstructured environment* and a motion planner is here required to computed how the vehicle should move to the desired location.

Before a maneuver can be planned and executed, the system needs to obtain a representation of the surrounding environment and understand where the vehicle is located in the world. These tasks are taken care of by the *perception* and *localization* layers, which use measurements from the vehicle's onboard sensors to construct this valuable information. A typical sensor platform on a self-driving vehicle is composed of multiple different sensors, such as global position system (GPS), internal measurement units (IMUs), RADAR sensors, LIDAR sensors and cameras. The localization layer fuses this information together with stored maps to estimate the vehicle's position, orientation and other important vehicle states (Skog and Händel, 2009). The perception layer makes use of the same sensor information to detect, classify and track different objects, such as people, other vehicles and important landmarks. Together, these layers provide the environmental representation in which motion planning and feedback control can be performed.

The objective of the motion planner is to compute a motion plan from the vehicle's current state to the desired goal state specified by the mission planner. The motion plan is a nominal trajectory or path that most often satisfies the system dynamics. It is computed such that the vehicle is predicted not to collide with any surrounding obstacles if executed with a small tracking error.

To guarantee safe execution of the motion plan, the feedback controller is designed to stabilize the vehicle around the nominal trajectory or path and to suppress disturbances and model errors. However, since the vehicle has limited sensor range, new obstacles may be detected by the perception layer while the motion plan is executed. In that case, the motion plan may need to be replanned, *e.g.*, if the previously computed motion plan is no longer safe to execute. In these situations, it is crucial that the motion planner is computationally efficient and the replanning phase does not induce instability in the closed-loop system consisting of the motion planner, the feedback controller and the controlled vehicle.

1.3 Objectives

There are two objectives of this thesis:

- To develop motion planning and feedback control frameworks for truck and trailer systems, and to validate their performance in practice.

- To develop structured design and analysis tools for verifying stability of the closed-loop system consisting of a controlled vehicle and a feedback controller executing a motion plan computed by a motion planner.

1.4 Thesis outline and contributions

The main work of this thesis is performed in the area of motion planning and feedback control for truck and trailer systems. It includes development of motion planning and feedback control frameworks, structured design tools for guaranteeing closed-loop stability and experimental validation of the proposed solutions through simulations, lab and field experiments. State estimation of a full-scale truck with a dolly-steered trailer is also considered for the case when the trailer is not equipped with any onboard sensors. The majority of the developed solutions are tested in both simulations and in practice on either a small-scale or a full-scale test platform.

Part I introduces the areas of nonholonomic systems, motion planning and feedback control to provide the reader with necessary background knowledge to understand the papers presented in Part II. Part I is concluded by Chapter 5 by summarizing the conclusions and discusses directions for future work.

Part II contains edited versions of four published papers and one submitted paper. A brief summary of each paper is given below.

Paper A: Path tracking and stabilization for a reversing general 2-trailer configuration using a cascade control approach

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

Summary: This work presents a cascaded control approach for path tracking of a reversing general 2-trailer. The unstable modes of the nonlinear system is linearized around circular equilibrium configurations and then stabilized using linear quadratic (LQ) control techniques together with gain-scheduling. A pure pursuit controller is designed on top in a cascaded control structure to allow for path following of piecewise linear reference paths. An easy to use driver aid is developed that can be used to manually plan complex maneuvers that the platform can execute. The system is tested in both simulation and on a small-scale test platform.

Background and contribution: The main contribution of this work was conducted by the author of this thesis during his Master's thesis project (Ljungqvist, 2015) in collaboration with his Master's thesis supervisor Niclas Evestedt and examiner Daniel Axehill. The lower-level gain-scheduled LQ controller is an extension of the work in Altafini et al. (2002) and the idea to use it together with a higher-level pure pursuit control for path tracking was developed by the author of the thesis. The implementation of the controller was performed by the author of this thesis and the idea to the user interface was a joint collaboration between all authors, where Niclas took care of the implementation. The hardware design of the small-scale test platform was accomplished in joint collaboration between the author of this thesis and Niclas as well as the experimental development and data collection. The majority of the writing of the manuscript was done by Niclas.

Paper B: Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT

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

Summary: This paper presents a probabilistic motion planning framework based on closed-loop rapidly-exploring random tree (CL-RRT) for the general 2-trailer configuration. The path following controller developed in Paper A is used to enable efficient closed-loop simulations of the system within the CL-RRT framework presented in Evestedt et al. (2015). The framework is evaluated in a series of simulation experiments in different kinds of environments and is shown to have a high success rate in finding motion plans in complex and constrained environments.

Background and contribution: The idea of performing closed-loop simulations of the general 2-trailer system within a CL-RRT framework was initiated from discussions between the Niclas Evestedt and Daniel Axehill. The concept was then developed by the author of this thesis during his Master's thesis project (Ljungqvist, 2015) where Niclas acted as supervisor and Daniel as examiner and supervisor. In an early stage, the author of this thesis and Niclas initiated a tight collaboration where the underlying CL-RRT platform was developed by Niclas (Evestedt et al., 2015) and the development as well as the integration of the stabilizing controller was performed by the author of this thesis. The experimental platform development and data collections were accomplished jointly between the author of this thesis and Niclas. The majority of the writing was done by Niclas and Daniel acted as supervisor and reviewed the manuscript.

Paper C: Path following control for a reversing general 2-trailer system

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

Summary: In this work, a path following controller for a reversing general 2-trailer is proposed for the case when the obtained motion plan is kinematically feasible to follow exactly. This is done by transforming the general 2-trailer system into a path coordinate system called the Frenet-Serret frame in which the control errors of the system are modeled in terms of its deviation from the nominal path. A stabilizing LQ controller with feedforward action is then design based on the Jacobian linearization of the error model around a straight nominal path. Given that the motion planner is constructing motion plans for a specified set of possible motions, the origin of the closed-loop linear parameter varying (LPV) control error system is shown to be an exponentially stable equilibrium point. The results are established by combining global optimization, theory from linear differential inclusions (LDIs) and linear matrix inequality (LMI) techniques. Furthermore, it is show that the established result implies that the origin of the nonlinear closed-loop control error system is an exponentially stable equilibrium point. The theoretical results

are verified in practice through simulations of the closed-loop system around an eight-shaped reference path.

Background and contribution: The idea to this work was initiated through discussion between the author of this thesis, Daniel Axehill and Anders Helmersson. Throughout the process, a tight collaboration between the authors were held. The modeling of the vehicle in the Frenet-Serret frame was performed by the author of this thesis as well as the theoretical derivations, implementations, numerical calculations and the written manuscript. Daniel and Anders acted as supervisors and reviewed the manuscript.

Paper D: On stability for state-lattice trajectory tracking control

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

Summary: This paper presents a systematic framework for analyzing stability of the closed-loop system consisting of a controlled vehicle and a feedback controller executing a motion plan computed by a lattice planner. When this motion planner is considered, it is shown that the closed-loop system can be modeled as a nonlinear hybrid system. Based on this, we propose a novel method for analyzing the behavior of the tracking error, how to design the low-level controller and how to potentially impose constraints on the motion planner, in order to guarantee that the tracking error is bounded and decays towards zero. The proposed method is applied on a truck and trailer system and the results are illustrated in two simulation examples.

Background and contribution: The idea to this work evolved after discussion between the author of this thesis and Daniel Axehill. Johan Löfberg was involved in some of the discussions and in particular in the development of Proposition 1. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, numerical calculations and the written manuscript. Daniel contributed throughout the process and acted as supervisors and reviewed the manuscript.

Paper E: A motion planning and control framework for a self-driving truck and trailer system

> O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo, and H. Pettersson. A motion planning and control framework for a self-driving truck and trailer system. *Journal of Field Robotics*, Under review.

Summary: This paper presents a complete motion planning and control solution for a truck with a dolly-steered trailer that can be used to automatically plan and execute difficult parking and obstacle avoidance maneuvers by combining forward and backward motion. A lattice-based motion planning framework is utilized in order to generate kinematically feasible and collision-free motion plans in real-time. To execute the motion plan, a path following controller is developed that stabilizes the lateral and angular control

error states of the vehicle. Moreover, a nonlinear observer for state estimation is developed which only utilize information from sensors that are mounted on the truck, making the system independent of additional trailer sensors. The proposed planning and control framework is implemented on a full-scale test vehicle and a series of field experiments are presented.

Background and contribution: This work summarizes a long project where the motion planner from Ljungqvist et al. (2017) and the path following controller from Paper C were implemented on a full-scale test platform. Many extensions from those works have been made to cope with the full-scale test platform and have mainly been performed by the author of this thesis together with Niclas Evestedt. Niclas started the implementation of the nonlinear observer before leaving academia. The author of this thesis took over and made many improvements to make the nonlinear observer work robustly for all relevant scenarios. The field experiments and the tuning of all modules were made by the author of this thesis together with Henrik Pettersson. The author of this thesis contributed with the majority of the work including theoretical derivations, numerical calculations, field experiments and the written manuscript. Niclas, Daniel Axehill and Marcello Cirillo contributed throughout the process and reviewed the manuscript.

1.5 Other publications

The following additional publications have been authored or co-authored by the author of this thesis:

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

G. Ling, K. Lindsten, O. Ljungqvist, J. Löfberg, C. Norén, and C. A. Larsson. Fuel-efficient model predictive control for heavy duty vehicle platooning using neural networks. In *Proceedings of the 2018 Annual American Control Conference (ACC)*, pages 3994–4001, June 2018.

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

1.6 Contributions

The author is particularly proud to present the following scientific contributions in this thesis.

- The development of the cascade control structure for a reversing general 2-trailer (Paper A) and to use of this solution within a CL-RRT framework (Paper B). To the authors knowledge, this was the first motion planning framework for a general 2-trailer system.
- The derivation of the control error model for the general 2-trailer system and the developed tool for analyzing stability of the closed-loop system, where the resulting constraints on the motion plan can be used by a motion planner to guarantee safe path execution. (Paper C).
- The modeling of the closed-loop system (a lattice planner, a feedback controller and a controlled vehicle) as a hybrid system and the structure which was exploited to develop a tailored controller synthesis and stability analysis tool to a priori guarantee stability of the closed-loop system. (Paper D)
- The development and experimental evaluation of a complete motion planning and control architecture for a full-scale truck with a dolly-steered trailer including a lattice planner, a path following controller and a nonlinear observer that is only utilizing information from sensors that are mounted on the truck. To the authors knowledge, this is the first complete motion planning and control architecture for a full-scale truck with a dolly-steered trailer that is demonstrated in practice. (Paper E)

2

Modeling of wheeled vehicles

This chapter is intended to present some models that are commonly used for motion planning and feedback control of wheeled vehicles during low-speed maneuvers. This chapter starts with a brief introduction to different modeling techniques. In Section 2.2, a short introduction to nonholonomic systems is presented. In Section 2.3 and 2.4, the kinematic bicycle model and the general n-trailer are presented, respectively.

2.1 Introduction

Modeling of wheeled vehicles has a long history and depending on the application, different modeling techniques are used (LaValle, 2006; Rajamani, 2011). Roughly speaking, a vehicle model is said to be *dynamic* if it is derived based on force balances, or *kinematic* if it is derived based on velocity constraints. These velocity constraints will be further referred to as *nonholonomic constraints*. For wheeled vehicles, these constraints naturally arise if the wheels of the vehicle are assumed to be rolling without slipping. During low-speed maneuvers on dry road surface conditions, a kinematic model is often sufficient to describe the behavior of the vehicle. On the contrary, during aggressive high-speed maneuvers, the dynamical effects becomes more dominant and a dynamic model is required to capture these effects. In this chapter, the basic tools for modeling the kinematic properties of the vehicle are presented, which is a popular approach in motion planning and control applications (LaValle, 2006; Paden et al., 2016; Spong et al., 2006). Here, only a brief introduction is provided and the reader is referred to, *e.g.*, (LaValle, 2006; Spong et al., 2006) for a more detailed coverage. A major reason why more advanced vehicle models with higher fidelity are usually not used for planning is that advanced models imply a high dimensional state-space. Since most practical applications require real-time performance, kinematic models with lower state dimension are commonly used (Paden et al., 2016). However, a more advanced vehicle model could preferably be used for simulation purposes (Lima, 2018; Rajamani, 2011).

2.2 Nonholonomic systems

A kinematic model of a wheeled vehicle is derived based on the assumption that the wheels of the vehicle are rolling without slipping. This implies that their exist velocity directions in which the vehicle cannot move. For systems subject to such velocity constrains, the system is called nonholonomic.

Let $q \in \mathbb{R}^n$ denote a configuration of the vehicle, *i.e.*, an n-dimensional vector of generalized coordinates. The configuration space $\mathcal{C}$ is defined as the manifold of all possible vehicle configurations and is assumed to be a smooth manifold (LaValle, 2006). For modeling of wheeled vehicles, we are particularly interested in constraints in the following form

$$g(q,\dot{q}) = 0. \tag{2.1}$$

These are constraints on the velocities of the system, *e.g.*, a car cannot move sideways. In some cases, velocity constraints can be explicitly integrable, giving rise to constraints in the form

$$h(q) = 0, \tag{2.2}$$

which are algebraic constraints in the configuration of the vehicle. Such constraints are said to be *holonomic* and the motion of the vehicle is thus restricted to a level surface of h. If a velocity constraint is not explicitly integrable the constraint is said to be *nonholonomic*. There exist different holonomic and nonholonomic constraints that naturally arise due to physical limitations of the system. Two common examples are

$$\begin{aligned} h(q) \leq 0: &\quad \text{Configuration inequality constraint} \\ h(\dot{q}) \leq 0: &\quad \text{Velocity inequality constraint} \end{aligned}$$

These are inequality constraints on the configuration q and the velocity $\dot{q}$ of the system and two examples of such constraints are limited steering angle and steering angle rate for a car. In the remainder of this chapter we focus on the special class of nonholonomic systems that have linear velocity constraints

$$\omega_i(q)\dot{q} = 0 \quad i = 1,\ldots,k < n, \tag{2.3}$$

where $\omega_i(q) \in \mathbb{R}^{1\times n}$. These constraints are called *Pfaffian constraints* and it is assumed that the ω_i's are smooth functions and linearly independent. An interpretation of the linear velocity constraints in (2.3) is that the vector fields of the system have to be orthogonal to each $\omega_i(q)$. We represent the linear velocity constraints in (2.3) on matrix form as $\omega(q)\dot{q} = 0$, where $\omega(q) \in \mathbb{R}^{k\times n}$.

Instead of expressing which directions the vehicle cannot move, it is more convenient to express which directions it can, *i.e.*, transform the constraints in (2.3) to explicit form $\dot{q} = f(q,u)$ (LaValle, 2006). Let $m = n-k > 0$ and choose $g_1,\ldots,g_m$ as a basis of right null-space of $\omega(q)$. By assigning each $g_j \in \mathbb{R}^n$ with a control signal u_j, we end up with a *control-affine driftless system*

$$\dot{q} = \sum_{j=1}^{m} g_j(q)u_j. \tag{2.4}$$

A common property for nonholonomic systems is that they are underactuated, *i.e.*, the number of control signals m is less than the dimension of the configuration-space n. Generally, a nonholonomic system can be modeled as a nonlinear system

$$\dot{x} = f(x,u), \tag{2.5}$$

where the state vector x is chosen as a subset of the configuration q or additional properties are also modeled, *e.g.*, $x = (q,\dot{q})$ when dynamics are also considered (LaValle, 2006). However, the degree of freedom m for the system will not change. Now, models of nonholonomic systems that are of particular importance for the contents of this thesis are derived.

2.3 The kinematic bicycle model

A vehicle model that is used to large extent in motion planning and control literature is the kinematic bicycle model (LaValle, 2006; Paden et al., 2016). During low-speed maneuvers, cars and trucks (Lima, 2018; Paden et al., 2016) have shown to be well described by this relatively simple model that is depicted in Figure 2.1. The vehicle is assumed to operate on a flat surface and front wheel steered with perfect Ackerman steering. Briefly, the Ackerman steering geometry makes the inner front wheel turn more than the outer front wheel in a corner. This implies that there exists an average steering angle α that approximates the two front steering angles. Denote wheelbase as L_1 and the longitudinal velocity of the rear axle of the vehicle as v_1. The kinematic bicycle model is derived based on the assumption that the front and the rear wheels of the vehicle are rolling without slipping.

Let the configuration of the vehicle be described by $q = (x_1, y_1, \theta_1, \alpha)$, where (x_1, y_1) is the position of the center of the vehicle's rear axle and θ_1 is its orientation. Under no-slip assumptions, the component of the velocity vector that is orthogonal to the rear wheels is zero which leads to the following linear velocity constraint:

$$\underbrace{\begin{bmatrix} -\sin\theta_1 & \cos\theta_1 & 0 & 0 \end{bmatrix}}_{\triangleq \omega_1(q)} \dot{q} = 0. \tag{2.6}$$

Repeating this argument for the front wheels implies that the following nonholonomic constraint also has to be satisfied

$$-\dot{x}_f \sin(\theta_1 + \alpha) + \dot{y}_f \cos(\theta_1 + \alpha) = 0. \tag{2.7}$$

The position for the center of the front axle (x_f, y_f) can be expressed in the configuration q as

$$x_f = x_1 + L_1 \cos\theta_1, \tag{2.8a}$$
$$y_f = y_1 + L_1 \sin\theta_1. \tag{2.8b}$$

By differentiating (2.8a)–(2.8b) with respect to time and inserting the result in (2.7), the second Pfaffian constraint becomes

$$\underbrace{\begin{bmatrix} -\sin(\theta_1+\alpha) & \cos(\theta_1+\alpha) & L_1\cos\alpha & 0 \end{bmatrix}}_{\triangleq \omega_2(q)} \dot{q} = 0. \tag{2.9}$$

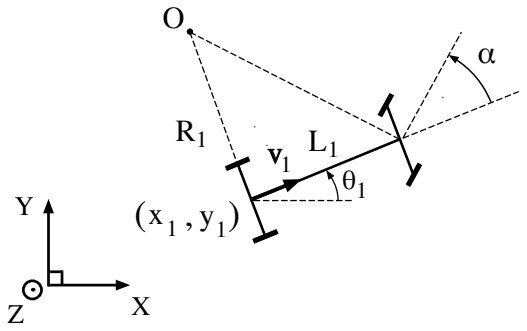


Figure 2.1: *Illustration of the kinematic bicycle model moving on a flat surface.*

Provided that the steering angle satisfies $|\alpha| < \pi/2$, the row vectors $\omega_1(q)$ and $\omega_2(q)$ are linearly independent and a basis of the right null space to $\omega(q) = [\omega_1(q)^T, \omega_2(q)^T]^T$ is

$$g_1 = \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \\ \frac{\tan\alpha}{L_1} \\ 0 \end{bmatrix}, \quad g_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.10}$$

Thus, the kinematic bicycle model can be written as a control-affine driftless system (2.4) with $m = 2$, and the control signals are the longitudinal velocity $u_1 = v_1$ and the steering angle rate $u_2 = \dot{\alpha}$.

In trajectory and path-following control applications during low-speed maneuvers, the dynamics in the steering angle is sometimes neglected and α is assumed to be directly controlled (Paden et al., 2016). For this case, define the state vector as $x = (x_1, y_1, \theta_1)$, the control signals as $u = (v_1, \alpha)$ and the resulting simplified version of the kinematic bicycle model is

$$\dot{x} = v_1 \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \\ \frac{\tan\alpha}{L_1} \end{bmatrix}. \tag{2.11}$$

Here, the steering angle α can be substituted with the truck's curvature κ which is defined as

$$\kappa = \frac{d\theta_1}{ds_x} = \frac{\tan\alpha}{L_1}, \tag{2.12}$$

which is inversely proportional to the turning radius R_1 of the vehicle, *i.e.*, $\kappa = 1/R_1$.

The Dubins version of the bicycle model (2.11) is obtained by constraining $v_1 \in \{0, 1\}$ and $\alpha = \{-\alpha_{\max}, 0, \alpha_{\max}\}$, which means that the vehicle is only allowed to move straight, maximum left or maximum right at constant forward speed $v_1 = 1$ (Dubins, 1957). The Reeds-Shepp version is obtained by allowing the vehicle to also travel in backward motion $v_1 \in \{-1, 0, 1\}$ (Reeds and Shepp, 1990).

Since v_1 enters bilinearly into the model in (2.11), a method known as time-scaling (Houska et al., 2011b) can be applied to eliminate the longitudinal speed $|v_1|$ from the model. Define $s_x(t)$ as the distance traveled by the rear axle of the vehicle, *i.e.*, $s_x(t) = \int_0^t |v_1(\tau)| d\tau$, then $\dot{s}_x = |v_1|$ and by applying the chain-rule, the model in (2.11) can be rewritten on spatial form as

$$\frac{dx}{ds_x} = \operatorname{sign}(v_1) \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \\ \kappa \end{bmatrix}. \tag{2.13}$$

This result is commonly used as motivation for decoupling the motion planning problem into path planning and velocity planning when low-speed maneuvers are considered. It is also a strong motivation for decoupling the vehicle controller into lateral and longitudinal control.

2.4 The general n-trailer

A system that is of special interest in this thesis is the so called general n-trailer (Altafini, 2001). This system consists of a truck that is connected to n passive trailers. The word "general" refers to that the connections are not assumed to be located at the longitudinal center of the proceeding vehicle's rear axle, see Figure 2.2. The control signals to this system are the longitudinal velocity v_1 for the rear axle of the truck and its steering angle α (or its curvature κ). Similar to the kinematic bicycle model, the model of the general n-trailer is also derived based on holonomic and nonholonomic constraints and a recursive formula is presented in Altafini (2001).

A schematic description of the system is provided in Figure 2.2. The truck is modeled as a kinematic bicycle model (2.11) and thus, its yaw-rate is $\dot{\theta}_1 = v_1\kappa$. Let (θ_i, v_i) denote the orientation and the longitudinal velocity of the $i-1$-th trailer. Define M_i as the distance from the rear axle of trailer $i-1$ to its off-axle hitch connection at trailer i. Let L_{i+1} denote the length of the i-th trailer. Assume that the angular rate $\dot{\theta}_i$ and longitudinal velocity v_i for trailer $i-1$ are known, then the recursive formulas for obtaining $\dot{\theta}_{i+1}$ and v_{i+1} for the i-th trailer are (Altafini, 2001):

$$\dot{\theta}_{i+1} = v_i \frac{\sin(\theta_i - \theta_{i+1})}{L_{i+1}} - \frac{M_i \cos(\theta_i - \theta_{i+1})}{L_{i+1}} \dot{\theta}_i, \tag{2.14a}$$

$$v_{i+1} = v_i \cos(\theta_i - \theta_{i+1}) + M_i \sin(\theta_i - \theta_{i+1}) \dot{\theta}_i. \tag{2.14b}$$

For details regarding the derivation, the reader is referred to Altafini (2001) or Ljungqvist (2015). Moreover, define $\beta_{i+1} = \theta_i - \theta_{i+1}$ as the relative angle between the $(i-1)$-th and the i-th trailer. Then, the relative angular rate is

$$\dot{\beta}_{i+1} = \dot{\theta}_i - \dot{\theta}_{i+1}. \tag{2.15}$$

By using the model in (2.11) for the leading truck, the model of the general n-trailer can now be derived by iteratively applying the recursive formulas in (2.14) starting from v_1 and $\dot{\theta}_1$. The state vector x can be composed of the relative angles $\beta_2, \beta_3, \ldots, \beta_{n+1}$, the

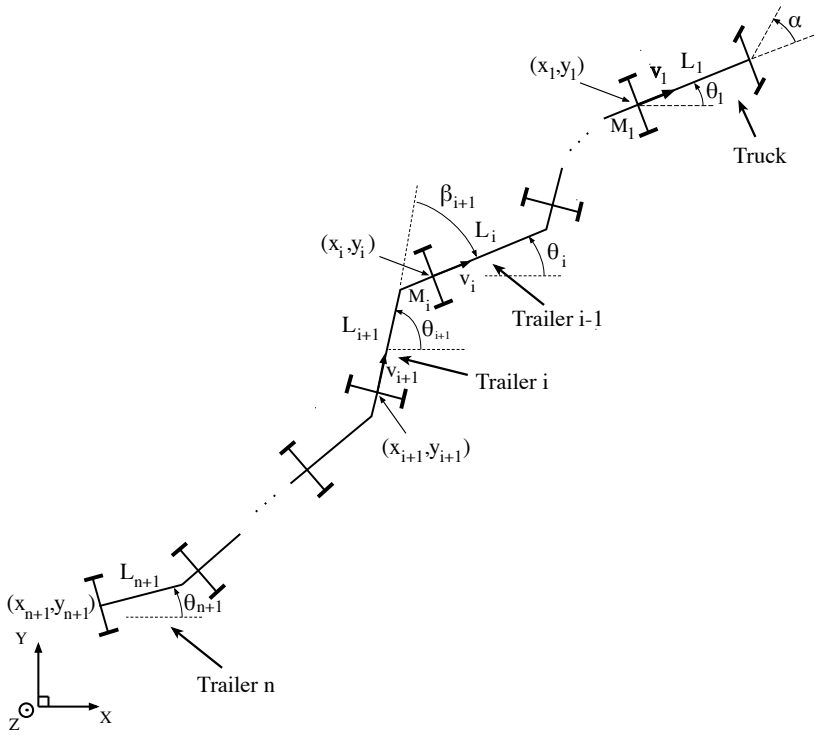


Figure 2.2: *A schematic description of the general n-trailer. The system consists of a truck that is connected to n passive trailers with off-axle hitch connections.*

position for the rear axle of the n-th trailer[1] (x_{n+1}, y_{n+1}) and its orientation θ_{n+1}, where (x_{n+1}, y_{n+1}) evolves as

$$\dot{x}_{n+1} = v_{n+1} \cos \theta_{n+1}, \tag{2.16a}$$

$$\dot{y}_{n+1} = v_{n+1} \sin \theta_{n+1}. \tag{2.16b}$$

Denote the state vector as $x = (x_{n+1}, y_{n+1}, \theta_{n+1}, \beta_2, \beta_3, \ldots, \beta_{n+1})$, the model of the general n-trailer can then be written in a compact form as

$$\dot{x} = v_1 f(x, \kappa), \tag{2.17}$$

where the control signals are the longitudinal velocity for the rear axle of the truck v_1 and its curvature κ. As for the kinematic bicycle model (2.11), time-scaling (Houska et al., 2011b) can be applied to eliminate the longitudinal speed $|v_1|$ dependence from the model.

For the special case with only on-axle hitching, *i.e.*, all $M_i = 0$, the model of the general n-trailer coincides with the standard n-trailer (Sørdalen, 1993). This system is

[1] In forward motion, it may be more natural to use the truck's absolute position and orientation instead of the n-th trailer's.

differentially flat (Rouchon et al., 1993) and can be converted into chained-form where the position of the rear axle of the n-th trailer (x_n, y_n) is used as the flat output (Sørdalen, 1993). The general 1-trailer is also differentially flat using a certain choice of flat output. However, when $n \geq 2$, the flatness property does not hold for the general n-trailer case (Rouchon et al., 1993).

The model of the general n-trailer is small-time locally controllable (Khalil and Grizzle, 1996), except from some singularities that are discussed in Altafini (2001). These singularities arise when the vehicle is arranged such that the truck can move while some of the trailers remain static. This chapter is concluded with the derivation of the model of a general 2-trailer that will be extensively used in Part II of this thesis.

A truck with a dolly-steered trailer

A truck with a dolly-steered trailer is a variant of the general 2-trailer with off-axle hitching at the truck, as illustrated in Figure 2.3. The system has an off-axle hitching ($M_1 \neq 0$) between the truck and the dolly and an on-axle hitching ($M_2 = 0$) between the dolly and the trailer. The truck is modeled as a kinematic bicycle model (2.11) and thus $\dot{\theta}_1 = v_1 \kappa$, where v_1 and κ are control signals. By applying the recursive formula in (2.14), we get

$$\dot{\theta}_2 = \frac{v_1 \sin\beta_2}{L_2} - \frac{M_1 \cos\beta_2}{L_2}\dot{\theta}_1 = v_1 \left(\frac{\sin\beta_2}{L_2} - \frac{M_1}{L_2}\cos\beta_2 \kappa \right), \tag{2.18a}$$

$$v_2 = v_1 \cos\beta_2 + M_1 \sin\beta_2 \dot{\theta}_1 = v_1 \cos\beta_2 \left(1 + M_1 \tan\beta_2 \kappa\right). \tag{2.18b}$$

By using the same recursive formula again with $M_2 = 0$, we obtain

$$\dot{\theta}_3 = \frac{v_2 \sin\beta_3}{L_3} = \frac{v_1 \sin\beta_3 \cos\beta_2}{L_3}\left(1 + M_1 \tan\beta_2 \kappa\right), \tag{2.19a}$$

$$v_3 = v_2 \cos\beta_3 = v_1 \cos\beta_3 \cos\beta_2 \left(1 + M_1 \tan\beta_2 \kappa\right). \tag{2.19b}$$

The dynamics for the position (x_3, y_3) of the rear axle of the last trailer is given by (2.16) where v_3 can be expressed in v_1 using (2.19b). By defining the state vector as $x = (x_3, y_3, \theta_3, \beta_3, \beta_2)$ and with $\dot{\beta}_2 = \dot{\theta}_1 - \dot{\theta}_2$ and $\dot{\beta}_3 = \dot{\theta}_2 - \dot{\theta}_3$, the model of this general 2-trailer system is

$$\dot{x}_3 = v_1 \cos\beta_3 \cos\beta_2 \left(1 + M_1 \tan\beta_2 \kappa\right) \cos\theta_3, \tag{2.20a}$$

$$\dot{y}_3 = v_1 \cos\beta_3 \cos\beta_2 \left(1 + M_1 \tan\beta_2 \kappa\right) \sin\theta_3, \tag{2.20b}$$

$$\dot{\theta}_3 = v_1 \frac{\sin\beta_3 \cos\beta_2}{L_3}\left(1 + M_1 \tan\beta_2 \kappa\right), \tag{2.20c}$$

$$\dot{\beta}_3 = v_1 \cos\beta_2 \left(\frac{1}{L_2}\left(\tan\beta_2 - M_1 \kappa\right) - \frac{\sin\beta_3}{L_3}\left(1 + M_1 \tan\beta_2 \kappa\right) \right), \tag{2.20d}$$

$$\dot{\beta}_2 = v_1 \left(\kappa - \frac{\sin\beta_2}{L_2} + \frac{M_1}{L_2}\cos\beta_2 \kappa \right). \tag{2.20e}$$

As discussed earlier, the system has singularities where it is no longer controllable (Altafini, 2001). To avoid them, the relative angles are bounded as $|\beta_i| < \pi/2$, $i = 1, 2$ and due to the off-axle hitching $M_1 \neq 0$, the following inequality

$$1 + M_1 \tan\beta_2 \kappa > 0, \tag{2.21}$$

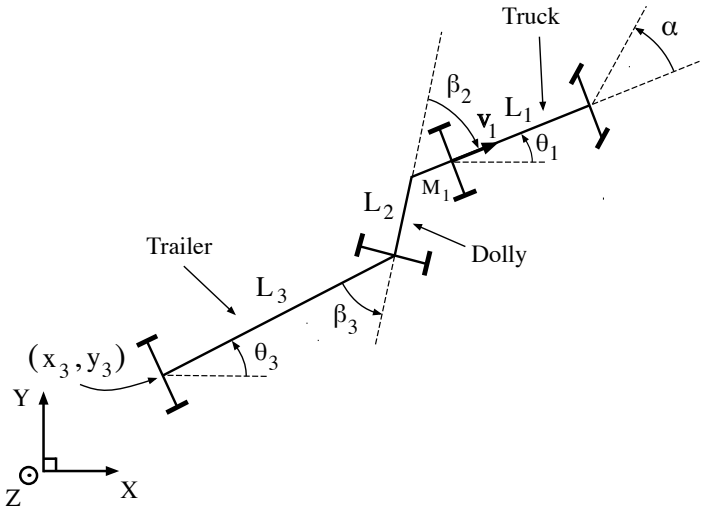
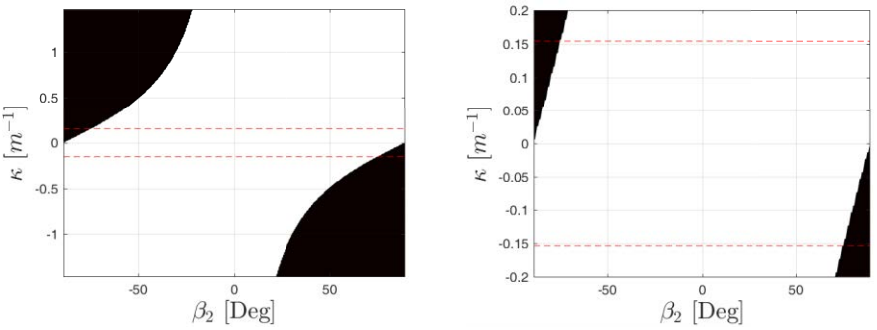


Figure 2.3: *A schematic illustration of a truck with a dolly-steered trailer.*

must hold. Together, these inequalities represent the domain of definition for the general 2-trailer in (2.20). From practical experience, these conditions are most often satisfied and a positive off-hitch $M_1 > 0$ actually enhance the maneuverability of the system.

To motivate this statement, we consider the full-scale test vehicle in Paper E. The wheelbase of the truck is $L_1 = 4.6$ m and the vehicle has a positive off-axle hitching $M_1 = 1.7$ m at the truck. The domain of definition for this system is the white area illustrated in Figure 2.4a. The steering angle α of the truck is limited to approximately $\alpha_{\max} = 45°$, which implies a limited curvature $|\kappa| \leq \kappa_{\max} = |\tan \alpha_{\max}|/L_1$ where $\kappa_{\max} = 0.15$ m^{-1}. These bounds are illustrated in the figure by red-dotted lines. A zoomed in version are illustrated in Figure 2.4b. Thus, the system is controllable if the relative an-



(a) *Domain of definition (white area).* ***(b)*** *A zoomed in version of Figure 2.4a.*

Figure 2.4: *The domain of definition for a general 2-trailer systems. The white area represent the region where the systems are controllable and the red-dotted lines represent the truck's curvature limitation.*

gle between the truck and the dolly β_2 satisfies $|\beta_2| \leq 70^\circ$. Such a large relative angle is rarely encountered in practice unless the vehicle has entered a *jack-knife state* while driving backwards. In that case, the vehicle must recover by driving forwards. However, if a larger range of steering angles is possible to use, or if the ratio M_1/L_1 increases, the constraint in (2.21) becomes more restrictive and therefore has to be taken into account during both motion planning and feedback control design.

3

Motion planning for self-driving vehicles

This chapter is dedicated to present motion planning techniques for systems subject to nonholonomic constraints. This chapter starts with a short introduction to the motion planning concept. In Section 3.2, the motion planning problem is presented as an optimal control problem (OCP) and reasons why this OCP is hard to solve by directly applying numerical optimal control methods is discussed. In Section 3.3, the general steps for solving a motion planning problem are explained and in Section 3.4, the basic tools for nonholonomic motion planning are presented. Section 3.5 starts by defining a general search-based motion planning algorithm and then, popular deterministic and probabilistic versions are presented.

3.1 Introduction

The concept of motion planning is and has been, an active research topic in different research fields for many decades and depending on the field, the meaning of planning differs. In the field of artificial intelligence, planning historically refers to finding a sequence of logical decisions or actions that transforms an initial world state to a desired goal state where vehicle dynamics is often neglected or very simplified (Russell and Norvig, 2016). In this thesis, this planning domain will be referred to as *mission planning* or *task planning*. In a self-driving vehicle, the mission planner acts as a high-level planner that, *e.g.*, specifies a mission to be executed by the low-level motion planning and control layer.

Here, motion planning is referred to as the problem of computing a trajectory or a path that satisfies the system dynamics, does not violate the vehicle's physically imposed constraints, does not collide with any obstacle and at the same time minimizes a specified performance measure.

From a control perspective, motion planning is an optimal control problem that in general is hard solve due to its non-convex nature. In self-driving vehicle applications, it is also required that a motion plan is computed, and possibly updated, within seconds

or even milliseconds. To fulfill these requirements, numerous of different motion planning algorithms have been proposed (Kuwata et al., 2009; LaValle, 1998; Paden et al., 2016; Pivtoraiko et al., 2009). Many of these algorithms originate from the robotics and computer science communities which has been adapted to also take system dynamics into account (LaValle, 2006).

This chapter focuses on motion planning for self-driving vehicles in *unstructured environments*, *e.g.*, parking lots or other open areas where rules are less specified and the vehicle's speed is low. This problem is fundamentally different from motion planning in *structured environments*, *e.g.*, high-way driving, where information of the road center line is used to enforce structure into the problem. This leads to a problem formulation that has strong similarities to *receding horizon control* (Lima et al., 2017b; Paden et al., 2016; Werling et al., 2012).

3.2 Problem formulation

The vehicle we are intended to plan a feasible motion for is assumed to be modeled as a time-invariant nonlinear system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t_I) = x_I, \tag{3.1}$$

where $x(t) \in \mathbb{R}^n$ denotes the vehicle states and $u(t) \in \mathbb{R}^m$ its control signals. The vehicle is assumed to have physical limitations on its states $x(t) \in \mathbb{X}$ and controls $u(t) \in \mathbb{U}$, where $\mathbb{X}$ and $\mathbb{U}$ are typically convex sets. The vehicle is operating in a world where static and possibly also moving obstacles are present. The regions which are occupied with obstacles $\mathbb{X}_{\text{obs}}(t)$ are usually divided into static obstacles $\mathbb{X}^s_{\text{obs}}$ and dynamic obstacles $\mathbb{X}^d_{\text{obs}}(t)$. The free-space where the vehicle is not in collision with any obstacle at time t is defined as $\mathbb{X}_{\text{free}}(t) = \mathbb{X} \setminus \mathbb{X}_{\text{obs}}(t)$. For now, we make the assumption that the set $\mathbb{X}_{\text{free}}(t)$ can be described analytically. Since the free-space $\mathbb{X}_{\text{free}}(t)$ is defined as the complement set of $\mathbb{X}_{\text{obs}}(t)$, it is in general a non-convex set. The objective function J to be minimized is often composed of different costs, *e.g.*, time duration and/or control energy.

The motion planning problem is defined as follows: Compute a feasible and collision-free motion plan, *i.e.*, a trajectory $(x_0(t), u_0(t))$, $t \in [t_I, t_G]$ that moves the vehicle from its initial state x_I to a desired goal state x_G, while minimizing the cost function J. This problem can be posed as an OCP in the following form

$$\begin{aligned} \underset{u_0(\cdot),\, t_G}{\text{minimize}} \quad & J = \int_{t_I}^{t_G} L(t, x_0(t), u_0(t))\, dt \\ \text{subject to} \quad & \dot{x}_0(t) = f(x_0(t), u_0(t)), \\ & x_0(t_I) = x_I, \quad x_0(t_G) = x_G, \\ & x_0(t) \in \mathbb{X}_{\text{free}}(t), \quad u_0(t) \in \mathbb{U}, \end{aligned} \tag{3.2}$$

where L is called the *Lagrange term* and specifies the performance measure to be minimized. It is assumed that $x_I \in \mathbb{X}_{\text{free}}(t)$ and $x_G \in \mathbb{X}_{\text{free}}(t)$, $\forall t \geq t_I$ to make the OCP in (3.2) well defined and that it always exists a terminal control signal u_T such that the vehicle remains at the goal $\forall t \geq t_G$ (LaValle, 2006). This assumption is equivalent to that the

vehicle ends up in an equilibrium point at the goal, *i.e.*, $f(x_0(t), u_T) = 0, \forall t \geq t_G$. This assumption implies that the motion plan can be extended for an arbitrary time duration. The terminal control signal is easy to construct for kinematic vehicle's by simply choosing the longitudinal velocity $v_{1,T} = 0$ which implies $f(x_0(t), u_T) = 0$. In some applications, the goal state x_G is replaced with a goal region $\mathbb{X}_G$ because the motion planner will not be able to reach the goal exactly (LaValle, 2006). The OCP in (3.2) is in general hard solve by directly invoking a state-of-the-art numerical optimal control solver (Houska et al., 2011b; Wächter and Biegler, 2006). Three fundamental reasons for this are:

1. Even if $\mathbb{X}_{\text{obs}}(t) = \emptyset$, the OCP in (3.2) is a non-convex optimization problem because of the nonlinear system dynamics. Thus, for complex systems, a proper warm-start is needed to even find a feasible solution.

2. $\mathbb{X}_{\text{free}}(t)$ is often constructed by fusing information from precomputed maps and onboard sensors on the vehicle and an analytical expression for the obstacle region $\mathbb{X}_{\text{obs}}(t)$ is therefore difficult to construct in practice.

3. $\mathbb{X}_{\text{free}}(t)$ is often non-convex and the OCP in (3.2) is therefore a combinatorial problem with possibly different classes of solution trajectories. A proper initial guess is therefore required to find a good locally optimal (or even a feasible) solution to the OCP in (3.2) (Zhang et al., 2017).

Besides the above mentioned problems, real-time performance and the absence of real-time certification for nonlinear solvers are also strong reasons why standalone numerical optimal control is rarely used in practice. However, numerical optimal control still plays an important role in many motion planning algorithms.

For many nonholonomic systems, the trajectory OCP in (3.2) can be converted into a path OCP when dynamic obstacles are neglected. In this thesis, the relationship between a path and a trajectory is defined as follows.

Definition 3.1. A path is represented as $(x_0(s)), u_0(s))$, $s \in [0, S_G]$ and a trajectory is a time-parametrized path $(x_0(s(t))), u_0(s(t)))$, $t \in [t_I, t_G]$.

Here, it is assumed that $s(t)$ is either monotonically increasing or decreasing in time. Thus, for wheeled vehicles, a path is split into separate segments depending on the direction of motion. Based on Definition 3.1, a trajectory can be constructed from a path by specifying the velocity $\dot{s}(t) = v_0(t)$ of which the path should be executed. For path planning of ground vehicles, the direction of motion is already determined by the path planner and what remains for the *velocity planner* is to compute the desired longitudinal speed $|v_0(t)|$. For systems that are differentially flat, a path can also be defined as a curve in the *flat outputs* (Paden et al., 2016). In that case, smoothness constraints have to be enforced on the curve such that the nominal states and control signals can be recovered (Luca et al., 1998).

3.3 A general solution concept

To solve, or rather to compute a suboptimal solution to the OCP in (3.2), motion planning algorithms are commonly deployed. A planning cycle for an autonomous vehicle is often

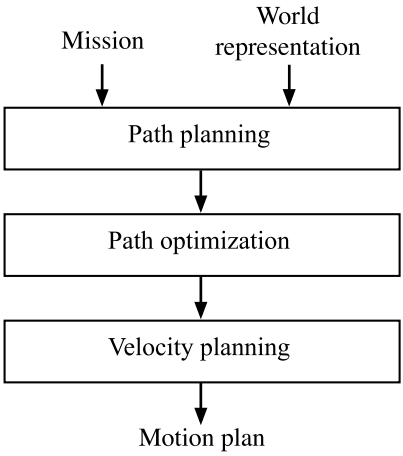


Figure 3.1: *The typical steps that are performed to compute a motion plan.*

divided into a couple of different phases, as illustrated in Figure 3.1 (LaValle, 2006). The perception and localization layer provides the motion planner with a compressed representation of the surrounding environment and information of the vehicle's current state. The high-level task planner feeds the low-level motion planning and control layer with a desired mission to be executed. Based on these inputs, a motion plan is computed through a series of steps. First, a collision-free path from the vehicle's current state to a desired goal state is computed. In this planning phase, the system dynamics is sometimes neglected and the computed path is then not kinematically feasible. The path can for example be composed of piecewise linear paths if a geometric planner is used during this step (LaValle, 1998, 2006; Likhachev et al., 2004). In the second step, the path is optimized, or smoothen, to generate a locally optimal motion plan that satisfies the system dynamics and its physically imposed constraints. Finally, a velocity profile is computed to obtain a trajectory. The velocity profile is computed such that the vehicle does not collide with any dynamic obstacle and such that the vehicle's maximally allowed velocities and accelerations, and other comfort and safety constraints are satisfied.

The computed motion plan $(x_0(\cdot), u_0(\cdot))$ is then sent to the vehicle controller for plan execution. The planning cycle is then repeated in case a new mission is received or if the current motion plan has become infeasible due to, *e.g.*, inaccurate prediction of dynamic obstacles or if a previously unseen obstacle is detected by the perception layer.

Ideally, a unified motion planning algorithm that performs all three above mentioned steps in one shot is preferred since separating the problem often leads to a suboptimal solution. This is however often intractable for most problem classes because of limited computation time. However, there exist efficient motion planning techniques where the vehicle's nonholonomic and physically imposed constraints are taken into account already when constructing the path and some popular techniques will be presented in this chapter.

3.4 Planning under differential constraints

Motion planning for nonholonomic systems is conceptually not different compared to motion planning for a holonomic system, *e.g.*, a quadcopter platform (Andersson et al.,

2018b). The motion planning techniques that are presented here are therefore also applicable for holonomic systems, as long as the dimension of the state-space is sufficiently small and the algorithmic specific assumptions for the system dynamics are satisfied. However, the velocity constraints that are inherited by a nonholonomic system make different phases of the algorithms more demanding compared to the holonomic case.

For unstructured environments, *e.g.*, parking lots or offloading/loading sites, a vast amount of different motion planning algorithms for ground vehicles exist and an overview can for example be found in Paden et al. (2016). Two popular branches for nonholonomic motion planning include *probabilistic methods* (Karaman and Frazzoli, 2011; Kuwata et al., 2009; LaValle and Kuffner, 2001) and *deterministic methods* (Cirillo et al., 2014; Pivtoraiko et al., 2009; Urmson et al., 2008) with different guarantees on the quality of the produced solution. Before presenting the planning algorithms, some important building blocks for these algorithms are presented.

3.4.1 Steering functions

Many motion planning algorithms require a method to generate a feasible trajectory between two nearby vehicle states (Karaman and Frazzoli, 2011; Pivtoraiko et al., 2009). Here, the surrounding obstacles are neglected and only the system dynamics $\dot{x} = f(x,u)$ and its physically imposed constraints $x \in \mathbb{X}$ and $u \in \mathbb{U}$ are considered. For some algorithms, this local trajectory generation problem needs to be solved online (Karaman and Frazzoli, 2011) and for others it is a procedure that is performed offline (Pivtoraiko et al., 2009). In the latter case, *position invariance* of the system is often exploited such that a precomputed trajectory can be translated and reused from all vehicle positions.

For the linear system case without physically imposed constraints, *Pontryagin's minimum principle* (PMP) (Pontryagin, 2018) can be used to derive analytical solutions (Werling et al., 2012). PMP can also be used to derive analytical solutions for some nonlinear systems, including the kinematic bicycle model (2.11). Two examples are the Dubins (Dubins, 1957) and the Reeds-Shepp (Reeds and Shepp, 1990) minimal path lengths which have been extended to continuous and continuously differentiable vehicle curvature in (Fraichard and Scheuer, 2004) and (Oliveira et al., 2018), respectively.

For systems that are differentially flat, efficient techniques for local trajectory generation can be used (Laumond et al., 1994; Murray and Sastry, 1991; Švestka and Vleugels, 1995; Tilbury et al., 1995). However, in all above examples, a general performance measure J_{BVP} is hard to optimize for and physically imposed constraints on the vehicle need to be checked afterwards for feasibility. In general, the problem of connecting two nearby vehicle states $x_i \in \mathbb{X}$ and $x_f \in \mathbb{X}$ without considering obstacles can be posed as an OCP in the following form

$$
\begin{aligned}
\underset{u_0(\cdot),\, t_f}{\text{minimize}} \quad & J_{\mathrm{BVP}} = \int_0^{t_f} L_{\mathrm{BVP}}(x_0(t), u_0(t))\, dt \\
\text{subject to} \quad & \dot{x}_0(t) = f(x_0(t), u_0(t)), \\
& x_0(0) = x_i, \quad x_0(t_f) = x_f, \\
& x_0(t) \in \mathbb{X}, \quad u_0(t) \in \mathbb{U}.
\end{aligned}
\tag{3.3}
$$

This OCP is sometimes referred to as a *boundary value problem* (BVP) and note its similarities to the formulation of the motion planning problem in (3.2). Here, the set of allowed vehicle states $\mathbb{X}$ is typically represented as a convex set, in contrast to (3.2) where $\mathbb{X}_{\text{free}}(t)$ is often non-convex. This implies that (3.3) is easier to find a locally optimal solution to. However, the OCP in (3.3) is still a non-convex problem due to the nonlinear system dynamics. In many applications, it is important that $u_0(t)$ is sufficiently smooth which is straightforward to handle by introducing an augmented state vector $\bar{x}_0(t) = (x_0(t), u_0(t), \dot{u}_0(t) \ldots, u_0^{(r-1)}(t))$ and view the r-th time derivative $u_0^{(r)}(t)$ as an artificial control signal. It is now possible to penalize the derivatives of the nominal control signal and enforce limitations on them in a structured way. The boundary values for $(u_0(t), \dot{u}_0(t) \ldots, u_0^{(r-1)}(t))$ are typically zero in order to guarantee that $u_0(t)$ is an $r-1$ times continuously differentiable function in time, even when multiple trajectories are combined during online planning (Ljungqvist et al., 2017).

The solution to (3.3) is a trajectory $(x_0(t), u_0(t))$, $t \in [0, t_f]$ that will be referred to as a *motion primitive* m_p and the solution satisfies

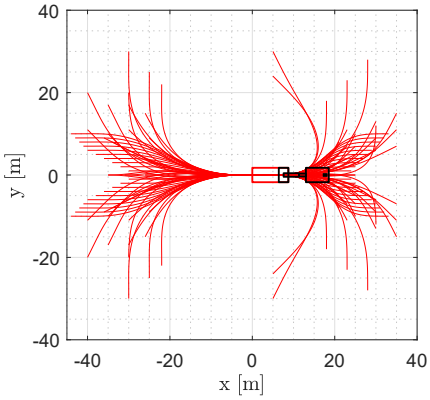
$$x_0(t_f) = x_0(0) + \int_0^{t_f} f(x_0(t), u_0(t)) dt. \tag{3.4}$$

Denote $x_k = x_0(0)$ and $x_{k+1} = x_0(t_f)$. For notional convenience, we define the *state transition equation* in (3.4) as

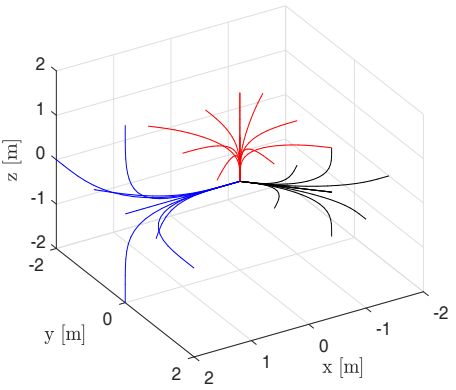
$$x_{k+1} = f_p(x_k, m_p). \tag{3.5}$$

The final vehicle state x_{k+1} when applying m_p from x_k is called a *successor state* of x_k. The Lagrange term L_{BVP} in (3.3) may be the same as the Lagrange term L in the motion planning problem in (3.2), or some of the terms are neglected in L_{BVP}, *e.g.*, only smoothness and comfort criteria are considered (Ljungqvist et al., 2017; Werling et al., 2012). It is not rare that completely different performance measures are used for the lower-level motion primitive generation (3.3) compared to the higher-level motion planning problem (3.2). In any case, the *stage cost* $J(m_p)$ is computed by integrating the resulting trajectory using the Lagrange term defined in the motion planning problem (3.2).

One approach to find a locally optimal solution to the OCP in (3.3) is to use state-of-the-art numerical optimal control software, *e.g.*, ACADO (Houska et al., 2011b) or Casadi (Andersson et al., 2018a). In these methods, the continuous-time optimal control problem is reformulated into a nonlinear programming (NLP) problem using for example multiple shooting combined with numerical integration. To solve the resulting NLP, a sequential quadratic programming (SQP) solver (Gill et al., 2005) or a nonlinear interior-point (IP) solver (Wächter and Biegler, 2006) can be used. Examples of different solutions for a truck with dolly-steered trailer (2.20) and for a quadcopter platform (Andersson et al., 2018b) are provided in Figure 3.2. For systems with complex dynamics, it can be problematic to even find a feasible solution to the BVP in (3.3) and a proper initial guess is crucial. Moreover, systems with unstable dynamics can cause numerical problems while solving the OCP in (3.3). One such example is the truck with a dolly-steered trailer (2.20) which is unstable while driving backwards. In Paper E, we present a method to avoid this issue for the truck and trailer case by establishing and exploiting symmetry properties of the system.



(a) *Precomputed motions for a truck with a dolly-steered trailer from where the vehicle is standing straight to different final states.*



(b) *Precomputed motions for a quadcopter platform from different initial velocities* $v_i = (v_{x,i}, v_{y,i}, v_{z,i})$ *to different final states.*

Figure 3.2: *Precomputed motions for a truck with a dolly-steered trailer (a) and for a quadcopter platform (b). Only the trajectories for the position of the rear axle of the trailer and the center of gravity for the quadcopter are plotted, respectively.*

3.4.2 Heuristics

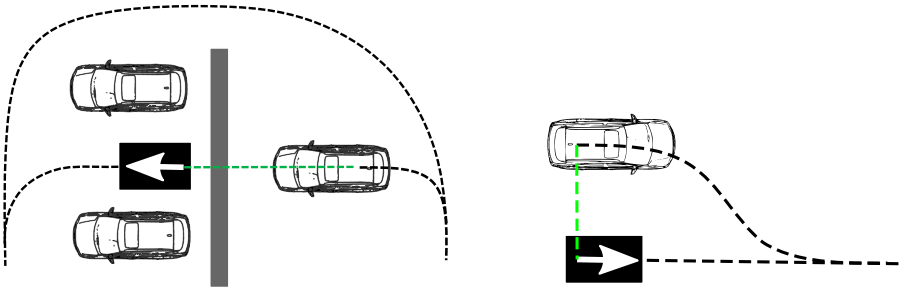
All sampling-based motion planning algorithms incrementally explores a sampled set $\mathbb{X}_s$ of the vehicle's state-space $\mathbb{X}$. This exploration is either performed in a deterministic or in a probabilistic fashion with the common goal of computing a motion plan to the desired goal. During this search process, an informative *heuristic function* h is needed for guidance (LaValle and Kuffner, 2001; Pivtoraiko et al., 2009). The heuristic function $h(x_k, x_G)$ is used to estimate the cost-to-go from a vehicle state $x_k \in \mathbb{X}_s$ to the goal state $x_G \in \mathbb{X}_s$. Let $h^*(x_k, x_G)$ denote the optimal cost-to-go for traveling from x_k to x_G and let a successor state $x_{k+1} \in \mathbb{X}_s$ represent a possible state transition from x_k with motion primitive m_p and stage cost $J(m_p)$. A heuristic function $h(x_k, x_G)$ is said to be

Admissible if for all $x_k \in \mathbb{X}_s$, the heuristic function satisfies $h(x_k, x_G) \leq h^*(x_k, x_G)$

Consistent if for all $x_k \in \mathbb{X}_s$, and all its successor states x_{k+1}, the heuristic function satisfies $h(x_k, x_G) \leq J(m_p) + h(x_{k+1}, x_G)$

The admissibility criterion implies that the heuristic cost always underestimates the optimal cost-to-go from x_k to x_G and the consistency criterion means that the heuristic cost from x_k to x_G cannot be larger than the stage cost $J(m_p)$ plus the heuristic cost from x_{k+1} to x_G, *i.e.*, the triangle inequality must hold. These criteria are necessary to satisfy if optimality guarantees for motion planning algorithms should be maintained.

When path planning is considered, the simplest heuristic function is the Euclidean distance. However, due to the nonholonomic constraints that are inherited by a nonholonomic system and the surrounding obstacles, the Euclidean distance may severely underestimate the true cost-to-go in many scenarios. Two illustrative examples for a passenger car are presented in Figure 3.3. In Figure 3.3a, the surrounding environment damage the



(a) The surrounding obstacles are not taken into account.

(b) The nonholonomic constraints are not taken into account.

Figure 3.3: *Illustration of two cases when the Euclidean distance (green dashed line) severely underestimate the true cost-to-go (black dashed line).*

estimated heuristic cost and in Figure 3.3b, the nonholonomic constraints make the Euclidean distance a poor estimate of the cost-to-go. A bad heuristic function may cause the motion planning algorithm to spend unnecessary computation time on exploring vehicle states that seem promising according to the heuristics, but are in reality bad candidates since the true cost-to-go from these states may be very expensive.

A common approach when designing heuristic functions is to relax the motion planning problem by removing certain constraints to make the problem simpler to solve. For the passenger car case, the problem in Figure 3.3b could be handled by computing the analytical solution for the Reeds-Shepp car (Reeds and Shepp, 1990) and use its optimal path-length as heuristic function. For the case in Figure 3.3a, the system dynamics could be removed and a geometric planning problem could be solved to compute an underestimate of cost-to-go. These two strategies are combined in Dolgov et al. (2010) where first a local trajectory generation problem without obstacles is solved, then the system dynamics is neglected and a geometric planning problem with obstacles is solved and the maximum of these estimates is used as heuristic cost. At a first glance, this proposal may seem computationally inefficient for a general nonholonomic system since a local trajectory generation problem without obstacles is a BVP (3.3) that has to be solved for each state expansion. Luckily, this problem can in many applications be performed offline on a finite grid of vehicle states and stored in a *heuristic look-up table* (HLUT) (Cirillo et al., 2014; Knepper and Kelly, 2006). The HLUT can then be preloaded into smart data structures and used during online planning to efficiently retrieve the heuristic function values for many problems in the vicinity of the vehicle.

Depending on the complexity of the surrounding environment, the simplified geometric planning problem may yield a good or a poor estimate of cost-to-go. For the problem in Figure 3.3b, the Euclidean distance and the solution to the geometric planning problem will yield the same poor estimate of cost-to-go. For maze-like environments, as in Figure 3.3a, the geometric planning problem will return a far better estimate of cost-to-go and will here guide the more computationally intense nonholonomic planner to not expand from states that leads into dead ends. To conclude, the development of a good and computationally efficient heuristic function is application dependent and depends on the surrounding environment the vehicle is intended to operate in.

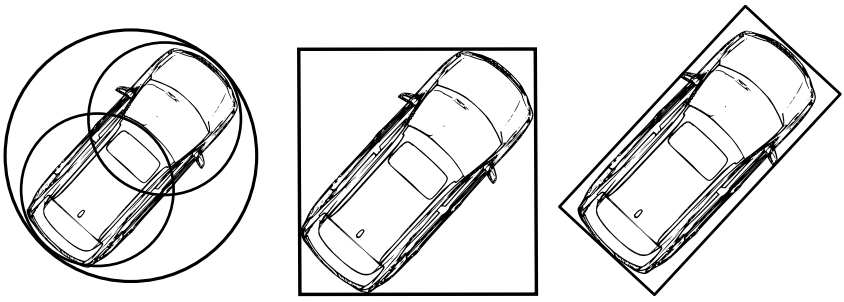


Figure 3.4: *Three different kinds of bounding regions: (a) circles, (b) axis-aligned bounding box, and (c) oriented bounding box.*

3.4.3 Collision detection

Efficient collision detection methods are crucial for any motion planning algorithm. The choice of method depends on how the surrounding obstacles $\mathbb{X}_{\text{obs}}(t)$ are represented and is invariant to the choice of motion planning algorithm. Typically, static obstacles $\mathbb{X}^s_{\text{obs}}$ and dynamic obstacles $\mathbb{X}^d_{\text{obs}}(t)$ are treated separately. Since the trajectory for a dynamic obstacle is unknown (unless the vehicles are communicating), its motion has to be predicted. This is usually done by a tracking filter in the perception layer where probabilistic state-space models are used for trajectory prediction of the dynamic obstacles. The details of these algorithms are out of the scope for this thesis and we refer to LaValle (2006) and the references therein for details.

Collision detection for static obstacles are typically performed using hierarchical methods where different bounding regions are used in each step. Figure 3.4 represents four different kinds of *bounding regions*: a bounding circle, multiple bounding circles, an axis-aligned bounding box (AABB) and an oriented bounding box (OBB). The complexity of performing collision checking for each type of bounding region is in increasing order where a circle is cheap and an oriented bounding box is costly. If the controlled vehicle is also represented as a bounding region (or a hierarchy of bounding regions), the criterion for collision is that the regions are intersecting. If the bounding circles are overlapping, AABB may be checked and finally OBB. As soon as one step reports no collision, the vehicle state is declared collision free. For multi-body vehicle's, separate bounding regions for each body are typically used (LaValle, 2006).

The collision detection procedure has to be performed thousands of times in a single planning cycle and thus precomputing certain steps significantly speeds up the planning process. A common way to perform 2D collision detection is to compute an occupancy grid map (Elfes, 1989) or an OctoMap for the 3D case (Hornung et al., 2013). The grid map is divided into cells where the obstacle's bounding regions are used to classify a cell as free or occupied[1]. Figure 3.5 illustrates how an occupancy gridmap is computed where a green cell means that a the cell is free and a blue cell means that the cell is occupied by an obstacle. The occupancy grid map can then be used to efficiently perform collision

[1] Probabilistic information can be incorporated in the cells instead of a binary representation.

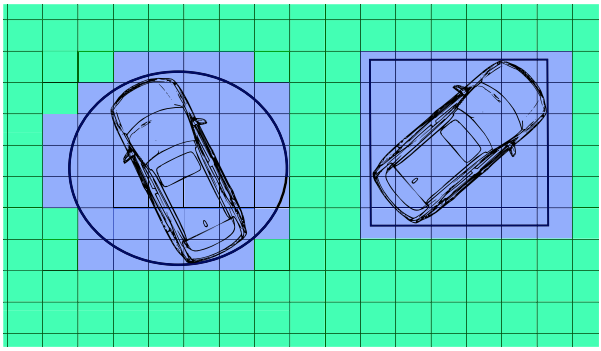
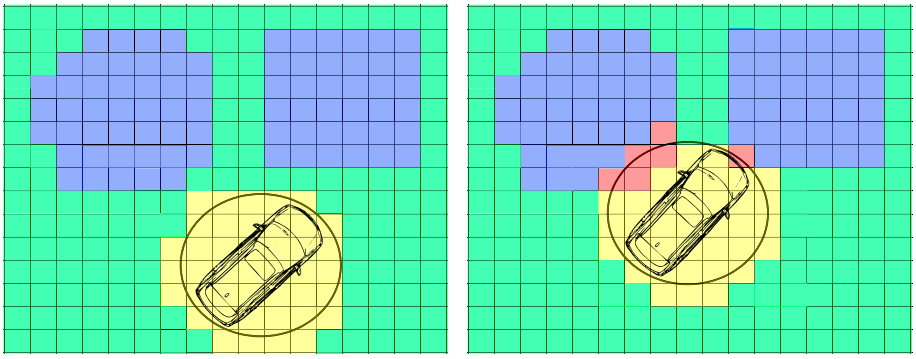


Figure 3.5: *Illustration of how obstacles with different bounding boxes are transformed to a binary occupancy grid map where blue means occupied and green means unoccupied.*

detection with the bounding region of the controlled vehicle, as illustrated in Figure 3.6. In this example, the vehicle state x in (a) is $x \in \mathbb{X}_{\text{free}}$ and in (b) $x \in \mathbb{X}_{\text{obs}}^{s}$ and hence, not in free space. Note that the collision detection procedure may be rather conservative if a single circle is used as bounding region. A motion primitive m_p is treated in analogy to a single state except that all states along the trajectory have to be collision free. In practice, only a finite set of states along the trajectory are used for collision detection. A motion primitive m_p applied from an initial state x_k is declared collision free if $c(x_k, m_p) \in \mathbb{X}_{\text{free}}$, otherwise it is declared as in collision.



(a) ***(b)***

Figure 3.6: *Collision detection using a binary occupancy grid map where a red cell means that the bounding circle of the controlled vehicle is intersecting an obstacle. In (a) the controlled vehicle is not in collision with any obstacle and in (b) the controlled vehicle is in collision with both obstacles.*

3.5 Search-based motion planning

We are now ready to present a general sampling-based motion planning framework for nonholonomic systems which is a direct generalization of geometric planning algorithms for which the system dynamics and kinematics are neglected (LaValle, 2006). For simplicity we assume only static obstacles in the reminder of this chapter, *i.e.*, $\mathbb{X}^d_{\text{obs}} = \emptyset$. A sampling-based motion planning algorithm is performing its search process in a directed[2] search graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where a vertex $x \in \mathcal{V}$ represents a vehicle state and an edge $m_p \in \mathcal{E}$ is a motion primitive that satisfies the system dynamics and its physically imposed constraints. This search graph may be incrementally constructed online or parts of the graph is precomputed offline. The general search process to compute a trajectory from the initial state $x_I \in \mathbb{X}_{\text{free}}$ to the goal state $x_G \in \mathbb{X}_{\text{free}}$ involves the following steps (LaValle, 2006):

1. **Initialization:** Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ represent a directed search graph, for which only $x_I \in \mathbb{X}_{\text{free}}$ is contained in the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$ is empty. The cost-to-come is $J(x_I, x_I) = 0$ and the estimated cost-to-go is $h(x_I, x_G)$.

2. **Choose expansion candidate:** Select a promising vehicle state $x_k \in \mathcal{V}$ by sorting $\mathcal{V}$ based on *e.g.* $J(x_k, x_I) + h(x_k, x_G)$, and a desired vehicle state $x_{k+1} \in \mathbb{X}_{\text{free}}$ to connect.

3. **Steering function:** Try to generate a motion primitive m_p from x_k to x_{k+1} by solving a BVP as in (3.3) such that the state transition $x_{k+1} = f_p(x_k, m_p)$ is achieved and the resulting trajectory $c(x_k, m_p) \in \mathbb{X}_{\text{free}}$. If this stage fails, then go back to Step 2.

4. **Insert in graph:** If $m_p \notin \mathcal{E}$, insert m_p. If $x_{k+1} \notin \mathcal{V}$, it is added with cost-to-come $J(x_I, x_{k+1}) = J(x_I, x_k) + J(m_p)$. If x_{k+1} already exists in $\mathcal{V}$ and $J(x_I, x_{k+1})$ is more expensive, x_{k+1} is not inserted. On the contrary, if $J(x_I, x_{k+1})$ is cheaper, a rewiring step may be performed where the current edge to x_{k+1} is replaced with the new edges.

5. **Check for a solution:** If $x_{k+1} = x_G$ (or $x_{k+1} \in \mathbb{X}_G$), backtrace from x_{k+1} to x_I and mark its trajectory as a solution candidate with cost $J(x_I, x_{k+1})$.

6. **Check for termination:** Go back to Step 2 unless a termination condition is satisfied. Otherwise, if multiple solution candidates are found, return the solution with minimum total cost J or return failure if no solution has been found.

With small modifications, most motion planning algorithms follow the above mentioned steps. Two motion planning algorithms will now be presented, but before having a closer look at these algorithms, some properties for motion planning algorithms are defined. A motion planning algorithm is said to be (LaValle, 2006):

Complete if the algorithm always finds a solution in finite time if one exists or terminates and reports failure if one does not exist.

[2] It is also possible to use a bidirectional or an undirected search graph where the planner is searching from x_I to x_G, from x_G to x_I and possibly from other states as well at the same time.

Resolution complete if the algorithm always finds a solution within a specified resolution in finite time if one exists or terminates and reports failure if one does not exist.

Probabilistically complete if a solution exists, the probability of finding one approaches one as the number of iterations tends to infinity.

Completeness is an attractive property for any motion planning algorithm. However, when motion planning for nonholonomic systems is considered, the completeness property is hard to establish for an algorithm and weaker properties, such as resolution or probabilistic completeness, have to be accepted.

3.5.1 Motion planning using state lattice

The main idea with lattice-based motion planning is to precompute the building blocks for the directed search graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ offline. These building blocks consist of vertices and edges that, when repeated, generate a search graph $\mathcal{G}$ that forms a regular and repeated pattern. This search graph is called a *state lattice* and, thanks to its structure, enables efficient graph search algorithms to be used during online planning (Pivtoraiko et al., 2009).

The set of possible vehicle states $x_k \in \mathcal{V}$ is specified using a regular discretization of the vehicle's state-space and the set of motion primitives $\mathcal{E}$ is precomputed offline by solving a large number of OCPs in the form of (3.3). Each motion primitive $m_p \in \mathcal{E}$ is a feasible trajectory that moves the vehicle from one discrete vehicle state $x_k \in \mathcal{V}$ to another $x_{k+1} \in \mathcal{V}$ in a bounded neighborhood in free-space. In other words, a motion primitive must comply with the specified state-space discretization. The framework assumes that the system is invariant to translation and rotation which implies that the motion primitives can be constructed from the position of the vehicle at the origin. A motion primitive can then be translated and reused from all other discrete positions in $\mathcal{V}$ during online planning. In general, all motion primitives are not applicable from each vehicle state $x_k \in \mathcal{V}$ and the motion primitives that can be used from x_k is denoted $\mathcal{P}(x_k) \subseteq \mathcal{E}$. The size of $\mathcal{P}(x_k)$ represents the *branching factor* from x_k and with increasing number of edges from each $x_k \in \mathcal{V}$, a good heuristic function becomes increasingly important during online planning.

By specifying the set of allowed vehicle states $x_k \in \mathcal{V}$ and precomputing the set of motion primitives $\mathcal{E}$, the continuous-time motion planning problem in (3.2) is approximated by a discrete-time OCP (3.6) where dynamic programming (DP) can be used to solve the problem.

$$
\begin{aligned}
\underset{\{m_p^k\}_0^{N-1},\, N}{\text{minimize}} \quad & J_{\text{DP}} = \sum_{k=0}^{N-1} J(m_p^k) && \text{(3.6a)}\\
\text{subject to} \quad & x_0 = x_I, \quad x_N = x_G, && \text{(3.6b)}\\
& x_{k+1} = f_p(x_k, m_p^k), && \text{(3.6c)}\\
& m_p^k \in \mathcal{P}(x_k), && \text{(3.6d)}\\
& c(m_p^k, x_k) \in \mathbb{X}_{\text{free}}. && \text{(3.6e)}
\end{aligned}
$$

The decision variables of the OCP in (3.6) are the motion primitive sequence $\{m_p^k\}_0^{N-1}$ and its length N. It is assumed that $x_I \in \mathcal{V}$ and $x_G \in \mathcal{V}$ to make the problem well defined. If $x_I \notin \mathcal{V}$ or $x_G \notin \mathcal{V}$, they have to be projected to their closest neighboring state in $\mathcal{V}$ using some distance metric. The projection of x_I to $\mathcal{V}$ should take the region of attraction for the closed-loop system, consisting of the vehicle controller and the controlled vehicle, into account. Otherwise, safe execution of the motion plan cannot be guaranteed. The set of applicable motion primitives from a certain vehicle state x_k is encoded in (3.6d) and a state transition $x_{k+1} = f_p(x_k, m_p^k)$ is only valid if the resulting trajectory remains in free space (3.6e). The construction of the building blocks for the search graph is now explained more thoroughly.

State lattice construction

The offline construction of the state lattice can be divided into three separate steps (a simplified example is illustrated in Figure 3.7):

1. Specify the resolution of the search space $\mathcal{V}$ by discretizing the vehicle's state-space.

2. Select which pairs of discrete vehicle states to connect based on the reachability of the system.

3. Generate the set of motion primitives $\mathcal{E}$ by solving the BVPs defined in the previous step.

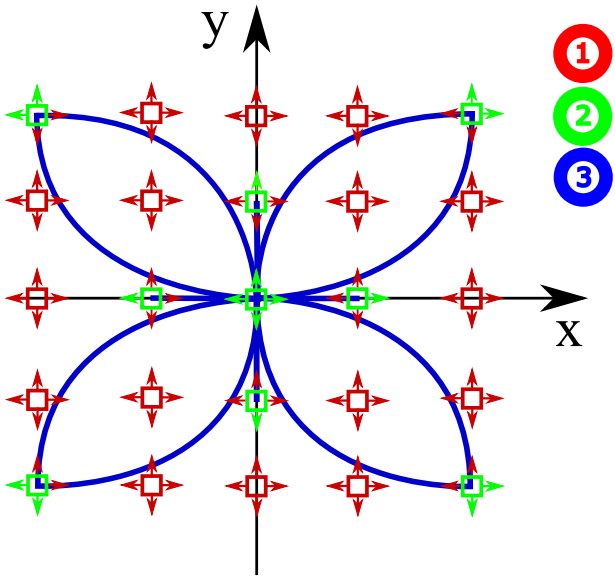


Figure 3.7: *An illustration of the three steps that are performed to generate the state lattice. (1) Discretize the state-space, (2) select which states to connect, (3) solve the BPVs.*

In the first step, the state-space discretization is performed based on the maneuverability of the system and the environment in which the vehicle is intended to operate. For path planning of ground vehicles in unstructured environments, the first step is to specify the fidelity for the position, orientation and steering angle of the vehicle (Pivtoraiko et al., 2009).

The second step is to select which pairs of discrete vehicle states to connect. Here, it is important to be aware of that the complexity of the discrete-time OCP in (3.6) scales exponentially in the number of applicable motion primitives $\mathcal{P}(x_k)$ from each discrete vehicle state x_k. Therefore, a careful selection of vehicle states to connect is most often performed. The selection should be performed such that the resulting *reachability graph* $\mathcal{G}_r(x_I, \mathcal{E})$ (the set of all possible trajectories from x_I given the motion primitive set $\mathcal{E}$) sufficiently resembles the *forward reachable set* of the vehicle from x_I.

The third step is to solve the specified BVPs, *e.g.*, using some of the optimization methods presented in Section 3.4.1. Examples of motion primitives $\mathcal{P}(x_k)$ from a selection of vehicle states x_k for a truck with a dolly-steered trailer and for a quadcopter platform are depicted in Figure 3.2 where Casadi (Andersson et al., 2018a) together with IPOPT (Wächter and Biegler, 2006) are used to solve the BVPs. To clarify these steps, Example 3.1 goes through them to construct a minimal state lattice for a kinematic bicycle model.

Example 3.1: The Kinematic bicycle model

The vehicle states for the Kinematic bicycle model (2.11) are the position $(x_1, y_1) \in \mathbb{R}^2$ and orientation $\theta_1 \in (-\pi, \pi]$. The control signals are the steering angle α and longitudinal velocity $v = 1$, *i.e.*, the vehicle is only allowed to move forward with constant longitudinal velocity. The position of the vehicle is discretized to a uniform grid with resolution r in both x and y. The orientation is discretized into four different orientations

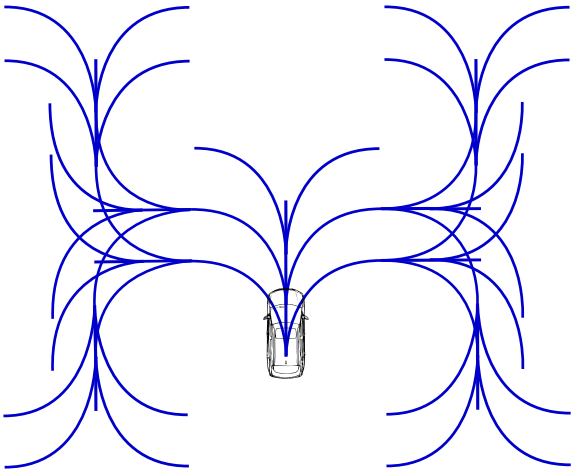


Figure 3.8: *The first three levels of the reachability graph $\mathcal{G}_r(x_I, \mathcal{E})$ from $x_I = (0, 0, \pi/2)$ where the vehicle is restricted to move according to the motion primitives in Example 3.1.*

$\Theta = \{-\pi/2, 0, \pi/2, \pi\}$. The steering angle α is constrained to zero at each vertex to make sure that it is continuous when multiple edges are combined during online planning.

The pair of discrete vehicle states to connect are illustrated in Figure 3.7 where a green box with a green arrow encodes that a discretized vehicle state is included. The resulting motion primitives are the blue paths where only $\pm 90^\circ$-turns and straight forward motions are generated from each initial orientation $\theta_i \in \Theta$ from the position of the vehicle at the origin. The first three levels of the resulting reachability graph $\mathcal{G}_r(x_I, \mathcal{E})$ from $x_I = (0, 0, \pi/2)$ are illustrated in Figure 3.8.

Online planning

During online planning, the discrete-time motion planning problem in (3.6) is solved by deploying efficient graph search methods, such as, A* (Hart et al., 1968) or anytime repairing A* (ARA*) (Likhachev et al., 2004). Since the motion planning algorithm is a *deterministic search* in a directed graph $\mathcal{G}$, the lattice planner automatically inherits the properties and guarantees of the chosen graph search algorithm. To further improve the online planning time, the motion primitive set $\mathcal{E}$ can be reduced in a systematic way using the reduction technique presented in Pivtoraiko and Kelly (2011). A motion primitive $m_p \in \mathcal{E}$ with stage cost $J(m_p)$ is removed if its state transition in free-space can be achieved by a combination of the other motion primitives in $\mathcal{E}$ with a combined total stage cost J_{comb} that satisfies $J_{comb} \leq \eta J(m_p)$, where $\eta \geq 1$ is a user specified constant. This procedure can be used to verify that redundant motion primitives do not exist in $\mathcal{E}$ by putting $\eta = 1$.

As discussed in Section 3.4.2, an informative heuristic function h is needed to guide the search algorithm during online planning where, *e.g.*, a precomputed HLUT can be used to quickly access the heuristic cost at runtime (Cirillo et al., 2014; Pivtoraiko et al., 2009).

The solution to the discrete-time OCP in (3.6) is an ordered sequence of N motion primitives $\{m_p^k\}_0^{N-1}$, *i.e.*, a feasible and collision-free trajectory $(x_0(t), u_0(t))$, $t \in [t_I, t_G]$ or a path $(x_0(s), u_0(s))$, $s \in [0, s_G]$ that minimizes the objective function J_{DP}. If A* is used as search method and enough planning time is given, the algorithm is guaranteed to return a solution that is *resolution optimal*. The online graph search using A* is presented in Algorithm 1, where the algorithm follows the steps presented in Section 3.5 with some minor adjustments. Firstly, since the motion primitive set is precomputed, no BVPs are solved online. Secondly, not only one but all possible edges $\mathcal{P}(x_k)$ from the best candidate vertex x_k are explored. In the algorithm, the *closedSet* stores the discrete vehicle states that have been expanded from and the *openSet* stores the vehicle states that have been expanded to but remains to be expanded from. The *openSet* is commonly referred to as the *frontier*. If the heuristic function h is admissible, the termination condition is trivial (line 10). If the best candidate vertex x_k from the *openList* is the goal, the optimal motion primitive sequence $\{m_p^{k^*}\}_0^{N^*-1}$ is found with cost $J(x_k, x_I)$ and the sequence is easily obtained by backtracing from x_k to x_I (line 11). Moreover, *the principle of optimality* ensures that every subtrajectory is optimal for its corresponding subproblem.

If $x_G \notin \mathcal{V}$ and an accurate goal state x_G is important, an OCP in the form of (3.3) can to be solved online to optimize the final part of the motion plan $(x_0(\cdot)), u_0(\cdot))$ such that

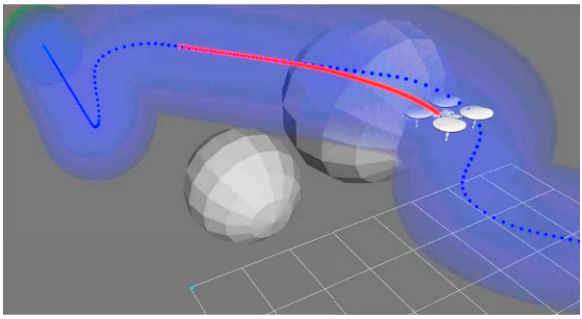
Algorithm 1 Online lattice planning using A*

Inputs: x_I, x_G and $\mathbb{X}_{\text{free}}$
$closedSet = \emptyset$
$openSet = \emptyset$
$x_I.arrival\_edge = 0$
$x_I.cost\_to\_come = 0$
$x_I.cost\_to\_go = h(x_I, x_G)$
$openSet.add(x_I)$
while $openSet \neq \emptyset$ **do**
    $x_k \leftarrow openSet.pop\_best\_candidate()$
    if $x_k = x_G$ **then**
        return $BACKTRACE(x_k)$
    $closedSet.add(x_k)$
    for each $m_p \in \mathcal{P}(x_k)$ **do**
        $x_{k+1} = f_p(x_k, m_p)$
        if $x_{k+1} \in closedSet$ **or** $c(m_p, x_k) \notin \mathbb{X}_{\text{free}}$ **then**
            continue
        $tentativeCost = J(x_k, x_I) + J(m_p)$
        if $x_{k+1} \notin openSet$ **then**
            $openSet.add(x_{k+1})$
        else if $tentativeCost \geq openSet(x_{k+1}).cost\_to\_come$ **then**
            continue
        else
            $openSet.swap(x_{k+1})$
        $x_{k+1}.parent = x_k$
        $x_{k+1}.arrival\_edge = m_p$
        $x_{k+1}.cost\_to\_come = tentativeCost$
        $x_{k+1}.cost\_to\_go = h(x_{k+1}, x_G)$
    end for
end while
return failure

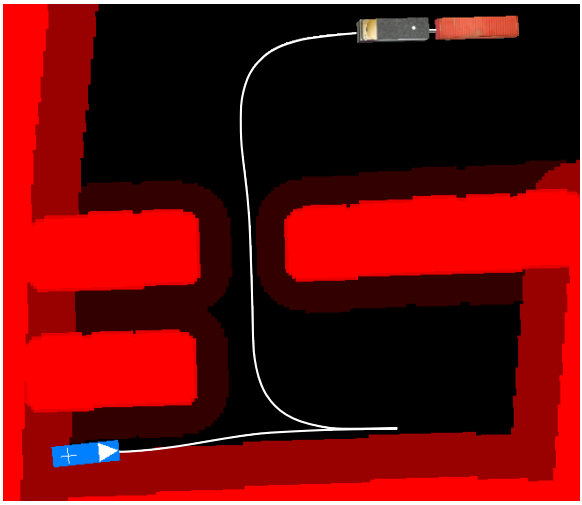
the goal state is reached exactly (Ljungqvist et al., 2017). The same approach can be used for the initial state x_I but is often omitted and the initial transient is left for the vehicle controller to suppress. Due to the algorithm's deterministic nature and real-time capabilities, motion planning using state-lattices has been used with great success on various robotic platforms (Andersson et al., 2018b; Cirillo et al., 2014; Pivtoraiko et al., 2009; Urmson et al., 2008). To illustrate the capability of lattice-based motion planning, two planning examples are illustrated in Figure 3.9 where in (a) is for a quadcopter platform presented in Andersson et al. (2018b) and in (b) is for a general 2-trailer from Paper E.

Other deterministic approaches

There exist other deterministic motion planning algorithms that rely on *input-space discretization* $u \in \mathcal{U}$ instead of state-space discretization. In this case, a model of the vehicle



(a)



(b)

Figure 3.9: *Two online planning examples using a lattice-based motion planner. In (a), trajectory planning for a quadcopter platform and in (b), path planning for a truck with a dolly-steered trailer. The blue and the white paths are the computed motion plans for the position of the quadcopter and the position of the rear axle of the trailer, respectively.*

is used online to simulate the system for Δt seconds, or Δs meters, using a constant or parametrized control signal u from $\mathcal{U}$. In general, the constructed motion primitive does not end up at a specified final state. This implies that the search graph becomes irregular which results in an exponentially exploding frontier during online graph search (Pivtoraiko et al., 2009). To resolve this, the state-space is often divided into cells where a cell is only allowed to contain one vertex. One motion planning algorithm that uses input-space discretization, also called control sampling, is the hybrid A* (Dolgov et al., 2010). Common for these approaches is that they lack all completeness and optimality guarantees and input-space discretization is not applicable for unstable systems, unless the simulations are performed in closed-loop with a stabilizing feedback controller.

On the contrary, control sampling may be more suitable for systems that are not po-

sition invariant, for systems that have a high-dimensional state-space (compared to the input-space) or if some vehicle states (except for the position of the vehicle) have to be discretized with high resolution. The latter case arise in dynamic on-road driving scenarios where trajectory planning is needed. A state-lattice approach is here impractical to use since the discretization of all possible vehicle speeds would lead to a huge search graph during online planning. An efficient algorithm that is tailored for on-road driving is the so-called spatiotemporal lattice formulation (McNaughton et al., 2011; Ziegler and Stiller, 2009) where parts of the vehicle's state-space is regularly discretized offline and other parts are implicitly discretized during online planning using control sampling.

3.5.2 Motion planning using RRT

Another popular motion planning algorithm is the so called *rapidly-exploring random tree* (RRT) (LaValle, 1998). Here, only a brief overview is presented and the reader is referred to, *e.g.*, Evestedt (2016) for more details. In contrast to the deterministic approaches presented in the previous section, the RRT algorithm is a probabilistic motion planning algorithm. The original RRT algorithm was developed for holonomic systems and was later extended to the nonholonomic case in LaValle and Kuffner (2001) where the algorithms follow the main steps presented in Section 3.5.

The RRT algorithm incrementally constructs a directed search tree[3] during online planning. The search tree is expanded by iteratively drawing random samples $x_{rand} \in \mathbb{X}$. A steering function is then used to connect from its nearest neighboring state x_{near} in the tree towards x_{rand}. If the resulting edge is collision-free, a new vertex is added to the tree where x_{near} is determined by sorting the vertices in the search tree using a heuristic function, *e.g.*, a look-up table for a nonholonomic system (LaValle and Kuffner, 2001; Pivtoraiko et al., 2009). If an efficient steering function exists, the RRT algorithm can handle a specified goal state x_G by performing goal sampling. Otherwise, the desired goal state has to be replaced with a goal region $\mathbb{X}_G$, *e.g.*, a ball centered around the goal state: $X_G = \{x \in \mathbb{X} \mid ||x - x_G||_Q \leq \varepsilon\}$ where $Q \succ 0$ and $\varepsilon > 0$.

For holonomic systems, the RRT algorithm is *probabilistically complete*, but is also guaranteed with probability one to converge to a suboptimal solution (Karaman and Frazzoli, 2010). For the nonholonomic case, the RRT algorithm lacks all completeness guarantees (Karaman and Frazzoli, 2010; Kunz and Stilman, 2015). An extension of the basic RRT is the so-called RRT* (Karaman and Frazzoli, 2011) where a rewiring procedure is performed in each tree expansion. In this step, the edges for the neighboring vertices in a ball around the new sample is rewired if a lower cost-to-come can be achieved. With this extension, RRT* is shown to be *probabilistically optimal* meaning that the algorithm will converge towards the optimal solution when the number of samples approaches infinity. However, the rewiring procedure involves solving several BVPs online. Therefore, its practical applicability is limited to systems where efficient BVP solvers exist.

Many modifications to the original RRT formulations have been proposed to speed up the search using different pruning strategies and heuristic functions that directs the sampling towards interesting regions. These methods are out of the scope of this thesis and a small selection of extensions can be found in (Gammell, 2017; Kalisiak and van de Panne, 2006; Nasir et al., 2013; Urmson and Simmons, 2003).

[3] A search tree is a special type of a search graph where each vertex only has one parent.

Anytime implementations of RRTs have been successfully deployed in dynamical environments by (Kuwata et al., 2009) using biased sampling of reference signals to a closed-loop system. The closed-loop RRT (CL-RRT) algorithm was developed to enable a longer tree expansion without solving expensive BVPs online, *i.e.*, the stabilizing controller is acting as a solver. Moreover, the CL-RRT framework is suitable for unstable systems and this is exploited in Paper B where a motion planner for a general 2-trailer system is presented.

4

Vehicle control techniques

This chapter presents different *feedback control techniques* for self-driving vehicles. In this context, the objective of the feedback controller is to ensure that the controlled vehicle executes a computed motion plan with a small control error even if disturbances and initial control errors may be present. Depending on the reference obtained from the motion planner, the feedback control problem will either be referred to as a *path following control problem* or a *trajectory tracking control problem*. We recall that the vehicle is assumed to be modeled as a time-invariant nonlinear system $\dot{x}(t) = f(x(t), u(t))$. During the control design, it is assumed that all vehicle states $x(t)$ are perfectly measured[1] and that the system is controllable, at least locally around the nominal trajectory (Khalil and Grizzle, 1996). Moreover, the motion plan is assumed to be feasible and collision-free, *i.e.*, it satisfies the system dynamics and does not collide with any surrounding obstacles. Here, the theory will be presented in continuous-time but most of the material has a discrete-time counterpart.

The remainder of this chapter is structured as follows. In Section 4.1, the trajectory tracking and path following control concepts are explained. In Section 4.2, the trajectory tracking problem is formally derived and some controller synthesis techniques are presented. In Section 4.3, path following for wheeled vehicles is presented together with some path following techniques tailored for self-driving vehicles.

4.1 Problem formulations

The role of the feedback controller is to stabilize the vehicle around a nominal path or trajectory computed by a motion planner. Even though the control tasks differs, the solution will be a feedback control-law. The control tasks are defined as follows:

[1] In practice, the vehicle states are obtained through onboard sensors on the vehicle. The sensor information is fused using nonlinear filtering techniques to obtain a state estimate $\hat{x}(t)$.

Trajectory tracking: The closed-loop system consisting of the controlled vehicle and the trajectory tracking controller should execute a nominal trajectory $(x_0(t), u_0(t))$ with a small tracking error $\bar{x}(t) = x(t) - x_0(t)$.

Path following: The closed-loop system consisting of the controlled vehicle and the path following controller should execute a nominal path $(x_0(s), u_0(s))$ with a small control error $\tilde{x}(t) = x(t) - x_0(s(t))$. Here, the path parameter $s(t)$ is defined as the actual progression of the vehicle with respect to the nominal path.

In *trajectory tracking*, the reference consists of nominal vehicle states $x_0(t)$ and controls $u_0(t)$ that are parametrized in time. In other words, the objective of the trajectory tracking controller is to stabilize the vehicle around a nominal state trajectory. Here, the nominal trajectory $(x_0(t), u_0(t))$ at each time instant is predetermined and is independent of the vehicle's current state $x(t)$. The trajectory tracking problem is illustrated in Figure 4.1a. Note that *point-stabilization* is a special case of trajectory tracking where the nominal trajectory is a single point.

In *path following*, the obtained reference from the motion planner consists of nominal vehicle states $x_0(s)$ and controls $u_0(s)$ that are parametrized in space rather than in time. In other words, the path following controller acts upon a space-parametrized control error that is implicitly parametrized in time. Given a vehicle state $x(t)$, the nominal vehicle states $x_0(s(t))$ and control $u_0(s(t))$ are calculated by projecting the position $p(t)$ of the vehicle onto the nominal path in $p_0(s)$.

For wheeled vehicles, the control signals typically consist of the steering angle $\alpha(t)$ and the longitudinal velocity $v_1(t)$ of the vehicle. The path-following controller uses the steering angle to stabilize the vehicle around the nominal path and the longitudinal velocity is controlled such that the nominal path is executed at a desired set speed $v_0(t)$. In this setup, the control task is said to be separated into *lateral* and *longitudinal* control. Figure 4.1b illustrates how a path following controller operates to stabilize the vehicle around the nominal path. For small control errors, a trajectory tracking controller and a path following controller performs similarly. However, if a large longitudinal tracking error is present in a sharp corner, a trajectory tracking controller can behave undesirably and take shortcuts since longitudinal and lateral control are coupled.

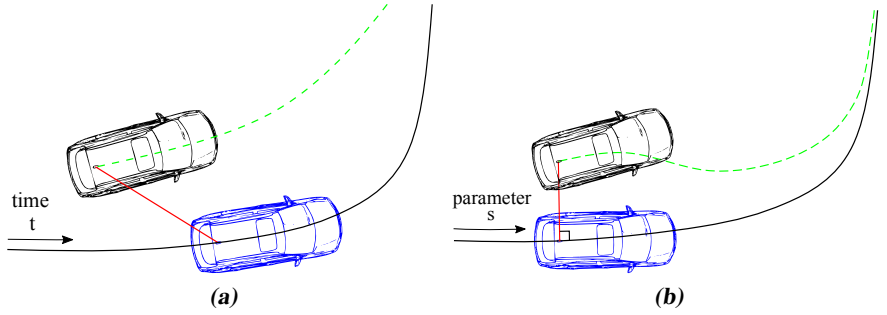


Figure 4.1: *Illustration of the difference between trajectory tracking (a) and path following control (b). The black vehicle is the controlled vehicle, the blue vehicle represents the reference vehicle, the red line is the positional control error and the green line represents a typical convergence behavior of the closed-loop system when trajectory tracking (a) and path following control (b) is used, respectively.*

4.2 Trajectory tracking control

In trajectory tracking control applications, the motion planner is assumed to provide the feedback controller with a dynamically-feasible and collision-free nominal trajectory $(x_0(t), u_0(t))$, $t \in [0,\infty)$[2]. Since the nominal trajectory is feasible, it can equivalently be written as

$$\dot{x}_0(t) = f(x_0(t), u_0(t)), \quad x_0(0) \text{ given.} \tag{4.1}$$

The objective of the trajectory tracking controller is to control the state trajectory $x(t)$ of the vehicle such that the nominal trajectory (4.1) is executed with a small tracking error $\bar{x}(t) = x(t) - x_0(t)$. Denote the control signal deviation as $\bar{u}(t) = u(t) - u_0(t)$, then the *tracking error system* becomes

$$\dot{\bar{x}}(t) = f(\bar{x}(t) + x_0(t), \bar{u}(t) + u_0(t)) - f(x_0(t), u_0(t)) \triangleq \bar{f}(t, \bar{x}(t), \bar{u}(t)), \tag{4.2}$$

where $(\bar{x}(t), \bar{u}(t)) = (0,0)$ is an equilibrium point since $\bar{f}(t,0,0) = 0$, $\forall t \geq 0$ and the initial tracking error $\bar{x}(0) = x(0) - x_0(0) = \bar{x}_0$ is given. Note that the nominal trajectory $(x_0(\cdot), u_0(\cdot))$ makes the tracking error system in (4.2) become time-varying. Define the feedback controller as $\bar{u}(t) = g(t, \bar{x}(t))$ where $g(t,0) = 0$. With this feedback control-law, the *closed-loop tracking error system* can compactly be written as

$$\dot{\bar{x}}(t) = \bar{f}(t, \bar{x}, g(t, \bar{x}(t))) \triangleq \bar{f}_{cl}(t, \bar{x}(t)), \quad \bar{x}(0) = \bar{x}_0, \tag{4.3}$$

where the origin $\bar{x}(t) = 0$ is an equilibrium point since $f_{cl}(t,0) = \bar{f}(t,0,0) = 0$, $\forall t \geq 0$. The closed-loop tracking error system in (4.3) is a nonlinear time-varying system and the objective is to design the feedback controller $\bar{u}(t) = g(t, \bar{x}(t))$ such that the origin of (4.3) obtains some of the following properties.

Definition 4.1 (Khalil and Grizzle (1996) Definition 4.4). The equilibrium point $\bar{x} = 0$ of (4.3) is

- *stable*, if, for each $\varepsilon > 0$, there is $\delta = \delta(\varepsilon, t_0) > 0$ such that

$$||\bar{x}(t_0)|| < \delta \Rightarrow ||\bar{x}(t)|| < \varepsilon, \quad \forall t \geq t_0 \geq 0. \tag{4.4}$$

- *uniformly stable* if, for each $\varepsilon > 0$, there is $\delta = \delta(\varepsilon) > 0$, independent of t_0, such that (4.4) is satisfied.

- *asymptotically stable* if it is stable and there is a positive constant $c = c(t_0)$ such that $\bar{x}(t) \to 0$ as $t \to \infty$, for all $||\bar{x}(t_0)|| < c$.

Other desirable stability properties of the origin includes *exponential stability* where the rate of convergence is upper-bounded by an exponential decay and *uniform asymptotic stability* which means that asymptotic stability around the origin is independent of the initial time (Khalil and Grizzle, 1996). In many applications, global stability properties

[2] In the previous chapter, we assumed that the motion plan is constructed such that the goal is an equilibrium point, *i.e.*, $f(x_0(t), u_0(t)) = 0$, $t \geq t_G$. This assumption implies that the motion plan can be extended to infinity where the vehicle is specified to remain it the goal state $\forall t \geq t_G$.

of the origin for the closed-loop tracking error system (4.3) is hard to establish and only local stability properties can be verified (Paden et al., 2016).

Generally, the problem of determining uniform stability (uniform asymptotic stability) of the closed-loop system in (4.3) is to find a *Lyapunov function candidate* $V(t,\bar{x}) \geq 0$, where its time derivative

$$\dot{V}(t,\bar{x}) = \frac{\partial V(t,\bar{x})}{\partial \bar{x}} \bar{f}_{cl}(t,\bar{x}(t)) + \frac{\partial V(t,\bar{x})}{\partial t}, \tag{4.5}$$

is non-positive (negative) for all $t \geq 0$ in a neighborhood around the origin (Khalil and Grizzle, 1996). For more details regarding stability of nonlinear systems, the reader is referred to (Khalil and Grizzle, 1996). Some techniques for designing the feedback control-law $\bar{u}(t) = g(t,\bar{x}(t))$ are now presented.

4.2.1 Trajectory tracking using linearization

Linear control techniques is a simple but yet efficient design tool for trajectory tracking of nonlinear systems where the nonlinear tracking error system (4.2) is first linearized around the nominal trajectory. Since the reference trajectory is dynamically-feasible, *i.e.*, it satisfies the system dynamics and its physically imposed constraints in $x_0(t) \in \mathbb{X}$ and $u_0(t) \in \mathbb{U}$, $\forall t \geq 0$. Assume that they are satisfied with a margin, *i.e*, there is room for the trajectory tracking controller to reject disturbances during the plan execution. Then, it is reasonable to neglect the constraints during the control design if the initial tracking error $\bar{x}(0)$ is small and the designed controller is able to locally stabilize the vehicle around the nominal trajectory. A first-order Taylor series expansion of the tracking error system (4.2) around the nominal trajectory yields

$$\dot{\bar{x}}(t) = A(t)\bar{x}(t) + B(t)\bar{u}(t), \quad \bar{x}(0) = \bar{x}_0, \tag{4.6}$$

where

$$A(t) = A(x_0(t),u_0(t)) = \left.\frac{\partial f(x(t),u(t))}{\partial x}\right|_{(x(t),u(t))=(x_0(t),u_0(t))}, \tag{4.7a}$$

$$B(t) = B(x_0(t),u_0(t)) = \left.\frac{\partial f(x(t),u(t))}{\partial u}\right|_{(x(t),u(t))=(x_0(t),u_0(t))}. \tag{4.7b}$$

The system in (4.6) is a *linear time-varying* (LTV) system. Here, we are interested in stabilizing the origin of (4.6) using a *linear state-feedback controller* with *feedforward action*

$$u(t) = u_0(t) + K(t)\bar{x}(t), \tag{4.8}$$

where $K(t) \in \mathbb{R}^{m\times n}$ is the *state-feedback gain*. With this state-feedback controller inserted in the nonlinear tracking error system (4.2), the closed-loop system is on the form in (4.3). The feedback gain $K(t)$ is designed such that the origin to the closed-loop tracking error system (4.3) obtains desired stability properties. By inserting the control-law (4.8) into the linearized tracking error system (4.6), the closed-loop system is

$$\dot{\bar{x}}(t) = (A(t) + B(t)K(t))\bar{x}(t) \triangleq A_{cl}(t)\bar{x}(t), \quad \bar{x}(0) = \bar{x}_0, \tag{4.9}$$

The origin of the linearized closed-lopp tracking error system in (4.9) can be shown to be globally exponentially stable using Theorem 4.1.

Theorem 4.1 (Khalil and Grizzle (1996)). *Let $A_{cl}(t)$ be continuous $\forall t \geq 0$. Suppose there exists a continuously differentiable $P(t) \succ 0$ that satisfies*

$$0 \prec c_1 I \preceq P(t) \preceq c_2 I, \quad \forall t \geq 0,$$

for positive constants c_1 and c_2, where $c_1 < c_2$. If $P(t)$ satisfies the Lyapunov inequality

$$\dot{P}(t) + P(t)A_{cl}(t) + A_{cl}^T(t)P(t) \preceq -Q, \quad \forall t \geq 0, \tag{4.10}$$

with $Q \succ 0$, then, the origin to (4.9) *is a globally exponentially stable equilibrium point.*

Proof: The proof follows directly from Theorem 4.10 in Khalil and Grizzle (1996). □

The condition in (4.10) is equivalent to the existence of a Lyapunov function $V(t,\bar{x}) = \bar{x}^T P(t)\bar{x}$ that satisfies $\dot{V}(t,\bar{x}) \leq -\bar{x}^T Q\bar{x}$, $\forall t \geq 0$, for the closed-loop LTV system in (4.9). In fact, exponential stability of the origin of the closed-loop LTV system in (4.9) is a sufficient condition for exponential stability of the origin of the nonlinear closed-loop system in (4.3). The result is formalized in Theorem 4.2.

Theorem 4.2 (Khalil and Grizzle (1996)). *Let $\bar{x}(t) = 0$ be an equilibrium point for the nonlinear system in* (4.3) *where $\tilde{f}_{cl} : [0,\infty) \times D \to \mathbb{R}^n$ is continuously differentiable, $D = \{\bar{x} \in \mathbb{R}^n \mid ||\bar{x}||_2 < r\}$, and the Jacobian matrix $A_{cl}(t)$ is bounded and Lipschitz on D, uniformly in $t \geq 0$. Then, the origin is an exponentially stable equilibrium point for the nonlinear system in* (4.3) *if it is an exponentially stable equilibrium point for the LTV system in* (4.9).

Proof: See Theorem 4.15 in Khalil and Grizzle (1996). □

Remark 4.1. Both Theorem 4.1 and 4.2 can be converted to the finite-time case. In that case, exponential stability is replaced with that the norm of the tracking error $||\bar{x}(t)||$ is upper bounded by a positive and exponentially decreasing function in time.

To conclude, if the feedback gain $K(t)$ is designed such that the closed-loop LTV system in (4.9) satisfies Theorem 4.1. Then, Theorem 4.2 ensures exponential stability of the origin for the nonlinear closed-loop system in (4.3). Now, methods for designing the feedback gain $K(t)$ such that Theorem 4.1 is satisfied are presented.

4.2.2 Robust control design using linear matrix inequalities

In some applications, a static feedback gain K is enough to stabilize the LTV system in (4.9). One way to make use of Theorem 4.1 is to assume that the pairs $[A(t),B(t)]$ lie in the *convex polytope* $\mathbb{P}$, $\forall t \geq 0$, where $\mathbb{P}$ is represented by its L vertices

$$[A(t),B(t)] \in \mathbb{P} = \mathbf{Co}\{[A_1,B_1],\ldots,[A_L,B_L]\}, \tag{4.11}$$

where **Co** denotes the convex hull. Denote the *polytopic linear differential inclusion* (LDI) (Boyd et al., 1994) as

$$\dot{\bar{x}}(t) \in \mathbb{P}\begin{bmatrix}\bar{x}(t)\\ \bar{u}(t)\end{bmatrix}, \quad \bar{x}(0) = \bar{x}_0. \tag{4.12}$$

We can interpret the polytopic LDI in (4.12) as describing a family of LTV systems including (4.6). This implies that a solution to (4.6) is also a trajectory of the LDI (4.12). Thus, with $\bar{u}(t) = K\bar{x}(t)$ in (4.12), global exponential stability of the closed-loop version of (4.12) implies global exponential stability of the closed-loop LTV system in (4.9). With a constant Lyapunov matrix $P(t) = P \succ 0$ in Theorem 4.1, the condition (4.10) for global exponential stability can be reformulated as a controller synthesis problem (Boyd et al., 1994) on the form of a *matrix inequality* feasibility problem in the variables $P \succ 0$ and $K \in \mathbb{R}^{m\times n}$

$$(A_i + B_i K)^T P + P(A_i + B_i K) \preceq -2\varepsilon P, \quad i = 1,\ldots,L, \tag{4.13}$$

where $\varepsilon > 0$ is a constant and $Q \succ 0$ in (4.10) has been replaced with $2\varepsilon P \succ 0$. Here, $\varepsilon > 0$ denotes the *decay-rate* of the Lyapunov function (Boyd et al., 1994). This matrix inequality is not jointly convex in P and K. However, if $\varepsilon > 0$ is fixed, with the bijective transformation $S = P^{-1} \succ 0$ and $Y = KP^{-1} \in \mathbb{R}^{m\times n}$, the matrix inequality in (4.13) can be rewritten as a *linear matrix inequality* (LMI) in S and Y (Wolkowicz et al., 2012):

$$SA_i^T + Y^T B_i^T + A_i S + B_i Y + 2\varepsilon S \preceq 0, \quad i = 1,\ldots,L. \tag{4.14}$$

In other words, it is an LMI feasibility problem to find a linear state-feedback controller $\bar{u}(t) = K\bar{x}(t)$ that satisfies condition (4.10) in Theorem 4.1. If S and Y are feasible solutions to (4.14), the *quadratic Lyapunov function* is $V(\bar{x}(t)) = \bar{x}(t)^T S^{-1}\bar{x}(t)$ and the linear state-feedback controller is $\bar{u}(t) = YS^{-1}\bar{x}(t)$. Moreover, by Theorem 4.2, the origin of the nonlinear closed-loop system in (4.3) is an exponentially stable equilibrium point. The feasibility problem in (4.14) can be converted into an optimization problem by, *e.g.*, using a nominal feedback gain K_{des} and optimize $||K - K_{\text{des}}||$ (Ljungqvist et al., 2018). One other possibility is to minimize the condition number of S^{-1} (Boyd et al., 1994; Ljungqvist et al., 2016).

If a single static state-feedback controller is not enough to stabilize the vehicle around the nominal trajectory, *gain-scheduling* can be used (Rugh and Shamma, 2000). Here, a set of working points is selected and a feedback-gain $K(x_0,u_0)$ is designed for each working point (x_0,u_0) (Rugh and Shamma, 2000). The resulting gain-scheduled linear feedback controller with feedforward action is

$$u(t) = u_0(t) + K(x_0(t),u_0(t))\bar{x}(t), \tag{4.15}$$

where the state-feedback gain is typically obtained using a convex-combination of neighboring working points to $(x_0(t),u_0(t))$. Unless the system matrices $A(x_0(t),u_0(t))$ and $B(x_0(t),u_0(t))$ are *slowly-varying*, stability for the resulting closed-loop LTV-system cannot be formally guaranteed (Rugh and Shamma, 2000). Thus, realistic simulations and experiments need to be performed in order to validate a gain-scheduled controller. This is however the case for any feedback control design in practice.

4.2.3 Linear quadratic control

In this section, we turn to the problem of designing the linear state-feedback controller $\bar{u}(t) = K(t)\bar{x}(t)$ in an optimal way. One possibility is to use *linear quadratic* (LQ) control techniques. The time-varying finite horizon LQ control problem is (Anderson and Moore, 2007):

$$\begin{aligned} \underset{\bar{u}(\cdot)}{\text{minimize}} \quad & ||\bar{x}(T)||^2_{P_N} + \int_0^T \left(||\bar{x}(t)||^2_{Q(t)} + ||\bar{u}(t))||^2_{R(t)} \right) dt \\ \text{subject to} \quad & \dot{\bar{x}}(t) = A(t)\bar{x}(t) + B(t)\bar{u}(t), \\ & \bar{x}(0) = \bar{x}_0 \text{ given}, \end{aligned} \tag{4.16}$$

where $T > 0$ is the prediction horizon, $P_N \succeq 0$ is the terminal cost, $R(t) \succ 0$ and $Q(t) \succeq 0$ are design matrices that are used to trade-off between large tracking errors and control signal deviations. Here, $A(t)$ and $B(t)$ describe the time-varying linearization (4.6) of the nonlinear tracking error system in (4.2) about the nominal trajectory $(x_0(\cdot), u_0(\cdot))$. Since $R(t) \succ 0$, the Hamiltion-Jacobi-Bellman (HJB) equation is strictly convex in $\bar{u}(t)$ and the optimal solution to the finite horizon LQ control problem is a linear state-feedback control law $\bar{u}(t) = K(t)\bar{x}(t)$ where the time-varying feedback gain is

$$K(t) = -R^{-1}(t)B^T(t)P(t), \tag{4.17}$$

and $P(t) = P^T(t)$ is obtained by solving a *matrix Riccati partial differential equation*

$$-\dot{P}(t) = A^T(t)P(t) + P(t)A(t) - P(t)B(t)R^{-1}B^T(t)P(t) + Q(t), \tag{4.18}$$

with boundary condition $P(T) = P_N$ (Kalman et al., 1960). Furthermore, the optimal cost for the finite horizon LQ control problem is $V(t,\bar{x}) = \bar{x}(t)^T P(t)\bar{x}(t)$. In general, an explicit solution to (4.18) does not exist but it can be computed numerically (Anderson and Moore, 2007; Zhou et al., 1996). The discrete-time version of (4.16) instead yields a solution $P(t_k)$ that satisfies a *matrix Riccati recursion* (Axehill, 2005; Nielsen, 2017).

The feedback controller defined in (4.17) and (4.18) can be deployed in practice using a *receding horizon control* strategy where the problem is solved over a moving time-window $[t, t+T]$. Here, a matrix Riccati partial differential equation (4.18), or a matrix Riccati recursion for the discrete time case (Axehill, 2005), has to be computed at each sampling instance.

If the matrices $A(x_0(t), u_0(t))$ and $B(x_0(t), u_0(t))$ are almost constant matrices, one single operating point (x_0, u_0) can be used to designing the feedback gain. This motivates to treat the LTV system as a *linear time-invariant system* given by the matrices $A(x_0, u_0)$ and $B(x_0, u_0)$. The time-invariant infinite horizon LQ control problem is

$$\begin{aligned} \underset{\bar{u}(\cdot)}{\text{minimize}} \quad & \int_0^\infty \left(||\bar{x}(t)||^2_Q + ||\bar{u}(t))||^2_R \right) d\tau \\ \text{subject to} \quad & \dot{\bar{x}}(t) = A\bar{x}(t) + B\bar{u}(t), \\ & \bar{x}(0) = \bar{x}_0 \text{ given}. \end{aligned} \tag{4.19}$$

Given that the pair (A,B) is controllable and $(A,Q^{1/2})$ is observable, this problem has an explicit solution and the optimal state-feedback is $\bar{u}(t) = K\bar{x}(t)$ where the feedback gain is

$$K = -R^{-1}B^T P, \tag{4.20}$$

and $P \succ 0$ is the unique solution the *algebraic Riccati equation* (ARE)

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \tag{4.21}$$

The optimal cost for (4.19) is $V(\bar{x}(t)) = \bar{x}(t)^T P\bar{x}(t)$ and the resulting closed-loop system $\dot{\bar{x}} = (A+BK)\bar{x}$ is exponentially stable (Anderson and Moore, 2007) where $V(\bar{x}(t))$ is a valid Lyapunov function candidate. The location of the poles to the closed-loop system are the negative eigenvalues to the *Hamiltonian matrix* (Anderson and Moore, 2007):

$$H = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix}. \tag{4.22}$$

Note that the solution P in (4.21) is often used as terminal cost P_N for the finite horizon LQ control problem in (4.16) where a working point (x_0, u_0) is selected for its derivation (Nielsen, 2017). Moreover, the location of the eigenvalues gives insight for how long prediction horizon T is needed to obtain an good trade-off between tracking performance and computational complexity. Infinite horizon LQ control is a popular feedback design tool due to its simplicity, robustness margins and stability guarantees (Anderson and Moore, 2007). However, if a *state observer* is used to estimate the states, all guarantees are lost (Doyle, 1978). The solution to the LQ problem is optimal in the sense that it minimizes the specified cost function given that the model is correct. However, in practice model uncertainties and other disturbances are always present. H_∞ control (Ravi et al., 1991; Zhou et al., 1996) is a structured design tool for taking such uncertainties and disturbances into account while synthesizing the feedback controller.

4.2.4 Nonlinear trajectory tracking techniques

An extensive amount of advanced nonlinear feedback control techniques has been proposed to address the trajectory tracking problem for nonholonomic systems. Here, only a brief selection is presented and for an overview, the reader is refereed to, *e.g.*, (Luca et al., 1998; Paden et al., 2016) and the references therein. For nonholonomic systems that are differentially flat, the trajectory tracking problem can be efficiently solved by using dynamic feedback compensation (Fliess et al., 1995). Other approaches involve converting the system into chained form (Samson, 1995) and apply linear control techniques in the new coordinate system (Luca et al., 1998). Trajectory tracking using back-stepping and control Lyapunov functions is proposed in Werling et al. (2010) to control a kinematic bicycle model. Some examples of trajectory tracking methods for truck and trailer systems with off-hitched trailers are LQ control approaches (Divelbiss and Wen, 1997) and nonlinear cascade-like control methods (Michałek and Pazderski, 2018). A well known problem with nonlinear control techniques is that the feedback gains can be hard to tune since physical insight may be lost after complex state and input transformations. Moreover, the performance of the resulting closed-loop system can be sensitive to model uncertainties.

Model predictive control

Maybe the most powerful tool for designing the feedback controller is to deploy online optimization techniques where an OCP is solved online at a specified sampling rate. For the previously presented control strategies, the physically imposed constraints on the states $x(t) \in \mathbb{X}$ and controls $u(t) \in \mathbb{U}$ are hard to take into account during the control design. This can be motivated during moderate driving conditions where the nominal trajectory is feasible. However, if the motion planner uses a too simplified vehicle model, it is not certain that the actual motion plan is feasible to execute. This can for example be the case during slippery road surface conditions or during emergency maneuvers where vehicle models with higher fidelity has to be used (Rajamani, 2011). To handle such situations, more responsibility is commonly laid on the trajectory tracking controller to take care of both local planning and control (Paden et al., 2016).

Model predictive control (MPC) (Garcia et al., 1989) is a general control design methodology which can be very efficient for such scenarios (Paden et al., 2016). Similar to the finite horizon LQ control problem in (4.16), an MPC controller uses a model of the vehicle to control (or plan) its trajectory in a receding horizon fashion. Here, a nonlinear model of the vehicle could be used and the vehicle's physical limitations can be taken into account during the control design.

To obtain feedback properties, the idea with MPC is to only apply the first portion of the control signal to the vehicle and continuously resolve the receding horizon OCP at every time step Δt using new state information $x(t)$. The MPC control-law $u_{\text{MPC}}(t)$ is computed by solving the *continuous-time nonlinear MPC problem*

$$\begin{aligned} \underset{u(\cdot)}{\text{minimize}} \quad & \Phi_T(\bar{x}(t+T)) + \int_t^{t+T} \left(||\bar{x}(\tau)||_Q^2 + ||\bar{u}(\tau))||_R^2 \right) d\tau \\ \text{subject to} \quad & \dot{x}(\tau) = f(x(\tau), u(\tau)), \\ & u(\tau) \in \mathbb{U}, \quad x(\tau) \in \mathbb{X}, \\ & x(t) \text{ given.} \end{aligned} \tag{4.23}$$

Here, T denotes the prediction horizon, $\Phi_T(\bar{x}(t+T)) \geq 0$ is the terminal cost, $Q \succeq 0$ and $R \succ 0$ are design matrices. The terminal cost is often chosen as $\Phi_T(\bar{x}(t+T)) = ||\bar{x}(t+T)||_P^2$ where $P \succ 0$ can be computed by solving a continuous-time algebraic Riccati equation (4.21) where the nonlinear system is linearized around a working point (x_0, u_0), *e.g.*, a straight reference trajectory for a wheeled vehicle. Similar to the design of the finite-horizon LQ controller, the location of the stable eigenvalues to the Hamiltonian matrix (4.22) gives insight for how long prediction horizon T is needed to obtain a good tracking performance.

In comparison to the motion planning problem in (3.2), the obstacle imposed constraints are commonly neglected at the trajectory tracking level and only the vehicle's physical limitations are considered. Instead, a quadratic penalty for the deviation from the nominal trajectory $(x_0(\cdot), u_0(\cdot))$ is used as performance measure.

The problem in (4.23) is a nonlinear optimal control problem that in general is very hard to solve to global optimality. However, efficient software exists for real-time deployment of approximate solution to the nonlinear MPC controller in (4.23). One example

is the ACADO code generation toolkit (Houska et al., 2011a) which can be used to automatically generate C-code for a highly efficient discrete-time version of the nonlinear MPC controller in (4.23). The generated code solves a sequence of quadratic programs using qpOASES (Ferreau et al., 2014), an active-set solver which can be efficiently warm-started. The idea is here to initialize the solver with the previous solution (shifted by Δt) and thereby decrease the number of iterations needed for the solver to converge to a locally optimal solution. Similar to H_∞ control (Zhou et al., 1996), robust MPC (Löfberg, 2003) is a structured design tool to take model uncertainties and disturbances into account while designing the MPC controller.

Trajectory tracking control using MPC was first utilized in the process industry to control slowly-varying systems (Rawlings, 2000). Advances in computing power as well as mathematical programming algorithms have now a days made MPC feasible for real-time deployment in self-driving vehicles and a review of existing works can be found in (Lima, 2018; Paden et al., 2016).

4.2.5 Trajectory tracking control using LQ control: An example

A trajectory tracking example for a kinematic bicycle model (2.11) is now presented. The state vector is $x(t) = (x_1(t), y_1(t), \theta_1(t))$, where (x_1, y_1) is the position of the rear axle of the vehicle and θ_1 is its absolute orientation. The control signals are $u(t) = (v_1(t), \kappa_1(t))$ where v_1 is the longitudinal velocity for the rear axle of the vehicle and κ_1 is the vehicle's curvature. Assume a nominal trajectory $\dot{x}_0(t) = f(x_0(t), u_0(t))$ is given. Denote the tracking error as $\bar{x}(t) = x(t) - x_0(t)$ and the control signal deviation as $\bar{u}(t) = u(t) - u_0(t)$, then the tracking error model can compactly be written as $\dot{\bar{x}}(t) = f_{cl}(t, \bar{x}(t), \bar{u}(t))$. Since the model is defined in a global coordinate system, it is not possible to design a static linear state-feedback controller $u(t) = u_0(t) + K\bar{x}(t)$ which stabilizes the tracking error system unless a state-transformation as in (Kanayama et al., 1990) is first performed. By the following change of basis

$$\begin{bmatrix} x_{1,e}(t) \\ y_{1,e}(t) \\ \theta_{1,e}(t) \end{bmatrix} = T(t, \bar{x}(t))\bar{x}(t) = \begin{bmatrix} \cos\theta_1(t) & \sin\theta_1(t) & 0 \\ -\sin\theta_1(t) & \cos\theta_1(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x}_1(t) \\ \bar{y}_1(t) \\ \bar{\theta}_1(t) \end{bmatrix}, \tag{4.24}$$

the tracking error system is (Derivation is provided in Appendix A)

$$\dot{x}_{1,e}(t) = y_{1,e}(t)v_1(t)\kappa_1(t) + v_1(t) - v_{1,0}(t)\cos\theta_{1,e}(t), \tag{4.25a}$$

$$\dot{y}_{1,e}(t) = -x_{1,e}(t)v_1(t)\kappa_1(t) + v_{1,0}(t)\sin\theta_{1,e}(t), \tag{4.25b}$$

$$\dot{\theta}_{1,e}(t) = v_1(t)\kappa_1(t) - v_{1,0}(t)\kappa_{1,0}(t). \tag{4.25c}$$

Represent the tracking error model in (4.25) as $\dot{x}_e = f_e(t, x_e, \bar{u})$. Clearly, $x_e(t) = 0$ is equivalent to $\bar{x}(t) = 0$ since $T(t, \bar{x}(t))$ is nonsingular ($\det T(t, \bar{x}) = 1$). Since the origin $(x_e, \bar{u}) = (0, 0)$ to (4.25) is an equilibrium point, a first-order Taylor-series expansion around the origin yields

$$\dot{x}_e = \begin{bmatrix} 0 & v_{1,0}(t)\kappa_{1,0}(t) & 0 \\ -v_{1,0}(t)\kappa_{1,0}(t) & 0 & v_{1,0}(t) \\ 0 & 0 & 0 \end{bmatrix} x_e(t) + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ \kappa_{1,0}(t) & v_{1,0}(t) \end{bmatrix} \bar{u}(t). \tag{4.26}$$

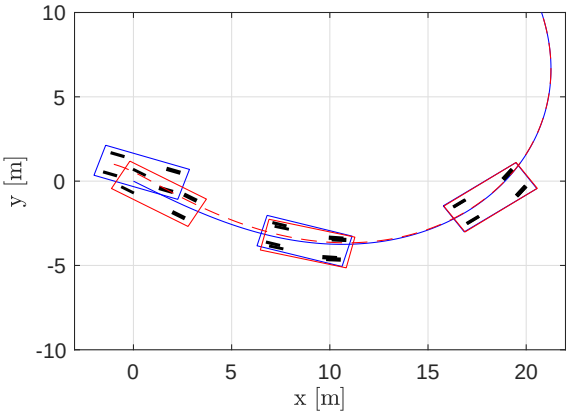


Figure 4.2: *Trajectory tracking for a kinematic bicycle model with initial tracking error $x_e = (-1, 1, 0.2)$. The controlled vehicle is plotted in blue and the nominal vehicle in red. The vehicles are plotted at $t = 0$, 7 and 14 s. The red dotted path displays the trajectory of the controlled vehicle's position during the simulation and the blue path illustrates the trajectory of the nominal position.*

This system is an LTV system with time-varying matrices $A(t)$ and $B(t)$ that depend on the nominal velocity $v_{1,0}(t)$ and curvature $\kappa_{1,0}(t)$ of the nominal trajectory. This system can also be viewed as a linear parameter-varying (LPV) system where $v_{1,0}(t)$ and $\kappa_{1,0}(t)$ are the time-varying parameters. As long as $v_{1,0} \neq 0$, the system is controllable but it can be difficult to stabilize the vehicle over a wide range of nominal curvature and velocity profiles. To handle this, finite horizon LQ control (4.16) or gain-scheduling techniques (4.15) could be used. As a simple example, a single infinite horizon LQ controller (4.19) is designed around the working point $(v_{1,0}, \kappa_{1,0}) = (1, 0)$. With the weight matrices $Q = \text{diag}([1, 1, 10])$ and $R = \text{diag}([10, 40])$, the LQ controller is given by

$$u(t) = u_0(t) - \begin{bmatrix} 0.3162 & 0 & 0 \\ 0 & 0.1581 & 0.7525 \end{bmatrix} x_e(t). \tag{4.27}$$

At this working point, it is clear that the longitudinal tracking error $x_{1,e}$ is stabilized by $v_1(t)$ and the lateral $y_{1,e}$ and angular $\theta_{1,e}$ tracking errors are stabilized by $\kappa_1(t)$. This is however not the case if the system is linearized around a working point $\kappa_{1,0} \neq 0$. The nominal trajectory is generated by simulating the vehicle using a sinusoidal nominal velocity and curvature profile

$$v_{1,0}(t) = 1 + A_v \sin(w_v t), \tag{4.28}$$
$$\kappa_{1,0}(t) = A_\kappa \sin(w_\kappa t), \tag{4.29}$$

with $w_v = w_\kappa = 0.1$ rad/s, $A_v = 0.5$ m/s and $A_\kappa = 0.1$ m^{-1}. To perform more realistic simulations, additional first-order systems for the curvature and the longitudinal velocity with time constants $T_\kappa = 0.2$ s and $T_v = 0.5$ s, respectively, are included. The simulation is depicted in Figure 4.2, where an initial tracking error $x_e(0) = (-1, 1, 0.2)$ is used to analyze how the feedback controller handles disturbance rejection. As can be seen from Figure 4.2, the tracking errors converge towards zero. In Figure 4.3, the control signals

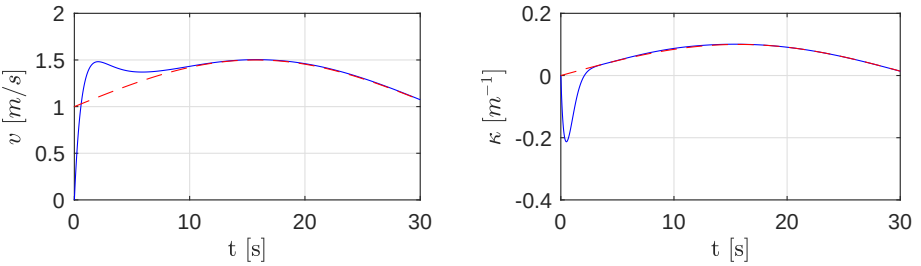


Figure 4.3: *The control signals (blue line) and their feedforwards (dashed red) during the trajectory tracking example.*

are plotted together with their nominal trajectories. As can be seen, the feedback part handles disturbance rejection and the feedforward part takes care of tracking.

4.3 Path following control

In self-driving vehicle applications, it is often convenient to separate the control problem into lateral and longitudinal control (Paden et al., 2016). The lateral controller is called a path following controller and its objective is to make sure that a nominal path $(x_0(s), u_0(s))$, $s \in [0, S_G]$ obtained from the motion planner is traversed with a small lateral and angular control error. After initial transients have been eliminated, the longitudinal controller is designed such that the motion plan is executed at a desired velocity, *i.e.*, $\dot{s}(t) = v_0(t)$. Thus, it is not crucial that the vehicle is located at a specific vehicle state in time. In the sequel, we will assume that the controlled system is a wheeled vehicle[3] represented by a state vector $x(t)$ that can be partitioned as $x(t) = (p(t), \theta_1(t), \eta(t))$, where $p(t) = (x_1(t), y_1(t))$ denotes the absolute position of the vehicle, $\theta_1(t)$ is its absolute orientation and $\eta(t)$ represents additional vehicle states, *e.g.*, n relative angles for the general n-trailer. Moreover, the position $p(t)$ of the vehicle is assumed to have a velocity vector $v_1(t)$ that is aligned with $\theta_1(t)$, *i.e.*, the position of the vehicle evolves as follows

$$\dot{p}(t) = v_1(t) \begin{bmatrix} \cos\theta_1(t) \\ \sin\theta_1(t) \end{bmatrix}. \tag{4.30}$$

Example of systems that satisfy these assumptions are the kinematic bicycle model (2.11) and the truck with a dolly-steered trailer (2.3).

The separation between lateral and longitudinal control is naturally performed by representing the vehicle in a path-coordinate system called the *Frenet-Serret frame* (Bishop, 1975). The Frenet-Serret frame is a coordinate system that moves along with the controlled vehicle. It is used to represent the position of the vehicle $p(t)$ in terms of its progression $s(t)$ along the path in $p_0(s)$ and its signed lateral distance $\tilde{z}(t)$ with respect to $p_0(s(t))$, see Figure 4.4. Here, $s(t)$ is defined as the orthogonal projection of the position

[3] Note that path following can be generalized to the 3D-case as well (Kaminer et al., 2006; Sujit et al., 2014).

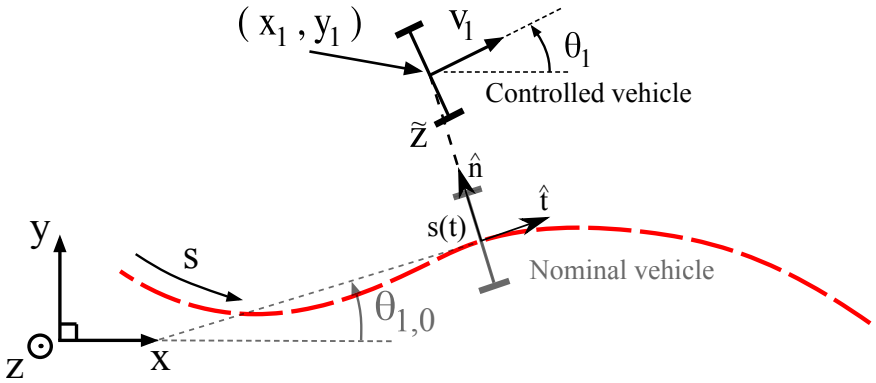


Figure 4.4: *A wheel vehicle in the Frenet-Serret frame where the red-dashed line represents the nominal path in* $(x_{1,0}(s), y_{1,0}(s))$*. The black vehicle illustrates the controlled system and the gray vehicle is the reference vehicle at* $s(t)$*.*

of vehicle $p(t)$ onto the nominal path in $p_0(s) = (x_{1,0}(s), y_{1,0}(s))$ at time t:

$$s(t) = \underset{s\in[0,s_G]}{\arg\min} ||p(t) - p_0(s)||_2. \tag{4.31}$$

Note that at $s(t)$, there exists a nominal vehicle state $x_0(s(t))$ and controls $u_0(s(t))$. Define the curvature $\kappa_0(s)$ of the nominal path in $p_0(s)$ as

$$\kappa_0(s) = \frac{d\theta_{1,0}(s)}{ds}, \tag{4.32}$$

which is assumed bounded for all $s \in [0, s_G]$. In the Frenet-Serret frame, the position of the vehicle $(x_1(t), y_1(t))$ is transformed to $(s(t), z(t))$ and its dynamics are given by (Luca et al., 1998)

$$\dot{s}(t) = v_1 \frac{\cos\tilde{\theta}_1}{1 - \kappa_0(s)\tilde{z}}, \tag{4.33a}$$

$$\dot{\tilde{z}}(t) = v_1 \sin\tilde{\theta}_1. \tag{4.33b}$$

where $\tilde{\theta}_1(t) = \theta_1(t) - \theta_{1,0}(s(t))$ denotes the orientation error of the vehicle. The transformation to the Frenet-Serret frame is defined in a tube around the nominal path in $p_0(s)$ for which $1 - \kappa_0(s)\tilde{z} > 0$. The width of this tube depends on the nominal curvature $\kappa_0(s)$ and when the curvature tends to zero (a straight line) $\tilde{z}$ can vary arbitrarily. Essentially, this coordinate transformation is well-defined if $|\tilde{z}| < |\kappa_0^{-1}|$ when $\tilde{z}$ and κ_0 have the same sign. Moreover, to guarantee $\dot{s}$ to be a monotonic function in time, the orientation error $\tilde{\theta}_1$ is assumed to satisfy $|\tilde{\theta}_1| < \pi/2$. With this assumption, $\dot{s} > 0$ along forward motion segments and $\dot{s} < 0$ along backward motion segments. Differentiating the orientation error $\tilde{\theta}_1(t) = \theta_1(t) - \theta_{1,0}(s(t))$ with respect to time yields

$$\dot{\tilde{\theta}}_1(t) = \dot{\theta}_1(t) + \dot{s}(t)\kappa_0(s(t)), \tag{4.34}$$

where the chain-rule has been applied and $\kappa_0(s(t))$ is defined in (4.32). The additional error states are defined as $\tilde{\eta}(t) = \eta(t) - \eta_0(s(t))$ and their corresponding differential equations $\dot{\eta}(t)$ are then obtained in analogy to (4.34). Define the control error states as $\tilde{x} = (\tilde{z}, \tilde{\theta}_1, \tilde{\eta})$ and the control signal deviation as $\tilde{u}(t) = u(t) - u_0(s(t))$, then the wheeled vehicle can compactly be described in the path-coordinate system as

$$\dot{s}(t) = f_s(s(t), \tilde{x}(t), \tilde{u}(t)), \tag{4.35a}$$

$$\dot{\tilde{x}}(t) = f_e(s(t), \tilde{x}(t), \tilde{u}(t)). \tag{4.35b}$$

The origin $(\tilde{x}(t), \tilde{u}(t)) = (0,0)$ to the control error model in (4.35b) is an equilibrium point since $f_e(s(t), 0, 0) = 0$ for all $s(t) \in [0, s_G]$. The model in (4.35a) describes the rate of which the nominal path is covered and at the origin, $\dot{s} = v_{1,0}(t)$.

The path following problem can now formulated as follows (Paden et al., 2016); Design the feedback controller $\tilde{u}(t) = g(s(t), \tilde{x}(t))$, with $g(s(t), 0) = 0$, such that the origin to the closed-loop control error system

$$\dot{\tilde{x}}(t) = f_e(s(t), \tilde{x}(t), g(s(t), \tilde{x}(t))) = \tilde{f}_{cl}(s(t), \tilde{x}(t)), \tag{4.36}$$

is, *e.g.*, exponentially stable and such that the nominal path is executed at a desired set-speed $v_{1,0}(t)$. With minor adjustments, the model in (4.36) can now be stabilized using any feedback control method presented in Section 4.2.

A path-following controller can easily be converted into a trajectory tracking controller by letting the longitudinal velocity of the vehicle control the progression $s(t)$ such that a nominal trajectory in $s_0(t)$ is tracked. Define the progression error as $\tilde{s}(t) = s(t) - s_0(t)$, then $\dot{s}_0(t) = v_{1,0}(t)$ and the longitudinal error model becomes

$$\dot{\tilde{s}}(t) = \dot{s}(t) - \dot{s}_0(t) = v_1 \frac{\cos\tilde{\theta}_1}{1 - \kappa_0(s)\tilde{z}} - v_{1,0}. \tag{4.37}$$

If $v_1(t)$ is a control signal, the input substitution $\mu(t) = v_1 \frac{\cos\tilde{\theta}_1}{1-\kappa_0(s)\tilde{z}} - v_{1,0}$ transforms the longitudinal error model into a single integrator system $\dot{\tilde{s}}(t) = \mu(t)$ where, *e.g.*, a proportional-integral-derivative (PID) controller can be designed to stabilize the longitudinal error model around $\tilde{s}(t) = 0$.

4.3.1 Pure pursuit controller

An early proposed solution to the path following control problem is the so-called *pure pursuit controller* (Coulter, 1992; Wallace et al., 1985). This controller uses a *lookahead distance* R to find its *lookahead point* $P(t) = (x_p(t), y_p(t))$ on the nominal path in $p_0(s)$. This point can be computed by finding the intersection between the reference path and a semi-circle with radius R that is centered at the vehicle's current position $(x_1(t), y_1(t))$, see Figure 4.5. The pure pursuit controller computes a desired curvature $\kappa(t)$ of the vehicle such that the lookahead point $P(t)$ is reached from the vehicle's current position $(x_1(t), y_1(t))$. However, since the lookahead point moves along with the vehicle, a smooth motion towards the path is obtained. The pure pursuit controller is schematically illustrated in Figure 4.5 and the derivation of the feedback-law can be found in, *e.g.*, Coulter (1992); Kuwata et al. (2009); Ljungqvist (2015). Define $\theta_e(t)$ as the orientation error of

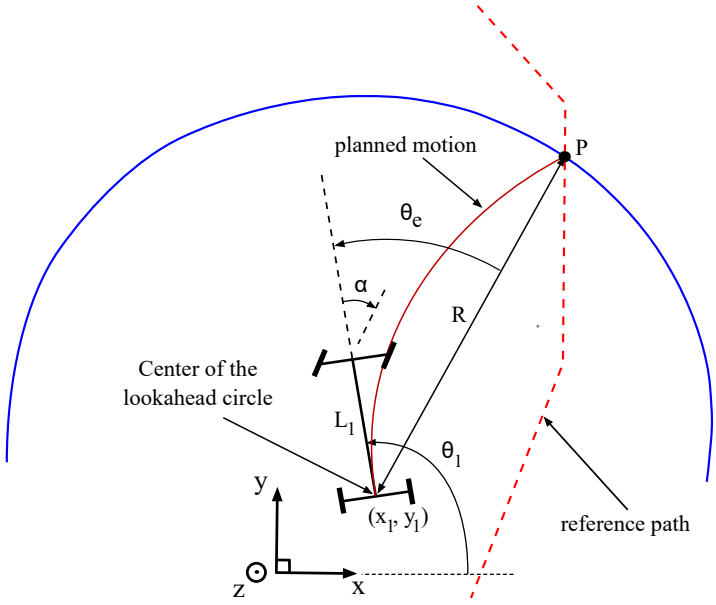


Figure 4.5: *Illustration of the pure pursuit controller. The controller uses a look-ahead distance R to compute the lookahead point P on the path where a feedback law is derived based on the look-ahead distance and the orientation error θ_e.*

the vehicle with respect to the lookahead point $P(t)$. In forward motion, $\theta_e(t)$ can be computed as

$$\theta_e(t) = \theta_1(t) - \text{atan2}(y_p(t) - y_1(t), x_p(t) - x_1(t)), \tag{4.38}$$

where $\text{atan2}(\Delta y, \Delta x)$ denotes the generalization of $\arctan(\Delta y/\Delta x)$ over all four quadrants. Using basic geometry, the pure pursuit feedback-law in forward motion is (Kuwata et al., 2009)

$$\kappa(t) = -\frac{2\sin\theta_e(t)}{R}, \tag{4.39}$$

where, the curvature of the vehicle maps to the steering angle as $\alpha(t) = \arctan(L_1\kappa(t))$.

In backward motion, the semi-circle is flipped and if the orientation error $\theta_{e,\text{rev}}(t)$ is defined as follows

$$\theta_{e,\text{rev}}(t) = \text{atan2}(y_1(t) - y_p(t), x_1(t) - x_p(t)) - \theta_1, \tag{4.40}$$

the same control-law as in (4.39) can be used in backward motion with $\theta_e(t)$ replaced by $\theta_{e,\text{rev}}(t)$. The design parameter for the pure pursuit controller is the lookahead distance R and the feedback-law becomes less aggressive with increasing value of R (Kuwata et al., 2009). However, its tracking performance degrades with increasing value of R. Commonly, the lookahead distance is parametrized with respect to the vehicle's nominal speed to trade off between smoothness during high speeds and improved tracking performance during low speeds (Kuwata et al., 2009).

The pure pursuit controller has been widely used in practice due to its simple implementation and satisfactory performance. For instance, two vehicles in the DARPA Grand Challenge (Buehler et al., 2007) and three vehicles in the DARPA Urban Challenge (Buehler et al., 2009) reported using a pure pursuit controller for path following. One of its strengths is that the nominal path does not necessarily need to be feasible to follow. However, the controller will never execute the nominal path with zero control error unless the nominal path is a straight line. To enhance the robustness properties of the closed-loop system, the center of the lookahead circle can be moved along with the longitudinal center of the vehicle, leading to a slightly adjusted feedback-law (Kuwata et al., 2008, 2009).

4.3.2 Nonlinear path following techniques

Path following control for nonholonomic systems has received a lot of attention from the control and robotic communities over the last decades. A vast amount of advanced nonlinear control techniques has been proposed where the natural approach of modeling the vehicle in the Frenet-Serret frame (Bishop, 1975) is usually explored (Altafini, 2003; David and Manivannan, 2014; Fliess et al., 1995; Lima, 2018; Paden et al., 2016). In recent years, path following using MPC has started to be widely used in practice for cars (Cairano et al., 2016; Plessen et al., 2018) and trucks (Lima et al., 2017a) and a survey of path following techniques for such systems can be found in Paden et al. (2016).

For truck and trailer systems with on-axle hitching (standard n-trailer), its flatness property can be explored to design path following controllers using feedback linearization techniques (Fliess et al., 1995; Rouchon et al., 1993). However, for truck an trailer systems with off-axle hitching (general n-trailer), feedback linearization is not applicable (Astolfi et al., 2004; Rouchon et al., 1993). To circumvent this problem, approximate solutions have been proposed (Bolzern et al., 1998), where a simplified reference vehicle that has similar stationary but different transient behavior is used for control design using feedback linearization.

Classical path following techniques for the general 1-trailer involve input-output linearization (Werling et al., 2014) and Lyapunov-based control design (Astolfi et al., 2004). The proposed path following controller in Werling et al. (2014) is evaluated in field experiments for a reversing car with a trailer. Path following in forward motion for the general n-trailer using input-output linearization is proposed in Altafini (2003). The proposed solution is however limited to forward motion since the introduced zero-dynamics become unstable in backward motion. Path following approaches in backward motion for the general n-trailer with only off-hitched trailers using nonlinear cascade-like control methods are proposed in (Michałek, 2014, 2017). Here, the path is assumed to be represented as an analytical equation of the nominal position of the last trailer and the proposed control methods do not need to find the closest distance to the path. However, only the position and orientation of the last trailer is followed by the controller.

Due to the complex dynamics of the general n-trailer, many path following solutions rely on Jacobian linearization of the control error model and then deploy linear control techniques (Altafini et al., 2002; David and Manivannan, 2014). A survey of path following techniques for truck and trailer systems can be found in David and Manivannan (2014).

4.3.3 Path following control using LQ control: An example

To conclude this chapter, a path following example for a kinematic bicycle model is presented. The model of the vehicle is presented in (2.11) where the state-vector is $x(t) = (x_1(t), y_1(t), \theta_1(t))$, and the control signals are $u(t) = (v_1(t), \kappa(t))$. Here, $v_1(t)$ is the longitudinal velocity and $\kappa(t) = \tan\alpha(t)/L_1$ is its curvature. The wheelbase of the vehicle is L_1 and $\alpha(t)$ is its steering angle. The kinematic bicycle model perfectly fits into the general path following framework presented in Section 4.3. The progression $s(t)$ is defined as the location of the rear axle of the vehicle $(x_1(t), y_1(t))$ onto its projection to the nominal path in $(x_{1,0}(s), y_{1,0}(s))$ at time t, see Figure 4.6. The orientation of the vehicle evolves as $\dot{\theta}_1(t) = v_1(t)\kappa(t)$ and by inserting this in (4.34), the complete model of the vehicle in the Frenet-Serret frame (4.35) is

$$\dot{s} = v_1 \frac{\cos\tilde{\theta}_1}{1-\kappa_0(s)\tilde{z}}, \tag{4.41a}$$

$$\dot{\tilde{z}} = v_1 \sin\tilde{\theta}_1, \tag{4.41b}$$

$$\dot{\tilde{\theta}}_1 = v_1 \left(\kappa - \frac{\kappa_0(s)\cos\tilde{\theta}_1}{1-\kappa_0(s)\tilde{z}} \right). \tag{4.41c}$$

The longitudinal dynamics is represented by (4.41a) and the control error model by (4.41b)–(4.41c). For simplicity, we only consider forward motion and assume that $v_{1,0}(t) > 0$ and $v_1(t) > 0$. In this example, the control error is $\tilde{x} = (\tilde{z}, \tilde{\theta}_1)$. Denote the controlled curvature deviation as $\tilde{\kappa} = \kappa - \kappa_0$ and control error model in (4.41b)–(4.41c) can in a compact form be written as

$$\dot{\tilde{x}} = v_1 f_e(s(t), \tilde{x}, \tilde{\kappa}), \tag{4.42}$$

where the origin $(\tilde{x}, \tilde{\kappa}) = (0,0)$ is an equilibrium point since $v_1(t) f_e(s(t), 0, 0) = 0$. Since $v_1(t)$ enters bilinearly into the control error model in (4.42), in analogy with Section 2.3, Time-scaling (Sampei and Furuta, 1986) can be applied to eliminate the speed dependence $|v_1(t)|$ from the model. Thus, during the design of the path following controller we assume, without loss of generality, that the longitudinal velocity $v_1(t)$ of the vehicle only takes on the value $v_1(t) = 1$ m/s[4].

The design of the path following controller is now equivalent to designing a stabilizing feedback controller $\kappa(t) = g(s(t), \tilde{x}(t))$ for the nonlinear system in (4.42) and any method presented in Section 4.2 can now be used with minor adjustments. As an example, we design the path following controller using robust linear control techniques. Since the origin to (4.42) is an equilibrium point, the Jacobian linearization is

$$\dot{\tilde{x}}(t) = A(s(t))\tilde{x} + B\tilde{\kappa}, \tag{4.43}$$

where

$$A(s(t)) = \left.\frac{\partial f_e}{\partial \tilde{x}}\right|_{(0,0)} = \begin{bmatrix} 0 & 1 \\ -\kappa_0^2(s(t)) & 0 \end{bmatrix}, \quad B = \left.\frac{\partial f_e}{\partial \tilde{\kappa}}\right|_{(0,0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4.44}$$

[4]Note that in practice, this assumption is only feasible for low-speed maneuvers since the model is derived based on no-slip conditions.

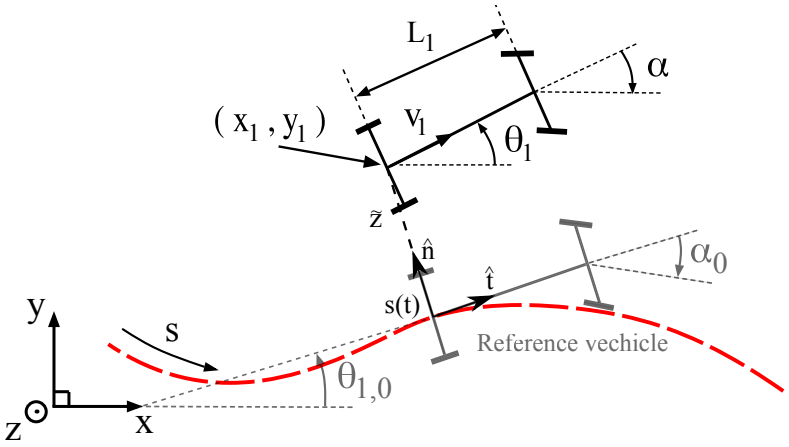


Figure 4.6: *The kinematic bicycle model in the Frenet-Serret frame where the red-dotted line represents the nominal path in $(x_{1,0}(s), y_{1,0}(s))$. The black vehicle illustrates the controlled system and the gray vehicle is the reference vehicle at $s(t)$.*

The objective is to design a static state-feedback controller with feedforward action $\kappa(t) = \kappa_0(s(t)) + K\tilde{x}(t)$ such that the closed-loop system

$$\dot{\tilde{x}}(t) = (A(s(t)) + BK)\tilde{x}(t), \tag{4.45}$$

is exponentially stable. Assume the nominal path is constructed such that the nominal curvature is bounded as $|\kappa_0(s(t))| \leq \kappa_{\max}$. Then, similar to (4.12), we can describe the linearized control error model in (4.43) as a polytopic LDI. The matrix B is a constant and the matrix $A(s(t))$ can be bounded in a polytope (4.11) with 2 vertices

$$A(s(t)) \in \mathbb{P} = \mathbf{Co}\{A_1, A_2\}, \tag{4.46}$$

where

$$A_1 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ -\kappa_{max}^2 & 0 \end{bmatrix}. \tag{4.47}$$

Similar to (4.14), with a constant decay rate $\varepsilon > 0$, the LMI feasibility problem to find a stabilizing feedback gain $K = YS^{-1}$ and a quadratic Lyapunov function $V(\tilde{x}) = \tilde{x}^T S^{-1} \tilde{x}$ with $S^{-1} \succ 0$ is

$$SA_i^T + Y^T B^T + A_i S + BY + 2\varepsilon S \preceq 0, \quad i = 1, 2. \tag{4.48}$$

It is desired that the feedback gain K inherits an infinite-horizon LQ controller's properties. A nominal LQ controller is thus designed around a straight reference path ($\kappa_0(s) = 0$) in forward motion using the weight matrices $R = 1$ and $Q = \frac{1}{20}\mathrm{diag}([1,1])$. The resulting feedback gain is $K_{\mathrm{LQ}} = -\begin{bmatrix} 0.22 & 0.70 \end{bmatrix}$. This controller is not guaranteed to stabilize the LPV system in (4.43) for all nominal curvatures. To establish stability guarantees, we can

minimize the difference between K and K_{LQ} subject to the LMI in (4.48). Since $S \succ 0$, this can be posed as a convex optimization problem[5] (Ljungqvist et al., 2018)

$$\underset{Y,S}{\text{minimize}} \quad ||Y - K_{\text{LQ}}S|| \tag{4.49}$$

$$\text{subject to} \quad (4.48) \text{ and } S \succ I. \tag{4.50}$$

The convex OCP in (4.49) can be solved using YALMIP (Löfberg, 2004) and with $\varepsilon = 0.2$ and $\kappa_{\max} = 0.3$ m^{-1}, the optimal objective function value to (4.49) is zero. This implies that $K = K_{\text{LQ}}$ and one quadratic Lyapunov matrix $P = S^{-1} \succ 0$ is

$$P = \begin{bmatrix} 0.16 & 0.17 \\ 0.17 & 0.63 \end{bmatrix}, \tag{4.51}$$

that guarantees that the closed-loop LTV system $\tilde{x}(t) = (A(s(t)) + BK)\tilde{x}(t)$ is exponentially stable as long as the nominal curvature is bounded as $|\kappa_0(s(t))| \leq \kappa_{\max}$. Moreover, with this feedback-law $\tilde{\kappa}(t) = K\tilde{x}(t)$, the origin of the closed-loop version of the control error model in (4.42) is an exponentially stable equilibrium point when $v_1(t) > 0$.

To illustrate how the path following controller handles disturbance rejection, a simulation of the nonlinear closed-loop system is performed. The reference path is generated using a constant longitudinal velocity profile $v_{1,0}(s) = 1$ and a sinusoidal curvature profile $\kappa_0(s) = 0.2\sin(0.1s)$. Figure 4.8 provides simulation results with an initial control error $\tilde{x}(0) = (\tilde{z}(0), \tilde{\theta}_1(0)) = (1, 0.2)$. As can be seen, the initial lateral and angular control errors are suppressed after roughly 15 s. The value of the quadratic Lyapunov function $V(\tilde{x}(t)) = \tilde{x}^T(t)P\tilde{x}(t)$ for the simulation is provided in Figure 4.7. As can be seen, the closed-loop system is exponentially stable. Note that, a similar control design can be performed for backward motion by simply changing the sign of the matrices $A(s(t))$ and B in (4.43). However, it is not possible to design the linear state-feedback controllers such that a common quadratic Lyapunov function exists in forward and backward motion. This will be shown in Paper D.

[5] If $||K - K_{\text{LQ}}|| = ||YS^{-1} - K_{\text{LQ}}|| = ||(Y - K_{\text{LQ}}S)S^{-1}|| = 0$, then $K = K_{\text{LQ}}$ since $S \succ 0$.

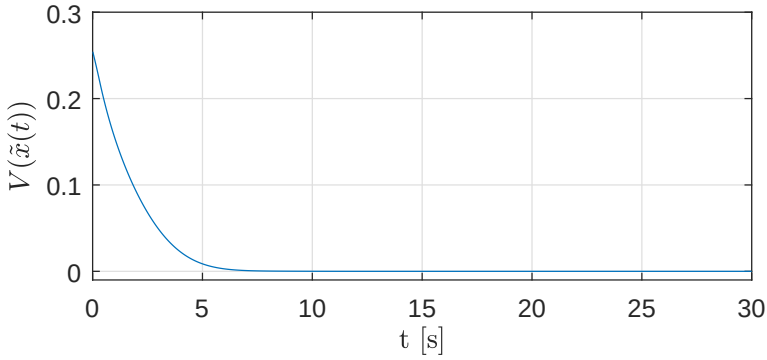
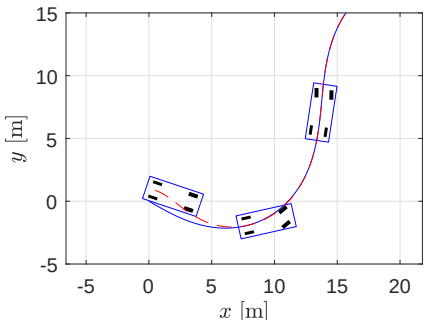
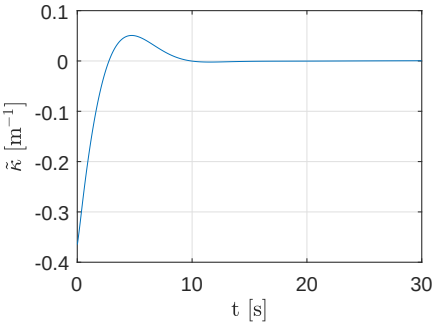


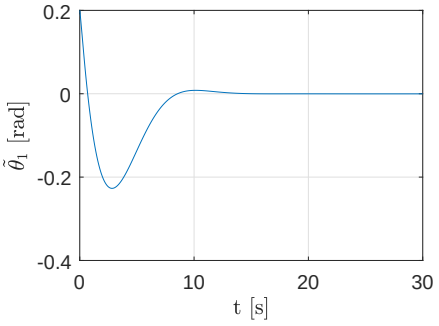
Figure 4.7: *The quadratic Lyapunov function $V(\tilde{x}(t)) = \tilde{x}^T(t)P\tilde{x}(t)$ during the simulation.*



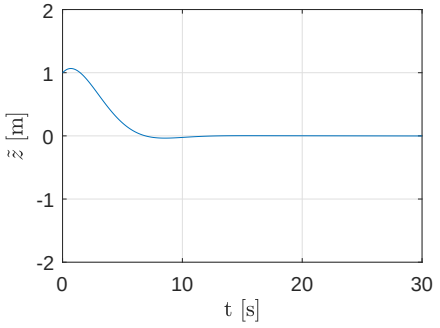
(a) The nominal path (blue line) and the motion of the controlled vehicle (red-dashed line).



(b) The feedback control signal $\tilde{\kappa}(t) = K\tilde{x}(t)$.



(c) The orientation control error $\tilde{\theta}_1(t)$.



(d) The lateral control error $\tilde{z}(t)$.

Figure 4.8: *Simulation results for the closed-loop system using the designed path following controller.*

5

Concluding remarks

In this chapter, a summary of the contributions in the papers that are included in this thesis is given and some directions for future work are discussed. The first part of this thesis gave an introduction to nonholonomic systems and some common motion planning and feedback control techniques of such systems. These techniques have then been used and modified for a number of practical problems and the results have been presented in the papers included in the thesis.

5.1 Summary of contributions

Broadly speaking, the main contributions of this thesis are within motion planning and control of truck and trailer systems in unstructured environments during low speed maneuvers. The first contribution is the development and modification of existing motion planning methods to enable its usage for complex truck and trailer systems. This contribution is contained in Paper B and E. The second contribution is the development of path following controllers for the same system. To develop tools for analyzing stability of the closed-loop system and to enforce constraints at the planning level in order to a priori guarantee safe path execution. This contribution is contained in Paper A, C and D. The third contribution is the deployment of the proposed method on a full-scale truck and trailer system and show that the proposed methods works in practice (Paper E).

In Paper A, a cascaded controller for path tracking of a reversing general 2-trailer is developed. The unstable modes of the nonlinear system is linearized around circular equilibrium configurations and stabilized using LQ control techniques together with gain-scheduling. A pure pursuit controller is designed on top in a cascaded control structure to enable path following of piecewise linear reference paths. A driver assist system is also presented that can be used to manually plan complex maneuvers that the platform can execute. The system is tested in both simulations and on a small-scale test platform.

In Paper B, a probabilistic motion planner for a general 2-trailer system is presented. The path following controller presented in Paper A is used to enable efficient closed-loop simulations of the vehicle within a CL-RRT framework. The framework is evaluated in a series of simulation experiments and is shown to have a high success rate in finding motion plans in complex and constrained environments.

In Paper C, a path following controller for a reversing general 2-trailer is proposed for the case when the obtained motion plan is kinematically feasible to follow exactly. The control errors of the system are modeled in terms of their deviation from the nominal path and a stabilizing LQ controller with feedforward action is then design based on the Jacobian linearization of the control error model. Given that the motion planner is constructing motion plans that satisfy certain properties, exponential stability of the origin for the control error model is established by combining global optimization, theory from LDIs and LMI techniques. The theoretical results are verified in practice through simulations of the closed-loop system around an eight-shaped reference path.

In Paper D, a systematic framework is presented for analyzing stability of the closed-loop system consisting of a controlled vehicle and a feedback controller, executing a motion plan computed by a lattice planner. When this motion planner is considered, it is shown that the closed-loop system can be modeled as a nonlinear hybrid system. Based on this, a novel method is presented for analyzing the behavior of the tracking error, how to design the low-level controller and how to potentially impose constraints on the motion planner in order to guarantee that the tracking error is bounded and decays towards zero. The proposed method is applied on a truck and trailer system and the results are illustrated in two simulation examples.

In Paper E, a complete motion planning and control solution for a truck with a dolly-steered trailer is presented. A state-lattice motion planner for a general 2-trailer system is developed, where a novel parametrization of the vehicle's state-space is proposed to make online planning tractable for real-time applications. A symmetry result is established that enables numerical optimal control to be used for generating the motion primitives. The path following control from Paper C is extended and deployed in practice. Moreover, a nonlinear observer for state estimation is developed which only utilizes information from sensors that are mounted on the truck, making the system independent of additional trailer sensors. The proposed planning and control framework is implemented on a full-scale test vehicle and a series of field experiments are presented.

5.2 Future work

In this section, some ideas for future work and extensions of the contributions that are presented in this thesis are summarized.

Motion planning for truck and trailer systems

The motion planning techniques for the general 2-trailer system that are presented in Paper B and E are possible to generalize to the general n-trailer case. Thus, the proposed techniques are possible to use for, *e.g.*, an additional trailer (Löfroth and Svensson, 2012). It would also be interesting to benchmark their performance with previously presented motion planning methods for the standard n-trailer.

The lattice planner presented in Paper E relies on a system expert that is manually specifying the BVPs to be solve for generating the motion primitive set. This can be problematic if system parameters are changed, *e.g.*, the trailer is switched, and manual retuning of the motion primitive set is needed. Since numerical optimal control is used to generate the motion primitives, it would be possible to specify the maneuvers at a higher level of abstraction to enable automatic retuning of the motion primitive set based on new system parameters.

On-road planning for truck and trailer systems would also be interesting to consider. In forward motion, the trailer angles are stable and do not necessarily need to be stabilized. However, at high speeds, a dynamic model of the vehicle may be used in order to generate motion plans that are dynamically feasible to execute. This would be straightforward to include in a CL-RRT framework, as long as it is possible to simulate the system. Another possibility is to leave more responsibility to the feedback controller to perform both local planning and feedback control.

Feedback control for truck and trailer systems

Also in this case, the path following techniques presented in Paper A and C are possible to generalize to the general n-trailer case. However, the region of attraction for the closed-loop system will become smaller with increasing number of passive trailers. To overcome this, active dolly and trailer steering would be interesting to consider.

More advance control methods, such as MPC, would be interesting to test and see how much can be gained by taking the vehicle's physical limitations, *e.g.*, limited steering angle, into account in the feedback control design. Robust lateral control of truck and trailer systems at high speeds is also an interesting research direction where, *e.g.*, wind disturbances and slippery road surface conditions are examples of external disturbances that the closed-loop system needs to be robust against.

Learning, stability and system architecture

Incorporation learning within a motion planning and control architecture is also an interesting direction of future research. If similar motion plans are used multiple times, it would be possible to adapt the controller and/or the motion plans in order to enhance the control performance of the closed-loop system.

Other important and interesting research questions involve stability and system architecture of the motion planning and feedback control layers. For example, how can closed-loop stability be ensured if the motion planner is performing replanning. These questions become even more interesting if different motion planners and feedback controllers are used in different parts of the environment.

Appendix

A

Derivation of the tracking error system

Here follows the derivation of the tracking error system in (4.25). The kinematic bicycle model is

$$\dot{x}(t) = v_1(t) \begin{bmatrix} \cos\theta_1(t) \\ \sin\theta_1(t) \\ \kappa(t) \end{bmatrix}, \tag{A.1}$$

where the state vector is $x(t) = (x_1(t), y_1(t), \theta_1(t))$ and the control signals are $u(t) = (v_1(t), \kappa_1(t))$. Represent the model in (A.1) as $\dot{x}(t) = f(x(t), u(t))$. The nominal trajectory is feasible and thus satisfies the state-equation $\dot{x}_0(t) = f(x_0(t), u_0(t))$. Denote the tracking error as $\bar{x}(t) = x(t) - x_0(t)$, by applying the proposed state transformation

$$\begin{bmatrix} x_{1,e}(t) \\ y_{1,e}(t) \\ \theta_{1,e}(t) \end{bmatrix} = T(t, \bar{x}(t))\bar{x}(t) = \begin{bmatrix} \cos\theta_1(t) & \sin\theta_1(t) & 0 \\ -\sin\theta_1(t) & \cos\theta_1(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x}_1(t) \\ \bar{y}_1(t) \\ \bar{\theta}_1(t) \end{bmatrix}, \tag{A.2}$$

and use the equality $\dot{x}_{1,0}\sin\theta_{1,0} = \dot{y}_{1,0}\cos\theta_{1,0}$, the tracking error system in (4.25) is

$$\begin{aligned}
\dot{x}_{1,e} &= (\dot{x}_1 - \dot{x}_{1,0})\cos\theta_1 + (\dot{y}_1 - \dot{y}_{1,0})\sin\theta_1 - (x_1 - x_{1,0})\dot{\theta}_1\sin\theta_1 + (y_1 - y_{1,0})\dot{\theta}_1\cos\theta_1 = \\
&= v_1 + y_{1,e}v_1\kappa_1 - \dot{x}_{1,0}\cos\theta_1 - \dot{y}_{1,0}\sin\theta_1 \\
&= v_1 + y_{1,e}v_1\kappa_1 - \dot{x}_{1,0}\cos(\theta_{1,0} + \bar{\theta}_1) - \dot{y}_{1,0}\sin(\theta_{1,0} + \bar{\theta}_1) \\
&= v_1 + y_{1,e}v_1\kappa_1 - \dot{x}_{1,0}(\cos\theta_{1,0}\cos\bar{\theta}_1 - \sin\theta_{1,0}\sin\bar{\theta}_1) - \dot{y}_{1,0}(\sin\theta_{1,0}\cos\bar{\theta}_1 + \cos\theta_{1,0}\sin\bar{\theta}_1) \\
&= v_1 + y_{1,e}v_1\kappa_1 - (\dot{x}_{1,0}\cos\theta_{1,0} + \dot{y}_{1,0}\sin\theta_{1,0})\cos\bar{\theta}_1 + (\dot{x}_{1,0}\sin\theta_{1,0} - \dot{y}_{1,0}\cos\theta_{1,0})\cos\bar{\theta}_1 \\
&= v_1 + y_{1,e}v_1\kappa_1 - v_{1,0}\cos\bar{\theta}_1,
\end{aligned} \tag{A.3}$$

$$\begin{aligned}
\dot{y}_{1,e} &= -(\dot{x}_1 - \dot{x}_{1,0})\sin\theta_1 + (\dot{y}_1 - \dot{y}_{1,0})\cos\theta_1 - (x_1 - x_{1,0})\dot{\theta}_1\cos\theta_1 - (y_1 - y_{1,0})\dot{\theta}_1\sin\theta_1 = \\
&= -x_{1,e}v_1\kappa_1 + \dot{x}_{1,0}\sin(\theta_{1,0} + \bar{\theta}_1) - \dot{y}_{1,0}\cos(\theta_{1,0} + \bar{\theta}_1) \\
&= -x_{1,e}v_1\kappa_1 - \dot{x}_{1,0}(\sin\theta_{1,0}\cos\bar{\theta}_1 + \sin\bar{\theta}_1\cos\theta_{1,0}) - \dot{y}_{1,0}(\cos\theta_{1,0}\cos\bar{\theta}_1 - \sin\theta_{1,0}\sin\bar{\theta}_1) \\
&= -x_{1,e}v_1\kappa_1 + (\dot{x}_{1,0}\sin\theta_{1,0} - \dot{y}_{1,0}\cos\theta_{1,0})\cos\bar{\theta}_1 + (\dot{x}_{1,0}\cos\theta_{1,0} - \dot{y}_{1,0}\sin\theta_{1,0})\sin\bar{\theta}_1 = \\
&= -x_{1,e}v_1\kappa_1 + v_{1,0}\sin\bar{\theta}_1, \qquad \text{(A.4)}
\end{aligned}$$

and

$$\dot{\theta}_{1,e} = \dot{\bar{\theta}}_1 = \dot{\theta}_1 - \dot{\theta}_{1,0} = v_1\kappa_1 - v_{1,0}\kappa_{1,0}. \qquad \text{(A.5)}$$

Together, equations (A.3)–(A.5) represent the tracking error system (4.25) in the new coordinate system.

Bibliography

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

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

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

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

J. A E Andersson, J. Gillis, G. Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018a.

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

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

D. Axehill. Applications of integer quadratic programming in control and communication, 2005.

R. L Bishop. There is more than one way to frame a curve. *The American Mathematical Monthly*, 82(3):246–251, 1975.

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

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

M. Buehler, K. Iagnemma, and S. Singh. *The 2005 DARPA grand challenge: the great robot race*, volume 36. Springer, 2007.

M. Buehler, K. Iagnemma, and S. Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

L. D. Burns. Sustainable mobility: a vision of our transport future. *Nature*, 497(7448): 181, 2013.

S. Di Cairano, U. V. Kalabić, and K. Berntorp. Vehicle tracking control on piecewise-clothoidal trajectories by mpc with guaranteed error bounds. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 709–714, Dec 2016. doi: 10.1109/CDC.2016.7798351.

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

C. R Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

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

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

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

J. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978.

L. E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989. ISSN 0018-9162. doi: 10.1109/2.30720.

Ernst and Young. Who's in the driving seat? how the rise of autonomous vehicles will transform the relationship between man and car. Technical report, 2015.

European Commission. Roadmap to a single european transport area – towards a competitive and resource efficient transport system. Technical report, March 2011.

European Road Transport Research Advisory Council. Multi-annual implementation plan for horizon 2020. Technical report, March 2013.

N. Evestedt. Sampling based motion planning for heavy duty autonomous vehicles, 2016.

N. Evestedt, D. Axehill, M. Trincavelli, and F. Gustafsson. Sampling recovery for closed loop rapidly expanding random tree using brake profile regeneration. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 101–106, June 2015.

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

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

H. J Ferreau, C. Kirches, A. Potschka, H. G Bock, and M. Diehl. qpoases: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, Dec 2014.

M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Design of trajectory stabilizing feedback for driftless flat systems. *Proceedings of the Third ECC, Rome*, pages 1882–1887, 1995.

T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, Dec 2004. ISSN 1552-3098.

D. J Gammell. *Informed anytime search for continuous planning problems*. PhD thesis, 2017.

C. F Garcia, D. M Prett, and M. Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

P. E Gill, W. Murray, and M. A Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

P. E Hart, N. J Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968.

A. Hornung, K. M Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

B. Houska, H. J Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 47(10):2279 – 2285, 2011a.

B. Houska, H. J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011b.

M. Kalisiak and M. van de Panne. Rrt-blossom: Rrt with a local flood-fill behavior. In *ICRA*, pages 1237–1242, 2006.

R. E Kalman et al. Contributions to the theory of optimal control. *Bol. Soc. Mat. Mexicana*, 5(2):102–119, 1960.

I. Kaminer, O. Yakimenko, A. Pascoal, and R. Ghabcheloo. Path generation, path following and coordinated control for timecritical missions of multiple uavs. In *Proceedings of the 2006 American Control Conference*, pages 4906–4913, June 2006. doi: 10.1109/ACC.2006.1657498.

Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 384–389 vol.1, May 1990. doi: 10.1109/ROBOT.1990.126006.

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104:2, 2010.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

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

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

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*, pages 233–244. Springer, 2015.

Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7166, 2008.

Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

J. P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5):577–593, Oct 1994. ISSN 1042-296X. doi: 10.1109/70.326564.

S. M. LaValle. Rapidly-exploring random trees a new tool for path planning. Technical report, 1998.

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

S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

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

P. F. Lima. *Optimization-Based Motion Planning and Model Predictive Control for Autonomous Driving: With Experimental Evaluation on a Heavy-Duty Construction Truck*. PhD thesis, KTH Royal Institute of Technology, Stockholm, 2018.

P. F. Lima, M. Nilsson, M. Trincavelli, J. Mårtensson, and B. Wahlberg. Spatial model predictive control for smooth and accurate steering of an autonomous truck. *IEEE Transactions on Intelligent Vehicles*, 2(4):238–250, Dec 2017a. ISSN 2379-8904.

P. F. Lima, R. Oliveira, J. Mårtensson, and B. Wahlberg. Minimizing long vehicles overhang exceeding the drivable surface via convex path optimization. In *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017b.

G. Ling, K. Lindsten, O. Ljungqvist, J. Löfberg, C. Norén, and C. A. Larsson. Fuel-efficient model predictive control for heavy duty vehicle platooning using neural networks. In *Proceedings of the 2018 Annual American Control Conference (ACC)*, pages 3994–4001, June 2018.

O. Ljungqvist. Motion planning and stabilization for a reversing truck and trailer system. Master's thesis, Linköping University, 2015.

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

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

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

O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo, and H. Pettersson. A motion planning and control framework for a self-driving truck and trailer system. *Journal of Field Robotics*, Under review.

J. Löfberg. *Minimax approaches to robust model predictive control*, volume 812. Linköping University Electronic Press, 2003.

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

C. Löfroth and G. Svensson. ETT – modulsystem för skogstransporter. Technical report, Skogforsk, 2012.

A. De Luca, G. Oriolo, and C. Samson. Feedback control of a nonholonomic car-like robot. In *Robot motion planning and control*, pages 171–253. Springer, 1998.

M. McNaughton, C. Urmson, J. M Dolan, and J-W. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Proceedings of the 2011 IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 4889–4895. IEEE, 2011.

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

M. M. Michałek. Cascade-like modular tracking controller for non-standard n-trailers. *IEEE Transactions on Control Systems Technology*, 25(2):619–627, March 2017. ISSN 1063-6536.

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

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

J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S Muhammad. Rrt*-smart: A rapid convergence implementation of rrt. *International Journal of Advanced Robotic Systems*, 10(7):299, 2013.

I. Nielsen. Structure-exploiting numerical algorithms for optimal control, 2017.

R. Oliveira, P. F Lima, M. Cirillo, J. Mårtensson, and B. Wahlberg. Trajectory generation using sharpness continuous dubins-like paths with applications in control of heavy duty vehicles. *arXiv preprint arXiv:1801.08995*, 2018.

B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016.

M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2172–2179, 2011.

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

M. G Plessen, D. Bernardini, H. Esen, and A. Bemporad. Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 26(1):38–50, 2018.

S. L Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.

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

R. Ravi, K. M Nagpal, and P. P Khargonekar. H-infinity control of linear time-varying systems: A state-space approach. *SIAM journal on control and optimization*, 29(6): 1394–1413, 1991.

J. B Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems*, 20 (3):38–52, 2000.

J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

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

W. J Rugh and J. S Shamma. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

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

C. Samson. Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1): 64–77, Jan 1995. ISSN 0018-9286.

I. Skog and P. Händel. In-car positioning and navigation technologies—a survey. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21, March 2009. ISSN 1524-9050. doi: 10.1109/TITS.2008.2011712.

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

M. W. Spong, S. Hutchinson, M. Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

P. B. Sujit, S. Saripalli, and J. B. Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems Magazine*, 34(1):42–59, Feb 2014. ISSN 1066-033X.

P. Švestka and J. Vleugels. Exact motion planning for tractor-trailer robots. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 2445–2450. IEEE, 1995.

S. Thrun. Toward robotic cars. *Commun. ACM*, 53(4):99–106, April 2010.

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

C. Urmson and R. Simmons. Approaches for heuristically biasing rrt growth. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1178–1183. IEEE, 2003.

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

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

R. S Wallace, A. Stentz, C. E Thorpe, H. P Moravec, W. Whittaker, and T. Kanade. First results in robot road-following. In *IJCAI*, pages 1089–1095. Citeseer, 1985.

M. Werling, L. Groll, and G. Bretthauer. Invariant trajectory tracking with a full-size autonomous road vehicle. *IEEE Transactions on Robotics*, 26(4):758–765, Aug 2010. ISSN 1552-3098.

M. Werling, S. Kammel, J. Ziegler, and L. Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

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

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

World Health Organization. Global status report on road safety. Technical report, October 2015.

X. Zhang, A. Liniger, and F. Borrelli. Optimization-based collision avoidance. *arXiv preprint arXiv:1711.03449*, 2017.

K. Zhou, J. Doyle, G. Comstock, Keith, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

J. Ziegler and C. Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *Proceedings of the 2019 IEEE/RSJ International conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE, 2009.

Part II

Publications

Publications

The publications associated with this thesis have been removed for copyright reasons. For more details about these see:

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-153892