Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments

Kristoffer Bergman
Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments

Kristoffer Bergman
Cover illustration: Motion planning in an unstructured environment, where a sampling-based motion planner is combined with receding horizon optimal control to compute an optimized trajectory.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
https://creativecommons.org/licenses/by-nc/4.0/

Linköping Studies in Science and Technology. Dissertations.
No. 2133

Exploiting Direct Optimal Control for Motion Planning in Unstructured Environments

Kristoffer Bergman

kristoffer.bergman@liu.se
www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping
Sweden

ISBN 978-91-7929-677-3 ISSN 0345-7524

Copyright © 2021 Kristoffer Bergman

Printed by LiU-Tryck, Linköping, Sweden 2021
To my family and friends!
Abstract

During the last decades, motion planning for autonomous systems has become an important area of research. The high interest is not the least due to the development of systems such as self-driving cars, unmanned aerial vehicles and robotic manipulators. The objective in optimal motion planning problems is to find feasible motion plans that also optimize a performance measure. From a control perspective, the problem is an instance of an optimal control problem. This thesis addresses optimal motion planning problems for complex dynamical systems that operate in unstructured environments, where no prior reference such as road-lane information is available. Some example scenarios are autonomous docking of vessels in harbors and autonomous parking of self-driving tractor-trailer vehicles at loading sites. The focus is to develop optimal motion planning algorithms that can reliably be applied to these types of problems. This is achieved by combining recent ideas from automatic control, numerical optimization and robotics.

The first contribution is a systematic approach for computing local solutions to motion planning problems in challenging unstructured environments. The solutions are computed by combining homotopy methods and direct optimal control techniques. The general principle is to define a homotopy that transforms, or preferably relaxes, the original problem to an easily solved problem. The approach is demonstrated in motion planning problems in 2D and 3D environments, where the presented method outperforms a state-of-the-art asymptotically optimal motion planner based on random sampling.

The second contribution is an optimization-based framework for automatic generation of motion primitives for lattice-based motion planners. Given a family of systems, the user only needs to specify which principle types of motions that are relevant for the considered system family. Based on the selected principle motions and a selected system instance, the framework computes a library of motion primitives by simultaneously optimizing the motions and the terminal states.

The final contribution of this thesis is a motion planning framework that combines the strengths of sampling-based planners with direct optimal control in a novel way. The sampling-based planner is applied to the problem in a first step using a discretized search space, where the system dynamics and objective function are chosen to coincide with those used in a second step based on optimal control. This combination ensures that the sampling-based motion planner provides a feasible motion plan which is highly suitable as warm-start to the optimal control step. Furthermore, the second step is modified such that it also can be applied in a receding-horizon fashion, where the proposed combination of methods is used to provide theoretical guarantees in terms of recursive feasibility, worst-case objective function value and convergence to the terminal state. The proposed motion planning framework is successfully applied to several problems in challenging unstructured environments for tractor-trailer vehicles. The framework is also applied and tailored for maritime navigation for vessels in archipelagos and harbors, where it is able to compute energy-efficient trajectories which complies with the international regulations for preventing collisions at sea.
Populärvetenskaplig sammanfattning

Ett av huvudmålen inom robotik och reglerteknik är att utveckla autonoma system. Dessa system har förmågan att tolka högnivåkommandon samt exekvera dem utan någon mänsklig interaktion. De autonoma systemen får bara en översiktlig beskrivning av vad uppgiften är, och det är sedan upp till dem att hitta lösningar till hur den ska genomföras. Några exempel på sådana system som utvecklas idag är självcörande bilar, drönare och industrirobotar. För att ett autonomt system ska kunna utföra sina uppgifter behöver flera olika tekniker integreras och samverka, såsom automatiserat resonerande, tolkning av omvärlden och reglering. Ett delproblem som behöver lösas är rörelseplanering (eng. *motion planning*), vilket handlar om att beräcka hur ett system ska förflytta sig mellan två punkter på ett säkert sätt. En viktig aspekt är att beräkningen tar hänsyn till hinder i den omgivande miljön och systemets begränsade möjligheter att förflytta sig. I denna avhandling är målet inte enbart att beräkna lösningar som tar hänsyn till systemens begränsningar, utan även att optimera ett användarvalt prestandamått. Några exempel på praktiskt användbara mått är minimering av energiförbrukning, sträcka, tid eller en kombination av dessa. Ur ett reglertekniskt perspektiv är det här problemet en typ av optimalt styrproblem (eng. *optimal control problem*). Fokus i den här avhandlingen ligger därför på att utveckla algoritmer för rörelseplaneringsproblem genom att kombinera metoder för att lösa optimala styrproblem med metoder från robotik och numerisk optimering.

Det första bidraget i den här avhandlingen är en algoritm som kombinerar en metod för att lösa olinjära ekvationssystem som kallas homotopi, med en metod anpassad för generella optimala styrproblem. I homotopimetoder löses en sekvens av delproblem. Först löses ett enkelt problem, sedan transformeras problemet stegvis till det ursprungliga, svåra problemet. Under transformationen utnyttjas lösningen från föregående delproblem som en startgissning till lösningen av nästa del.

Det andra bidraget är ett optimeringsramverk som ska användas i förberedande beräkningar till en planeringsmetod från robotik som kallas lattice-baserade rörelseplanerare (eng. *lattice-based motion planners*). Dessa planerare söker efter lösningar till rörelseplaneringsproblem genom att kombinera delrörelser från ett bibliotek av förberäknade rörelser. Tidigare har framtagandet av biblioteket inneburit ett tidskrävande arbete för en domänexpert, men med hjälp av det föreslagna ramverket kan det istället automatiseras.

Det sista bidraget är ett planeringsramverk som kombinerar metoder från robotik och optimal styrning för att lösa rörelseplaneringsproblem. Huvudidén bygger på att utnyttja styrkorna hos både lattice-baserade rörelseplanerare och algoritmer för optimal styrning. I ett första steg löser den lattice-baserade planeraren problemet i en diskretiserad sökrymd, där både prestandamåttet som optimeras samt rörelsemodellen är valda till att sammanfalla med det efterföljande optimala styrningssteget. Denna kombination säkerställer att den lattice-baserade planeraren tillhandahäller en lösning som är mycket lämplig att använda som utgångspunkt för det optimala styrningssteget. Det resulterande planeringsramverket applikeras med framgångsrika resultat på rörelseplaneringsproblem för lastbilar med släp samt fartyg.
Acknowledgments

First of all, I would like to thank my supervisor Assoc. Prof. Daniel Axehill for your continuous support and encouragement during the last years. Thank you for your never-ending enthusiasm, and for always motivating me to do my best! A special thanks to Dr. Oskar Ljungqvist for our fruitful research collaborations and discussions during the past years. Your ability to put research ideas into practice has been a true source of inspiration. Thank you Dr. Jonas Linder for providing an interesting research application, and for the research collaborations that this application resulted in. I would also like to thank my co-supervisor Prof. Torkel Glad for your assistance.

Thank you Prof. Svante Gunmarson, former head of Division at Automatic Control, and Assoc. Prof. Martin Enqvist, the current head of division. You both have done an excellent work to create a working environment that makes everyone feel welcome!

This thesis has been greatly improved by feedback from Lic. Daniel Arnström, Lic. Per Boström-Rost, Herman Ekwall, Anja Hellander, Lic. Fredrik Ljungberg and Dr. Oskar Ljungqvist. I really appreciate the time you spent on proofreading parts of this thesis. Furthermore, I thank Assoc. Prof. Gustaf Hendeby and Dr. Henrik Tidefelt for providing the template that has been used to write this thesis.

I would like to thank current and former colleagues at Automatic Control for contributing to such a friendly working atmosphere. Even though the last year has been different in many aspects, you have made fika breaks, lunch walks and other activities a pleasure! A special thanks to my officemate and friend Per Boström-Rost, for making time at the university and home office more fun, and for letting me beat you at squash and fantasy football! I also greatly appreciate the informal discussions over the past years where we have been able to exchange research ideas.

This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. WASP did not only contribute with funding, but also an invaluable network with people from all over the world. Thank you all as batch one students for making courses, conferences and international study trips more fun, and to all the WASP seniors who have arranged them! Furthermore, I would like to thank Saab AB for financing my studies, and my manager Lic. Torbjörn Crona for believing in me and giving the chance to take this fantastic opportunity.

Finally, I would like to thank family: my sister Emelie, my father Lars and my mother Maria, for your love and encouragement. The journey taking me to this point would not have been possible to accomplish without your support. Matilda, there are no words to describe how much you mean to me, but I will give it a try. You are both the love of my life and my best friend. Thank you for all the adventures we have shared together, and for your endless love and support. I look forward to spending the rest of my life with you.

Linköping, March 2021
Kristoffer Bergman
# Contents

Notation \hspace{1cm} xv  

## I Background  

### 1 Introduction  

1.1 Background and motivation ......................................... 3
1.2 Problem formulation .................................................. 3
1.3 Contributions ......................................................... 6
1.4 Thesis outline ......................................................... 7

### 2 Nonlinear optimization  

2.1 Nonlinear programming problem ................................... 13
2.2 Optimality conditions ............................................... 13
2.3 Algorithms for solving nonlinear equations ..................... 14
   2.3.1 Newton’s method ................................................ 14
   2.3.2 Continuation/homotopy methods ............................... 17
2.4 Algorithms for solving nonlinear programs ..................... 18
   2.4.1 Sequential quadratic programming ........................... 18
   2.4.2 Nonlinear interior point ..................................... 19
2.5 Globalization techniques ........................................... 20
   2.5.1 Merit function ................................................. 20
   2.5.2 Filter ........................................................... 20
   2.5.3 Line search .................................................... 21
   2.5.4 Trust region ................................................... 21

### 3 Optimal control  

3.1 Problem formulation .................................................. 23
3.2 Direct methods for optimal control ............................... 23
   3.2.1 Direct single shooting ....................................... 23
   3.2.2 Direct multiple shooting ..................................... 24
   3.2.3 Direct collocation ............................................ 24
3.3 Model predictive control ............................................ 24
4 Optimal motion planning
   4.1 Nonlinear system models .......................... 31
      4.1.1 Dynamics and kinematics ...................... 31
      4.1.2 Holonomic or nonholonomic .................. 32
   4.2 Problem formulation .............................. 32
   4.3 Sequential solution concept ...................... 33
   4.4 Sampling-based motion planning ................... 36
      4.4.1 Steering function ............................. 36
      4.4.2 Collision detection ............................ 38
      4.4.3 Expansion strategy ............................ 39
   4.5 Lattice-based motion planning .................... 41
      4.5.1 State-lattice construction ..................... 42
      4.5.2 Online planning ............................... 45

5 Concluding remarks ................................. 49
   5.1 Summary of contributions ......................... 49
   5.2 Conclusions .................................... 51
   5.3 Future work .................................... 52

Bibliography .................................... 53

II Publications

A Combining Homotopy Methods and Numerical Optimal Control to Solve Motion Planning Problems .......................... 63
   1 Introduction ..................................... 65
      1.1 Related work ..................................... 66
      1.2 Main contribution ................................ 67
   2 Optimal control preliminaries ..................... 68
      2.1 Solving the optimal control problem .................. 68
      2.2 Nonlinear programming formulation .................. 69
      2.3 Sequential quadratic programming .................. 69
   3 Continuation/homotopy methods ..................... 70
   4 Sequential homotopy quadratic programming ............. 72
   5 Obstacle classification ................................ 74
   6 Examples of obstacle quadratic programming ............. 75
   7 Numerical results ................................ 76
      7.1 Motion models ................................ 77
      7.2 Experimental results ........................ 78
   8 Conclusions and future work ......................... 81
   Bibliography .................................... 82

B Improved Optimization of Motion Primitives for Motion Planning in State Lattices .......................... 85
   1 Introduction ..................................... 87
   2 Preliminaries ................................... 89
## Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control

1 **Introduction** .................................................. 110
2 **Problem formulation** ........................................ 112
   2.1 The optimal path planning problem ....................... 112
   2.2 Lattice-based path planner ............................... 113
3 **Bilevel optimization** ........................................ 115
   3.1 Bilevel optimization problem reformulation ............. 117
   3.2 Analysis of solution properties using bilevel arguments ... 118
4 **Improvement using optimal control** ....................... 120
5 **Numerical results** ........................................... 122
   5.1 Vehicle models ............................................ 123
   5.2 State-space discretizations ............................. 124
   5.3 Maneuver specifications ................................. 124
6 **Conclusions and future work** ............................ 130

**Bibliography** .................................................. 131

---

## An Optimization-Based Receding Horizon Trajectory Planning Algorithm

1 **Introduction** .................................................. 137
2 **Problem formulation** ........................................ 139
3 **Receding horizon planning** ................................ 140
   3.1 Receding horizon planning formulation .................. 141
   3.2 Feasibility, optimality and convergence .................. 142
4 **A practical algorithm** ...................................... 144
   4.1 Solving the receding horizon planning problem .......... 145
   4.2 Adjusted receding horizon planning formulation ......... 145
   4.3 Feasibility, optimality and convergence ................ 146
   4.4 Algorithm ................................................ 147
5 **Simulation study** ............................................ 148
   5.1 Vehicle model ............................................ 149
   5.2 Lattice-based motion planner ............................ 149
   5.3 Simulation results ...................................... 150
6 **Conclusions and Future Work** ............................ 153

**Bibliography** .................................................. 155
### E  An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments  

| Section | Title | Page |
|---------|-------|------|
| 1 | Introduction | 160 |
| 2 | Ship modeling and problem formulation | 161 |
| 3 | Lattice-based motion planner | 164 |
| 3.1 | State-space discretization | 164 |
| 3.2 | Motion primitive computation | 165 |
| 3.3 | Collision avoidance and environmental cost | 166 |
| 3.4 | Resulting graph-search problem | 167 |
| 4 | Optimization-based improvement | 168 |
| 4.1 | Collision avoidance and environmental cost | 168 |
| 4.2 | Improvement optimization problem | 171 |
| 5 | Simulation study | 172 |
| 6 | Conclusions and Future Work | 175 |
| | Bibliography | 178 |

### F  A COLREGs-Compliant Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments  

| Section | Title | Page |
|---------|-------|------|
| 1 | Introduction | 184 |
| 2 | Ship modeling | 185 |
| 3 | COLREGs-compliant motion planning | 187 |
| 3.1 | COLREGs representation | 187 |
| 3.2 | Problem formulation | 190 |
| 4 | Optimization-based motion planner | 190 |
| 4.1 | Lattice-based motion planner | 191 |
| 4.2 | Receding-horizon improvement | 192 |
| 4.3 | Proposed CCMP algorithm | 194 |
| 5 | Simulation study | 195 |
| 6 | Conclusions and Future Work | 200 |
| | Bibliography | 202 |
## Notation

### Notations

| Notation | Meaning |
|----------|---------|
| $\mathbb{R}$ | Set of real numbers |
| $\mathbb{Z}$ | Set of integers |
| $\mathbb{Z}_{i,j}$ | Set of integers $\{i,i+1,\ldots,j-1,j\}$ |
| $[a,b]$ | Interval of real numbers $x \in \mathbb{R}$ such that $a \leq x \leq b$ |
| $z^T$ | Transpose of vector $z \in \mathbb{R}^n$ |
| $\nabla f$ | Gradient of the function $f : \mathbb{R}^n \to \mathbb{R}$ |
| $\nabla_z f$ | Gradient of the function $f(z,x) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ with respect to $z$ |
| $\nabla^2_{zz} f$ | Hessian of the function $f(z,y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ with respect to $z$ |
| $z \triangleq y$ | Definition of the vector $z$ as equal to $y$ |
| $z \succeq y$ | $z_i \geq y_i$ holds for each pair of components of $z$ and $y$ |
| $Q \succeq 0$ | Matrix $Q$ is positive semidefinite |
| $\|z\|_2$ | Euclidean norm of vector $z$ |
| $|S|$ | Cardinality of a set $S$ |
| $O(g(z))$ | Complexity growth dominated by $g(z)$ when $\|z\|_2 \to \infty$ |
## Abbreviations

| Abbreviation | Meaning |
|--------------|---------|
| BVP          | Boundary Value Problem |
| CCTOC        | Constrained Continuous-Time Optimal Control |
| HJB          | Hamilton-Jacobi-Bellman |
| HLUT         | Heuristic Look-Up Table |
| IP           | Interior Point |
| KKT          | Karush-Kuhn-Tucker |
| LICQ         | Linear Independence Constraint Qualification |
| MINLP        | Mixed-Integer Nonlinear Programming |
| MPC          | Model Predictive Control |
| NLP          | Nonlinear Programming |
| ODE          | Ordinary Differential Equation |
| OCP          | Optimal Control Problem |
| PMP          | Pontryagin’s Maximum Principle |
| RHP          | Receding Horizon Planning |
| RRT          | Rapidly-exploring Random Tree |
| SHQP         | Sequential Homotopy Quadratic Programming |
| SQP          | Sequential Quadratic Programming |
| SST          | Stable Sparse RRT |
| QP           | Quadratic Programming |
Part I

Background
In recent years, the development of autonomous systems has rapidly increased. Many autonomous systems need to be able to move from one location to another. The problem of planning such a movement is commonly referred to as a motion planning problem, where the objective is to find a feasible motion plan that brings a system from its current state to a terminal state while avoiding obstacles in the surrounding environment. The focus in this thesis is to develop algorithms that are able to solve such problems. This chapter gives an introduction to the field of motion planning, provides a summary of the contributions, and outlines the remaining parts of the thesis.

1.1 Background and motivation

One of the main goals in the fields of robotics and control is to develop systems that are autonomous. These systems should be able to interpret high-level descriptions of tasks and execute them without any human intervention. The high-level tasks only contain information of what to achieve, and it is up to the algorithms within the autonomous system to find solutions to the tasks. Today, autonomous systems are developed to be used in a wide variety of applications such as security and surveillance, aerospace, manufacturing, underwater operations, automotive and much more. For an autonomous system to work as intended, several technologies such as sensing and perception, automated reasoning and control need to be integrated [Latombe, 2012]. Within the research of these technologies, multiple subproblems arise that need to be solved, where one is motion planning.

A popular approach in classical motion planning has been to ignore the system dynamics during planning and instead focus on how to reach the desired terminal state by translating and rotating a body without colliding with obstacles. This problem is also referred to as the piano mover’s problem [LaValle, 2006]. For
Introduction

Figure 1.1: Examples of applications where motion planning is required.

classical motion planning, there are mainly two general solution approaches that have been used, namely combinatorial and sampling-based planning approaches. Combinatorial approaches compute graph representations of the free space that exactly represent the original problems, while sampling-based approaches sample the state space and conduct discrete searches using these samples. In recent years, motion planning for dynamical systems that are subject to differential constraints has become an important area of research. This is not the least due to the high interest in systems such as self-driving cars, unmanned aerial vehicles and robotic manipulators. Two applications involving this type of systems are illustrated in Figure 1.1.

This thesis addresses motion planning problems for dynamical systems that operate in unstructured environments. In such environments, no prior reference such as the center line of a lane on the road is available for the system. Instead, it is the responsibility of the motion planner to compute a reference in terms of a motion plan between an initial state and a desired terminal state. Important aspects are that the planned motion both has to satisfy the dynamical constraints imposed on the system and be safe to execute, i.e., not lead to collision with obstacles. Most of the successful motion planning methods for dynamical systems in unstructured environments are sampling-based, because little can be done using combinatorial methods when constraints on the motion of the system in terms of differential equations are present. Many sampling-based methods solve motion planning problems by concatenating sequences of motion segments, which are commonly referred to as motion primitives [LaValle, 2006]. These can be computed either online or offline depending on the choice of method. The main strength of sampling-based approaches is that they are good at solving combinatorial aspects of the problem, such as selecting on which side to pass obstacles. Furthermore, they can be designed such that they are compatible with any collision detection algorithm [Lindemann and LaValle, 2005]. The main drawback with these methods is that they suffer from the curse of dimensionality as the dimension of the search space to explore grows.

In this thesis, the objective is not to find any feasible motion plan between the initial and terminal state. Instead, the objective is to find solutions that also optimize a performance measure, e.g., solutions that minimize the traveled distance, duration, energy consumption or risk of collision with obstacles. When a performance measure is included in the motion planning problem formulation, the
term *optimal motion planning* problem is commonly used [LaValle, 2006]. From a control perspective, the optimal motion planning problem is an instance of an optimal control problem.

In optimal control theory, the fields of optimization and automatic control are combined to compute optimal control inputs to the system that is being controlled. It dates back to the development of calculus of variations, but regained interest during the 1960’s when it was successfully used for planning optimal trajectories in aerospace applications [Sargent, 2000]. There exist many different methods to solve an optimal control problem, where perhaps the most used one is numerical optimization. Many of these methods seek only a local solution, i.e., one where the performance measure is lower than for any other feasible nearby solution. The main reason is that a global solution is usually both difficult to characterize and to locate [Nocedal and Wright, 2006], and hence intractable to find in reasonable time. In this thesis, the search for a solution is limited to locally optimal solutions.

An increasingly used approach to compute optimized trajectories is to develop algorithms based on direct methods for optimal control, e.g., [Mellinger and Kumar, 2011, Schulman et al., 2014, Zhang et al., 2018]. Similar techniques have also been used to compute trajectories in a receding horizon fashion, such as the work presented in [Diehl et al., 2006, Gao et al., 2010, Patel and Goulart, 2011, Rosolia et al., 2017]. The increased interest is mainly due to

- the development of efficient and reliable optimization algorithms,
- increased computational resources, and
- the ability to systematically encode problem constraints such as restrictions on how the system can move and actuator limitations within an optimal control framework [Maciejowski, 2002].

The main difficulty with applying these methods to motion planning problems is that the corresponding problems usually are challenging nonconvex optimization problems. The performance of an optimization algorithm for solving such problems is strongly dependent on the provided initialization, i.e., where the algorithm starts the search for a solution [Nocedal and Wright, 2006]. Hence, there is a significant risk of computing a solution which is bad from a practical point of view, or not being able to find a solution at all, if the algorithms are naively applied to the motion planning problem using standard initialization strategies.
1.2 Problem formulation

The main focus of this thesis is to develop optimization-based algorithms that can reliably be applied to motion planning problems. In particular, this thesis deals with motion planning problems for nonlinear dynamical systems that operate in complex unstructured environments. The aim is to investigate and provide answers to the following questions:

- How can the strengths of sampling-based methods for motion planning and direct methods for optimal control be combined in a structured way?
- How can direct methods for optimal control be exploited to improve the generation of motion primitives for sampling-based motion planning?
- How can collision avoidance be ensured when solving motion planning problems with direct methods for optimal control?

1.3 Contributions

The main contributions of this thesis are within the area of optimal motion planning for complex dynamical systems operating in cluttered and unstructured environments. The results are mainly achieved by combining ideas from automatic control, numerical optimization and robotics. All developed methods are evaluated in extensive simulation studies on different types of dynamical systems including cars, tractor-trailer vehicles and marine vessels. In summary, the contributions of this thesis are:

- A systematic approach for solving motion planning problems, where a homotopy method and direct methods for optimal control are combined to solve problems with challenging geometry of the feasible set (Paper A).
- A framework for automatic generation of motion primitives for lattice-based motion planners. Given a parameterized family of systems, the user only needs to specify which types of so-called maneuvers that are relevant for the considered system family. These maneuvers can then be reused for all system instances. (Paper B)
- A unified optimization-based path planning approach, where lattice-based path planning and direct methods for optimal control are tightly combined to solve advanced path planning problems (Paper C).
- An optimization-based receding horizon trajectory planning algorithm with theoretical guarantees in terms of recursive feasibility, worst-case objective function value and convergence to the desired terminal state (Paper D). The algorithm is also tailored and extended to account for the specific needs of marine vessels (Paper E).
- An optimization-based motion planner for autonomous vessels that complies with the international regulations for preventing collisions at sea (COLREGS) (Paper F).
1.4 Thesis outline

This thesis is divided into two parts, where Part I contains background material and Part II contains edited versions of the published papers.

Part I – Background

The first part introduces the optimal motion planning problem and presents relevant theoretical background material, which includes nonlinear optimization, optimal control and motion planning. The main purpose of this part is to provide a theoretical foundation for the publications presented in Part II.

Part II – Publications

The second part of this thesis contains edited versions of the publications listed below. Here, a summary and background for each publication is given.

Paper A: Combining Homotopy Methods and Numerical Optimal Control to Solve Motion Planning Problems

Kristoffer Bergman and Daniel Axehill. Combining homotopy methods and numerical optimal control to solve motion planning problems. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 347–354, 2018.

This paper received the best application paper award at the conference.

Summary: In Paper A, it is shown how a state-of-the-art method from numerical optimal control can be combined with a homotopy method into a method that solves motion planning problems with challenging geometry of the feasible set. The method is combined with a novel structure-exploiting constraint parameterization introduced in this paper. It is illustrated that the proposed method manages to efficiently solve problems where a traditional state-of-the-art numerical optimal control method fails to compute any solution at all. Furthermore, it significantly outperforms an optimizing sampling-based motion planner, both in terms of computational performance as well as trajectory quality.

Background: This work was initiated through discussions between the author of this thesis and Daniel Axehill. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations and writing the manuscript. Daniel contributed with technical discussions and reviewed the manuscript.
Paper B: Improved Optimization of Motion Primitives for Motion Planning in State Lattices

Kristoffer Bergman, Oskar Ljungqvist, and Daniel Axehill. Improved optimization of motion primitives for motion planning in state lattices. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2307–2314, 2019.

Summary: In Paper B, it is shown how direct methods for optimal control can be used to improve the motion primitive generation for a lattice-based motion planner. The procedure of manually specifying the connectivity is both time consuming and non-trivial since it is heavily application dependent. In this paper, a framework that automates this procedure is proposed. Based on a number of user-defined so-called maneuvers, the framework simultaneously selects an optimized connectivity for the system under consideration and computes optimal motions to obtain the motion primitive set. The maneuvers can be reused to compute the motion primitive set for any instance in a parameterized family of systems, e.g., cars, buses or trucks parameterized by platform dimensions and actuator limitations.

Background: This work was initiated through discussions between the author of this thesis, Oskar Ljungqvist and Daniel Axehill. The author of this thesis contributed with the majority of the work including the development of the motion primitive generation framework, numerical evaluations and writing the manuscript. Daniel and Oskar contributed with technical discussions and reviewed the manuscript.

Paper C: Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control

Kristoffer Bergman, Oskar Ljungqvist, and Daniel Axehill. Improved path planning by tightly combining lattice-based path planning and optimal control. *IEEE Transactions on Intelligent Vehicles*, 6(1):57–66, 2021a.

Summary: In Paper C, a lattice-based path planner and direct methods for optimal control are combined to compute locally optimal solutions to path planning problems. It is shown how the solution from a lattice-based path planner can be improved by initializing a direct optimal control solver that efficiently solves the problem to local optimality. The main idea behind the proposed approach is that the lattice-based path planner solves the combinatorial aspects (e.g., on which side to pass obstacles) while direct optimal control is used to improve the continuous aspects of the solution keeping the combinatorial part fixed. It is shown in numerical examples that the proposed approach is able to compute solutions with generally lower objective function value compared to previously used initialization strategies or lattice-based planners alone. Furthermore, the approach results in reduced computation times with reliable convergence of the optimal control solver.
**Background:** This work was initiated through discussions between the author of this thesis, Oskar Ljungqvist and Daniel Axehill. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations and writing the manuscript. Daniel and Oskar contributed with technical discussions, assisted in the development of the theoretical derivations and reviewed the manuscript.

**Paper D: An Optimization-Based Receding Horizon Trajectory Planning Algorithm**

Kristoffer Bergman, Oskar Ljungqvist, Torkel Glad, and Daniel Axehill. An optimization-based receding horizon trajectory planning algorithm. In *Proceedings of 21st IFAC World Congress*, 2020a.

**Summary:** Paper D presents an optimization-based receding horizon trajectory planning algorithm for dynamical systems operating in unstructured and cluttered environments. The proposed approach is a two-step procedure, where a motion planning algorithm is used in the first step to find a feasible, but suboptimal, trajectory. This initial trajectory is then improved in the second step based on receding horizon optimization, which performs local trajectory refinement over a sliding time window. The second step uses the initial trajectory in a novel way to provide theoretical guarantees in terms of recursive feasibility, objective function value, and convergence.

**Background:** This work was initiated through discussions between the author of this thesis, Oskar Ljungqvist and Daniel Axehill, as an extension to the work presented in Paper C. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations and writing the manuscript. Torkel Glad contributed mainly in the development of the theoretical derivations in the paper. Daniel and Oskar contributed with technical discussions, assisted in the development of the theoretical derivations and reviewed the manuscript.

**Paper E: An Optimization-Based Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments**

Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axehill. An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5283–5290, 2020b.

**Summary:** In Paper E, a two-step optimization-based motion planner is proposed for autonomous maneuvering of ships in constrained environments such as harbors. The motion planner is based on the algorithm in Paper D, where a lattice-based motion planner is used in the first step. Both steps of the algorithm use a high-fidelity model of the ship to plan feasible and energy-efficient trajectories. Moreover, a novel algorithm is proposed for automatic computation of spatial safety envelopes.
around the trajectory computed by the lattice-based planner. These safety envelopes are used in the second step to obtain collision-avoidance constraints which complexity scales well with an increased number of obstacles. Finally, the planner is evaluated with successful results in a simulation study for autonomous docking problems in a model of the Cape Town harbor.

**Background:** This work was initiated through discussions between the author of this thesis, Oskar Ljungqvist, Jonas Linder and Daniel Axehill. The idea was to extend and tailor the work in Paper D for the specific needs of ships. The author of this thesis contributed with the majority of the work including theoretical derivations, implementations, evaluations and writing the manuscript. Jonas provided the nonlinear model of the ship, reviewed the manuscript and contributed with technical discussions. Daniel and Oskar contributed with technical discussions and reviewed the manuscript.

**Paper F: A COLREGs-Compliant Motion Planner for Autonomous Maneuvering of Marine Vessels in Complex Environments**

Kristoffer Bergman, Oskar Ljunqvist, Jonas Linder, and Daniel Axehill. A COLREGs-compliant motion planner for autonomous maneuvering of marine vessels in complex environments. Under review for possible publication in IEEE Journal of Oceanic Engineering. Available at arXiv: https://arxiv.org/abs/2012.12145v2, 2021b.

**Summary:** An enabling technology for future sea transports is safe and energy-efficient autonomous maritime navigation in narrow environments with other marine vessels present. In Paper F, a two-step optimization-based motion planner is proposed that obeys the rules specified in the international regulations for preventing collisions at sea (COLREGs). The motion planner is based on the work presented in Paper E. To comply with the rules specified in COLREGs, the lattice-based planner is augmented with discrete states that represent what type of COLREGs situations that are active with respect to other nearby vessels. The applicability of the proposed motion planner is demonstrated in a simulation study, where it computes energy-efficient and COLREGs-compliant trajectories. The results also show that the motion planner is able to prevent complex collision situations from occurring.

**Background:** The idea to this work was initiated through discussions between the author of this thesis, Oskar Ljungqvist, Jonas Linder and Daniel Axehill. Throughout the process, the author of this thesis and Oskar maintained a tight collaboration. The implementation was accomplished jointly between the author of this thesis and Oskar. The author of this thesis contributed with writing the manuscript, including theoretical derivations and numerical evaluations. Oskar, Jonas and Daniel contributed with technical discussions and reviewed the manuscript.
Other publication

The following additional publication has been co-authored by the author of this thesis:

Oskar Ljungqvist, Kristoffer Bergman, and Daniel Axehill. Optimization-based motion planning for multi-steered articulated vehicles. In Proceedings of 21st IFAC World Congress, 2020.
Nonlinear optimization algorithms are the basis of many methods to solve optimal control problems. This chapter contains a brief overview of the field and introduces some of the most important concepts in the area. The material in this chapter is inspired by [Nocedal and Wright, 2006] and [Boyd and Vandenberghe, 2004], which are extensive references to nonlinear and convex optimization.

## 2.1 Nonlinear programming problem

Consider an optimization problem given in the form:

\[
\begin{align*}
\text{minimize} \quad & L(z) \\
\text{subject to} \quad & f_i(z) \leq 0, \quad i \in \mathcal{I} \\
& g_j(z) = 0, \quad j \in \mathcal{E},
\end{align*}
\]

where \( z \in \mathbb{R}^n \) is the optimization variable, \( L : \mathbb{R}^n \rightarrow \mathbb{R} \) is the objective function, \( f_i : \mathbb{R}^n \rightarrow \mathbb{R}, \ i \in \mathcal{I} \) and \( g_j : \mathbb{R}^n \rightarrow \mathbb{R}, \ j \in \mathcal{E} \) are smooth functions representing the inequality and equality constraint functions, respectively. Finally, \( \mathcal{I} \) with cardinality \( |\mathcal{I}| = p \) and \( \mathcal{E} \) with cardinality \( |\mathcal{E}| = m \) are finite sets of indices with \( \mathcal{I} \cap \mathcal{E} = \emptyset \). This type of optimization problem is referred to as a nonlinear programming (NLP) problem if at least one of the involved functions is nonlinear. An important subclass of optimization problems in the form of (2.1) is the class of convex optimization problems, which has several special properties. Such problems are used in many areas such as automatic control systems, estimation and signal processing, statistics and finance [Boyd and Vandenberghe, 2004]. Furthermore, they are also used within many algorithms for solving general NLPs [Nocedal and Wright, 2006]. To introduce the notion of a convex optimization problem, the following definitions are required:
Definition 2.1 (Feasible set). The feasible set $\Omega$ is defined as the set $\Omega := \{z \in \mathbb{R}^n | f_i(z) \leq 0, i \in \mathcal{I}; g_j(z) = 0, j \in \mathcal{E}\}$.

Definition 2.2 (Convex set). A set $\Omega$ is convex if for any two points $z_1, z_2 \in \Omega$ and any $\theta \in [0, 1]$ it holds that
\[
\theta z_1 + (1 - \theta) z_2 \in \Omega.
\] (2.2)

Definition 2.3 (Convex function). A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if its domain $\mathcal{D} \subseteq \mathbb{R}^n$ is a convex set and if for all pairs of $z_1, z_2 \in \mathcal{D}$ and any $\theta \in [0, 1]$ it holds that
\[
f(\theta z_1 + (1 - \theta) z_2) \leq \theta f(z_1) + (1 - \theta) f(z_2).
\] (2.3)

The optimization problem in (2.1) is referred to as a convex optimization problem, if the objective function $L$ and the inequality constraint functions $f_i$ are all convex functions and the equality constraint functions $g_j$ are affine [Boyd and Vandenberghe, 2004].

2.2 Optimality conditions

In this section, first and second-order optimality conditions for $z^* \in \Omega$ to be a local minimizer of (2.1) are stated. To be able to pose these optimality conditions, the Lagrangian of the problem in (2.1) needs to be defined, which is given by:
\[
\mathcal{L}(z, \lambda, \nu) \triangleq L(z) + \sum_{i \in \mathcal{I}} \lambda_i f_i(z) + \sum_{j \in \mathcal{E}} \nu_j g_j(z).
\] (2.4)

The Lagrangian is an augmented representation of the objective function which contains weighted sums of the constraint functions. The weights are called Lagrange multipliers, or dual variables, and are given by the vector $\lambda$ with components $\lambda_i, i \in \mathcal{I}$ for the inequality constraint functions and the vector $\nu$ with components $\nu_j, j \in \mathcal{E}$ for the equality constraint functions. Furthermore, the active set at a point $z$ is defined as:

Definition 2.4 (Active set). The active set $\mathcal{A}(z)$ at a feasible point $z \in \Omega$ of (2.1) consists of the constraint indices corresponding to the constraints that hold with equality [Nocedal and Wright, 2006], i.e.,
\[
\mathcal{A}(z) = \{ i \in \mathcal{I} | f_i(z) = 0, \} \cup \mathcal{E}.
\]

Finally, the linear independence constraint qualification (LICQ) is given by:

Definition 2.5 (Linear independence constraint qualification). Given a point $z$ and the corresponding active set $\mathcal{A}(z)$, the linear independence constraint qualification (LICQ) holds if the gradients of the active constraints,
\[
\nabla f_i(z), i \in \mathcal{A}(z), \quad \nabla g_j(z), j \in \mathcal{A}(z),
\]
are linearly independent.
With these definitions, it is possible to pose the first-order optimality conditions as [Nocedal and Wright, 2006]:

**Theorem 2.1.** Suppose that \( z^* \) is a local solution to (2.1), that the functions \( L, f_i \) and \( g_j \) in (2.1) are continuously differentiable, and that LICQ holds at \( z^* \). Then, there exist Lagrange multiplier vectors \( \lambda^*, \nu^* \) with components \( \lambda^*_i \in \mathbb{R} \) and \( \nu^*_j \in \mathbb{R} \), such that the following conditions are satisfied at \( (z^*, \lambda^*, \nu^*) \):

\[
\begin{align*}
\nabla_z \mathcal{L}(z^*, \lambda^*, \nu^*) &= 0, \\
f_i(z^*) &\leq 0, \quad i \in \mathcal{I} \\
g_j(z^*) &= 0, \quad j \in \mathcal{E} \\
\lambda^*_i &\geq 0, \quad i \in \mathcal{I} \\
\lambda^*_i f_i(z^*) &= 0, \quad i \in \mathcal{I}.
\end{align*}
\]

**Proof:** See Section 12.4 in [Nocedal and Wright, 2006].

The conditions in Theorem 2.1 are known as the Karush-Kuhn-Tucker (KKT) conditions and are necessary conditions for local optimality of (2.1). In the special case of a convex optimization problem, these conditions are not only necessary for a locally optimal solution, but also sufficient for a globally optimal solution. The conditions in (2.5e) are the complementarity conditions, which imply that either an inequality constraint is active, i.e., \( f_i(z^*) = 0 \), or that the corresponding Lagrange multiplier \( \lambda^*_i \) is zero, or possibly both [Nocedal and Wright, 2006]. The following special case of complementarity will be used later in this thesis:

**Definition 2.6 (Strict complementarity).** Given a local solution \( z^* \) to (2.1) and Lagrange multipliers satisfying the KKT conditions (2.5), the strict complementarity condition is said to hold if exactly one of \( \lambda^*_i \) and \( f_i(z^*) \), \( i \in \mathcal{I} \) is zero in (2.5e).

The following definition classifies the state of the inequality constraints according to the value of the corresponding Lagrange multiplier:

**Definition 2.7 (Strongly active constraint).** Let \( z^* \) be an optimal solution to the optimization problem in (2.1), where the KKT conditions are satisfied with Lagrange multipliers \( \lambda^* \) associated with the inequality constraints \( f_i, \ i \in \mathcal{I} \). A constraint \( f_i(z) \) is then said to be strongly active if \( f_i(z^*) = 0 \) and \( \lambda^*_i > 0 \) [Nocedal and Wright, 2006].

To be able to pose the second-order optimality conditions, the set of linearized feasible directions is required. It is defined as:

**Definition 2.8 (Linearized feasible directions).** Given a point \( z \in \Omega \) and the corresponding active set \( \mathcal{A}(z) \) (as defined in Definition 2.4), the set of linearized feasible directions \( \mathcal{F}(z) \) is

\[
\mathcal{F}(z) = \{ d \in \mathbb{R}^n \mid d^\top \nabla g_j(z) = 0, \ \forall j \in \mathcal{E}; \ d^\top \nabla f_i(z) \leq 0, \ \forall i \in \mathcal{A}(z) \cap \mathcal{I} \}.
\]
feasible direction \( w \in F(z^*) \) will result in either an increased or constant first-order approximation of the objective function value, i.e., \( w^T \nabla L(z^*) \geq 0 \). For directions where \( w^T \nabla L(z^*) = 0 \), first-order information is not sufficient to determine if the objective function value decreases for a nonconvex optimization problem. With second-order information, it is sometimes possible to resolve this issue [Nocedal and Wright, 2006]. In the following theorem, sufficient second-order conditions are given to ensure that \( z^* \) is a local solution to (2.1):

**Theorem 2.2.** Suppose that for some feasible \( z^* \in \Omega \) of (2.1), there are Lagrange multipliers \( (\lambda^*, \nu^*) \) such that the KKT conditions in (2.5) are satisfied. Furthermore, suppose that

\[
w^T \nabla^2_{zz} L(z^*, \lambda^*, \nu^*) w > 0, \quad \forall w \in \mathcal{C}(z^*, \lambda^*), \ w \neq 0.
\]  

Then, \( z^* \) is a strict local solution to (2.1), i.e., there exists a neighborhood \( N \) of \( z^* \) such that \( L(z) > L(z^*) \) for all \( z \in N \cap \Omega, \ z \neq z^* \).

**Proof:** See Section 12.5 in [Nocedal and Wright, 2006].

In Theorem 2.2, \( \mathcal{C}(z^*, \lambda^*) \) represents the critical cone which is defined as

\[
\mathcal{C}(z^*, \lambda^*) \triangleq \{ w \in F(z^*) | w^T \nabla f_i(z^*) = 0, \forall i \in A(z^*) \text{ with } \lambda^*_i > 0 \}.
\]  

(2.8)

From this definition and (2.5a) in the KKT conditions, it can be concluded that the critical cone is the set of directions from \( F(z^*) \) where first-order derivative information is insufficient to determine if the objective function value will increase or decrease [Nocedal and Wright, 2006].

### 2.3 Algorithms for solving nonlinear equations

The basis of many methods for solving NLP problems in the form of (2.1) is to be able to solve a system of nonlinear equations defined by:

\[
r(z) = 0,
\]

(2.9)

where the function \( r : \mathbb{R}^n \to \mathbb{R}^n \) defines the system of equations. A vector \( z^* \) is called a solution, or root, if (2.9) is satisfied for \( z^* \), i.e., if \( r(z^*) = 0 \) [Nocedal and Wright, 2006]. This section outlines two methods that can be used to find such solutions.

#### 2.3.1 Newton’s method

The perhaps most well-known method for solving systems of nonlinear equations is Newton’s method. It is an iterative method that searches locally for a solution to (2.9) by constructing a linear model around the current iterate \( z_k \). By letting \( J(z) \in \mathbb{R}^{n \times n} \) denote the Jacobian of \( r \), a linear model of \( r \) locally around \( z_k \) is:

\[
M_k(p) = r(z_k) + J(z_k)p,
\]

(2.10)
where \( p \in \mathbb{R}^n \) is the search direction. Since the aim is to find a root to (2.9), the standard Newton’s method computes the next iterate \( z_{k+1} \) by the following steps:

\[
\begin{align*}
    p_k &= -J(z_k)^{-1}r(z_k), \\
    z_{k+1} &= z_k + p_k,
\end{align*}
\]

which is repeated until \( r(z_{k+1}) = 0 \). Note that (2.11a) corresponds to solving \( M_k(p) = 0 \). This method is shown to converge fast to a solution, under the assumption that the initial iterate \( z_0 \) is sufficiently close to a solution. There exist several variants of the standard Newton’s method where the rate of convergence is improved. However, Newton-based methods are not guaranteed to converge to a solution to (2.9) if \( J(z) \) is singular in the region of interest [Nocedal and Wright, 2006].

### 2.3.2 Continuation/homotopy methods

One way to solve difficult root-finding problems in the form of (2.9) is to use so-called homotopy, or continuation methods [Allgower and Georg, 2012]. Instead of solving the original problem directly, one can set up an easy problem for which the solution is more or less obvious, and then successively transform the problem towards the original and more difficult one. This is most easily illustrated using an example:

---

**Example 2.1: Homotopy method**

If the problem is to find a solution to the nonlinear system of equations in (2.9), the homotopy method instead focuses on finding a solution to the homotopy map

\[
h(z, \gamma) = \gamma r(z) + (1 - \gamma)\hat{r}(z),
\]

where \( \gamma \in [0, 1] \) is the *homotopy parameter*, and \( \hat{r} : \mathbb{R}^n \to \mathbb{R}^n \) is a user-defined function. The basic idea is to solve a sequence of \( n + 1 \) problems \( h(z, \gamma_k) = 0 \), for \( \gamma_k = \gamma_0 < \gamma_1 < \ldots < \gamma_n = 1 \). If solutions are relatively easy to find for \( \hat{r}(z) \), it should be easy to find a first iterate \( z_0 \). Then for any problem \( k \in [0, n - 1] \), given a starting point \( z_k \) that is the solution to \( h(z, \gamma_k) = 0 \), the next solution \( z_{k+1} \) is calculated by solving \( h(z, \gamma_{k+1}) = 0 \). This can be done using an iterative method such as Newton’s method, with \( z_k \) as initial iterate. The method is able to compute the next solution if the difference between \( \gamma_k \) and \( \gamma_{k+1} \) is small enough. In the last iteration, a solution to \( h(z, 1) = r(z) = 0 \) is found, i.e., to the original system of equations [Allgower and Georg, 2012].

---

The trajectory of points for which \( h(z, \gamma) = 0 \) is called the *homotopy path* or *zero path*. In many problems, it is not sufficient to monotonically increase the homotopy parameter \( \gamma \) to transform a solution from \( h(z, 0) = 0 \) to \( h(z, 1) = 0 \), e.g., the homotopy path in Figure 2.1. In such cases, predictor-corrector path-following methods can be used, which instead aim at solving

\[
h(z(s), \gamma(s)) = 0.
\]
Here, $s \in \mathbb{R} \geq 0$ is a variable that represents the arc length along the homotopy path [Nocedal and Wright, 2006].

2.4 Algorithms for solving nonlinear programs

In general, a local solution to an NLP problem is found using iterative schemes. The major difference between different iterative solvers is how the search direction and step length are computed. When computing local solutions to general NLP problems in the form of (2.1), there are mainly two families of optimization methods that are used: sequential quadratic programming (SQP) methods and nonlinear interior-point (IP) methods. They both aim at finding a (locally optimal) solution which satisfies the KKT conditions in Theorem 2.1, but approach the problem in different ways [Nocedal and Wright, 2006]. This section gives a brief introduction to these methods.

2.4.1 Sequential quadratic programming

One of the most used methods for solving general NLPS are SQP methods, which are based on solving quadratic subproblems iteratively. In a basic SQP method, the search direction $p_k \in \mathbb{R}^n$ is computed by approximating the NLP in (2.1) by a quadratic program (QP) at iterate $(z_k, \lambda_k, \nu_k)$. The iterate is given by the optimization variable $z_k$ and the Lagrange multipliers $(\lambda_k, \nu_k)$ of the last iterate. By linearizing the constraint functions and constructing a quadratic approximation of the objective function in (2.1) at this iterate, the following QP is obtained [Nocedal

Figure 2.1: A one-dimensional example of a homotopy path which can not be followed by monotonically increasing the homotopy parameter $\gamma$ from 0 to 1.
and Wright, 2006]:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} p_k^T B_k p_k + \nabla L(z_k)^T p_k \\
\text{subject to} & \quad \nabla f(z_k)^T p_k + f(z_k) \preceq 0, \\
& \quad \nabla g(z_k)^T p_k + g(z_k) = 0.
\end{align*}
\]

Here, \( f : \mathbb{R}^n \to \mathbb{R}^p \) and \( g : \mathbb{R}^n \to \mathbb{R}^m \) are introduced to compactly represent the inequality and equality constraint functions from (2.1), respectively. \( B_k \) denotes the approximation of the Hessian of the Lagrangian to the \text{nlp} in (2.1). For an exact Newton method, the exact Hessian is used, i.e., \( B_k = \nabla^2_{zz} L_k \). The Jacobians of the inequality and equality constraint functions, respectively, are given by \( \nabla f \) and \( \nabla g \). They are defined as:

\[
\begin{align*}
\nabla g(z) &= [\nabla g_1(z), \nabla g_2(z), \ldots, \nabla g_m(z)], \\
\nabla f(z) &= [\nabla f_1(z), \nabla f_2(z), \ldots, \nabla f_p(z)].
\end{align*}
\]

Finally, \( \nabla L \) is the gradient of the objective function in (2.1).

Due to the equivalence between \text{SQP} and Newton’s method, the search direction \( p_k \) can either be defined as the solution to the \text{QP} in (2.14), or as the search direction generated by Newton’s method for the corresponding \text{NLP} problem (2.1) for any fixed working set (i.e., the set of constraints that are imposed to hold with equality while all others are ignored) [Nocedal and Wright, 2006]. The following theorem assures under which conditions the \text{SQP} method is able to correctly identify the optimal active set:

**Theorem 2.3.** Suppose that \( z^* \) is a local solution to (2.1). At this point, the \text{KKT} conditions in Theorem 2.1 are satisfied for some Lagrange multipliers \((\lambda^*, \nu^*)\). Furthermore, suppose that \text{LICQ} (Definition 2.5), the strict complementary condition (Definition 2.6) and the second-order sufficient conditions (Theorem 2.2) hold at \((z^*, \lambda^*, \nu^*)\). Then, if the iterate \((z_k, \lambda_k, \nu_k)\) is sufficiently close to \((z^*, \lambda^*, \nu^*)\), there is a local solution to the \text{SQP}-subproblem (2.14) whose active set is the same as the active set \( A(z^*) \) of the \text{NLP} in (2.1).

**Proof:** See proof of Theorem 2.1 in [Robinson, 1974].

Once the optimal active set is identified (and not changed in subsequent iterations), what remains is basically Newton’s method on a nonlinear system of equations (see Section 2.3.1) with fast convergence to the solution. Furthermore, far from the solution, an \text{SQP} method is usually able to improve the estimate of the optimal active set and thus guide the iterates toward the solution. A major advantage with \text{SQP} methods based on active-set strategies is that they can be efficiently warm started if a good initial iterate is available. This can be useful if, e.g., a sequence of similar problems are to be solved [Nocedal and Wright, 2006].

### 2.4.2 Nonlinear interior point

Most nonlinear \text{IP} methods associate the problem of solving (2.1) with the following barrier problem:
minimize \( L(z) - \mu \sum_{i \in \mathcal{I}} \log(s_i) \)
subject to \( f_i(z) + s_i = 0, \quad i \in \mathcal{I} \)
\( g_j(z) = 0, \quad j \in \mathcal{E}, \)

where the inequality constraints from (2.1) are reformulated as equality constraints by introducing non-negative slack variables \( s_i \geq 0 \) with \( s_i \in \mathbb{R} \). The barrier parameter \( \mu \) is given by a positive scalar, and \( \log(\cdot) \) represents the natural logarithm function. The inequality constraints \( s_i \geq 0 \) are not explicitly included in the problem since the additional term \( \mu \sum \log s_i \) will force the components \( s_i \) from getting close to zero.

The solution to the barrier problem in (2.16) does not exactly coincide with the solution to the actual problem to be solved in (2.1), due to the modified objective function. The barrier approach instead aims at finding solutions to (2.16) for a sequence of positive barrier parameters \( \{\mu_k\} \), with \( \mu_k \to 0 \) for increasing values of \( k \). Hence, in the limit, the problem is equivalent to (2.1). Due to this, interior-point methods can also be seen as homotopy methods, where \( \mu \) represents the homotopy parameter. The term interior point is derived from that early barrier methods did not introduce slack variables, but instead used

\[ L(z) - \mu \sum_{i \in \mathcal{I}} \log(f_i(z)), \]

as objective function to prevent iterates from leaving the feasible region. However, this formulation requires an initial iterate \( z_0 \) that is feasible with respect to the inequality constraints. Most modern nonlinear IP methods can start from any initial iterate and remain interior only with respect to the constraints \( s_i \geq 0 \) [Nocedal and Wright, 2006].

An advantage of using IP methods instead of SQP methods is that the combinatorial aspect of identifying the correct active set at the optimal solution is avoided. Hence, the linear system to be solved at each iteration always has the same structure and structure-exploiting techniques can easily be applied. Due to this property, IP methods usually outperform SQP methods on large-scale problems [Nocedal and Wright, 2006]. A drawback is that they are not warm-start friendly due to numerical sensitivities for small perturbations on (2.16) when \( \mu \) is close to zero [Nocedal and Wright, 2006].

### 2.5 Globalization techniques

In this section, some approaches to increase the area of convergence are briefly outlined. It should not be confused with global optimization, which aims at finding the global minimizer of a problem. To measure the progress of iterations, either merit functions or filter techniques are applied. The aim with these methods is to handle the trade-off between decreasing the objective function value and reducing the constraint violation. To ensure that progress is made in every iteration, either
line-search or trust-region methods are used. The use of these methods is important for enabling NLP algorithms to converge from initializations that are not in the neighborhood of a local solution [Nocedal and Wright, 2006].

2.5.1 Merit function

A merit function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a function that produces a scalar measure of progress for the iterates computed during the search for a solution to (2.1). For an unconstrained optimization problem, the trivial choice of merit function is the objective function itself. When constraints are present, the merit function should also consider the constraint violation. One commonly used choice of merit function is the $\ell_1$ penalty function given by:

$$
\phi(z) = L(z) + \sigma \left( \sum_{i \in I} [f_i(z)]^+ + \sum_{j \in E} |g_j(z)| \right), 
$$

(2.18)

where $[x]^+ = \max(0, x)$ [Nocedal and Wright, 2006]. The scalar $\sigma$ represents the penalty parameter which determines the trade-off between the importance of reducing constraint violation or decreasing the objective function value. One important property of this choice of merit function is that it is an exact merit function, i.e., that there exists a large enough $\sigma$ such that a local minimizer to (2.18) is also a local solution to the corresponding NLP in (2.1) [Nocedal and Wright, 2006]. Thus, in a standard descent procedure, it is required that $\phi(z_{k+1}) < \phi(z_k)$. Otherwise the step is rejected or modified using either a line search procedure or a trust region, which are further described below in this section.

2.5.2 Filter

Filter techniques separate the goals of minimizing the objective function value and the constraint violation. By defining a measure of infeasibility according to:

$$
a(z) = \sum_{i \in I} [f_i(z)]^+ + \sum_{j \in E} |g_j(z)|, 
$$

(2.19)

this separation can be written as the multi-objective optimization problem:

$$
\text{minimize } L(z) \quad \text{and} \quad \text{minimize } a(z).
$$

(2.20)

A pair of the objective function value and measure of the constraint violation $(L(z_l), a(z_l))$ for a point $z_l$ is said to dominate another pair $(L(z_k), a(z_k))$ if both $L(z_l) \leq L(z_k)$ and $a(z_l) \leq a(z_k)$. An iterate $z_k$ is thus accepted by the filter if the pair $(L(z_k), a(z_k))$ is not dominated by any other pair that have previously been generated and added to the filter. If $z_k$ is accepted, all pairs that are dominated by $z_k$ in the filter are removed. To obtain good practical performance, several enhancements to this simple filter are needed, which are thoroughly described in [Nocedal and Wright, 2006].
2.5.3 Line search

An algorithm that employs a line-search strategy starts the solution procedure by computing a search direction $p_k$ using, e.g., one of the optimization algorithms described in Section 2.4. Then, the algorithm searches along this direction from the current iterate $z_k$ to a new iterate $z_{k+1} = z_k + \alpha_k p_k$ with a lower objective function (or merit function) value. Most line-search algorithms require that the search direction is a descent direction, which ensures that it is possible to reduce the objective function (or merit function) value in this direction. Thus, the problem of computing a step length $\alpha_k \in \mathbb{R}$ becomes a one-dimensional optimization problem:

$$
\min_{\alpha_k > 0} L(z_k + \alpha_k p_k). \tag{2.21}
$$

Solving (2.21) exactly is usually too expensive. Instead, an inexact backtracking line-search procedure is often used. The procedure starts by evaluating the full step with $\alpha_k = 1$. If this is unsuccessful, $\alpha_k$ is iteratively reduced until a sufficient decrease in the objective function is obtained. One popular approach to guarantee sufficient decrease is to ensure that the Wolfe conditions are satisfied for the candidate step length $\alpha_k$. The Wolfe conditions are satisfied when the following inequalities hold:

$$
L(z_k + \alpha_k p_k) \leq L(z_k) + c_1 \alpha_k \nabla L(z_k)^T p_k, \tag{2.22a}
$$

$$
\nabla L(z_k + \alpha_k p_k)^T p_k \geq c_2 \nabla L(z_k)^T p_k, \tag{2.22b}
$$

where $c_1$ and $c_2$ are scalar parameters such that $0 < c_1 < c_2 < 1$ [Nocedal and Wright, 2006].

2.5.4 Trust region

In trust-region methods, an additional constraint is added to the optimization problem which limits the current step $p_k$ such that $z_{k+1}$ is in a region sufficiently close to the previous iterate $z_k$. Within this region, the subproblem to be solved at $z_k$ is a good approximation of the actual NLP in (2.1), which means that the computed step can be trusted. Thus, in contrast to line-search methods, trust-region methods choose both the search direction and the step size simultaneously [Nocedal and Wright, 2006]. Using the QP subproblem in (2.14) as an example, an SQP algorithm that employs a trust region instead solves the following optimization problem at $z_k$:

$$
\begin{align*}
\minimize_{p_k} & \quad \frac{1}{2} p_k^T B_k p_k + \nabla L(z_k)^T p_k \\
\text{subject to} & \quad \nabla f(z_k)^T p_k + f(z_k) \preceq 0, \\
& \quad \nabla g(z_k)^T p_k + g(z_k) = 0, \\
& \quad \|p_k\|_2 \leq \Delta_k,
\end{align*} \tag{2.23}
$$

where the scalar $\Delta_k > 0$ is called the trust-region radius. To obtain fast and reliable convergence, the trust-region radius $\Delta_k$ is typically modified after each iteration such that progress can be ensured [Nocedal and Wright, 2006].
Optimal control theory deals with the problem of computing optimal control inputs to a dynamical system such that a user-defined performance measure is minimized. In this chapter, the general continuous-time optimal control problem (OCP) is posed, and some commonly used solution strategies are discussed with focus on direct methods. Finally, an introduction to model predictive control is given.

### 3.1 Problem formulation

The traditional optimal control problem is to compute an optimal control input to a dynamical system such that an objective function is minimized, subject to initial and terminal constraints [Speyer and Jacobson, 2010]. By also enforcing the solution to satisfy certain inequality constraints, the problem is a constrained continuous-time optimal control problem, and is defined in its general form as:

$$\begin{align*}
\text{minimize} & \quad \Phi(x(t_f)) + \int_{t_i}^{t_f} \ell(x(t), u(t)) \, dt \\
\text{subject to} & \quad x(t_i) = x_s, \\
& \quad \dot{x}(t) = f(x(t), u(t)), \quad t \in [t_i, t_f] \\
& \quad c(x(t), u(t)) \leq 0, \quad t \in [t_i, t_f] \\
& \quad \Psi(x(t_f)) = x_f,
\end{align*}$$

(3.1)

where $x_s \in \mathbb{R}^n$ and $x_f \in \mathbb{R}^n$ are the initial and terminal states, $x(\cdot)$ is the state trajectory, $u(\cdot)$ is the control-input trajectory, $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is the function that describes the system model, $c : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$ represents the path constraints and contains all the $p$ inequality constraint functions that determine the feasible region. Moreover, $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\ell : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ form the objective function.
and $\Psi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the terminal constraint function. The final time $t_f$ may, or may not, be an optimization variable [Speyer and Jacobson, 2010]. For notational convenience, the term optimal control problem (OCP) is used when referring to a problem of this type further on in this thesis.

The solution to the OCP in (3.1) is a state and control-input trajectory

$$(x(t), u(t)), \quad t \in [t_i, t_f]$$

where the state trajectory is given by:

$$x(t) = x(t_i) + \int_{t_i}^{t} f(x(\tau), u(\tau)) d\tau, \quad t \in [t_i, t_f].$$ (3.2)

The OCP in (3.1) can be solved by applying any of the three approaches Hamilton-Jacobi-Bellman (HJB), indirect methods or direct methods [Diehl et al., 2006] (as illustrated in Figure 3.1). In the first approach, the principle of optimality is used to obtain a partial differential equation in state space, known as the HJB equation, which represents the continuous counterpart to the discrete-time dynamic programming approach. However, due to the curse of dimensionality, this approach is only applicable for small-scale problems [Diehl et al., 2006]. Indirect methods use the necessary conditions of optimality of the OCP in (3.1), described by Pontryagin’s maximum principle (PMP), to obtain a boundary value problem (BVP) that is given by a set of ordinary differential equations (ODEs) [Bryson and Ho, 1975]. In general, this BVP needs to be solved numerically, which makes it common to refer to this method as a “first optimize, then discretize” approach. Finally, direct methods solve the infinite-dimensional OCP by first reformulating it into a finite-dimensional NLP problem. After this reformulation, it is possible to use any of the optimization methods described in Chapter 2 to solve the problem, making it a “first discretize, then optimize” approach [Diehl et al., 2006].

In this thesis, direct methods for solving OCPs are used. The main reasons for this choice of method are that indirect formulations are often hard to solve in practice [Betts, 2010], and that there exist several software packages for solving the reformulated NLP problem with these methods, such as ACADO [Houska et al., 2011],

**Figure 3.1:** Different approaches to solve constrained continuous-time optimal control problems such as (3.1) [Diehl et al., 2006].
3.2 Direct methods for optimal control

In direct methods for solving OCPs as the one in (3.1), the infinite number of decision variables are finitely parameterized such that it is possible to transform the problem in (3.1) into an NLP. In this section, a brief description of three well-known direct methods are given: direct single shooting, direct multiple shooting and direct collocation. The notation and material in this section is inspired by [Diehl et al., 2006].

3.2.1 Direct single shooting

All shooting methods use an embedded ODE solver which makes it possible to rewrite, or even eliminate, the constraints of the continuous-time dynamics in (3.1) such that they can be handled by a general NLP solver. This is done by parameterizing the control signal $u(t)$. One of the most widely used approaches is letting the control signal be constant on each phase of a uniform discretization of time with $\{t_k\}_{k=0}^N$, i.e., $t_k = t_0 + k\Delta t$ and $t_0 = t_i$, where $\Delta t$ is an optimization variable. This parameterization can be written as:

$$u(t; z) = u_k, \quad t \in [t_k, t_{k+1}]$$  \hspace{1cm} (3.3)

with $z = \{u_0, u_1, \ldots, u_{N-1}\}$.

In direct single shooting the continuous-time dynamics constraints are removed by forward integration of the system, starting at the initial state $x_s$ and applying the parameterized control input in (3.3), which results in the state trajectory $x(t; z)$. Thus, the OCP in (3.1) can be posed as:

$$\min_{z, \Delta t} \Phi(x(t_N; z)) + \int_{t_i}^{t_f} \ell(x(t; z), u(t; z))dt$$  \hspace{1cm} (3.4)

subject to $g(x(t_k; z), u(t_k; z)) \preceq 0, \quad k \in \mathbb{Z}_{0,N}$

$$\Psi(x(t_N; z)) = x_f,$$

where both the state trajectory and the integral in the objective function can be computed using numerical integration techniques such as Runge-Kutta methods [Betts, 2010]. Hence, it is possible to use state-of-the-art ODE solvers with an adaptive step size for error control [Diehl et al., 2006]. The problem is now in the form of a standard NLP and can be solved, e.g., using any of the methods in Section 2.4. The main drawbacks with this method are that the ODE solution $x(t; z)$ can be severely nonlinear with respect to $z$, and that unstable systems are difficult to handle due to numerical sensitivities [Diehl et al., 2006].
3.2.2 Direct multiple shooting

In direct multiple shooting, the problem is formulated as a collection of \( N \) phases. By assuming, as in the case of single shooting, piecewise constant control signals \( u_k \), the ODE is in multiple shooting separately solved on each interval, starting from an artificial initial state value \( s_k \):

\[
\begin{align*}
\dot{x}_k(t) &= f(x_k(t), u_k), \quad t \in [t_k, t_{k+1}] \\
x_k(t_k) &= s_k.
\end{align*}
\]

This approach produces trajectory sequences \( x_k(t; s_k, u_k) \) for each phase, where the semicolon indicates that the trajectories are determined by the initial state values \( s_k \) and the constant control input \( u_k \). The NLP decision variables are thus given by \( z = \{s_0, u_0, s_1, u_1, \ldots, s_N\} \) [Diehl et al., 2006], which gives the following NLP formulation:

\[
\begin{align*}
\text{minimize} \quad \Phi(s_N) + \sum_{k=0}^{N-1} \ell_k(s_k, u_k) \\
\text{subject to} \quad s_0 &= x_s, \\
s_{k+1} &= f_k(s_k, u_k), \quad k \in \mathbb{Z}_{0:N-1} \\
g(s_k, u_k) &\preceq 0, \quad k \in \mathbb{Z}_{0:N} \\
\Psi(s_N) &= x_f.
\end{align*}
\]

Here, the constraints \( s_{k+1} = f_k(s_k, u_k) \triangleq x_k(t_{k+1}; s_k, u_k) \) are the so-called continuity constraints that connect the subsequent phases. The objective function is here given by a sum of the numerically computed integrals for each phase [Diehl et al., 2006]. One obvious difference in multiple shooting compared to single shooting is that the dimension of the problem increases. However, sparsity in the direct multiple shooting formulation can be utilized for faster computations. Some of the benefits of using direct multiple shooting instead of direct single shooting are that it is possible to initialize both the state and control-input trajectories, it is more numerically stable such that unstable systems can be better handled and that it can robustly handle path and terminal constraints [Betts, 2010].

3.2.3 Direct collocation

Another widely used approach is the direct collocation method. In this method, the OCP is, similarly as in the shooting methods, discretized on a grid \( \{t_k\}_{k=0}^N \). Here, each interval \( [t_k, t_{k+1}] \), \( t_{k+1} - t_k = \Delta t \) is denoted a collocation interval. On each collocation interval, \( m \) collocation points \( \{t_k^{(1)}, \ldots, t_k^{(m)}\} \) are chosen and the state trajectory is approximated by a polynomial:

\[
p_k(t; v_k) = \sum_{i=0}^{m} v_{k,i} t^i,
\]

where \( v_k \in \mathbb{R}^m \) is the coefficient vector. At the collocation points, equality constraints are added to ensure that the derivative of the polynomial approximation
of the state trajectory coincides with the function \( f(x,u) \) that describes the system model. By again representing the initial state values with \( s_k \), and assuming that the control signal \( u_k \) is constant on each collocation interval, these constraints are given by:

\[
s_k = p_k(t_k; v_k) = \sum_{i=0}^{m} v_{k,i} t^i, \quad (3.8a)
\]

\[
f(p_k(t_k^{(1)}; v_k), u_k) = p'_k(t_k^{(1)}; v_k) = \sum_{i=1}^{m} iv_{k,i}(t_k^{(1)})^{i-1}, \quad (3.8b)
\]

\[
\vdots
\]

\[
f(p_k(t_k^{(m)}; v_k), u_k) = p'_k(t_k^{(m)}; v_k) = \sum_{i=1}^{m} iv_{k,i}(t_k^{(m)})^{i-1}. \quad (3.8c)
\]

By letting the constraints on each collocation interval in (3.8) be compactly represented as \( b_k(s_k, v_k, u_k) = 0 \), and the decision variables be represented by:

\[
z = \{s_0, v_0, u_0, \ldots, s_{N-1}, v_{N-1}, u_{N-1}, s_N\}, \quad (3.9)
\]

the following NLP is obtained:

\[
\begin{align*}
\text{minimize} & \quad \Phi(s_N) + \sum_{k=0}^{N-1} \ell_k(s_k, v_k, u_k) \\
\text{subject to} & \quad s_0 = x_s, \\
& \quad s_{k+1} = p_k(t_{k+1}; v_k), \quad k \in \mathbb{Z}_{0,N-1} \\
& \quad b_k(s_k, v_k, u_k) = 0, \quad k \in \mathbb{Z}_{0,N-1} \\
& \quad g(s_k, u_k) \preceq 0, \quad k \in \mathbb{Z}_{0,N} \\
& \quad \Psi(s_N) = x_f.
\end{align*} \quad (3.10)
\]

where \( \ell_k(s_k, v_k, u_k) \) denotes the numerical approximation of the objective function on interval \( k \). The obtained NLP is a large, sparse problem where sparsity exploiting solvers can be used for faster computations. The major drawback is that the NLP needs to be re-gridded if adaptive discretization error control should be used. This means that the dimension of the NLP is required to be changed, which is not the case for single and multiple shooting since they can use adaptive ODE solvers [Diehl et al., 2006].

### 3.3 Model predictive control

After (3.1) is solved, e.g., by using any of the techniques in Section 3.2, the computed control-input trajectory \( u(\cdot) \) can be used to control the system towards the terminal state \( x_f \). This approach is commonly referred to as **open-loop control**. However, due to modeling errors and unforeseen external disturbances, the system
will most likely not reach the desired terminal state when the open-loop control law is applied.

In model predictive control (MPC), a closed-loop control law is evaluated point-wise by repeatedly solving an OCP in a receding horizon fashion over a user-defined prediction horizon $T$. Due to this, MPC is sometimes referred to as receding horizon control. At each time instant $t_k$, the MPC control law is computed for an estimated (or measured) current state $\hat{x}(t_k)$ by solving the following OCP:

$$\min_{x_k(t), u_k(t)} \Phi_T(x_k(t_k + T)) + \int_{t_k}^{t_k+T} \ell_{\text{MPC}}(x_k(t), u_k(t)) \, dt$$

subject to

$$\dot{x}_k(t) = f(x_k(t), u_k(t)), \quad t \in [t_k, t_k + T]$$

$$c(x_k(t), u_k(t)) \leq 0, \quad t \in [t_k, t_k + T]$$

$$x_k(t_k + T) \in \mathcal{X}_f,$$

where $\Phi_T : \mathbb{R}^n \rightarrow \mathbb{R}$ is the terminal cost function and $\ell_{\text{MPC}} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ is the stage cost function, which are both minimized by the controller. Furthermore, $\mathcal{X}_f \subseteq \mathbb{R}^n$ is the terminal constraint set. After solving (3.11), only the initial part of the obtained control input trajectory, i.e., $u(t) = u^*_k(t)$, $t_k \leq t < t_{k+1}$, is executed. Then, the procedure of solving (3.11) is repeated at the next time instant $t_{k+1}$ with an updated estimate of the current state $\hat{x}(t_{k+1})$. This receding horizon closed-loop control strategy makes it possible to suppress both modeling errors and various external disturbances acting on the system [Maciejowski, 2002].

The selection of the cost functions and the terminal constraint set in (3.11) are crucial to ensure stability properties in terms of recursive feasibility and convergence for the controller [Mayne et al., 2000]. If a reference trajectory $(\hat{x}(\cdot), \hat{u}(\cdot))$ is available, a popular choice is to define $\hat{x}_k(t) = x_k(t) - \bar{x}(t)$ and $\hat{u}_k(t) = u_k(t) - \bar{u}(t)$ and use the following cost functions in the controller:

$$\ell_{\text{MPC}}(\hat{x}_k(t), \hat{u}_k(t)) = \|\hat{x}_k(t)\|_Q^2 + \|\hat{u}_k(t)\|_R^2,$$  

$$\Phi_T(\hat{x}_k(t_k + T)) = \|\hat{x}_k(t_k + T)\|_P^2.$$  

(3.12a)  

(3.12b)

Here, $Q \succeq 0$, $R > 0$ and $P \succeq 0$ are design matrices. One popular choice of $P$ is to use the solution of the continuous-time algebraic Riccati equation when the system is linearized at a working point $(\bar{x}_w, \bar{u}_w)$. In this case, the terminal cost will represent an approximation of the remaining cost-to-go for an infinite-horizon linear quadratic (LQ) controller [Findeisen and Allgöwer, 2002]. To ensure stability, one approach is to select $\mathcal{X}_f$ such that the system is sufficiently close to the working point $(\bar{x}_w, \bar{u}_w)$ at the end of the horizon [Mayne et al., 2000].

Instead of tracking a reference trajectory, it is possible to use the same performance measure as in the original problem formulation (3.1), i.e., to let

$$\ell_{\text{MPC}}(x_k, u_k) = \ell(x_k, u_k).$$  

(3.13)

This is commonly referred to as economic MPC, which was first introduced as a control strategy within the process industry [Rawlings and Amrit, 2009]. The main
idea is that the desired control objective is not necessarily to stabilize the system at a given set-point or around a pre-computed reference trajectory, but rather to optimize economic considerations such as the energy consumption. A survey on different selections of terminal constraints and modifications of the cost functions to ensure stability for an economic MPC controller are given in, e.g., [Faulwasser et al., 2018].
The meaning of the term *optimal motion planning* depends on the field of subject and the applications [LaValle, 2006]. In this thesis, the term refers to the problem of finding a feasible and locally optimal trajectory for a nonlinear system from an initial state to a terminal state in an environment that includes obstacles. In the literature, there is a vast variety of different methods that can be used to solve motion planning problems and produce feasible or optimized trajectories [Latombe, 2012, LaValle, 2006, Paden et al., 2016]. This chapter introduces the optimal motion planning problem and defines some important terms within the field. Furthermore, a commonly used sequential solution concept for motion planning problems is presented, which is followed by an overview of sampling-based motion planning techniques.

### 4.1 Nonlinear system models

The function $f(x,u)$ that describes the system model in (3.1) can have several different properties which influence the difficulty of finding a solution to the motion planning problem. In this section, some important system model characteristics in motion planning for nonlinear systems are briefly discussed.

#### 4.1.1 Dynamics and kinematics

Kinematics is the branch of classical mechanics which describes the motion of a system without considering the forces and accelerations that cause a motion [Murray et al., 1994]. Hence, a system model is in this thesis denoted as *kinematic* if it is derived based on velocity constraints only. On the other hand, the field of dynamics studies how applied forces and torques affect the motion of a system, such as Newton’s second law of motion [Murray et al., 1994]. Hence, it is reasonable...
that a dynamic model should consider these forces and torques. However, in the
definition of *kinodynamic motion planning* [Donald et al., 1993], it is also used to
denote that limitations on higher-order derivatives, such as accelerations and/or
jerks, are considered in the motion planning phase [LaValle, 2006]. Using this
definition, a system model is denoted *dynamic* in this thesis if any higher-order
derivatives are considered in the system model.

### 4.1.2 Holonomic or nonholonomic

Holonomic and nonholonomic properties are related to what type of constraints on
the velocity of the system that are present. A constraint is said to be holonomic
(or integrable) if it restricts the attainable states of the system to a smooth hy-
persurface, and otherwise the constraint is said to be nonholonomic. This implies
that holonomic constraints can be converted to constraints that do not involve any
time derivatives of the system states [Murray et al., 1994]. One example of a non-
holonomic system is a car. The car is unable to move sideways or rotate in place,
but it is well known that it is possible to park a car at any position and orienta-
tion. Thus, the constraints imposed on the car are nonholonomic since the system
is able to attain any feasible state, even though the motion is *locally* restricted.
For a detailed description of holonomic and nonholonomic properties, the reader is
referred to [Murray et al., 1994]. Planning the motion for a nonholonomic system
is often a complicated task due to the locally restricted motion capability [Li and
Canny, 2012].

### 4.2 Problem formulation

In this thesis, continuous-time nonlinear systems in the form

\[
\dot{x}(t) = f(x(t), u(t)), \quad x(t_i) = x_s, \tag{4.1}
\]

are considered, where \(x_s\) is the initial state at time \(t_i\), while \(x \in \mathbb{R}^n\) and \(u \in \mathbb{R}^m\)
denote the state and control input of the system, respectively. These are subject
to the following constraints:

\[
x \in \mathcal{X} \subseteq \mathbb{R}^n, \quad u \in \mathcal{U} \subseteq \mathbb{R}^m, \tag{4.2}
\]

where \(\mathcal{X}\) and \(\mathcal{U}\) describe the physical constraints on the states and controls, which
typically are described by convex sets. Furthermore, the system should not collide
with obstacles in the surrounding environment, where an obstacle can be either
static or dynamic. The region occupied by static obstacles is \(\mathcal{X}_{\text{obst}}^s\), and the time-
varying region for dynamic obstacles is \(\mathcal{X}_{\text{obst}}^d(t)\). Thus, the complete obstacle
region is defined as the union of static and dynamic obstacles

\[
\mathcal{X}_{\text{obst}}(t) = \mathcal{X}_{\text{obst}}^s \cup \mathcal{X}_{\text{obst}}^d(t) \subset \mathbb{R}^n. \tag{4.3}
\]

Using this definition, the state space is constrained as:

\[
x(t) \in \mathcal{X}_{\text{free}}(t) = \mathcal{X} \setminus \mathcal{X}_{\text{obst}}(t). \quad (4.4)
\]
This constraint is in general non-convex since $\mathcal{X}_{\text{free}}$ is defined as the complement set of $\mathcal{X}_{\text{obst}}$.

The optimal motion planning problem can now be defined as the problem of computing a feasible (i.e., satisfying (4.1)-(4.4)) state and control input trajectory $(x(\cdot), u(\cdot))$ that moves the system from $x_s \in \mathcal{X}_{\text{free}}(t_i)$ to a desired terminal state, $x_f \in \mathcal{X}_{\text{free}}(t_f)$, while a performance measure $J$ is minimized. This problem can be posed as the following continuous-time OCP:

$$\begin{align*}
\text{minimize} & \quad J = \int_{t_i}^{t_f} \ell(x(t), u(t)) dt \\
\text{subject to} & \quad x(t_i) = x_s, \quad x(t_f) = x_f, \quad (4.5) \\
& \quad \dot{x}(t) = f(x(t), u(t)), \quad t \in [t_i, t_f] \\
& \quad x(t) \in \mathcal{X}_{\text{free}}(t), \quad t \in [t_i, t_f] \\
& \quad u(t) \in \mathcal{U}, \quad t \in [t_i, t_f].
\end{align*}$$

In some applications, the equality constraint for the terminal state is represented by a terminal region $x(t_f) \in \mathcal{X}_f$ [LaValle, 2006], but in this thesis the focus is on problems where it is desired to reach the terminal state exactly.

Since the motion planning problem (4.5) is an OCP, it can be solved using optimal control techniques described in Chapter 3. The advantages of using optimal control are that state and control constraints can easily be incorporated in the problem formulation, and that it is straightforward to change the nonlinear model of the system and the performance measure (i.e., objective function), and to define and update problem parameters. In recent years, the popularity of these methods have increased for real-time online motion planning due to increased computational resources and the development of robust nonlinear optimization algorithms [Bitar et al., 2020, Martinsen et al., 2019, Schulman et al., 2014, Zhang et al., 2020, Ziegler et al., 2014].

The main challenge with optimization-based approaches for solving (4.5) is that the constraint $x(t) \in \mathcal{X}_{\text{free}}(t)$ introduces the combinatorial aspect of selecting which side to pass an obstacle. Hence, there usually exist several possible classes of solutions which makes the choice of initialization strategy crucial for a reliable and fast convergence. Another challenge to handle is the nonlinear system model, which further complicates the problem. This combination makes the motion planning OCP (4.5) in general hard to solve by directly invoking a solver based on direct methods for optimal control. Instead, motion planning algorithms are often used to compute a suboptimal solution [LaValle, 2006]. In the following sections, some commonly used motion planning approaches are discussed.

### 4.3 Sequential solution concept

To compute a suboptimal solution to the problem in (4.5), many motion planning algorithms divide the problem into three steps: path planning, path optimization and velocity planning [LaValle, 2006] which are solved sequentially. The inputs to
the motion planner are a terminal state provided by a higher-level task planner and a description of the environment provided by the perception and localization subsystem. Based on these inputs, the motion planner computes a motion plan following the three steps illustrated in Figure 4.1.

The path planning step consists of computing a collision-free path from the initial state to the terminal state. By introducing the path parameter \( s \geq 0 \), where \( s \) represents the progression along the path, this step traditionally aims at computing a path \( x(s) \) using a classical planning algorithm. In this step, the constraints on the motion of the system are usually neglected and instead handled in later steps. In the second step, the path from the path planning step is transformed, or smoothened, such that the path satisfies constraints on the system based on, e.g., wheel motions for a nonholonomic vehicle [LaValle, 2006]. In the path planning and optimization steps, dynamic obstacles are disregarded since time is not considered in these steps. The final step is performed by the velocity planner. Since the direction of motion is already decided by the previous steps, what remains for the velocity planner is to compute a speed profile \( \dot{s}(t) \) along each path segment. It is computed such that the system does not collide with any dynamic obstacles while minimizing some performance measure, e.g., related to comfort or smoothness, and also respecting velocity and acceleration constraints. The relation between a path and a trajectory can thus be described as follows

**Definition 4.1 (Path trajectory relation).** A path is represented as \((x(s), u(s)), s \in [0, s_g] \) and a trajectory is a time-parameterized path \((x(s(t)), u(s(t))), t \in [t_i, t_f] \).

In the literature, several approaches to solve these steps have been suggested. One approach is to use B-splines or Bezier curves to compute smooth paths for differentially flat systems, either to smoothen a sequence of waypoints, for example generated by a classical path planner [Piazzi et al., 2007, Yang and Sukkarieh,
or as steering functions within a sampling-based motion planner [Banzhaf et al., 2018, Oliveira et al., 2018a]. The use of these methods is computationally efficient since the model of the system can be described analytically. However, these methods are not applicable to non-flat systems, such as, e.g., many truck and trailer systems [Rouchon et al., 1993]. Furthermore, it is difficult to optimize the path with respect to a general performance measure. Another popular method, which is used in this thesis, is to formulate the problem as an OCP and use direct methods for optimal control to transform the problem to an NLP. Due to non-convex constraints that represent the obstacles and the nonlinear system model, a well-informed initialization strategy is crucial to converge to a good local optimum [Nocedal and Wright, 2006]. A straightforward initialization strategy is to use linear interpolation between the initial state and terminal state [Rao, 2009]. This can however often lead to convergence issues in cluttered environments, which is shown in both Paper A and [Zhang et al., 2018]. A more sophisticated initialization strategy is to use the solution from a sampling-based path planner. In this case, the solution from the path planning step is used as initialization from where the NLP solver can converge to a locally optimal solution. In previous work in the literature, the vehicle models used by the path planner for initialization are simplified; either they disregard the system dynamics completely (classical motion planning) [Campos-Macías et al., 2017, LaValle, 2006], or partially (respecting only kinematic constraints) [Andreasson et al., 2015, Zhang et al., 2018]. The use of a classical motion planning initialization strategy is shown in [Zhang et al., 2020] to cause problems for vehicles with nonholonomic constraints (illustrated here in Figure 4.2). Initializations based on a planned path using a kinematic model will in general be infeasible in the actual problem to be solved, and potentially not continuously transformable into a feasible solution which will make it impossible for the NLP solver to converge to a feasible solution [Stoneman and Lampariello, 2014]. Furthermore, the objective function in the sampling-based planner can only consider states that are represented in the chosen simplified model description, which might cause it to find an initialization far from a local minimum.

A unified motion planning algorithm that handles all of the involved steps simultaneously is desirable, since the sequential approach often leads to suboptimal solutions [LaValle, 2006]. The next section provide a detailed description of sampling-based motion planning, which is a widely used motion planning approach for dynamical systems.
4.4 Sampling-based motion planning

The perhaps most commonly used approach for solving motion planning problems is to apply sampling-based motion planners. This section outlines a general framework which corresponds to a generalization of sampling-based algorithms used for classical motion planning where the system dynamics and kinematics are neglected [LaValle, 2006]. In general, the sampling-based algorithms search for a solution to (4.5) by constructing a directed graph $G = (\mathcal{V}, \mathcal{E})$ incrementally online, where a vertex $x_k \in \mathcal{V}$ represents a state of the system and an edge $e_k \in \mathcal{E}$ represents a motion that satisfies all the constraints imposed on the system given in (4.5). A general sampling-based motion planning algorithm is outlined in Algorithm 4.1. In the following sections, some parts of the algorithm are described in detail.

4.4.1 Steering function

The main difference between Algorithm 4.1 and classical sampling-based motion planning algorithms is that a steering function (step 3) is required to be able to compute a feasible motion between two nearby states. The output from the steering function is a feasible motion $e_k$ from $x_k$ to $x_{k+1}$ with an associated cost $J(e_k)$. This cost is used in the graph search to compute the cost to reach states from $x_s$ and is in the end used to determine the optimal solution from $x_s$ to $x_f$. For general nonlinear systems and performance measures, the steering function
Algorithm 4.1: Sampling-based motion planning

1: **Initialization**: Let \( G = (V, E) \) represent a directed graph, where \( V \) contains the initial state \( x_s \) and \( E \) is empty.
2: **Expansion**: Choose a vertex \( x_k \in V \) to expand and select a desired system state \( x_{k+1} \in X_{\text{free}}(t) \) to connect with \( x_k \).
3: **Steering function**: Try to compute a collision-free and dynamically feasible motion \( e_k \) from \( x_k \) to \( x_{k+1} \). If this stage fails, return to step 2.
4: **Insertion**: Insert \( x_{k+1} \) in \( V \) and \( e_k \) in \( E \) if \( x_{k+1} \) is not a duplicate, or if \( x_{k+1} \in V \) and the current cost-to-come to \( x_{k+1} \) is higher than the candidate cost. Otherwise, return to step 2.
5: **Solution check**: If \( x_{k+1} = x_f \), mark the candidate trajectory from \( x_s \) to \( x_f \) as a feasible solution to the problem.
6: **Termination check**: If a termination condition is satisfied, return the solution with the lowest cost. Otherwise, return to step 2.

requires the solution of an OCP in the form of

\[
\begin{align*}
\text{minimize} & \quad \int_{t_k}^{T} \ell(x(t), u(t))\,dt \\
\text{subject to} & \quad x(t_k) = x_k, \quad x(T) = x_{k+1}, \\
& \quad \dot{x}(t) = f(x(t), u(t)), \quad t \in [t_k, T] \\
& \quad x(t) \in X_{\text{free}}(t), \quad t \in [t_k, T] \\
& \quad u(t) \in U, \quad t \in [t_k, T],
\end{align*}
\tag{4.6}
\]

for each pair of states \((x_k, x_{k+1})\) to connect. This subproblem is often easier to solve than the complete problem (4.5) since the distance between \( x_k \) and \( x_{k+1} \) is typically much shorter than the distance between \( x_s \) and \( x_f \). Many steering functions neglect the collision avoidance constraint \( x(t) \notin X_{\text{obst}}(t) \) and instead rely on a separate collision detection step, which is described in Section 4.4.2. This further simplifies the OCP to be solved, and makes the sampling-based motion planning algorithms compatible with any representation of the environment [LaValle, 2006]. However, unnecessary computations will occur when trajectories that collide with obstacles are selected.

One approach for general nonlinear systems is to use a steering function that only approximately solves (4.6). This is done by selecting a discrete set of available actions in an offline phase, i.e., before attempting to solve the motion planning problems. The resulting set of trajectories is called the motion primitive set, where the motion primitives represent dynamically feasible motions. When a state \( x_k \) has been selected for expansion, these motion primitives can be used by the steering function to, e.g., bring the system as close as possible to \( x_{k+1} \) when being applied from \( x_k \). Some prior work where motion primitives have been used within sampling-based motion planners are [Frazzoli et al., 2002, Howard and Kelly, 2007, Ljungqvist et al., 2019, Pivtoraiko et al., 2009]. A related approach
is to let a closed-loop controller act as the steering function, where the objective for the controller is to steer the system towards the next state $x_{k+1}$. This is done by forward simulation of the system, either for a pre-determined duration or until the system is sufficiently close to $x_{k+1}$ [Kuwata et al., 2008]. As a consequence, this method can reliably be applied to unstable systems, where the controller is used to stabilize the system [Evestedt et al., 2016]. The main drawback is that the use of these types of inexact steering functions limits the sampling-based algorithm to find solutions within a terminal region, i.e., $x_f \in \mathcal{X}_f$, since it can not be guaranteed that it is possible to reach the terminal state $x_f$ precisely.

For linear systems without any other imposed constraints on the state and control signal, PMP can be applied to derive analytical solutions to the steering function OCP (4.6) [Werling et al., 2012]. Analytical solutions also exist for some nonlinear systems, such as shortest path solutions for car-like vehicles based on Dubins [Dubins, 1957] or Reeds-Shepp [Reeds and Shepp, 1990] car models. Recently, these results have been extended to account for higher-order differentiability on the path curvatures such as the work in [Fraichard and Scheuer, 2004, Oliveira et al., 2018b]. Furthermore, there exist analytical solutions for differentially flat systems, such as multirotors [Mellinger and Kumar, 2011], where it is possible to compute polynomial functions in the flat outputs to generate candidate solutions between two states [LaValle, 2006]. The main benefit of these methods is that the steering function can be computed fast. However, it is difficult to optimize a general performance measure, and to encode path constraints imposed on the non-flat outputs of the system.

### 4.4.2 Collision detection

An important part of step 3 in Algorithm 4.1 is to be able to efficiently detect if the motion computed by the steering function leads to a collision with obstacles. As discussed in Section 4.4.1, collision avoidance is usually decoupled from the steering function in order to be compatible with any representation of the environment. Moreover, static and dynamic obstacles are often treated differently. This is since future states of the dynamic obstacles need to be predicted, which is usually performed by a separate tracking and prediction module. The details of this module are outside the scope of this thesis, but interested readers are referred to work such as [Batkovic et al., 2018, Deo et al., 2018, Hubmann et al., 2018, Wiest et al., 2012].

A popular approach for collision detection is to use bounding regions to check for collision between the system and surrounding obstacles. Some commonly used representations of bounding regions are illustrated in Figure 4.3. The use of circular bounding regions is the most simple and computationally cheap representation. As an example, a circle at $p_1 = (x_1, y_1)$ with radius $r_1$ does not intersect a circle at $p_2 = (x_2, y_2)$ with radius $r_2$ if

$$\|p_1 - p_2\| > r_1 + r_2,$$

which can be used as a criterion to ensure collision avoidance. For elongated bodies such as marine vessels, or multi-body systems such as tractor-trailer systems,
hierarchical representations can be used. Each body is then decomposed into a
tree, where a vertex in the tree represents a bounding region which contains a
subset of the body, see, e.g., Figure 4.3a [LaValle, 2006]. To detect collision when
using a tighter approximation of the bodies, e.g., the convex hull in Figure 4.3b, is
in general more computationally expensive, as it involves computing the distance
between two general convex sets $\mathcal{A}$ and $\mathcal{B}$. This is given as the solution to the
following optimization problem:

$$\text{dist}(\mathcal{A}, \mathcal{B}) = \inf\{\|u - v\| \mid u \in \mathcal{A}, v \in \mathcal{B}\},$$

(4.8)

where $u - v$ can be interpreted as the smallest transition that puts $\mathcal{A}$ and $\mathcal{B}$ in
contact [Schulman et al., 2014]. One algorithm that can be used to compute this
distance between two general convex sets is the Gilbert-Johnson-Keerthi (GJK)
algorithm [Bergen, 1999].

Another commonly used approach for collision detection is to discretize the
environment into cells that are either classified as occupied or free, such as occu-
pancy grid maps in 2D [Elfes, 1989] or OctoMaps in 3D [Hornung et al., 2013].
In these approaches, the system is considered to be in collision if any part of the
system’s body is within a cell that is classified as occupied. The main benefit of
this approach is its simplicity: it is both fast to evaluate and straightforward to
implement when the environment is explored using sensors such as lidar and/or
radar [Hornung et al., 2013]. The occupancy grid representation can be extended
by associating each cell with a real-valued cost, rather than a binary representation.
This makes it possible to promote trajectories computed by the sampling-based
motion planner that keeps a safety distance to obstacles [Ferguson and Likachev,
2008], which is illustrated in Figure 4.4.

4.4.3 Expansion strategy

The order in which states $x_k$ are selected for expansion in step 2 of Algorithm 4.1 is
a critical choice for the performance of a sampling-based motion planner [LaValle,
Figure 4.4: Illustration of the conceptual difference between using a binary occupancy grid map (a) and a real-valued cost map (b). The use of an occupancy grid map results in a trajectory that is close to the boundary of the obstacle in the middle. This is avoided in (b) where the cost map is used to promote trajectories that keep a safety distance to obstacles.

2006]. The state to expand from, $x_k$, and the desired state to connect to, $x_{k+1}$, are selected by either randomized or deterministic sampling strategies [LaValle, 2006]. Randomized sampling methods draw random samples of $x_{k+1}$ from $X_{\text{free}}$ and then finds the nearest neighbor $x_k \in V$. A steering function is then used which aims at connecting these states, as described in Section 4.4.1. A randomized sampling strategy typically leads to tree-shaped search spaces [LaValle, 2006]. Many sampling-based motion planning algorithms with random sampling are based on the standard Rapidly-exploring Random Tree (rRT) algorithm [Lavalle, 1998], which was originally developed for solving motion planning problems without considering system dynamics or kinematics. Several methods based on the rRT algorithm have been proposed to account for system models, e.g., [Frazzoli et al., 2002, LaValle and Kuffner Jr, 2001]. Another extension is rRT* for dynamical systems [Karaman and Frazzoli, 2013, Stoneman and Lampariello, 2014], where a rewiring step of the search tree is added such that asymptotic optimality guarantees can be obtained [Karaman and Frazzoli, 2013]. A drawback of rRT* for dynamical systems is that the steering function requires the solution of an OCP between each pair of states that are selected for connection or rewiring online, a process which for a general nonlinear system can be computationally demanding or even intractable. In [Kuwata et al., 2008], rRT is instead used to sample the input to a closed-loop system with an underlying controller (CL-rRT), i.e., using an inexact steering function. On the downside, no optimization of the search tree by rewiring (as used in rRT*) can be performed with CL-rRT since states cannot be connected exactly with this choice of steering function. For a more detailed description of sampling-based motion planning with random sampling, the reader is referred to the rRT-related references [Karaman and Frazzoli, 2013, Kuwata et al., 2008, Lavalle, 1998, Stoneman and Lampariello, 2014] and [Evestedt, 2016, Ljungqvist, 2019].
For deterministic sampling strategies, a heuristic function \( h(x_k, x_f) \) is usually used to approximate the cost-to-go, i.e., the cost to reach \( x_f \) from \( x_k \in \mathcal{V} \). This value, together with the cost-to-come, i.e., the cost to reach \( x_k \) from the initial state, guides the expansion step to select the most promising candidate \( x_k \in \mathcal{V} \) to expand. If the heuristic function is a good approximation of the actual optimal cost-to-go from \( x_k \) to \( x_f \), here denoted \( h^*(x_k, x_f) \), the search conducted in Algorithm 4.1 will be able to find a solution faster [Hart et al., 1968]. To ensure that the algorithms are able to provide optimality guarantees, the heuristic function needs to satisfy some properties. Let \( J(e_k) \) denote the cost of a feasible motion \( e_k \) from \( x_k \) to \( x_{k+1} \). Then, optimality guarantees are maintained if the heuristic function \( h \) has the following properties [Korf and Reid, 1998]:

1. Admissible: \( h(x_k, x_f) \leq h^*(x_k, x_f) \), \( x_k \in \mathcal{V} \).
2. Consistent: \( h(x_k, x_f) \leq J(e_k) + h(x_{k+1}, x_f) \), \( x_k \in \mathcal{V} \) with successor state \( x_{k+1} \in \mathcal{V} \).

Most admissible heuristic functions are also consistent. Furthermore, an inconsistent but admissible heuristic function can be transformed into a consistent heuristic function that is a more accurate approximation than the original one [Korf and Reid, 1998]. In this thesis, a deterministic sampling strategy in terms of a lattice-based motion planner is used. This type of motion planner and the reason for choosing it is further described in the next section.

### 4.5 Lattice-based motion planning

The key idea with a lattice-based motion planner is to restrict the control inputs to a discrete subset of the admissible inputs and hence transform the optimal motion planning problem in (4.5) to a discrete graph search problem [Pivtoraiko and Kelly, 2005]. This transformation can be performed since the attainable states can be represented by a lattice structure, i.e., neighboring states can be reached by addition or subtraction of motion segments. There are two different approaches that are used to generate a lattice for motion planning [Howard et al., 2008]:

1. Control sampling: The control space is sampled in a way such that the resulting sampling in the state space has desired properties in terms of discrepancy and dispersion. This strategy relies on forward simulations of the system online.

2. State lattice: First, a desired state-space discretization \( \mathcal{X}_d \) is selected. Then, an OCP solver is used offline to connect several states in the discretized state space.

In this thesis, the state-lattice methodology will be used. The state-lattice formulation has mainly been used for position invariant systems operating in unstructured environments. The benefits of using a state lattice are that it is possible to design the state-space discretization depending on the application, and the potentially complex relation of how the control signal affects the states of the system is handled offline by the OCP solver. The latter also means reduced online computations.
since no forward simulations of the system are needed during online planning [Pivtoraiko and Kelly, 2011]. It is also much easier to create regular lattices, which will cover a larger volume of the search space in fewer samples [LaValle, 2006].

The use of a regular state lattice will lead to that the graph $G = (V, E)$ in Algorithm 4.1 becoming cyclic, since many combinations of edges $e \in E$ will arrive at the same state $x_k \in V$. This enables the possibility of using cyclic graph search methods such as bidirectional search and exact pruning for a faster online planning [Pivtoraiko and Kelly, 2011].

### 4.5.1 State-lattice construction

The aim of the state-lattice construction is to compute a motion primitive set $P$ which contains feasible motions that move the system between different discretized states $x_d \in \mathcal{X}_d$. This set is then used to solve the motion planning problems by concatenating a sequence of compatible motion primitives. Thus, the motion primitives act as precomputed steering functions within the sampling-based motion planner outlined in Algorithm 4.1, where the successor states are deterministically sampled offline. The state-lattice construction procedure can be divided into three different steps which are all performed offline [Pivtoraiko et al., 2009] (illustrated in Figure 4.5):

1. Discretize the state space to obtain a discrete search space $\mathcal{X}_d$.
2. Select which pairs of states to connect in the discretized representation.
3. Solve the OCPs defined in the previous step to compute the motion primitive set $P$. 

**Figure 4.5:** An illustration of the three steps in the state-lattice construction. (1) Discretize the state space, (2) select connectivity, (3) solve resulting OCPs.
The first step is to decide how the state space should be discretized, i.e., how to choose \( X_d \). This is done by selecting which fidelity of the state space that should be used in the search space for the planner. As an example for car-like vehicles, the first step could be to determine the fidelity of the position, orientation and steering angle of the vehicle [Pivtoraiko et al., 2009].

The second step is to select which pairs of states to connect in the discretized search space. If the system is position invariant, it is only required to compute motion primitives from states positioned at the origin. These motion primitives can then be translated and reused at all positions on the grid [Pivtoraiko et al., 2009]. Furthermore, if the system is orientation invariant, rotational symmetries of the system can be exploited such that it is only required to compute motion primitives from a few initial headings. These motions can then be mirrored and/or rotated to the other initial headings to obtain the full motion primitive set [Pivtoraiko and Kelly, 2005]. Ideally, all possible initial states positioned at the origin should be connected directly to all terminal states in the discrete state space. However, this is intractable in practice since the required memory usage, offline computation and online search time (with obstacles present) would be intractable. Instead, a smaller number of connections are chosen, such that the resulting reachability graph \( G_r(x_s, P) \) (which encodes the set of all possible trajectories from \( x_s \) given \( P \)) sufficiently well resembles the true forward reachable set of the system [LaValle, 2006].

The procedure of how to select which states to connect is application dependent; systems that are more agile or systems working in environments with a lot of obstacles should have a dense connectivity selection. However, it is also a trade-off between optimality and online computation time, since the number of motion primitives defines the branching factor online, i.e., the number of edges that are explored in the graph for each node expansion. Hence, the complexity of the graph search problem to be solved online scales exponentially with the number of applicable motion primitives \( P(x) \) from each discrete state \( x \in X_d \) [Pivtoraiko et al., 2009].

Finally, the ocp's defined by the second step need to be solved to obtain the motion primitive set. In this thesis, a motion primitive \( m \in P \) is defined as:

\[
m = (x_m(t), u_m(t)) \in X \times U, \ t \in [0, T_m],
\]

and represents a feasible trajectory that moves the system from an initial state \( x_m(0) = x_{start} \in X_d \) to a terminal state \( x_m(T_m) = x_{final} \in X_d \) by applying the control inputs \( u_m(\cdot) \), where \( x_{start} \in X_d \) and \( x_{final} \in X_d \) are given by the connectivity selection in the previous step. The motion primitives are computed by solving the following ocp:

\[
\begin{align*}
\minimize_{x(\cdot), u(\cdot), T} & \int_{t=0}^{T} \ell(x(t), u(t)) \, dt \\
\text{subject to} & \quad x(0) = x_{start}, \quad x(T) = x_{final}, \\
& \quad x(t) = f(x(t), u(t)), \quad t \in [0, T] \\
& \quad x(t) \in X, \quad u(t) \in U, \quad t \in [0, T]
\end{align*}
\]

(4.9)
for each combination of $x_{\text{start}}$ and $x_{\text{final}}$ from the connectivity step. Since these calculations are performed offline, the obstacle constraints, $x(t) \notin X_{\text{obst}}(t)$ in (4.5), are so far unknown and hence disregarded in (4.10). Instead, they are considered in the collision detection step (see Section 4.4.2). The cost function can be chosen as any smooth function of the states and control signals. For autonomous vehicles, one function that is commonly used is:

$$\ell(x(t), u(t)) = 1 + \|x(t)\|_Q^2 + \|u(t)\|_R^2,$$

where the weight matrices $Q \in \mathbb{R}^{n \times n} \succeq 0$ and $R \in \mathbb{R}^{m \times m} \succeq 0$ are used to determine the trade-off between time duration (captured by the integral of the first term in (4.11)) and other measures such as smoothness of a motion [Bitar et al., 2020, Ljungqvist, 2019, Zhang et al., 2020].

One approach to solve the OCPs given in (4.10) is to use direct methods for optimal control, which is a popular method for computing motion primitives [Andersson et al., 2018, Howard and Kelly, 2007, Ljungqvist et al., 2017, Pivtoraiko et al., 2009]. In this thesis, state-of-the-art optimal control software applicable to general nonlinear systems is used to solve the OCPs. In these methods, the continuous-time optimal control problem in (4.10) is reformulated as an NLP problem using methods described in Chapter 3. An example of a resulting motion primitive set for a car-like vehicle is shown in Figure 4.6.
4.5.2 Online planning

After the motion primitive set is computed, it is possible to approximate the OCP in (3.1) with the following discrete OCP:

\[
\begin{align*}
\text{minimize} & \quad J_{\text{lat}} = \sum_{k=1}^{N} L_m(m_k) \\
\text{subject to} & \quad x_1 = x_s, \quad x_{N+1} = x_f, \quad t_1 = t_i, \\
& \quad x_{k+1} = f_m(x_k, m_k), \quad k \in \mathbb{Z}_{1,N} \\
& \quad t_{k+1} = t_k + T_{m_k}, \quad k \in \mathbb{Z}_{1,N} \\
& \quad m_k \in \mathcal{P}(x_k), \quad k \in \mathbb{Z}_{1,N} \\
& \quad c(x_k, m_k, t) \notin \mathcal{X}_{\text{obst}}(t), \quad k \in \mathbb{Z}_{1,N}, \quad t \in [t_k, t_{k+1}].
\end{align*}
\]

The decision variables are the motion primitive sequence \( \{m_k\}_{k=1}^{N} \) and its length \( N \). The cost for each motion primitive, which here corresponds to a stage, is

\[
L_m(m) = \int_{0}^{T_m} \ell(x_m(t), u_m(t))dt.
\]

The state-transition equation in (4.12c) defines the successor state \( x_{k+1} \) after \( m_k \) is applied from \( x_k \) and (4.12d) defines the corresponding time at \( x_{k+1} \). Furthermore, the constraint in (4.12e) ensures that \( m_k \) is selected from the set of applicable motion primitives at \( x_k \). Finally, the constraint in (4.12f) ensures that system does not collide with obstacles during the time interval \([t_k, t_{k+1}]\) when \( m_k \) is applied from \( x_k \).

The problem in (4.12) can be solved online by using efficient graph search methods such as \( A^* \) [Hart et al., 1968], anytime repairing \( A^* \) (ARA*) [Likhachev et al., 2004], etc., making it real-time applicable [Ljungqvist et al., 2017, Pivtoraiko et al., 2009]. Since standard graph search techniques can be used, the lattice-based motion planner inherits the properties and optimality guarantees of the chosen graph search method. Another benefit is that the solution is guaranteed to be feasible with respect to the system model since the motion primitives have been computed by solving (4.10). On the downside, the solutions from a lattice-based planner often suffer from discretization artifacts [Andreasson et al., 2015, Oliveira et al., 2018a], making it desirable to smoothen the state-lattice solution. Another limitation with graph search is that it is only possible to plan from and to states within the selected state-space discretization [LaValle, 2006].

An \( A^* \) search algorithm, inspired by Algorithm 1 in [Evestedt, 2016], tailored for a lattice-based motion planner is outlined in Algorithm 4.2, where the aim is to compute a trajectory that brings the system from \( x_s \) to \( x_f \). The algorithm maintains two sets of nodes: one to be able to keep track of which states that have been visited, \( V_{\text{closed}} \), and one which gathers all the nodes left for exploration, \( V_{\text{open}} \). The nodes in \( V_{\text{open}} \) are sorted according to an estimate of the objective function value represented by \( J_{\text{best}} \). This estimate is calculated as the sum of the cost-to-come defined by \( J_{\text{come}} \), and an estimate of the cost-to-go given by the heuristic
function $h(x, x_f)$. Hence, the node associated with the most promising state is always selected for expansion. Once a node $n_k \in V_{\text{open}}$ has been selected for expansion, each possible feasible motion primitive from the associated state, i.e., $m \in P(n_k, x)$, is applied from $n_k, x$ and the resulting state $x_{\text{next}}$ and corresponding time $t_{\text{next}}$ are calculated (lines 9-10 in Algorithm 4.2). Now, four possible scenarios can occur:

1. If a node associated with $x_{\text{next}}$ is already in the set of closed nodes $V_{\text{closed}}$, the path in the graph between $n_k, x$ and $x_{\text{next}}$ can be pruned since the best solution from $x_s$ to $x_{\text{next}}$ is already known (lines 11-12 in Algorithm 4.2).

2. If the path between $n_k, x$ and $x_{\text{next}}$ is in collision, represented by the constraint $c(n_k, x, m, t) \in \mathcal{X}_{\text{obst}}(t), t \in [n_k, t, t_{\text{next}}]$, the current path in the graph is pruned and the search continues, since the motion is infeasible (lines 13-14 in Algorithm 4.2).

3. If no node associated with $x_{\text{next}}$ exists in $V_{\text{open}}$, a new node for expansion is added to $V_{\text{open}}$ with associated state $x_{\text{next}}$ and time $t_{\text{next}}$, costs $J_{\text{come}}$ and $J_{\text{est}}$, parent node $n_k$ and the applied motion primitive $m$ (lines 18-25 in Algorithm 4.2).

4. If a node $n_j \in V_{\text{open}}$ associated with $x_{\text{next}}$ already exists, but the new cost-to-come $J_{\text{come}}$ is lower than the cost-to-come value to $x_{\text{next}}$ from $n_k, x$, the node is updated with new costs, parent node, motion primitive and time (lines 26-30 in Algorithm 4.2). Otherwise, the previous path in the graph to get to $x_{\text{next}}$ has a lower cost-to-come and the current candidate path in the graph is pruned.

The search for a solution is terminated when a node associated with $x_f$ is selected for expansion, and the solution is obtained by backtracking the parent nodes until a node with the initial state $x_s$ is reached (lines 5-6 in Algorithm 4.2). Optimality of the solution can be ensured since the list of open nodes is sorted according to the estimated objective function value. Hence, under the assumption that the heuristic function is both admissible and consistent, it is not possible to find any better solution with the current motion primitive set that brings the system from $x_s$ to $x_f$ [Hart et al., 1968].

As stated in Section 4.4, the online computational efficiency of a motion planner based on deterministic sampling relies on a well-informed heuristic function to guide the search towards the solution. One alternative that is compatible with lattice-based motion planning is to use a precomputed heuristic look-up table (HLUT). The HLUT typically represents a perfect estimate of the cost-to-go in free space [Knepper and Kelly, 2006]. The HLUT is computed offline by solving a large number of motion planning problems (4.12) in an obstacle-free environment from all initial states $x_{\text{start}} \in \mathcal{X}_d$ positioned at the origin to, e.g., all states $x_{\text{final}} \in \mathcal{X}_d$ positioned within a square centered around the origin with a user-defined side length $\rho$, where the size of the HLUT for a sufficient online planning performance is application dependent. In [Knepper and Kelly, 2006], some techniques for reducing the required memory usage and computation time for storing and generating the HLUT can be found.
Algorithm 4.2: A\textsuperscript{*} search tailored for lattice-based motion planning

1: Initialization: $n_s.x \leftarrow x_s$, $n_s.t \leftarrow t_i$, $\mathcal{V}_{\text{open}} \leftarrow n_s$, $\mathcal{V}_{\text{closed}} \leftarrow \emptyset$.
2: while $\mathcal{V}_{\text{open}} \neq \emptyset$ do
3:     $n_k \leftarrow \mathcal{V}_{\text{open}}.\text{pop}()$
4:     $\mathcal{V}_{\text{closed}}.\text{push}(n_k)$
5:     if $n_k.x = x_f$ then
6:         return back\_track($n_k$)
7:     end if
8:     for each $m \in \mathcal{P}(n_k.x)$ do
9:         $x_{\text{next}} \leftarrow f_m(n_k.x, m)$
10:        $t_{\text{next}} \leftarrow n_k.t + T_m$
11:        if any $n_j.x = x_{\text{next}}$, $n_j \in \mathcal{V}_{\text{closed}}$ then
12:            continue
13:        else if $c(n_k.x, m, t) \in \mathcal{X}_{\text{obst}}(t), t \in [n_s.t, t_{\text{next}}]$ then
14:            continue
15:        else
16:            $J_{\text{come}} \leftarrow n_k.J_{\text{come}} + L_m(m)$
17:            $J_{\text{est}} \leftarrow J_{\text{come}} + h(x_{\text{next}}, x_f)$
18:            if not any $n_j.x = x_{\text{next}}$, $n_j \in \mathcal{V}_{\text{open}}$ then
19:                $n_{\text{next}}.x \leftarrow x_{\text{next}}$
20:                $n_{\text{next}}.t \leftarrow t_{\text{next}}$
21:                $n_{\text{next}}.J_{\text{come}} \leftarrow J_{\text{come}}$
22:                $n_{\text{next}}.J_{\text{est}} \leftarrow J_{\text{est}}$
23:                $n_{\text{next}}.\text{parent} \leftarrow n_k$
24:                $n_{\text{next}}.\text{motion} \leftarrow m$
25:                $\mathcal{V}_{\text{open}}.\text{push}(n_{\text{next}})$
26:            else if any $n_j.x = x_{\text{next}}$ and $J_{\text{come}} < n_j.J_{\text{come}}$, $n_j \in \mathcal{V}_{\text{open}}$ then
27:                $\mathcal{V}_{\text{open}}.\text{update\_cost}(n_j, J_{\text{come}}, J_{\text{est}})$
28:                $\mathcal{V}_{\text{open}}.\text{update\_parent}(n_j, n_k)$
29:                $\mathcal{V}_{\text{open}}.\text{update\_motion}(n_j, m)$
30:                $\mathcal{V}_{\text{open}}.\text{update\_time}(n_j, t_{\text{next}})$
31:            end if
32:        end if
33:     end for
34: end while
Concluding remarks

In this thesis, optimal motion planning for dynamical systems operating in unstructured environments is studied. The focus is to develop reliable optimization-based algorithms that can be applied to problems of this type. Here, the main contributions are summarized, general conclusions are given, and possible future research directions are presented.

5.1 Summary of contributions

This section summarizes the contributions of the papers in Part II. In Paper A, the problem of initializing an OCP solver for motion planning is addressed by successfully transforming the problem environment. It is shown that by combining the homotopy method and the SQP method, a method that has promising properties to solve motion planning problems involving environments with challenging geometries of the feasible set is obtained. A proof-of-concept implementation of the algorithm is evaluated in numerical experiments on several motion planning problems. The results show that the proposed method extends the applicability of direct optimal control methods, by solving problems that could not be solved at all by these methods before when using standard initialization strategies.

The work in Paper B proposes an automatic optimization-based motion primitive generation framework for lattice-based motion planning. Based on user-defined so-called maneuvers, the suggested framework computes the motion primitive set for a parameterized family of systems. Examples of such systems are cars, buses or trucks parameterized by platform dimensions and actuator limitations. It is shown that the proposed framework enables use of the same maneuver definitions for all system instances, which gives a fast computation of the motion primitive set regardless of which system in the family that is being considered. In numerical experiments, the new method is shown to clearly outperform existing
related methods, both in terms of performance as well as generality.

Paper C presents a unified optimization-based path planning approach, based on a novel combination of a lattice-based path planner and direct optimal control. The first step of the proposed approach consists of using a lattice-based path planner. This step uses a motion primitive set, which is optimized using the framework presented in Paper B, to find a dynamically feasible solution to the path planning problem in a discretized search space. This solution is then used as a well-informed warm-start in a second improvement step where optimal control techniques are used to compute a locally optimal solution. In order to tightly couple the two steps, the lattice-based path planner uses a vehicle model and objective function that are chosen to coincide with those used in the second improvement step. This proposed combination makes, in a structured way, use of sampling-based methods’ ability to solve combinatorial parts of the problem and optimal control-based methods’ ability to obtain locally optimal solutions. The motion planning approach is further developed in Paper D. In this paper, the second step is modified such that it can be applied in a receding-horizon fashion. Theoretical guarantees in terms of recursive feasibility, worst-case objective function value and convergence to the terminal state are established by utilizing the combination of methods proposed in Paper C. The resulting motion planning algorithm is applied with successful results in several problems for car-like and tractor-trailer vehicles.

In Paper E, the motion planning algorithm proposed in Paper D is tailored and extended to account for the specific needs of marine vessels. As a part of the first step, the motion primitive generation framework in Paper B is used to compute a set of optimized motion primitives that are able to account for the complex dynamical model of the vessel. Furthermore, an algorithm to compute spatial safety envelopes along each sample of the trajectory from the lattice-based planner is proposed. These safety envelopes are represented using convex polytopes and are used in the second step to ensure collision avoidance. The proposed motion planner is evaluated with successful results in a simulation study for autonomous docking problems in a model of the Port of Cape Town.

In Paper F, the work in Paper E is extended to handle scenarios where other marine vessels are present. In such scenarios, the motion planner needs to compute trajectories that obey the rules specified in the international regulations for preventing collisions at sea (COLREGS). This introduces the additional combinatorial aspect of selecting a COLREGS-compliant trajectory when passing other nearby vessels. This aspect is handled by augmenting the lattice-based planner with discrete states that represent what type of COLREGS situations that are active at each time instant. In the second step, the problem is solved to local optimality in a receding-horizon fashion with the discrete COLREGS state fixed. The proposed motion planner is evaluated in a model of the Helsinki archipelago, where it successfully computes trajectories that comply with COLREGS. The results also show the motion planner’s ability to avoid complex collision situations from occurring at all.
5.2 Conclusions

In this thesis, algorithms that efficiently compute locally optimal solutions to motion planning problems in unstructured environments have been developed. A common theme has been to combine methods from automatic control, numerical optimization and robotics, and to utilize the strengths of the methods within each field. Motion planning algorithms from the robotics community are good at handling the combinatorial aspects of the problem, such as selecting on which side to pass obstacles. However, they suffer from the curse of dimensionality when computing optimized solutions to problems where the search space to explore grows. Optimal control algorithms on the other hand scale better with higher state-space dimensions and do not suffer from discretization artifacts. However, when being applied without a well-informed warm-start to a nonconvex problem such as the general motion planning problem, optimal control algorithms are in danger of getting stuck in a bad local minimum, or not being able to find a solution at all.

In the thesis, the strengths of the methods have been utilized in a tightly coupled two-step approach, where the second optimal control-based step is warm-started with a solution from the first sampling-based motion planning step. The motion primitives used in the first step are computed offline. By exploiting direct methods for optimal control, an optimization-based framework for motion primitive generation is developed that simultaneously optimizes the motions and the terminal states. This removes the need for a time-consuming manual tuning or an exhaustive search-based generation of the motion primitive set. Furthermore, the vehicle model and objective function in the motion primitive generation are chosen to coincide with those used in the second step. This makes it possible to provide theoretical guarantees for the second step in terms of feasibility and worst-case objective function value. Hence, the developed algorithms can reliably be applied for motion planning in challenging unstructured environments.

Another important aspect in the development of reliable algorithms for motion planning is to ensure collision avoidance. Two different approaches for representing the obstacles within a solver based on direct methods for optimal control have been investigated and used in the thesis. The first one is to represent the obstacles with smooth constraints such as circles and ellipsoids. However, the solver might not be able to find a solution if no well-informed initialization is available. In such cases, one alternative proposed in the thesis is to successively transform the representation of the obstacles from an easily solved problem to the original one. The second approach, which can be used when a well-informed initialization is available, is to represent the free space locally by constructing safety envelopes. These envelopes can be represented using, e.g., convex polytopes. Previously presented methods for computing safety envelopes have mainly been applied to vehicles where the body of each vehicle can be reasonably described by a single bounding circle. The thesis proposes a new method that can reliably be applied to elongated vehicles such as ships and multi-body vehicles such as tractor-trailer systems.
5.3 Future work

The results presented in the thesis can be extended in several ways. One possible extension of the results in Paper A is to integrate and implement the proposed algorithm in an NLP solver suited for OCPs with long prediction horizons. Another important future work is to develop an obstacle detection and classification framework, and to automatically select suitable homotopy maps for the obstacles (i.e., how the obstacles are “introduced”). A final extension to these results is to investigate if it is possible to integrate the homotopy map of the obstacle constraints with the homotopy-based functionality that is already implemented for the barrier parameter in IP-based optimization solvers.

Future work for the results presented in Paper B includes to develop an extension which automatically handles the discretization (completely or partially) of the state space, and to compute motion primitives that are also optimized with respect to expected potential obstacles in the environment.

For the tightly coupled two-step motion planning framework presented in Paper C–F, future work includes taking environmental disturbances such as wind and ocean currents into account when computing a trajectory. It would also be interesting to incorporate learning in the motion planner, e.g., by using learning-based MPC approaches such as the ones in [Hewing et al., 2020] in combination with a sampling-based motion planner. It would be interesting to investigate if this combination of methods can be used to update uncertain model parameters during execution. These new parameter values can then be used by the sampling-based motion planner to compute more accurate motion plans. It is expected that this will lead to less conservative motion plans compared to the ones obtained using robust planning techniques such as [Majumdar and Tedrake, 2017, Singh et al., 2018]. Another extension is to investigate the possibility of developing a heuristic function in the lattice-based planner that is able to capture the influence of both static and dynamic obstacles. A final interesting topic for future research is to investigate if it is possible to use a similar motion planning approach in tight combination with a task planner to solve problems that involve both continuous-time dynamics and discrete decisions, such as the problem considered in [Srivastava et al., 2014].
Eugene L Allgower and Kurt Georg. Numerical continuation methods: an introduction, volume 13. Springer Science & Business Media, 2012.

Olov Andersson, Oskar Ljungqvist, Mattias Tiger, Daniel Axehill, and Fredrik Heintz. Receding-horizon lattice-based motion planning with dynamic obstacle avoidance. In 2018 IEEE Conference on Decision and Control (CDC), 2018.

Henrik Andreasson, Jari Saarinen, Marcello Cirillo, Todor Stoyanov, and Achim J Lilienthal. Fast, continuous state path smoothing to improve navigation accuracy. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 662–669, 2015.

Holger Banzhaf, Nijanthan Berinpanathan, Dennis Nienhüser, and J Marius Zöllner. From $G^2$ to $G^3$ continuity: Continuous curvature rate steering functions for sampling-based nonholonomic motion planning. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 326–333, 2018.

Ivo Batkovic, Mario Zanon, Nils Lubbe, and Paolo Falcone. A computationally efficient model for pedestrian motion prediction. In 2018 European Control Conference (ECC), pages 374–379, 2018.

Gino van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. Journal of graphics tools, 4(2):7–25, 1999.

Kristoffer Bergman and Daniel Axehill. Combining homotopy methods and numerical optimal control to solve motion planning problems. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 347–354, 2018.

Kristoffer Bergman, Oskar Ljungqvist, and Daniel Axehill. Improved optimization of motion primitives for motion planning in state lattices. In 2019 IEEE Intelligent Vehicles Symposium (IV), pages 2307–2314, 2019.

Kristoffer Bergman, Oskar Ljungqvist, Torkel Glad, and Daniel Axehill. An optimization-based receding horizon trajectory planning algorithm. In Proceedings of 21st IFAC World Congress, 2020a.
Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axehill. An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments. In 2020 59th IEEE Conference on Decision and Control (CDC), pages 5283–5290, 2020b.

Kristoffer Bergman, Oskar Ljungqvist, and Daniel Axehill. Improved path planning by tightly combining lattice-based path planning and optimal control. IEEE Transactions on Intelligent Vehicles, 6(1):57–66, 2021a.

Kristoffer Bergman, Oskar Ljunqvist, Jonas Linder, and Daniel Axehill. A COLREGs-compliant motion planner for autonomous maneuvering of marine vessels in complex environments. Under review for possible publication in IEEE Journal of Oceanic Engineering. Available at arXiv: https://arxiv.org/abs/2012.12145v2, 2021b.

John T Betts. Practical methods for optimal control and estimation using nonlinear programming. SIAM, 2010.

Glenn Bitar, Andreas B Martinsen, Anastasios M Lekkas, and Morten Breivik. Two-stage optimized trajectory planning for ASVs under polygonal obstacle constraints: Theory and experiments. IEEE Access, 8:199953–199969, 2020.

Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, Cambridge, UK, 2004. ISBN 0521833787.

Arthur E. Bryson and Yu-Chi Ho. Applied optimal control: optimization, estimation and control. Hemisphere Publishing Corp., 1975.

Christof Büskens and Dennis Wassel. The ESA NLP solver WORHP. In Modeling and Optimization in Space Engineering, volume 73, pages 85–110. Springer New York, 2013.

Leobardo Campos-Macías, David Gómez-Gutiérrez, Rodrigo Aldana-López, Rafael de la Guardia, and José I Parra-Vilchis. A hybrid method for online trajectory planning of mobile robots in cluttered environments. IEEE Robotics and Automation Letters, 2(2):935–942, 2017.

Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. How would surround vehicles move? a unified framework for maneuver classification and motion prediction. IEEE Transactions on Intelligent Vehicles, 3(2):129–140, 2018.

Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. In Fast Motions in Biomechanics and Robotics, pages 65–93. Springer, 2006.

Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. Journal of the ACM (JACM), 40(5):1048–1066, 1993.

Lester 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.
Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Niclas Evestedt. *Sampling Based Motion Planning for Heavy Duty Autonomous Vehicles*. Licentiate thesis, Linköping University, 2016.

Niclas Evestedt, Oskar Ljungqvist, and Daniel Axehill. Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3690–3697, 2016.

Timm Faulwasser, Lars Grüne, Matthias A Müller, et al. Economic nonlinear model predictive control. *Foundations and Trends® in Systems and Control*, 5(1):1–98, 2018.

Dave Ferguson and Maxim Likachev. Efficiently using cost maps for planning complex maneuvers. In *Proceedings of the Workshop on Planning with Cost Maps, IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, volume 11, pages 119–141, 2002.

Thierry Fraichard and Alexis Scheuer. From Reeds and Shepp’s to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, 2004.

Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of guidance, control, and dynamics*, 25(1):116–129, 2002.

Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *ASME 2010 Dynamic Systems and Control Conference*, pages 265–272, 2010.

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

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

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.
Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. ACADO toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

Thomas M Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, 26(2):141–166, 2007.

Thomas M Howard, Colin J Green, Alonzo Kelly, and Dave Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.

Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE Transactions on Intelligent Vehicles*, 3(1):5–17, 2018.

Sertac Karaman and Emilio Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *2013 IEEE International Conference on Robotics and Automation*, pages 5041–5047, 2013.

Ross A Knepper and Alonzo Kelly. High performance state lattice planning using heuristic look-up tables. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3375–3380, 2006.

Richard E Korf and Michael Reid. Complexity analysis admissible heuristic search. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence/Tenth Conference on Innovative Applications of Artificial Intelligence*, pages 305–310, 1998.

Yoshiaki Kuwata, Justin Teo, Sertac Karaman, Gaston Fiore, Emilio Frazzoli, and Jonathan P How. Motion planning in complex environments using closed-loop prediction. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

Steven M LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, UK, 2006.

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

Zexiang Li and John F Canny. *Nonholonomic motion planning*, volume 192. Springer Science & Business Media, 2012.
Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In Advances in Neural Information Processing Systems 16, pages 767–774, 2004.

Stephen R Lindemann and Steven M LaValle. Current issues in sampling-based motion planning. In Robotics Research: The Eleventh International Symposium, pages 36–54, 2005.

Oskar Ljungqvist. On motion planning and control for truck and trailer systems. Licentiate thesis, Linköping University, 2019.

Oskar Ljungqvist, Niclas Evestedt, Marcello Cirillo, Daniel Axehill, and Olov Holmer. Lattice-based motion planning for a general 2-trailer system. In 2017 IEEE Intelligent Vehicles Symposium (IV), 2017.

Oskar Ljungqvist, Niclas Evestedt, Daniel Axehill, Marcello Cirillo, and Henrik Pettersson. A path planning and path-following control framework for a general 2-trailer with a car-like tractor. Journal of Field Robotics, 36(8):1345–1377, 2019.

Oskar Ljungqvist, Kristoffer Bergman, and Daniel Axehill. Optimization-based motion planning for multi-steered articulated vehicles. In Proceedings of 21st IFAC World Congress, 2020.

Jan Marian Maciejowski. Predictive control: with constraints. Pearson Education, 2002.

Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. The International Journal of Robotics Research, 36(8): 947–982, 2017.

Andreas B Martinsen, Anastasios M Lekkas, and Sebastien Gros. Autonomous docking using direct optimal control. IFAC-PapersOnLine, 52(21):97–102, 2019.

David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. Automatica, 36 (6):789–814, 2000.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In 2011 IEEE International Conference on Robotics and Automation, pages 2520–2525, 2011.

Richard M Murray, Zexiang Li, and S Shankar Sastry. A mathematical introduction to robotic manipulation. CRC Press, 1994.

Jorge Nocedal and Stephen J Wright. Numerical Optimization. Springer, 2006.

Rui Oliveira, Marcello Cirillo, and Bo Wahlberg. Combining lattice-based planning and path optimization in autonomous heavy duty vehicle applications. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 2090–2097, 2018a.
Rui Oliveira, Pedro F Lima, Marcello Cirillo, Jonas Mårtensson, and Bo Wahlberg. Trajectory generation using sharpness continuous dubins-like paths with applications in control of heavy-duty vehicles. In *2018 European Control Conference (ECC)*, pages 935–940, 2018b.

Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

Rushen B Patel and Paul J Goulart. Trajectory generation for aircraft avoidance maneuvers using online optimization. *Journal of Guidance, Control, and Dynamics*, 34(1):218–230, 2011.

Aurelio Piazzi, Corrado Guarino Lo Bianco, and Massimo Romano. $\eta^3$-splines for the smooth path generation of wheeled mobile robots. *IEEE Transactions on Robotics*, 23(5):1089–1095, 2007.

Mihail Pivtoraiko and Alonzo Kelly. Efficient constrained path planning via search in state lattices. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, pages 1–7, 2005.

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

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

Elijah Polak. An historical survey of computational methods in optimal control. *SIAM review*, 15(2):553–584, 1973.

Anil V Rao. A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009.

James B Rawlings and Rishi Amrit. Optimizing process economic performance using model predictive control. In *Nonlinear Model Predictive Control*, pages 119–138. Springer, 2009.

James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

Stephen M Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms. *Mathematical programming*, 7(1):1–16, 1974.

Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, pages 5115–5120, 2017.
Pierre Rouchon, Michel Fliess, Jean Lévine, and Philippe Martin. Flatness, motion planning and trailer systems. In *Proceedings of 32nd IEEE Conference on Decision and Control*, volume 3, pages 2700–2705, 1993.

RWH Sargent. Optimal control. *Journal of Computational and Applied Mathematics*, 124(1-2):361–371, 2000.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

Sumeet Singh, Mo Chen, Sylvia L Herbert, Claire J Tomlin, and Marco Pavone. Robust tracking with model mismatch for fast and safe planning: an sos optimization approach. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 545–564, 2018.

Jason L Speyer and David H Jacobson. *Primer on optimal control theory*. SIAM, 2010.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014.

Samantha Stoneman and Roberto Lampariello. Embedding nonlinear optimization in RRT* for optimal kinodynamic planning. In *Proceedings of the 53rd IEEE Conference on Decision and Control*, pages 3737–3744, 2014.

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

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

Jürgen Wiest, Matthias Höffken, Ulrich Kreßel, and Klaus Dietmayer. Probabilistic trajectory prediction with gaussian mixture models. In *2012 IEEE Intelligent Vehicles Symposium*, pages 141–146, 2012.

Kwangjin Yang and Salah Sukkarieh. An analytical continuous-curvature path-smoothing algorithm. *IEEE Transactions on Robotics*, 26(3):561–568, 2010.

Xiaojing Zhang, Alexander Liniger, Atsushi Sakai, and Francesco Borrelli. Autonomous parking using optimization-based collision avoidance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4327–4332, 2018.

Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 2020.
Julius Ziegler, Philipp Bender, Thao Dang, and Christoph Stiller. Trajectory planning for bertha—a local, continuous method. In 2014 IEEE Intelligent Vehicles Symposium Proceedings, pages 450–457, 2014.
Part II

Publications
Papers

The papers 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-174175
M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.
H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.
A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.
P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.
J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.
M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.
U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.
A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.
N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.
K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.
M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.
F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.
V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.
M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.
F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.
J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.
J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.
J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.
O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.
R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.
D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.
R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.
J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.
E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.
M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.
T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.
I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.
J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

D. Axehill: Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.

G. Hendeby: Performance and Implementation Aspects of Nonlinear Filtering. Thesis No. 1161, 2008. ISBN 978-91-7393-979-9.

J. Sjöberg: Optimal Control and Model Reduction of Nonlinear DAE Models. Thesis No. 1166, 2008. ISBN 978-91-7393-964-5.

D. Törnqvist: Estimation and Detection with Applications to Navigation. Thesis No. 1216, 2008. ISBN 978-91-7393-785-6.

P-J. Nordlund: Efficient Estimation and Detection Methods for Airborne Applications. Thesis No. 1231, 2008. ISBN 978-91-7393-720-7.

H. Tidefelt: Differential-algebraic equations and matrix-valued singular perturbation. Thesis No. 1292, 2009. ISBN 978-91-7393-479-4.

H. Ohlsson: Regularization for Sparseness and Smoothness — Applications in System Identification and Signal Processing. Thesis No. 1351, 2010. ISBN 978-91-7393-287-5.

S. Moberg: Modeling and Control of Flexible Manipulators. Thesis No. 1349, 2010. ISBN 978-91-7393-289-9.

J. Wallén: Estimation-based iterative learning control. Thesis No. 1358, 2011. ISBN 978-91-7393-255-4.

J. D. Hol: Sensor Fusion and Calibration of Inertial Sensors, Vision, Ultra-Wideband and GPS. Thesis No. 1368, 2011. ISBN 978-91-7393-197-7.

D. Ankelhed: On the Design of Low Order H-infinity Controllers. Thesis No. 1371, 2011. ISBN 978-91-7393-157-1.

C. Lundquist: Sensor Fusion for Automotive Applications. Thesis No. 1409, 2011. ISBN 978-91-7393-023-9.

P. Skoglar: Tracking and Planning for Surveillance Applications. Thesis No. 1432, 2012. ISBN 978-91-7519-941-2.

K. Granström: Extended target tracking using PHD filters. Thesis No. 1476, 2012. ISBN 978-91-7519-796-8.

C. Lyzell: Structural Reformulations in System Identification. Thesis No. 1475, 2012. ISBN 978-91-7519-800-2.

J. Callmer: Autonomous Localization in Unknown Environments. Thesis No. 1520, 2013. ISBN 978-91-7519-620-6.

D. Petersson: A Nonlinear Optimization Approach to H2-Optimal Modeling and Control. Thesis No. 1528, 2013. ISBN 978-91-7519-567-4.

Z. Sjanic: Navigation and Mapping for Aerial Vehicles Based on Inertial and Imaging Sensors. Thesis No. 1533, 2013. ISBN 978-91-7519-553-7.

F. Lindsten: Particle Filters and Markov Chains for Learning of Dynamical Systems. Thesis No. 1530, 2013. ISBN 978-91-7519-559-9.
P. Axelsson: Sensor Fusion and Control Applied to Industrial Manipulators. Thesis No. 1585, 2014. ISBN 978-91-7519-368-7.

A. Carvalho Bittencourt: Modeling and Diagnosis of Friction and Wear in Industrial Robots. Thesis No. 1617, 2014. ISBN 978-91-7519-251-2.

M. Skoglund: Inertial Navigation and Mapping for Autonomous Vehicles. Thesis No. 1623, 2014. ISBN 978-91-7519-233-8.

S. Khoshfetrat Pakazad: Divide and Conquer: Distributed Optimization and Robustness Analysis. Thesis No. 1676, 2015. ISBN 978-91-7519-050-1.

T. Ardeshiri: Analytical Approximations for Bayesian Inference. Thesis No. 1710, 2015. ISBN 978-91-7685-930-8.

N. Wahlström: Modeling of Magnetic Fields and Extended Objects for Localization Applications. Thesis No. 1723, 2015. ISBN 978-91-7685-903-2.

J. Dahlin: Accelerating Monte Carlo methods for Bayesian inference in dynamical models. Thesis No. 1754, 2016. ISBN 978-91-7685-797-7.

M. Kok: Probabilistic modeling for sensor fusion with inertial measurements. Thesis No. 1814, 2016. ISBN 978-91-7685-621-5.

J. Linder: Indirect System Identification for Unknown Input Problems: With Applications to Ships. Thesis No. 1829, 2017. ISBN 978-91-7685-588-1.

M. Roth: Advanced Kalman Filtering Approaches to Bayesian State Estimation. Thesis No. 1832, 2017. ISBN 978-91-7685-578-2.

I. Nielsen: Structure-Exploiting Numerical Algorithms for Optimal Control. Thesis No. 1848, 2017. ISBN 978-91-7685-528-7.

D. Simon: Fighter Aircraft Maneuver Limiting Using MPC: Theory and Application. Thesis No. 1881, 2017. ISBN 978-91-7685-450-1.

C. Veibäck: Tracking the Wanders of Nature. Thesis No. 1958, 2018. ISBN 978-91-7685-200-2.

C. Andersson Naesseth: Machine learning using approximate inference: Variational and sequential Monte Carlo methods. Thesis No. 1969, 2018. ISBN 978-91-7685-161-6.

Y. Jung: Inverse system identification with applications in predistortion. Thesis No. 1966, 2018. ISBN 978-91-7685-171-5.

Y. Zhao: Gaussian Processes for Positioning Using Radio Signal Strength Measurements. Thesis No. 1968, 2019. ISBN 978-91-7685-162-3.

R. Larsson: Flight Test System Identification. Thesis No. 1990, 2019. ISBN 978-91-7685-070-1.

P. Kasebzadeh: Learning Human Gait. Thesis No. 2012, 2019. ISBN 978-91-7519-014-3.

K. Radnosrati: Time of flight estimation for radio network positioning. Thesis No. 2054, 2020. ISBN 978-91-7929-884-5.

O. Ljungqvist: Motion planning and feedback control techniques with applications to long tractor-trailer vehicles. Thesis No. 2070, 2020. ISBN 978-91-7929-858-6.

G. Lindmark: Controllability of Complex Networks at Minimum Cost. Thesis No. 2074, 2020. ISBN 978-91-7929-847-0.
