DISS. ETH NO. 24509

Motion Planning in Dynamic Environments

with Application to Self-Driving Vehicles

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Ulrich Schwesinger

Dipl.-Ing. Electrical Engineering/Information Technology,
Karlsruhe Institute of Technology

born on September 12, 1984

citizen of Freiburg im Breisgau, Germany

accepted on the recommendation of

Prof. Dr. Roland Y. Siegwart, Examiner
Prof. Dr. Andreas Krause, Co-examiner

2017

Institute for Robotics and Intelligent Systems, Autonomous Systems Lab
ETH Zurich
Switzerland

© 2017 Ulrich Schwesinger. All rights reserved.

Abstract

This thesis is concerned with the development of trajectory planning approaches targeting autonomous driving applications in dynamic environments shared with other traffic participants. The goal is to enable mobile robots to operate in challenging environments, characterized by narrow spaces and close proximity of other agents.

With their broad range of private and commercial applications reaching from logistics to valet parking to name a few, driverless vehicles have gained increasing attention in the past years. The complexity of automated driving, however, differs widely with respect to the application domain. In contrast to automated highway driving where the set of admissible actions and different traffic participants is usually small, driverless vehicles in urban environments are faced with a broad variety of semantic information and action choices. In addition, the challenging system kinematics and rotation-variant shape of cars call for system-compliant motion planning algorithms and accurate collision detection with both static elements and other traffic participants in the world. Most state-of-the-art approaches divide the motion planning problem in dynamic environments into prediction and planning. First, the future evolvement of the world is predicted, and in the following the autonomous agent seeks a plan to avoid predicted obstacles at any time. The fact that other traffic participants will adapt their future actions depending on the presence of the robot is neglected in order to make the challenging problem of real-time motion planning in dynamic environments tractable. This simplification, however, gives rise to over-conservative behavior of the autonomous agent, as the latter is rendered responsible for collision avoidance alone. Especially in more densely populated environments with few or weak traffic rules, neglecting the possibility of interaction between the autonomous agent and other traffic participants may lead to reduced navigation performance.

This thesis addresses two aspects of motion planning for autonomous vehicles. First, the thesis contributes a *non-cooperative*, sampling-based motion planning approach, suitable for autonomous vehicle navigation in narrow environments such as parking lots in valet parking applications. The second part is dedicated to *cooperative* motion planning to improve upon navigation in the presence of other agents.

The sampling-based motion planning approach, developed in the first part of the thesis, focuses on *planning highly system-compliant motions* in order to minimize the divergence between the motion planning and execution layer, and fast and accurate *time-aware collision detection* to test trajectories of objects of arbitrary shape for collision. Both system-compliant motion planning as well as accurate collision detection is mandatory for narrow environments in order to ensure safety, and avoid the loss of rare navigation space by conservative approximations of collision shapes. The algorithm employs a low-dimensional sampling space to generate inherently drivable candidate motions. The sampling space is hereby designed to yield expressive candidate motions for the task of navigation on a road network. The planning and collision detection modules are both extensively tested on a self-driving vehicle prototype operating in mixed traffic on several indoor parking garages and outdoor parking lots. Hereby, the automated vehicle is solely equipped with low-cost sensors such as cameras and ultrasonic sensors.

The motion planning approach above treats other agents in a non-cooperative fashion. First, predictions of the future motion of objects are obtained. Afterwards, motion planning is casted to the problem of finding a collision-free trajectory with respect to the fixed object predictions. It is shown from statistics of extensive real-world experiments, that—despite good performance in static environments and selected dynamic scenarios—the non-cooperative approach is especially vulnerable to noise in velocity estimates of other agents when navigating in close proximity. In the absence of rich semantic information to stabilize long-term object predictions, this thesis advocates the assumption of cooperative behavior as an autarkic mechanism to alleviate over-conservative robot behavior in close proximity to other agents. A *cooperative probabilistic planning approach* based on a maximum entropy model of joint navigation behavior is pursued, where typical per agent motion patterns, as well as joint navigation patterns are captured via feature functions. Parameters of the model are learned from demonstrated behavior in order to circumvent the cumbersome and difficult process of manual parameter tuning. Motion planning with the probabilistic model results in a joint trajectory optimization problem, in which the most likely collaborative solution is sought. As the model includes feature functions for static obstacle avoidance of all agents as well as features for agent interaction compliant with social norms, complex joint navigation scenarios can be forecasted. This thesis uniquely demonstrates real-world cooperative planning with the maximum entropy model in an uncontrolled public space. The results obtained support that cooperative planning is mandatory to address over-conservative mobile robot behavior in environments shared with other traffic participants.

Zusammenfassung

Diese Dissertation befasst sich mit der Entwicklung von Algorithmen zur Trajektorienplanung für Anwendungen im Bereich des autonomen Fahrens. Der Fokus liegt hierbei auf der Trajektorienplanung in dynamischen Umgebungen, die mit anderen Verkehrsteilnehmern geteilt werden. Diese Arbeit soll es mobilen Robotern ermöglichen sich in herausfordernden Umgebungen fortzubewegen, die nur wenig Raum bieten und Navigation in der Nähe anderer Agenten erfordern.

Durch ihr breites Spektrum an privaten und kommerziellen Anwendungsgebieten, in der Logistik oder automatisierten Parkdienstleistungen um einige Beispiele zu nennen, haben selbstfahrende Fahrzeuge in den letzten Jahren vermehrt an Aufmerksamkeit gewonnen. Der Schwierigkeitsgrad des automatisierten Fahrens unterscheidet sich je nach Anwendungsgebiet jedoch stark. Im Gegensatz zu automatisiertem Fahren auf Autobahnen, wo die verschiedenen Handlungsmöglichkeiten und anzutreffenden Verkehrsteilnehmer gering sind, werden automatisierte Fahrzeuge in urbanen Gebieten mit einer Vielzahl von semantischen Informationen und Handlungsmöglichkeiten konfrontiert. Hinzu kommt, dass das anspruchsvolle Bewegungsmodell und die rotationsvariante Form der Fahrzeuge systemkonforme Algorithmen zur Bewegungsplanung und akkurate Kollisionsdetektion mit sowohl statischen Hindernissen als auch anderen Verkehrsteilnehmern in der Welt erfordern. Die meisten aktuellen Ansätze unterteilen die Bewegungsplanung in dynamischen Umgebungen in Prädiktion und Planung. Zuerst wird die zukuftige Entwicklung der Welt geschätzt und anschließend sucht der autonome Agent einen Bewegungsplan, welcher prädizierte Hindernisse zu jeder Zeit vermeidet. Hierzu wird vernachlässigt, dass andere Verkehrsteilnehmer ihre Aktionen in Abhängigkeit von der Präsenz des autonomen Agenten adaptieren, um das herausfordernde Problem der Bewegungsplanung in dynamischen Umgebungen in Echtzeit lösbar zu machen. Diese Vereinfachung führt jedoch zu übermäßig konservativem Verhalten des autonomen Agenten, da angenommen wird, dass Letzterer alleinig für die Kollisionsvermeidung zuständig ist. Vor allem in Umgebungen mit einer Vielzahl anderer Verkehrsteilnehmer und wenigen oder nicht klar definierten Verkehrsregeln kann die Vernachlässigung der Möglichkeit zur Interaktion zu einer verringerten Navigationsqualität führen.

In dieser Dissertation werden zwei Aspekte der Bewegungsplanung für autonome Fahrzeuge behandelt. Zuerst wird ein *nicht-kooperativer*, sampling-basierter Ansatz entwickelt, der speziell für die Navigation in engen Umgebungen, wie sie in etwa auf Parkplatzgeländen für automatisiertes Parken zu finden sind, geeignet ist. Der zweite Teil der Dissertation befasst sich mit *kooperativer* Bewegungsplanung um die Navigation in Anwesenheit anderer Verkehrsteilnehmer, vor allem Fußgänger, zu verbessern.

Der sampling-basierte Ansatz, der im ersten Teil der Dissertation entwickelt wird, fokussiert sich zum einen auf die *Planung von besonders konformen Bewegungen* im Bezug auf das Fahrzeugmodell um die Abweichung zwischen dem Bewegungsplanungsmodul und der realen Plattform zu minimieren. Zum anderen wird ein schneller Algorithmus zur *Kollisionsdetektion von zeitparametrisierten Trajektorien* entwickelt, der keine Annahmen bezüglich der Form der Objekte trifft. Sowohl systemkonforme Bewegungsplanung als auch akkurate Kollisionsdetektion ist in engen Umgebungen unabdingbar um zum einen Sicherheit zu gewährleisten, und zum anderen zu vermeiden, dass durch eine konservative Approximation der Objekte wertvoller Platz für die Navigation verloren geht. Der Algorithmus verwendet einen niedrigdimensionalen Sampling-Raum um direkt fahrbare Trajektorienvorschläge zu generieren. Der Sampling-Raum ist hierbei so gestaltet, dass aussagekräftige Trajektorienvorschläge für das Fahren entlang eines Straßennetzwerkes entstehen. Der Bewegungsplanungsansatz wird intensiv auf einem selbstfahrenden Versuchsträger im Mischverkehr auf mehreren Parkplatzgeländen und Parkgaragen getestet. Hierbei ist das selbstfahrende Fahrzeug lediglich mit kostengünstiger Sensorik wie Kameras und Ultraschallsensoren ausgestattet.

Der oben genannte Bewegungsplanungsansatz behandelt andere Agenten in einer nicht-kooperativen Art und Weise. Zuerst werden Prädiktionen der Objektbewegungen erstellt, und anschließend wird die Bewegungsplanung als das Problem behandelt eine kollisionsfreie Trajektorie durch die vorgegebenen Objektprädiktionen zu finden. Mit Hilfe von gewonnenen Statistiken aus umfangreichen Experimenten wird gezeigt, dass der nicht-kooperative Ansatz trotz guter Ergebnisse in statischen und ausgewählten dynamischen Szenarien speziell anfällig gegenüber Rauschen in Geschwindigkeitsschätzungen anderer Agenten ist, wenn das Fahrzeug in kleinen Abständen zu diesen navigieren muss. In Ermangelung reichhaltiger semantischer Informationen um Langzeitprädiktionen dieser zu stützen wird in dieser Dissertation die Annahme von kooperativem Verhalten als ein autarker Mechanismus zu Grunde gelegt um übermäßig konservatives Verhalten des Roboters zu reduzieren. Hierfür wird ein *kooperativer, probabilistischer Planungsansatz* basierend auf einem Maximumentropie-Modell, welches gemeinsames Navigationsverhalten beschreibt, verfolgt. Typische

Bewegungsmuster von Agenten, sowie gemeinsames Navigationsverhalten werden durch Merkmalsfunktionen beschrieben. Die Parameter des Modells werden aus Demonstrationen gelernt, um mühsames und schwieriges manuelles Einstellen dieser zu umgehen. Die Bewegungsplanung mit dem probabilistischen Modell resultiert in einem gemeinsamen Trajektorienoptimierungsproblem, bei dem die wahrscheinlichste kooperative Lösung gesucht wird. Da das Modell Merkmalsfunktionen für die statische Hindernisvermeidung aller Agenten sowie Merkmalsfunktionen für die Interaktion zwischen Agenten, konform mit gesellschaftlichen Normen, beinhaltet, können komplexe Szenarien mit mehreren Agenten vorhergesagt werden. In dieser Dissertation wird zum ersten Mal kooperative Bewegungsplanung mit dem Maximumentropie-Modell in einem unbeaufsichtigten, öffentlichen Raum demonstriert. Die gewonnenen Resultate unterstreichen die Notwendigkeit kooperativer Bewegungsplanung um übermäßig konservatives Verhalten mobiler Roboter in Umgebungen zu vermeiden, die mit anderen Verkehrsteilnehmern geteilt werden.

Acknowledgements

Before all else, I would like to thank Prof. Roland Siegwart for the opportunity to conduct the research for this thesis at the Autonomous Systems Lab and his invaluable supervision and trust. Not only did he create an inspiring atmosphere at work, but beyond that a great team spirit. I furthermore want to thank Prof. Andreas Krause for his generous offer to co-supervise this thesis. In addition, my gratitude goes to Dr. Paul Furgale for his excellent supervision and guidance, as well as his company during countless trips and hours of autonomous travel, as well as Prof. Cédric Pradalier for his support during the beginning of my doctorate. I thank Dr. Gabriel Agamennoni for sharing his machine learning expertise and his dedicated help in the design of the cooperative planning framework.

I am fortunate, that I could work with extraordinary people throughout the years. I thank Dr. Martin Rufli for sharing his expertise in motion planning and his spirit of doing things "the right way". Hannes Sommer unconditionally provided valuable support, not exclusively limited to questions related to mathematics and programming. He also provided great software that made this work possible. I thank Mark Pfeiffer and Hannes Sommer for the collaborative development and implementation of the interaction-aware motion planner, and Mathias Bürki for supporting me in numerous test drives with our autonomous car. The adventurous autonomous drives on Zurich's public streets with Dr. Philipp Krüsi's robot ARTOR and the motion planner developed in this thesis will stay in my memory. I would furthermore like to thank—in alphabetical order—Dr. Javier Alonso-Mora, Dr. Andreas Breitenmoser, Dr. Philipp Krüsi, Dr. Janosch Nikolic, Mark Pfeiffer, Jörn Rehder, Dr. Martin Rufli, and Hannes Sommer, who I shared an office with over the years, for fruitful discussions and good company.

My gratitude also goes to Luciana Borsatti and Cornelia Della Casa, for the administration in the background, and Regina Notz for supporting me in the scientific lead of the V-Charge project during the final year. I also want to thank all people involved in the V-Charge project, too many to list by name. It was never an easy, but always exciting and rewarding journey to build up an autonomous car from scratch. I also

appreciate the support of all the anonymous helpers during the public demonstrations of the V-Charge car and the testing phases leading up to them.

I am also thankful to the people of the mobility and transport team at ETH Zurich, especially Jan Anders Viridén, for their support with the organization and maintenance of our autonomous car, and their help for our sometimes obscure request when testing.

Special thanks go to Mark Pfeiffer, Prof. Javier Alonso-Mora, and Hannes Sommer for taking the time to provide helpful feedback to this thesis at hand.

Finally, I want to thank my family, friends and better half for their love and support. You kept reminding me that self-driving cars are by far not the most important thing in the world.

Zurich, December 14, 2016 *Ulrich Schwesinger*

Financial Support

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No. 269916, V-Charge, and grant agreement No. 610603, EUROPA2.

Contents

Abstract **i**

Zusammenfassung **iii**

Acknowledgements **vii**

1 Introduction **1**
1.1 Motivation and Objectives 2
1.2 A Survey on Motion Planning for Self-Driving Cars 4
1.2.1 Software Architectures 5
1.2.2 State-Lattices 6
1.2.3 Randomized Sampling 8
1.2.4 Local Trajectory Rollout 9
1.2.5 Prediction of Other Traffic Participants 9
1.2.6 Interaction-Aware Planning 10
1.2.7 Adaptive/Trainable Motion Planning 11
1.2.8 Summary and Conclusion 12
1.3 Terminology 13
1.4 Approach 14
1.5 Organization 20

2 Contributions **21**
2.1 System-Compliant Motion Planning 21
2.2 Collision Detection in Space-Time 22
2.3 Cooperative Motion Planning via Maximum Entropy Models 23
2.4 List of Publications 25
2.5 Supervised Student Projects 27
2.6 Project Management 28

3 Onlane Motion Planning for Autonomous Cars **29**
3.1 Context and Related Work 29

3.2 Robotic Platform 30
3.3 Motion Planning Framework 31
3.3.1 System Architecture 32
3.3.2 Algorithm 33
3.3.3 System Modeling 37
3.3.4 Controller Design 40
3.3.5 Collision Detection 43
3.3.6 Objective Function 44
3.3.7 Situation-Specific Adaptations 49
3.3.8 Movement Predictions for Traffic Participants 49
3.4 Experimental Results 51
3.4.1 System-Compliant Planning in Narrow Environment 51
3.4.2 Navigation in Simulated Traffic Scenario 53
3.4.3 Mitigation of Crossing Pedestrian 56
3.4.4 Platooning 58
3.4.5 Long-Term Automated Driving Statistics 58
3.5 Summary and Discussion 62

4 Time-Aware Collision Detection 65
4.1 Context and Related Work 65
4.2 Approach 68
4.2.1 Isolated Configuration Query 70
4.2.2 Tree-vs-Tree Query 71
4.2.3 Time Discretization and Temporal Extent of Primitives 71
4.2.4 Convex Hull Representation of Trajectories 73
4.3 Results 73
4.4 Summary and Discussion 77

5 Cooperative Motion Planning via Maximum Entropy Models 79
5.1 Introduction 79
5.2 Related Work 81
5.3 The Principle of Maximum Entropy 86
5.4 Distribution over Composite Trajectories 90
5.5 Feature Functions 92
5.5.1 Motion Features 94
5.5.2 Interaction Features 101
5.5.3 Observation Features 101
5.6 Baseline Model 102
5.7 Model Training 103

5.7.1 The Loss Function 103
5.7.2 Computation of Feature Expectations 105
5.7.3 Stochastic Optimization and Incremental Learning 112
5.7.4 Model Selection 114
5.8 Online Motion Planning 115
5.9 Implementation Details 118
5.10 Experimental Results 120
5.10.1 Experimental Platform Setup 120
5.10.2 Model Training 124
5.10.3 Online Planning 139
5.11 Summary & Discussion 146

6 Conclusion and Future Directions 149

Bibliography 153

Curriculum Vitae 175

List of Publications 177

1

Introduction

Since the advent of mobile robotics in the late 1940's, sparked by William Walter Grey's *Machina speculatrix*, autonomous systems have greatly developed. Their capabilities to perform complex tasks to support us in unwelcome or dangerous duties grew ever since with advances in electronics and sensory devices, computing power, and algorithmic design. However, as opposed to robotic manipulators, appointed to perform repetitive factory tasks with a high level of precision, mobile robots have not yet experienced the widespread use. Nowadays, one finds mobile robots gradually taking on delivery tasks for industrial warehouse automation, but truly autonomous mobile systems have hardly found their way into everyday applications.

The slow adoption of mobile systems in our daily routine may stem from the fact that these systems and humans will have to share common workspaces, and by that, must be equipped with mechanisms to enable safe and efficient coexistence. Paying close attention to the own decision-making process in the presence of other agents when driving a car or roaming the streets, one may notice that the ability to reason about others' intentions plays a crucial role for this. However, the increase in complexity of the motion planning problem is immense when moving from static environments, over dynamic but fully predictable environments, to environments shared with other agents where the robot must base its actions on such reasoning. The richness of information that humans exploit for this purpose, is to this extent not available on a robot.

Despite the challenges, mobile robotic systems offer great potential—reaching far beyond pure commercial benefits. Agricultural robotics may increase productivity to feed the world. Autonomous search and rescue units can find accident victims in inaccessible areas and provide rudimentary treatment. Mobile robots can support overburdened hospital staff or physically impaired persons in their daily routines. And energy-efficient self-driving vehicles can help in our fight against climate change, and reduce lethal as well as non-lethal traffic accidents.

Especially self-driving vehicles have fascinated mankind for a long time. Yet it is fact that—despite the recent hype—automated driving features at the present time are mainly restricted to driver assistance systems or to low-complexity environments such as highways. We argue that this can be accounted to the major challenge of reasoning among other decision-making traffic participants. This thesis thus aims to develop motion planning algorithms for automated vehicles with a focus on navigation in shared workspaces.

1.1 Motivation and Objectives

In order to take on tasks not limited to restricted and controlled domains, autonomous mobile robots eventually need to navigate among and interact with human traffic participants. Motion planning hereby constitutes an essential task in the algorithmic processing chain. It is the final instance which aggregates all information available to the robot and computes feasible motion commands for the platform. In most applications, the motion plans have to navigate the robot from its start location to its destination while circumnavigating obstacles extracted from its sensor(s). Many applications call for the adherence to further criteria such as safety distances, smooth behavior, compliance with traffic rules or, in shared workspaces, mitigation of other traffic participants. Additionally, motion plans have to respect the physical capabilities of the robotic platform, referred to as system-compliant motion planning.

System-compliant motion planning is mandatory for constrained platforms such as cars, but challenging even in static environments. Motion plans violating the vehicle's kinodynamic constraints cannot be faithfully executed by the platform and the discrepancy between the motion planning and execution layer might lead to suboptimal, if not even dangerous behavior. Especially for car-like vehicles at low speeds, the constraints dominate the platform's motion capabilities. Motion planning for such kinodynamically constrained systems has been widely studied, yet the excessive computation time of the methods originating from this research complicates the use for online planning, in which the robot has to be able to quickly react to unforeseen events.

Depending on the geometry of the robot and the characteristics of the environment the robot has to navigate in, the need for *fast and accurate collision detection* might add to the complexity of the motion planning problem. As collision detection often comprises the major computational burden for motion planning algorithms (LaValle, 2006), this part may become a bottleneck if approximation of the robot's geometry with simple shapes is not acceptable due to space constraints in tight environments.

Those situations are encountered for self-driving vehicles with elongated collision geometries operating in tight spaces such as parking lots.

Especially *dynamic, shared environments* call for sufficiently fast planning cycles, as the future evolvement of the environment can never be exactly forecasted. However, moving from static to dynamic environments, the motion planning problem gets considerably more difficult as the time dimension, which can be safely neglected in static environments, now has to be taken into account (Fiorini et al., 1998). In most applications of interest, dynamic elements in the world represent other (decision-making) agents, and suddenly the robot's designer is faced with the question in which way the robot should interact with those other agents. The defensive approach, in which the robot tries to interfere as little as possible with the other agents' future movement, is problematic, as it will cause the robot to "freeze" as soon as the number of other agents or the uncertainty about their future motion increases (Trautman and Krause, 2010). Vice versa, the opposite approach of completely ignoring other agents is obviously not an option as it yields antisocial and potentially dangerous behavior. The robot thus has to take into account the presence of other agents as well as the fact that it itself represents an entity that other agents will react to. This type of *interaction-aware motion planning* problem is notoriously hard, as the robot has to reason about other agents' possible future motion, conversely dependent on its own. In this setting, historically used performance metrics optimized by the robot such as travel time or path length have to be complemented by new elements related to the social compliance of its behavior. Research in this field is still young and only few approaches developed therein have shown to be capable of operating a robot in a real-world setting.

The goal of this thesis is to address these aforementioned shortcomings to pave the way for—but not limited to—future self-driving cars in complex, dynamic environments. The thesis is embedded in the V-Charge (Automated Valet Parking and Charging for e-Mobility)[1] project, targeting fully automated driving in mixed traffic on parking lots with low-cost sensors only. The project's goal is to boost the end-user acceptance of electric vehicles by offering an intelligent valet parking and charging functionality to compensate for their two main deficiencies of reduced driving ranges and increased refueling times. For valet parking, the aforementioned issues of system-compliant motion planning (important for low-speed navigation on parking lots) in tight spaces with mixed traffic have to be addressed. Figure 1.1 depicts the V-Charge self-driving vehicle operating in three typical environments to highlight these challenges. In this thesis, algorithms capable of guiding the self-driving vehicle in mixed-traffic will be developed, implemented and tested on the real platform

[1] `http://www.v-charge.eu`

for experimental validation. The scope of this thesis is however not limited to the autonomous driving domain, and approaches developed are as well applied to other platforms navigating in shared environments.

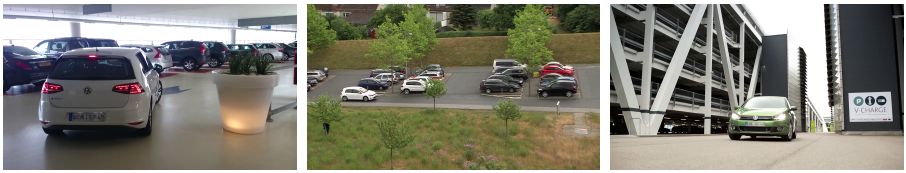


Figure 1.1: The V-Charge (Automated Valet Parking and Charging for e-Mobility) self-driving vehicle in three of the parking lots and parking garages it was tested in. The narrow environments call for highly system-compliant planning and accurate collision detection. For future operation in mixed traffic with pedestrian activity and human-operated cars, this thesis investigates cooperative motion planning approaches. *Left:* Excellence parking garage, Shiphol Airport, Amsterdam. *Middle:* Mobile Life Campus, Wolfsburg, Germany. Credit: Volkswagen AG. *Right:* Bosch Parkhaus, Stuttgart, Germany. Credit: Robert Bosch GmbH.

1.2 A Survey on Motion Planning for Self-Driving Cars

At this point, we consider it necessary to make the reader familiar with the related work and state of the art in motion planning for self-driving cars. It will allow us to position the approach and contributions of this thesis within the scientific field. This section is additionally intended as a brief introduction to various problem settings arising in automated driving.

The general motion planning problem involves the computation of a feasible curve through the robot's free configuration space from its initial configuration to a goal configuration. As there exist in general more than a single feasible curve, usually an additional objective to be optimized is specified. This objective may trade off different desiderata such as for instance to maximize safety distances to obstacles, maximize smoothness, minimize fuel consumption, or minimize travel time. When planning in dynamic environments, the robot's configuration space needs to include a notion of time. The free configuration space, i.e. the set of admissible configurations of the robot, usually encodes collision-free configurations with respect to obstacles, however may also describe other holonomic constraints. Already at this point, the formulation of the problem may be discussed controversially. It is not carved in stone, that the solution must lie in the robot's free configuration space, i.e. must be collision-free. In case collisions are inevitable, the robot may have to trade off the

severity of such, hence rendering collisions rather a factor in the objective than a hard constraint.

1.2.1 Software Architectures

Motion planning for automotive applications is especially subject to differential (non-holonomic) constraints of the platform. Slow-moving cars need to obey the "rolling-without-slipping" condition, that prevent them from moving sideways instantaneously. The inherent difficulty of the motion planning problem described led to hierarchical software architectures for automated driving, comprised of *mission planning*, *maneuver planning*, *trajectory planning* and *control*. On higher levels, solutions to simplified problem instances are computed and passed to lower levels for refinement within a narrower scope. As an example, in order to solve the problem of going from A to B in a city, first a route on a low-fidelity road network is planned (*mission planning*), neglecting obstacles and constraints of the platform. While following the route, a *maneuver planning* module performs high-level decisions, such as stopping at an intersection or changing lanes. *Trajectory planning*, often subdivided into *path planning* and *velocity planning*, then computes a feasible trajectory over a limited time horizon, taking into account platform constraints, obstacles and other traffic participants. As existing approaches still struggle to incorporate highly detailed system models in the *trajectory planning* stage, *control* algorithms on the lowest level account for remaining modeling errors of the *trajectory planning* module. Since *maneuver planning*, *trajectory planning* and *control* is executed while driving, algorithms employed therein must provide real-time capabilities in order to react timely to unforeseen events. Predictions about the evolution of the environment lose their validity rapidly. This necessitates continuous re-planning, which, due to the difficulty of the planning problem, can only be satisfied locally.

The example given above describes the planning task in a structured city environment. Not all environments a self-driving car may find itself in bear such a clear structure. For instance, open spaces as parking lots often do not provide obvious roads, which is why software architectures often have dedicated high-level planning modules for structured and unstructured environments.

Within this thesis, and accordingly within the survey presented in this section, we focus on the low-level modules dedicated to *trajectory planning* and *control*. As *trajectory planning* corresponds to the original motion planning problem, improving on such will reduce the need for, or at least the gap to, other modules within the architecture.

The three dominating methods for system-compliant motion planning are (a) state-lattices, (b) Rapidly-Exploring Random Trees (RRTs), and (c) variants of the Dynamic Window Approach (DWA) . In the following, we will give a brief summary of approaches utilizing these methods or variants thereof, with a focus on related work targeting autonomous driving. We will furthermore discuss approaches to object prediction, required for planning in dynamic environments, as well as work on interaction-aware and adaptive planning techniques. A more detailed differentiation to the approaches taken in this thesis will be presented later in Sections 3.1, 4.1 and 5.2.

1.2.2 State-Lattices

State-lattices, originally presented in Pivtoraiko and Kelly (2006) and later refined in Pivtoraiko, Knepper, et al. (2009) and Pivtoraiko and Kelly (2011), construct an implicit directed graph in the vehicle's state-space (cf. Figure 1.2). The state space is discretized and a system-compliant two-point boundary value solver (e.g. Dubins (1957), Reeds et al. (1990), or Kelly et al. (2003)) establishes system-compliant connections between states in an offline precomputation stage. During execution, the graph is searched for the minimum cost path, for instance with Dijkstra's algorithm (Dijkstra, 1959), $A^\star$ (Hart et al., 1968) or the more sophisticated anytime dynamic $A^\star$ (Likhachev, Ferguson, et al., 2005). State-lattice planners are able to handle kinodynamic constraints of the robot and provide resolution completeness and optimality guarantees, however they suffer from the curse of dimensionality in higher-dimensional state-spaces, describing the fact that the number of discretized states grows exponentially with the dimensionality of the state-space and leading to a combinatorial explosion. This renders the approach unsuitable for *highly* system-compliant planning in real-time, as desired continuity in certain state variables requires those to be included in the description of the state-space. Additionally, solving the two-point boundary value problem for kinodynamically constrained platforms is far from being trivial and often requires simplifications of the vehicle model during construction of the graph. Yet, the offline construction of motions comes with opportunities to save online computation time, as it allows for instance for the precomputation of information facilitating online collision detection (Pivtoraiko, Knepper, et al., 2009).

A state-lattice planner was used on Carnegie Mellon's vehicle in the DARPA Urban Challenge (Likhachev and Ferguson, 2008; Ferguson, Howard, et al., 2008 and Urmson et al., 2008) for path planning in unstructured areas. However computation time in the range of up to several seconds necessitated an underlying faster local trajectory planner. A variant of state-lattice search, named hybrid $A^\star$, that counteracts

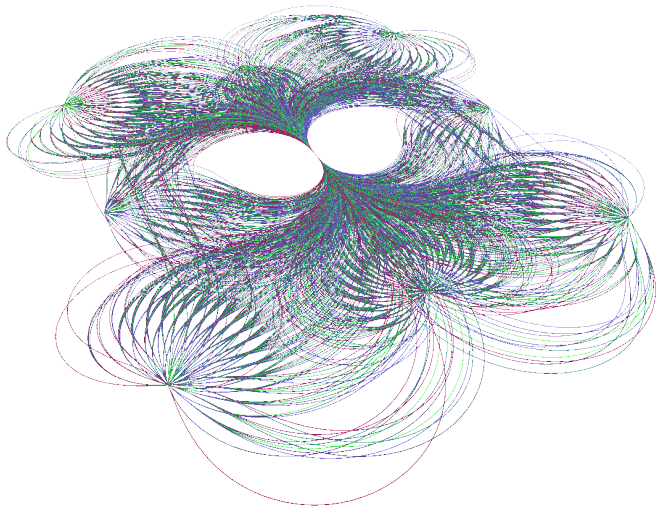


Figure 1.2: An exemplary state-lattice for a car-like vehicle containing 2D position, heading and curvature in the description of its state-space. Kinematic constraints of the vehicle are enforced by construction of feasible edges in the graph.

artifacts arising from the discretization of the state-space by retaining an association with underlying continuous states was used in Dolgov et al. (2010) for planning in static, unstructured areas. This modification however negates optimality guarantees, and resulting paths still had to be smoothed in a post-processing step to attenuate the effects of discretization artifacts. To overcome the issue of potentially bad alignment of the lattice structure, especially the heading dimension, in structured environments, Rufli et al. (2010) proposed a bendable lattice in state- and input-space able to deform around a reference path.

State-lattices, in their original form, are designed for static environments. In order to apply the concept in dynamic environments, Ziegler et al. (2009) span a lattice in the vehicle's state-time space with quintic polynomials that likewise adapt to the shape of the road. Unfortunately, the benefits of state-lattices to perform offline preprocessing are negated when the lattice structure depends on online information such as the shape of a reference path. Yet Ziegler et al. (2009) report very fast computation times of 20 ms, however, to the best of our understanding, without considering static obstacles or accurate collision shapes of dynamic obstacles. Quintic polynomials as motion primitives additionally constrain the type of system models this approach is

suitable for. McNaughton et al. (2011) demonstrate that spatio-temporal state-lattices may achieve real-time capabilities with static and dynamic obstacles utilizing a GPU (graphics processing unit) implementation, however with low fidelity in higher-order system states and ignoring the rotation of other vehicles during dynamic collision detection.

1.2.3 Randomized Sampling

In contrast to the deterministic sampling of the state space employed in state-lattices, probabilistic sampling schemes prove superior when dealing with high-dimensional state spaces. In the late 1990s, randomized sampling techniques gained popularity due to the work of LaValle (1998). The presented Rapidly-Exploring Random Tree (RRT) algorithm expands a tree in the robot's state space by biasing it towards large Voronoi regions. This guarantees a quick exploration of the state space and probabilistic completeness. However, due to the resulting tree structure, no optimality guarantees can be provided. This issue was later tackled by Karaman et al. (2009) in the Rapidly-Exploring Random Graph (RRG) algorithm by building an incremental graph, and in the RRT* algorithm (Karaman et al., 2011) via a local tree rewiring procedure.

RRT methods have been successfully applied to the motion planning problem for cars. Macek et al. (2006) use low-dimensional RRT search to a goal region specified by a higher-level waypoint planner in conjunction with spline smoothing to compensate for jerky solutions produced by randomized search. Results are provided in simulation for a dynamic, urban traffic scene utilizing circular approximations of other traffic participants and constant velocity predictions. The authors however mention issues regarding computation time, even for this simple setup. In Jeon et al. (2013), the RRT* algorithm is used to plan trajectories for a car-like vehicle, considering maximum frictional tire forces. However, reported computation times of several seconds prohibit online application and separation of path and velocity computation renders the approach unsuitable for systems with pronounced dynamic steering constraints. Dynamic objects were neglected and static collision detection restricted to lane boundary checks. Garrote et al. (2014) employ the RRT* algorithm to navigate a vehicle along a reference path in a simulated dynamic environment. Dynamic objects are approximated with circular collision shapes and their future motion is assumed to follow a constant velocity model. Unfortunately the authors do not provide any runtimes to assess the real-time applicability of the implementation.

1.2.4 Local Trajectory Rollout

The Dynamic Window Approach (DWA) by Fox et al. (1997) was among the first algorithms to introduce a system model into the planning process. Exhaustive search takes place in the system's input space, where the assumption is made that chosen inputs are held constant over a certain time interval into the future. Fox et al. originally targeted differential-drive platforms for which system inputs are given by translational and rotational velocity in a two-dimensional input space. Maximum velocity and acceleration constraints are respected via a rectangular window in the velocity search space centered at the robot's current velocity.

The simplistic assumption of constant future system inputs is clearly a limiting factor. However—with increasing computational power over the years—more versatile future maneuvers could be evaluated and DWA variants were among the most prominent online trajectory planning algorithms at DARPA's Urban Challenge autonomous vehicle contest in 2007. This one-step trajectory rollout scheme is used in Ferguson, Howard, et al. (2008); Hundelshausen et al. (2008); Montemerlo et al. (2008); Bohren et al. (2008); Werling et al. (2011); L. Ma et al. (2012) and Broggi et al. (2012), proving its applicability during real-world tests. The methods locally explore alternative driving maneuvers by either sampling in action- or in state-space. The latter is more suited for navigation in structured environments as samples may be tailored towards the lane structure. Ferguson, Howard, et al. (2008) create local paths via the two-point boundary value solver (Kelly et al., 2003) used for graph construction within their state-lattice planner, and later overlay different velocity profiles. Dynamic objects are predicted with multiple hypotheses utilizing road network information (Ferguson, Darms, et al., 2008). Hundelshausen et al. (2008) use different sets of circular arcs for varying vehicle speeds, however not influenced by the lane structure and thus less expressive for onlane driving. Other traffic participants were not considered as dynamic. Montemerlo et al. (2008) use a vehicle simulation undergoing different steering parameters. Whether dynamic objects were considered in the trajectory planning stage is unclear. Werling et al. (2011) construct minimal jerk trajectories through quintic polynomials in the Frenet frame, i.e. a moving frame along a smooth reference curve. The authors account for dynamic objects over time via collision detection with conservative bounding circles. L. Ma et al. (2012) employ cubic Bézier curves.

1.2.5 Prediction of Other Traffic Participants

Motion prediction for other traffic participants can be categorized into *physics-based*, *maneuver-based*, and *interaction-aware* prediction techniques. Lefèvre et al. (2014)

give a comprehensive overview about related work regarding these approaches. For the sake of brevity we only summarize the main aspects and refer the reader to their work for references.

Physics-based models, such as constant velocity or constant acceleration models, constitute the simplest form of prediction models. The future movement is solely determined by the laws of physics. They are widely used, as they are computationally cheap and easily incorporate kinematic or dynamic system models.

The longer the prediction horizon, motion is less and less governed by laws of physics and increasingly more influenced by the environment and the driver's intent or goal. Note that motion prediction may often be casted as a motion planning problem, thus the distinction between both is blurred. *Maneuver-based* prediction models incorporate an estimate of the driver's intent to perform a certain maneuver. Estimates are predominantly obtained through supervised learning techniques such as artificial neural networks, support vector machines (SVM), or hidden Markov models. Maneuver-based prediction models are mainly suitable for structured environments with a small corresponding action set such as intersections and usually provide better long-term predictions than physics-based models. The work of Karasev et al. (2016) is notable, as it obtains stable long-term predictions by considering both environmental constraints and agents' intents.

In both physics-based and maneuver-based prediction methods, objects are treated independently of each other. Taking inter-agent dependencies into account leads to more accurate long-term predictions within *interaction-aware* methods. In addition to the references provided in Lefèvre et al. (2014), we see parallels to interaction-aware models in tracking applications (Pellegrini, Ess, Schindler, et al., 2009 and Pellegrini, Ess, and Van Gool, 2010) and crowd simulation (Treuille et al., 2006; K. H. Lee et al., 2007 and Berg, Patil, et al., 2008). Tracking greatly benefits from more accurate object predictions in the case of occlusions, and crowd simulation software likewise aims at creating realistic joint motion patterns.

1.2.6 Interaction-Aware Planning

Human drivers possess the capability of anticipating the consequences of their own actions on the actions of other traffic participants. Without exception, the above mentioned motion planning approaches neglect mutual influences of traffic participants on each other, if other agents are considered at all in a dynamic setting. They treat the vehicle for which the plan is acquired as non-existent from the perspective of other traffic participants, and hence as exclusively responsible for collision avoidance. The defensive behavior arising from that may quickly lead to an inability to move at all

when the number of surrounding traffic participants increases (Trautman and Krause, 2010). This negligence of cooperation exists presumably because existing approaches already struggle with computational issues when planning for a single agent in the presence of kinodynamic constraints and obstacles. Yet it is not only the increased complexity of the motion planning problem itself, but also the necessity to formulate a modified objective, that more heavily depends on "soft" factors describing socially accepted behavior. Deriving models for well accepted social behavior is a problem with interdisciplinary character, with connections to psychology and social sciences.

Only a handful of real-time capable, interaction-aware approaches have emerged throughout the last decade, none of them to our best knowledge applied to car navigation. The Optimal Reciprocal Collision Avoidance (ORCA) formulation (Berg, Guy, et al., 2011) extends the pioneering work of Fiorini et al. (1998) on Velocity Obstacles (VO) to account for joint collision avoidance effort. Based on the assumption of linear future motion and known preferred velocities of all agents, a linear program in relative velocity space is formulated, which can be solved efficiently for thousands of agents. Constraints in velocity space imposed by other agents are shifted to account for mutual avoidance efforts. ORCA is derived from the Reciprocal Velocity Obstacles (RVO) formulation by (Berg, Lin, et al., 2008), extending it with a suitable linearization procedure of the non-convex velocity constraints found in RVO to yield an efficient program. The method was later extended by Alonso-Mora et al. (2013) to account for nonholonomic constraints, and a recursive filter was suggested by Kim et al. (2014) to estimate the agents' preferred velocities in an online fashion.

In contrast to the deterministic RVO approach, Trautman and Krause (2010) and Kuderer, Kretzschmar, Sprunk, et al. (2012) formulate a probability distribution over composite trajectories. Interactions of agents are captured through coupling terms. Kretzschmar et al. (2014) show superior predictive quality of the latter probabilistic approach over RVO. We will provide a more detailed overview concerning interaction-aware motion planning in Section 5.2.

1.2.7 Adaptive/Trainable Motion Planning

Every motion planner possesses at least a few tuning parameters that affect the planner's performance or alter its resulting behavior. Silver et al. (2013) showed that, with algorithms becoming more and more complex in general, human engineers may encounter difficulties in manually determining (near-)optimal parameters. Additionally, different situations and user preferences may call for different parameter sets, making manual tuning more and more cumbersome.

In the last decade, particularly reinforcement learning (RL) and inverse reinforcement learning (IRL) methods have emerged as the methods of choice. In the RL setting, a direct mapping from state to action is learned given a predefined reward function, whereas IRL aims to reveal the underlying reward function from observed expert behavior. Riedmiller, Montemerlo, et al. (2007) use RL in conjunction with an artificial neural net to learn a model-free path controller for a car. Silver et al. (2013) use IRL to learn driving styles and maneuver preferences from human demonstrations in unstructured environments. Kuderer, Gulati, et al. (2015) build on the framework of maximum entropy IRL (Ziebart, Maas, Bagnell, et al., 2008) to learn characteristic driving styles for on-road driving from human demonstration. Previously the authors employed the framework in a cooperative planning problem (Kuderer, Kretzschmar, Sprunk, et al., 2012; Kuderer, Kretzschmar, and Burgard, 2013 and Kretzschmar et al., 2014). Sadigh et al. (2016) employ IRL to learn a human reward function for driving, and use the reward function for interaction-aware planning incorporating game-theoretic aspects. Very recent approaches as in Bojarski et al. (2016) and Santana et al. (2016) attempt to learn an end-to-end mapping from sensor inputs to control actions with deep networks architectures.

1.2.8 Summary and Conclusion

Katrakazas et al. (2015) compiled an excellent up-to-date survey on motion planning techniques for self-driving vehicles. The authors highlight shortcomings and provide suggestions for future research directions. The identified shortcomings are:

- system-compliant trajectory planning in the presence of acceleration, braking and steering constraints
- too simplistic approximations of obstacle shapes for collision detection
- dealing with imperfect sensory data
- missing interaction-awareness in mixed traffic
- missing real-world evaluation

These findings are in line with the overview of the state of the art given in this section. Incorporating detailed vehicle dynamics in a real-time capable framework is still challenging. If planning in dynamic environments is considered at all, often too simplistic assumptions on the objects' shapes and future motions are made. Most often, experiments are only provided in simple simulation environments. Research and results on trainable, interaction-aware planning methods are in an experimental stage at the forefront of current research.

In line with these findings, this thesis especially addresses the shortcomings of system-compliant planning, collision detection and interaction-aware planning in dynamic

environments. Taking algorithms out of simulation and lab environments is a major effort that we took in order to analyze their suitability for real-world application. All algorithms presented in this thesis were applied to autonomous platforms interacting with other uninformed traffic participants in realistic environments.

1.3 Terminology

In this section we want to discuss briefly—on a high level—some specific terminology used within this thesis due to the absence of a standard terminology.

Local/Partial and global motion planning In contrast to *global planning*, where a motion from start to the ultimate goal state is computed, *local planning* usually limits the horizon of the motion planner in either space or time to make complicated problems approachable within a reasonable computation time. Global planning approaches often operate on simplified versions of the full motion planning problem, neglecting for instance dynamics of the environment or kinodynamic constraints of the platform. In contrast, local or partial planning methods must operate with a sufficient level of detail since they usually constitute the lowest layer responsible for computing final control commands.

Real-time/Online capability Real-time software refers to programs that must execute with guaranteed upper bounds on execution time. Within this thesis, we use the term real-time rather to express that a motion planning algorithm must produce an answer "fast enough" to be suitable for navigating a mobile robot through an only partially predictable environment. What "fast enough" means, depends on the environment's dynamics and on the capabilities of the sensors employed. In this thesis, we propose to call algorithms with a runtime of up to 500 ms, similar to a descent human reaction time (Green, 2000), as real-time capable. This definition is without question motivated by our focus on low-speed applications. Interchangeably, we will use the term *online*.

Reactive and predictive planning In this thesis, we will distinguish between *reactive planning* and *predictive planning*. We will follow the definition that reactive planning approaches compute *a single, immediate action* based on the current context. In contrast, predictive planners compute a *sequence of actions*, usually while employing a model to forecast the evolution of the environment into the future.

Cooperative/Interaction-aware planning We describe motion planning approaches as *cooperative* which make use of the assumption that other agents in the environment are aware of the presence of the robot and will interact with it in a helpful, non-hostile manner. While in the strict sense of the word, *interaction-aware* solely describes the

algorithm's awareness that other agents can be interacted with and does not specify whether this interaction is friendly or not, we will use this term interchangeably with cooperative. Consequently, a motion planner ignoring the potential for interaction will be called *non-cooperative* or *non-interaction-aware*.

1.4 Approach

This section summarizes the approaches taken to address the scientific challenges outlined above related to (a) system-compliant motion planning in real-time, (b) accurate collision detection for planning in narrow environments, and (c) motion planning in shared environments.The thesis will first present a non-interaction-aware motion planning strategy, built around traditionally employed tree-search methods, and suitable for kinodynamically constraint platforms operating in tight navigation spaces. The approach was amongst others motivated by project requirements, as the motion planning layer needed to be available at an early stage, and similar approaches have proven their suitability (cf. Section 1.2.4). The advantages and drawbacks of this framework will be discussed, and—as a logical next step to improve on the navigation performance in the close presence of other traffic participants—we present an interaction-aware framework based on a maximum entropy model of joint navigation behavior.

Onlane Motion Planning for Autonomous Cars

The planning approach developed in Chapter 3 is designed to guide a mobile robot along a reference path, mitigating static obstacles and dynamic objects, while easily accounting for arbitrary system dynamics and actuator constraints. A reference path is often simple to obtain in structured environments as found in driving applications where a sequence of lanes to be followed can be easily computed on a topological road network. In unstructured environments, a reference path can still be computed via global optimization techniques such as RRT planners (LaValle, 1998) or state-lattice planners (Pivtoraiko and Kelly, 2006). Hereby, the optimization is usually performed on a low-fidelity version of the problem, neglecting for instance dynamic constraints of the robot and dynamic elements in the scene to remain computationally feasible. Also, some specific applications like T&R (Teach & Repeat), where the robot's task is specifically to navigate along a previously driven route, naturally give rise to a reference path. Indeed, the onlane motion planner developed in this thesis was applied to both lane following (Schwesinger, Versari, et al., 2015) and T&R (Krüsi et al., 2015) for the mobile platforms depicted in Figure 1.3.

Figure 1.3: The onlane motion planner developed in this thesis operated both the rough terrain robot *ARTOR* (left, credits: Krüsi et al. (2015)), and the self-driving vehicle *Wolle* (right, credits: Volkswagen AG).

The onlane motion planner operates in a (deterministic) sampling-based, non-cooperative and receding horizon fashion. Sampling-based optimization allows to achieve a high degree of system-compliance as explained in a moment, whereas the limited time horizon is a necessary compromise to achieve real-time capability. Sampling-based, as opposed to gradient-based optimization, also finds global optima and offers the advantage of an easier objective function design as even non-smooth functions are handled. However, sampling-based motion planning with a high level of (kino-dynamic) system-compliance is challenging, as planning takes place in a large state-space. With the limited computation time budget for online planning in dynamic environments, the sampling procedure must generate expressive candidate motions to avoid the waste of computation time on samples with a very low probability of contributing to the optimal solution. Therefore, generating samples whose endpoints align with a reference path is popular among existing planning frameworks for automated driving (Urmson et al., 2008; Bacha et al., 2008; Montemerlo et al., 2008; Bohren et al., 2008 and Werling et al., 2011). By doing so, one reduces the full state-space to a manifold embedded in such, and avoids the danger of entering unsafe end states. In contrast to previous work, where candidate motions are constructed from geometric primitives that have to be tested for compliance with the system model later, we obtain inherently system-compliant samples via a forward integration of the robot's (closed-loop) system dynamics. The planner directly outputs a sequence of system states and control inputs. The sequence of system states may be used for feedback control, albeit in general the quality and rate of computed control inputs is good enough for feedforward control only. For instance, on the mobile robot ARTOR (cf. Figure 1.3, left), an underlying trajectory controller was completely omitted. Diverse samples are generated by employing a simulated path controller to regulate the simu-

lated system onto laterally shifted versions of the reference path. As a path controller usually exists in a robotic navigation framework, and by that a description of the system model to tune the controller, our approach is easily implemented and transferred to different platforms.

In Section 3.3.6, we propose an objective function comprised of different desiderata such as a small deviation from the reference path and a reference speed profile and safety distances to static obstacles. In order to assess the risk of candidate motions with respect to other traffic participants, we employ the time-to-collision (TTC) metric (D. N. Lee, 1976) due to its reported usefulness as risk indicator (Lefèvre et al., 2014). Dynamic object predictions for the TTC metric are computed using road network information for other vehicles encountered. A physics-based constant velocity model is employed for pedestrians, due to the absence of labeled pedestrian walkways or other useful semantics of the environment. With small situation-specific adaptations to the objective function, behaviors like platooning and yielding to crossing pedestrians emerge naturally from our framework.

Time-aware Collision Detection with Dynamic Objects

The TTC metric used in the onlane motion planner requires the detection of collisions of candidate ego trajectories with predicted trajectories of other agents in space and time. Collision detection in general constitutes the main computational burden for motion planning algorithms (LaValle, 2006). Therefore, the few existing frameworks capable of real-time motion planning in space-time apply conservative simplifications to the objects' collision geometries (Macek et al., 2006; Werling et al., 2011 and Garrote et al., 2014) or neglect their orientation (McNaughton et al., 2011 and Lawitzky et al., 2012) in order to speed up collision tests. In our case, both approaches turned out to be unacceptable for navigation in tight, densely occupied parking lot environments. A conservative approximation of vehicles' collision shapes with outer bounding circles eliminates too much valuable space for navigation, which is especially noticeable when navigating through narrow parking rows.

Therefore, we develop an approach in Chapter 4 based on bounding volume hierarchies (BVHs) in workspace-time space, which allows us to replace the largest number of expensive oriented box tests with cheaper collision tests. It provides fast queries if and at which point in time a candidate ego trajectory collides with any object prediction, without making any restrictive assumptions on the objects' movement or collision geometry. By specifying an extent of collision objects in the time dimension, one is able to conveniently enforce a minimum time-gap. This way, the ego vehicle will not visit a position in the world that was or will be occupied by another traffic

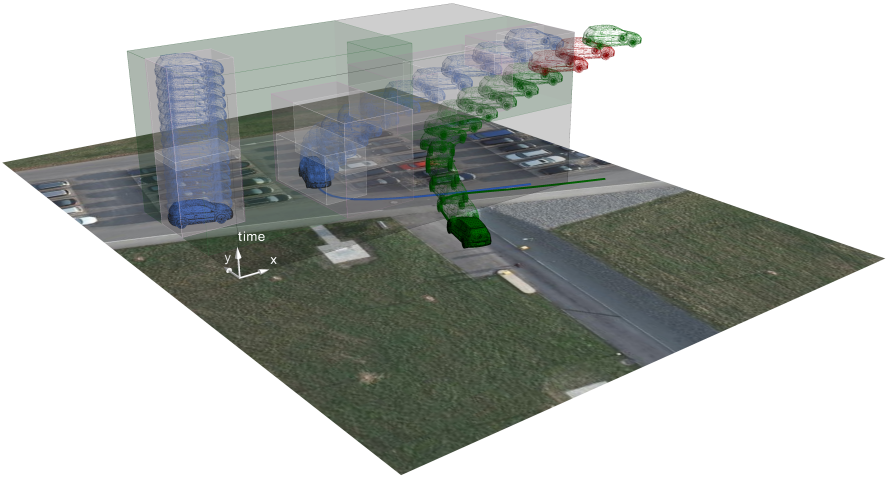


Figure 1.4: Exemplary bounding volume hierarchy using axis-aligned bounding boxes in workspace-time space. The horizontal plane represents the vehicle's 2D workspace, the vertical axis represents time. Candidate ego trajectories (green) can be tested efficiently for collision (red) against predicted dynamic object trajectories (blue). For instance, a collision of the ego vehicle with the stationary vehicle on the left at any point in time can be ruled out with a single axis-aligned bounding box overlap test.

participant shortly before or after. An exemplary situation is depicted in Figure 1.4, where the collision query reports a violation of the minimum time-gap with another vehicle.

Cooperative Motion Planning

Ignoring interactions in the planning stage is without doubt a simplification, yet often performed to make planning in dynamic environments approachable given real-time constraints. If at all, interactions are taken into account on a higher, discrete decision level for highly structured environments (Lawitzky et al., 2012 and Schwarting et al., 2014). However, we argue that these approaches fail to generalize to unstructured environments, where the small discrete action sets are not sensible.

A significant amount of research still focuses on obtaining discrete predictions of agents' decisions, for instance whether another vehicle will turn left or right. Therefore, mainly environmental semantic features are leveraged, ignoring interactions

between individual agents in the world. Later those predictions are used in a non-cooperative planning and decision framework. For highly structured environments equipped with strong traffic rules, this method might give acceptable results as interactions are dominated by those rules. Again we argue that these methods will fail as soon as automated driving is tackled in environments with less structure and traffic rules.

Figure 1.5: A challenging scenario for mobile robot navigation. An autonomous agent must interact with surrounding pedestrians and bicyclists in an unstructured navigation space with fuzzy traffic rules. Source: www.gehlpeople.com/cases/new-road-brighton-uk

We are convinced that, in order to eventually realize automated driving in environments such as depicted in Figure 1.5, motion planning must be approached within a holistic, interaction-aware framework in a continuous state- and action space. In addition, we advocate the use of cooperative planning to cope with noise-affected measurements of other agents' positions and velocities—an issue that will become important once cooperative algorithms make it out of simulation frameworks onto real platforms equipped with cheap sensors. In Section 5.10.3.2 we show that the assumption of cooperative behavior indeed reduces the negative effects of sensor noise on the reaction of the ego robot. Without the assumption of cooperative behavior, and limited available semantic information to constrain future positions of other agents, measurement noise can invoke harsh (and from an outside view unnecessary) reactions of the robot when the noise suggests that other agents might enter its path. We

also believe that adaptable models are important to reduce maintenance effort of such, and allow the models to adjust to new situations.

For that, we learn a maximum entropy probability distribution over joint navigation behavior on a trajectory level from real-world demonstrations of interacting agents, building on the work of Abbeel and Ng (2004); Ziebart, Maas, Bagnell, et al. (2008) and Kuderer, Kretzschmar, Sprunk, et al. (2012). The maximum entropy distribution is a natural choice to resolve the ambiguity that many different distributions may explain the demonstrations observed. Among these distributions, the maximum entropy distribution is the most uniform one, subject to the constraint that the expectations of a set of measurable functions—or feature functions—match the empirical expectations of the demonstrations. In the maximum entropy distribution, each feature comes with a linear weight that is optimized within the training stage such that the model best explains demonstrated joint navigation behavior. The cooperative aspect of the approach is introduced via feature functions acting on trajectories of multiple agents simultaneously. Most importantly, the trade-off between different desiderata can be learned from real-world demonstrations, relieving one from time-consuming manual adjustments of the model and naturally equipping the robot with a human-like behavior.

Figure 1.6: The pedestrian assistant robot *Obelix* in its role as a guide for visitors. The robot serves as the platform for experimental validation of the maximum entropy cooperative planning framework.

The cooperative planning and learning approach is implemented on the pedestrian assistance robot *Obelix* of the European project EUROPA2 (European Robotic Pedes-

trian Assistant 2), using solely onboard sensing for both tasks. This robotic platform enables real-world experimental validation of the framework in (mildly) crowded environments for which permissions for a self-driving car would hardly be granted at the moment. Figure 1.6 depicts the robot at an exhibition in Switzerland in its role as a guide for visitors of the event.

1.5 Organization

After summarizing the contributions of this thesis in Chapter 2, the remainder of the thesis will dive into the details of the different elements briefly outlined above. Chapter 3 describes the partial onlane planning framework, developed to navigate a self-driving car along a reference path. Chapter 4 provides details regarding the collision detection approach for dynamic objects employed in the onlane planner. Chapter 5 describes the interaction-aware motion planner based on a maximum entropy distribution to capture joint navigation behavior. Finally, we conclude the thesis in Chapter 6 and provide potential directions for future research.

2

Contributions

This thesis makes the following three main contributions:

- **System-compliant motion planning:** A novel framework for highly system-compliant planning for low-speed application in structured, narrow environments. The approach was published in Schwesinger, Rufli, et al. (2013) and Schwesinger, Versari, et al. (2015).
- **Collision detection in space-time:** A fast collision detection approach based on bounding volume hierarchies in workspace-time space, published in Schwesinger, Siegwart, et al. (2015).
- **Real-world cooperative motion planning with maximum entropy models:** A flexible probabilistic model for interaction-aware motion planning is trained from demonstrations of cooperative behavior. Our work in this context was published in Pfeiffer et al. (2016).

We provide a compressed summary of these contributions in Section 2.1, Section 2.2 and Section 2.3. Chapter 6 lists all publications that originated from the doctoral studies. In Section 2.5, we give an overview of supervised student projects, and Section 2.6 illustrates some achievements of the related V-Charge (Automated Valet Parking and Charging for e-Mobility) project during the period in which the author of this thesis engaged as scientific coordinator.

2.1 System-Compliant Motion Planning

We develop a novel partial motion planning framework for highly system-compliant planning in structured, narrow environments. In Section 1.2.8, we highlighted that

existing approaches face difficulties in capturing complex system constraints characteristic for autonomous cars (including steering angle, steering rate, and acceleration limits, as well as reaction delays) explicitly. Most of the related sampling-based approaches are based on geometric primitives instead of actual vehicle models, separating the computation of velocity profiles from the geometric construction of the path. This separation may result in conflicts with non-holonomic platform constraints—especially at low speeds. Consequently, trajectories need to be validated in a post-processing step—which may in turn lead to the pruning of a considerable amount of candidate motions.

The approach developed in this thesis fully addresses these important real-world platform and environmental constraints without having to revert to model inversion techniques. It proposes a *generic* sampling-based framework requiring only (i) a (potentially even non-smooth) reference path, (ii) a system description suitable for numerical forward propagation, and (iii) a control law. Its output consists of a system-compliant command and state sequence, which may be employed for feed-forward control of the platform directly. Real-time capability, while planning in a high-dimensional configuration-time space, is retained through a reference path aligned manifold (Werling et al., 2011). With our approach, we uniquely propose a method to exploit the guidance provided by a reference path, while allowing for any system model suitable for forward simulation.

The proposed motion planning framework was successfully implemented on two mobile robotic platforms. The robot *ARTOR* implemented the framework in the context of adaptive route following in a Teach & Repeat (T&R) setting. We refer the reader to Krüsi et al. (2015) for results. On the V-Charge self-driving car, the framework successfully performed autonomous navigation in narrow, dynamic parking lot environments, using low-cost perception sensors only. We demonstrate in simulation and on the V-Charge car that behaviors such as platooning and yielding to crossing pedestrians naturally emerges from the framework, particularly through space-time collision detection. Our extensive real-world tests performed with the framework on mixed-traffic parking lots provide important experimental evidence that interaction-aware planning techniques will be necessary to improve on the robustness of motion planning in close proximity to other traffic participants.

2.2 Collision Detection in Space-Time

This thesis develops a bounding volume hierarchy (BVH) data structure in workspace-time space for fast collision detection of candidate trajectories in sampling-based motion planning frameworks for dynamic environments. Existing real-time applicable

techniques make restricting assumptions or simplifications either on the object's motion (Lawitzky et al., 2012) or the object's shape (Werling et al., 2011). Our approach provides fast responses to collision queries with almost no restrictions on particular motions or shapes of the objects and is thus well suited for application in a broad variety of planning environments. Though the idea of using partitioning techniques in workspace-time is not new (Cameron, 1984; Cameron, 1985 and Cameron, 1990), we are not aware of any application within an online motion planning framework, especially not within the autonomous driving domain. This thesis highlights various implementation considerations and provides detailed runtime statistics to facilitate decisions regarding the usage of this approach within other online planners operating in dynamic environments. We see great potential of the method for application in spatio-temporal state-lattices, where the accelerating data structures can be precomputed in an offline stage.

2.3 Cooperative Motion Planning via Maximum Entropy Models

In this thesis, a trainable, cooperative motion planner using a probabilistic model of joint agent behavior is developed. It extends the ideas developed in Kuderer, Kretzschmar, Sprunk, et al. (2012) to ultimately enable cooperative navigation with the model in realistic real-world environments. The most important contribution in this context is the training, application and evaluation of the approach in realistic environments shared with other uninstructed pedestrians without the need for external sensing devices. The model is trained with motion tracks of pedestrians recorded in public spaces with onboard sensors of the robot only. Uniquely, we demonstrate online navigation with the learned interaction-aware model among pedestrians in realistic everyday situations. The trainable, interaction-aware model and the self-contained perception system allows for independent, autonomous agents equipped with social etiquette, that can adapt to new environments.

Several contributions in detail made this achievement possible. The described contributions may lack the required context for the reader not familiar with the related work on cooperative maximum entropy models at the current point in the document. We kindly ask those readers to refer to Chapter 5 for details and return to this section afterwards.

- Instead of planning trajectories to a goal position within a fixed distance and optimizing for the travel time as shown in Kuderer, Kretzschmar, Sprunk, et al., 2012, we adapt the formulation to a receding horizon formulation with a

fixed time horizon and variable end positions of trajectories. This eliminates the requirement to find a set of fixed waypoints for agents with an unknown destination and facilitates the deployment of the robot in unmapped environments. It furthermore removes the strong commitment towards a high-level solution from a global waypoint planner.

- We explore the use of stochastic optimization techniques in the training stage of the model in order to cope with the substantial training time, and to address potentially non-stationary objectives as found in life-long learning settings.
- In Kretzschmar et al. (2014), a mixture distribution over homotopically distinct composite trajectories is used. However, homotopies vanish in the receding horizon formulation employed in this thesis. We investigate an extension to standard Markov chain Monte Carlo (MCMC) designed for multimodal densities to compute sample approximations of feature expectations (required for the training step) directly without the need to manually enumerate different homotopies.
- We contribute to the design of C^1-continuous features for cooperative navigation, that enables the use of faster Quasi-Newton optimization techniques during online planning.

We evaluate the approach under realistic settings in order to reduce the gap between academic research and real-world application. The benefit of interaction-aware motion planning is assessed during navigation in a public space, in which the robot traveled a total distance of over 4 km. We are able to show improved navigation performance with the cooperative model in comparison to a non-cooperative counterpart among pedestrians instructed to behave normally around the robot. However, the high attention the robot received from uninstructed pedestrians and their interest in testing its basic braking functionality revealed a substantial difference between human-human interaction used as training data for the robot's behavioral model, and human-robot interaction. In another extensive experiment, we show the potential of cooperative planning to reduce over-conservative robot behavior in the presence of substantial measurement noise. The assumption of cooperation enables the robot to predict a participation of other agents in mutual collision avoidance, and with that alleviates drastic reactions invoked in non-cooperative planners through noisy object detections. Finally, we highlight improvable parts of our developed model based on the experience with pedestrians' behavior recorded at a variety of places, and hope to direct future research with this.

2.4 List of Publications

This section provides a complete list of publications, in chronological order, that were published during these doctoral studies.

Publications Included in This Thesis

The following publications are related to the main contributions of this thesis. Results of those publications are included in this document at hand.

- Schwesinger, U., Rufli, M., Furgale, P., and Siegwart, R. (2013). "A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 391–396. DOI: 10.1109/IVS.2013.6629500
- Schwesinger, U., Siegwart, R., and Furgale, P. (2015). "Fast Collision Detection Through Bounding Volume Hierarchies in Workspace-Time Space for Sampling-Based Motion Planners". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–68. DOI: 10.1109/ICRA.2015.7138981
- Schwesinger, U., Versari, P., Broggi, A., and Siegwart, R. (2015). "Vision-only Fully Automated Driving in Dynamic Mixed-traffic Scenarios". *it - Information Technology* 57.4, pp. 231–242. DOI: 10.1515/itit-2015-0005
- Pfeiffer, M., Schwesinger, U., Sommer, H., Galceran, E., and Siegwart, R. (2016). "Predicting Actions to Act Predictably: Cooperative Partial Motion Planning with Maximum Entropy Models". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329

Related Publications

The results of this thesis directly contributed to the following related work. However, no content is included here.

- Furgale, P., Schwesinger, U., Rufli, M., Derendarz, W., Grimmett, H., Mühlfellner, P., Wonneberger, S., Rottmann, J. T. S., Li, B., Schmidt, B., Nguyen, T. N., Cardarelli, E., Cattani, S., Brüning, S., Horstmann, S., Stellmacher, M., Mielenz, H., Köser, K., Beermann, M., Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Iser, R., Triebel, R., Posner, I., Newman, P., Wolf, L., Pollefeys, M.,

Brosig, S., Effertz, J., Pradalier, C., and Siegwart, R. (2013). "Toward Automated Driving in Cities using Close-to-Market Sensors: An Overview of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*. Gold Coast, Australia, pp. 809–816. DOI: 10.1109/IVS.2013.6629566

- Derendarz, W., Iser, R., Mühlfellner, P., Wonneberger, S., Nguyen, T. N., Brosig, S., Effertz, J., Mielenz, H., Mimura, K., Beermann, M., Schwesinger, U., Rufli, M., Furgale, P., Timpner, J., Rottmann, S., Wolf, L., Häne, C., Heng, L., Lee, G., Pollefeys, M., Grimmet, H., Triebel, R., Posner, I., Newman, P., Cardarelli, E., Cattani, S., Bertozzi, M., Brüning, S., Horstmann, S., Stellmacher, M., and Siegwart, R. (2013). "Vollautomatisches Laden und Fahren auf öffentlichen Parkplätzen". In: *Internationaler Kongress Elektronik im Fahrzeug*. VDI-Berichte 2188. Baden-Baden, Germany: VDI Wissensforum GmbH

- Furgale, P., Krüsi, P., Pomerleau, F., Schwesinger, U., Colas, F., and Siegwart, R. (2014). "There and Back Again—Dealing with highly-dynamic scenes and long-term change during topological/metric route following". *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*

- Krüsi, P., Bücheler, B., Pomerleau, F., Schwesinger, U., Siegwart, R., and Furgale, P. (2015). "Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching". *Journal of Field Robotics* 32.4, pp. 534–564. DOI: 10.1002/rob.21524

- Schwesinger, U., Bürki, M., Timpner, J., Rottmann, S., Wolf, L., Paz, L. M., Grimmett, H., Posner, I., Newman, P., Häne, C., Heng, L., Lee, G. H., Sattler, T., Pollefeys, M., Allodi, M., Valenti, F., Mimura, K., Goebelsmann, B., Derendarz, W., Mühlfellner, P., Wonneberger, S., Waldmann, R., Grysczyk, S., Last, C., Brüning, S., Horstmann, S., Bartholomäus, M., Brummer, C., Stellmacher, M., Pucks, F., Nicklas, M., and Siegwart, R. (2016). "Automated Valet Parking and Charging for e-Mobility: Results of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden, pp. 157–164. DOI: 10.1109/IVS.2016.7535380

Other Publications

The author contributed to the following publication, unrelated to the work presented in this thesis.

- Bircher, A., Alexis, K., Schwesinger, U., Omari, S., Burri, M., and Siegwart, R. (2017). "An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees". *Robotica* 35.6, pp. 1327–1340. DOI: 10.1017/S0263574716000084

2.5 Supervised Student Projects

This section provides a list of all student projects supervised by the author as part of the doctoral studies.

Master Thesis

Six months full time research project

- Wilko Schwarting, 2016: "Parallel Autonomy in Automated Vehicles: Trajectory Generation with Real-time Obstacle Avoidance and Human Input Optimization"
- Andre Widmer, 2014: "Learning Socially Compliant Navigation for Robot Path Planning"
- Martin Zellweger, 2013: "Local Motion Planning for Trajectory Following in Dynamic Environments"

Semester Thesis

Semester-long part time research project during the master studies

- Florian Tschopp, 2016: "360° Object Detection and Tracking with 3D Laser and Camera Data"
- Vaibhav Bhatla, 2013: "Motion Pattern Learning for Traffic Participants using Gaussian Processes"

Bachelor Thesis

Semester-long part time research project

- Timon Homberger, 2014: "Path Optimization for an Artistic Autonomous Sand Drawing Robot"
- Wilko Schwarting, 2014: "Cooperative Motion Planning in Dynamic Highway Traffic"

- Sebastian Verling & Julian Zilly, 2013: “Modeling and Control of a VTOL Glider”
- Basil Weibel, 2013: “Design and Modeling of an Efficient Wing-Only Vertical Take-Off and Landing Airplane”

Studies on Mechatronics

Semester-long introduction to the basics of non-experimental scientific research

- Wilko Schwarting, 2014: “Cooperative Motion Planning in Dynamic and Uncertain Environments”

Focus Project

Two-semester product development project in a team undergraduate students

- Simon Kober, Daniel Lehmann, Sebastian Verling, Remo Büchele, Basil Weibel, Felix Winkler, Julian Zilly, Marc Faude, 2012 - 2013: “PacFlyer - The VTOL Glider”

2.6 Project Management

In the period of January, 2015 - September 2015, the author of this thesis engaged as scientific coordinator of the European Project V-Charge. During this period, the project was publicly demonstrated to a technical audience at Volkswagen’s *Mobile Life Campus* in Wolfsburg, Germany, and to international media in the *Excellence Parking* garage at Amsterdam Shiphol airport. The project was featured in the Euronews Futuris episode “Look, no hands! the self-driving car on the road to reality” (09/2015), and awarded 2015 with the *Connected Car Award* by the German magazine *Auto Bild*.

3

Onlane Motion Planning for Autonomous Cars

This chapter incorporates material from the following first author publications:

Schwesinger, U., Rufli, M., Furgale, P., and Siegwart, R. (2013). "A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 391–396. DOI: 10.1109/IVS.2013.6629500

Schwesinger, U., Versari, P., Broggi, A., and Siegwart, R. (2015). "Vision-only Fully Automated Driving in Dynamic Mixed-traffic Scenarios". *it - Information Technology* 57.4, pp. 231–242. DOI: 10.1515/itit-2015-0005

3.1 Context and Related Work

Related work on *local* motion planning & motion control is commonly separated into approaches that compute solution curves directly, and such that require a reference path (or trajectory) for guidance. In the former case, classic collision avoidance methods (Borenstein et al., 1991 and Fox et al., 1997) have gradually been replaced by state lattice graphs (Pivtoraiko and Kelly, 2006 and Rufli et al., 2010) and Rapidly-Exploring Random Tree (RRT) (LaValle, 1998) methods. Nonetheless, these methods remain restricted to approximately four dimensional state spaces if online operation is desired—thereby limiting the fidelity of the agent models they can faithfully reproduce. Consequently, they typically find application in situations where no external structure can be extracted for guidance, such as off-road or on large-scale parking lots. If a reference path (or trajectory) is available for guidance, higher-dimensional state-space representations become approachable. Road and lane information in particular enact narrow constraints on heading and curvature values. A common approach is

thus to align the endpoints of local trajectory samples with the reference path (Urmson et al., 2008; Bacha et al., 2008; Montemerlo et al., 2008; Bohren et al., 2008 and Werling et al., 2011). This technique simultaneously reduces the search complexity and overcomes the danger of entering unsafe end states. Most of these approaches are based on geometric primitives instead of actual vehicle models, separating the computation of velocity profiles from the geometric construction of the path (Bacha et al., 2008; Montemerlo et al., 2008; Bohren et al., 2008 and Ziegler et al., 2009). This separation may result in conflicts with non-holonomic platform constraints—especially at low speeds. Consequently, trajectories need to be validated in a post-processing step, which may in turn lead to the pruning of a considerable amount of candidate motions (Werling et al., 2011). In contrast, the local on-road planner described in Urmson et al. (2008) generates motion samples by solving two-point boundary value problems subject to the dynamics of a detailed vehicle model directly (Howard et al., 2007). Unfortunately, this method is prone to convergence issues related to the non-invertibility of non-holonomic vehicle models.

We thus observe that existing approaches face difficulties in capturing system constraints characteristic for autonomous cars (including steering angle, steering rate, and acceleration limits, as well as reaction delays) explicitly. Furthermore, non-holonomic constraints cannot be integrated—nor inverted—in closed form. The approach developed in this thesis fully addresses these important platform constraints, and is detailed in the following sections. We proceed with a short digression describing the self-driving vehicle platform constituting the basis for the work in this chapter.

3.2 Robotic Platform

The V-Charge (Automated Valet Parking and Charging for e-Mobility) self-driving car prototype, depicted in Figure 1.3, is based on a Volkswagen e-Golf platform. It is equipped with front- and rear facing stereo cameras with a horizontal field of view (FOV) of 45° and 120°, respectively. Four 1.3 Mpx monocular fisheye cameras with a nominal FOV of 185°, synchronously triggered at 12.5 Hz, provide 360° surround view. The prototype wide-angle rear stereo camera uses a 5 cm wide baseline and the same fisheye lenses as the monocular cameras. 12 ultrasonic sensors are used for close-range obstacle detection. The sensor setup depicted in Figure 3.1 is complemented by stock odometers. In the beginning of the project, two traditional combustion engine cars were used, equipped with an identical sensor setup as described above. The more evolved longitudinal dynamics of the combustion engine cars contributed towards our decision to work on a highly-system compliant motion

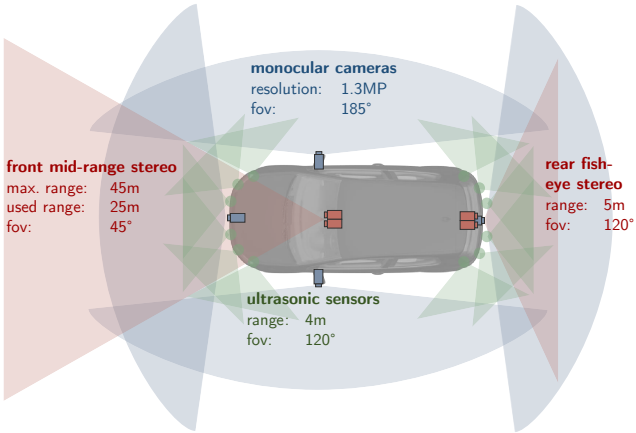


Figure 3.1: Schematic coverage of the sensor suite employed on the V-Charge self-driving car. Monocular fisheye cameras provide a 360° surround view. Front- and rear facing stereo cameras deliver direct depth information for obstacle avoidance. 12 stock ultrasonic sensors cover the close-range areas at the front and the back of the car.

planning framework. Within the project, the maximum speed of the vehicle is limited to 2.78 m/s = 10 km/h in the low-level control units of the car. The desired speed is usually set to 1.6 m/s = 5.8 km/h in order to allow safety personal to conveniently follow on foot.

3.3 Motion Planning Framework

This section elaborates on the motion planning framework, as employed on our self-driving car platform. First, we sketch a high-level overview of the algorithmic processing chain in Section 3.3.1. The main contribution of this section consists of a novel framework for partial motion planning along reference paths, detailed in Section 3.3.2. In its generic form, the framework is suitable for a wide range of systems and not limited to car-like platforms. We then proceed with an application of the framework to our use case, automated driving. Sections 3.3.3 and 3.3.4 describe a system model and a control law, suitable to be used for car-like vehicles within the framework. In Section 3.3.5 we briefly outline the collision detection with respect to the static environment, dynamic collision detection is described in Chapter 4. Section 3.3.8 describes the non-cooperative object prediction employed in the framework. In Section 3.3.6, the objective function optimized by the framework is presented, before we proceed to the experimental results in Section 3.4. The final

Section 3.5 discusses the results and motivates our research on cooperative motion planning.

3.3.1 System Architecture

A high-level overview of the algorithmic processing chain—reduced to the relevant part for the partial onlane planner presented in this thesis—is depicted in Figure 3.2, starting with the sensor data (left) and eventually resulting in system inputs to the vehicle (right). The key sensors to perceive the environment are the front stereo camera

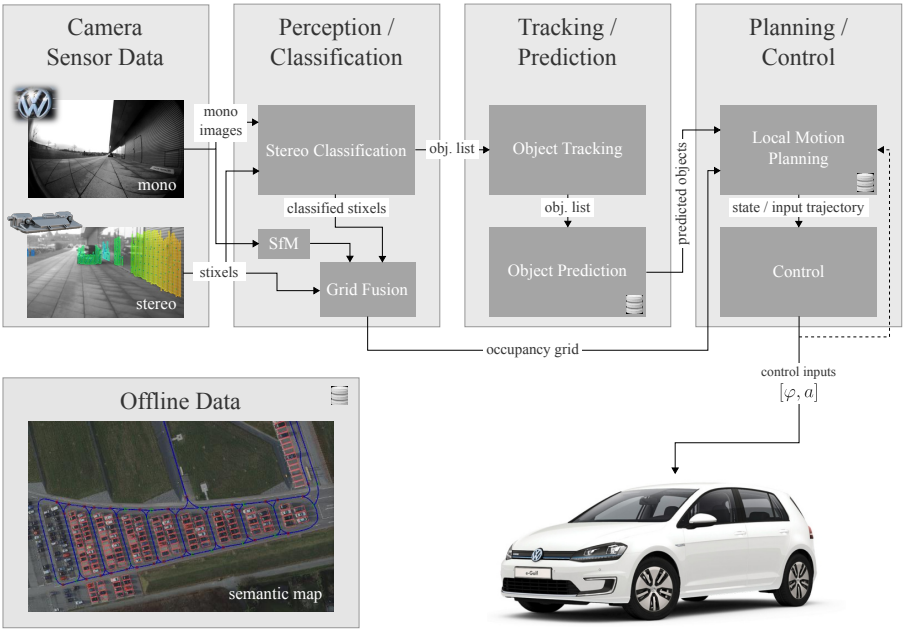


Figure 3.2: Overview of software modules involved in the onlane motion planning framework. The processing chain reaches from raw sensor data (left) to the system inputs sent to the platform (right).

as well as the front and side monocular cameras. The monocular cameras provide the inputs to a structure from motion module (SfM) from Häne et al. (2015). The 2.5D stixel output of the front stereo sensor (colored bars) is on the one hand used as an input to a *Grid Fusion* module. This module creates a two-dimensional (2D) binary occupancy grid of the environment, containing static information only. On the other hand—together with the images from the front monocular camera—stixels form the

input of the *Stereo Classification* module, which groups them to objects and assigns class labels to them from a database of known object classes (Schwesinger, Versari, et al., 2015). Stixels assigned to dynamic objects are removed from the grid in order to have a clear separation between static and dynamic elements in the world. We found this separation of uttermost importance, as the vehicle would clearly try to circumnavigate static elements early. Stixels misclassified as static, for instance originating from a leading car while driving, would invoke lateral reactions of the following automated vehicle. In the *Object Tracking* module, the classified objects are tracked with an extended Kalman filter (EKF), applying suitable motion models for propagation based on the classification result. This module was provided by Volkswagen AG in the context of the V-Charge project. Once classified and tracked, predictions of the future movement of objects are generated (*Object Prediction*), taking into account the object's dynamic model and further infrastructure information such as the road network stored in a semantic map database (Grimmett et al., 2015). The *Local Motion Planning* module, described in the following sections, computes a safe motion plan based on the latest perceptual information and forwards the plan in form of a time-stamped sequence of reference states plus feed-forward system inputs to an underlying *Control* module. Speed limits are annotated in the semantic map. The *Control* module takes care of correcting remaining modeling errors from the *Local Motion Planning* module. The system inputs to the vehicle computed by the *Control* module are given by steering wheel angle, φ, and translational acceleration, a. They are fed back to the local motion planner to correctly propagate the system states during the expected cycle time of the planner and to model system dead times, i.e. the time needed for an input to show a response at the system output.

The rear fisheye stereo sensor as well as the ultrasonic sensors (not displayed) are used for close-range obstacle detection during parking maneuvers. The ultrasonic sensors additionally serve as the input to an emergency brake module. The vehicle is localized through a monocular visual localization system (Mühlfellner et al., 2015) relative to a pre-built map containing sparse visual landmarks.

3.3.2 Algorithm

The proposed planning framework consists of a path-aligned manifold embedded in the vehicle's full state space, a system model, and a control law. Search operates on a tree that is constructed online. Nodes, n, represent particular vehicle state / time pairs, $(\mathbf{x}, t)$, while edges (synonymously candidate motions), e, form system-compliant trajectory segments. An outline of a single planning cycle is portrayed in Algorithm 1 and explained in the following.

Algorithm 1 One cycle of the partial onlane planner

Require: Initial state $\mathbf{x}_0$, current time t_0, commands $\mathbf{u}_0$ of previous solution, system model $\mathbf{f}$, control law $\mathbf{g}$, objective J, and auxiliary controller variable A (default $A = \emptyset$)

```
N ← ∅,  E ← ∅,
n ← (x_0, t_0),  N ← N ∪ {n}
(n_0, e_0) ← simulate(n, f, g, T_cycle, u_0)
N ← N ∪ {n_0},  E ← E ∪ {e_0}
for l = 1 → desired tree depth do
    for all nodes n at tree level l do
        for j = 1 → |M × A| do
            ξ = (d_ref, Δv_ref, a) ← drawSample (M × A)
            v_ref(s) = v_path(s) + Δv_ref
            (n̂, ê) ← expand(n, f, g, d_ref, v_ref(s), a, T_sim)
            N ← N ∪ {n̂},  E ← E ∪ {ê}
        end for
    end for
end for
computeCosts(E, N, J)
n* ← minimum cost node at last tree level
e* ← edge sequence leading to n*
return e*
```

The search routine operates online at a fixed rate in accordance with environmental requirements. During initialization, any remaining nodes in N are removed. Then, the most recent system state, $\mathbf{x}_0$, obtained from the localization module is inserted at the root of the tree at time t_0. To account for the planning time, T_{cycle}, during which the planner is busy, the root vertex is expanded according to the initial segment of the solution found during the previous search iteration (if available). Actual search thus begins from $(\mathbf{x}_1, t_0 + T_{\text{cycle}})$.

During node expansion, successor edges are constructed by (numerical) forward propagation of a general user supplied vehicle model, $\mathbf{f}$, with

$$\mathbf{x}(t + \Delta t) = \mathbf{f}([\mathbf{x}(t), \dot{\mathbf{x}}(t), \ldots], \mathbf{u}(t), t, \Delta t) \ . \tag{3.1}$$

Uniquely, this allows to incorporate arbitrary system descriptions—even non-linear and non-invertible ones that are hard to work with in competing approaches. A potential dead time, i.e. the duration until the system responds to a command, can be model in the user-supplied system model, $\mathbf{f}$. In order to contain the size of the state space when using high-dimensional system representations, a control-based approach building on terminal manifolds is enforced. The notion of the terminal manifold (Werling et al., 2011) refers to a reduced subspace of the state-time space towards which

node expansion is directed. In our framework it takes the form of heading and curvature alignment (θ and κ) with respect to a user-supplied reference path,

$$\mathbf{z}(s) = \begin{bmatrix} \theta(s) - \theta_{\text{ref}}(s) \\ \kappa(s) - \kappa_{\text{ref}}(s) \end{bmatrix} = \mathbf{0} \; \forall s > s_0 \,, \tag{3.2}$$

leaving the desired lateral offset to the reference path, d_{ref}, as well as the translational velocity, v_{ref}, at the curvilinear abscissa, s, as free variables. The alignment is enforced for all states ahead of the current curvilinear abscissa, s_0. Along the reference path, the manifold, $\mathbf{z}$, is then sampled into a set of submanifolds $M = D \times V$, built from a discrete set of lateral path offsets, D, and offsets from a reference velocity profile, V. If no velocity profile is provided with the reference path, the profile can be assumed to be constant zero without loss of generality and offsets restricted to the non-negative real line. Note that position is not sampled and consequently M contains rather submanifolds of $\mathbf{z}$, which naturally gives rise to a tree in the system's full state space. An auxiliary set A enables the user to modulate for instance the aggressiveness of trajectory segments via different controller gains. During node expansion, samples $\boldsymbol{\xi} = (d_{\text{ref}}, \Delta v_{\text{ref}}, \mathbf{a})$ are iteratively drawn from the Cartesian product $M \times A$ of the set of submanifolds and auxiliary variables, and $(d_{\text{ref}}, v_{\text{ref}}(s))$ is used as target state for closed-loop system control. Note that the system's reference velocity, $v_{\text{ref}}(s)$, is a function of the curvilinear abscissa along the reference path, and computed through linear combination of a reference velocity profile, $v_{\text{path}}(s)$, with the desired offset, Δv_{ref}. The function `expand` uses a user-supplied controller, $\mathbf{g}$, with $\mathbf{u}(t) = \mathbf{g}(\mathbf{x}(t), d_{\text{ref}}, v_{\text{ref}}(s), \mathbf{a})$ to regulate the forward-simulated system model $\mathbf{f}$ towards $(d_{\text{ref}}, v_{\text{ref}}(s))$ over a fixed time period, T_{sim}. The controller hereby serves as a link between the task specification—namely reaching a certain submanifold—and the potentially complex system model. The newly retrieved state and its incoming trajectory segment are pushed back onto Q as a node / edge-pair, $(\hat{n}, \hat{e})$.

Once the tree has been expanded to a desired search depth (or maximum planning time has elapsed), edge and accumulated node costs are computed given the objective J. Then, the lowest cost node at maximal search depth is identified and back-traced to the root node $(\mathbf{x}_0, t_0)$, which results in the construction of the solution trajectory, $e^\star$. Depending on the fidelity of the system model employed during edge construction, inputs corresponding to the solution trajectory may be directly executed on the platform as feed-forward commands. This was realized on both platforms the partial onlane planner has been employed on (cf. Figure 1.3). On the V-Charge self-driving car platform an additional underlying trajectory controller provided further feedback stabilization utilizing the corresponding time-stamped system states of the solution.

Figure 3.3 illustrates a search tree created using a coarse sampling of the terminal manifold. For illustrative purposes, only one velocity offset and three lateral offsets

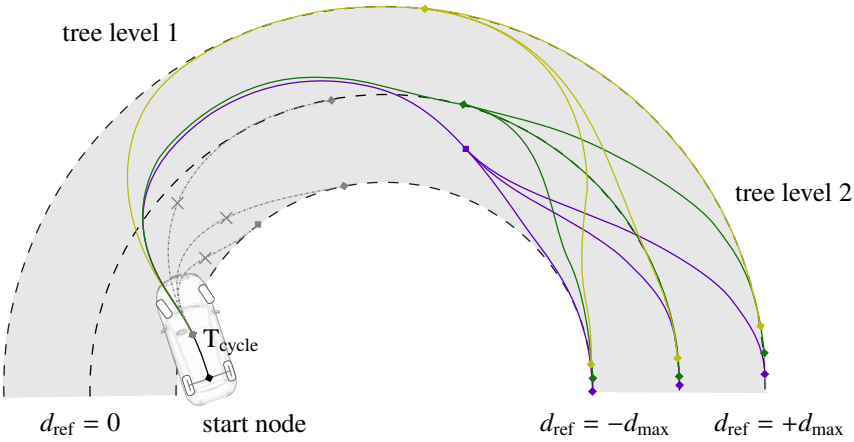


Figure 3.3: An exemplary tree composed of system-compliant trajectory segments. The first edge is appended to compensate for the expected planning cycle time and corresponds to the initial part of the solution found during the previous search iteration. For subsequent tree levels, the terminal manifold is (for illustrative purposes) discretized into a coarse set of three lateral offsets $\{-d_{max}, 0, d_{max}\}$ and a single velocity offset. Segments in gray illustrate edges generated in competing approaches in the absence of a detailed vehicle model that have to be pruned after construction due to a violation of vehicle constraints.

are explored in a tree of depth two (ignoring the tree level corresponding to the segment compensating for the cycle time of the planner). In the example's scenario, competing approaches relying on a geometric construction of trajectory segments would need to prune numerous candidate motions due to the sharp turn. Our method instead generates by construction inherently system-compliant candidate motions that approach the submanifold samples as close as possible given the system's kinematic and dynamic constraints.

While we described a deterministic sampling scheme and exhaustive node expansion, the basic idea of closed-loop system control towards samples drawn from a path-aligned manifold can as well be combined with randomized sampling techniques. This would allow for a more diverse simulation of maneuvers, i.e. switches between submanifolds, over the time horizon, as the exponential complexity of the exhaustive tree search outlined here clearly limits the maximum depth. We refrained from randomized search however as the behavior is not predictable given the small number of samples possible to process in real-time. Instead we chose a single tree level to obtain most fine-grained actions within the part of the tree actually executed. Edges within deeper tree levels rather serve to sample the future cost and are never executed due to

replanning. It was very rarely the case that complicated obstacle configurations called for a greater diversity of simulated maneuvers. For heavily cluttered environments, one could consider a combination of initial deterministic and exhaustive search in order to guarantee the exploration of sufficiently diverse immediate actions and avoid jittering solutions over subsequent planning cycles, and randomized search to more efficiently sample the future costs.

3.3.3 System Modeling

In this section's short excursion, the goal is to derive a low-fidelity kinodynamic system model for car-like platforms, suitable for efficient forward simulation within the partial onlane planner. The scope is limited to the domain where the vehicle is operated far away from its stability limits and on surfaces with sufficient traction such as paved roads, to ensure that effects like tire slip may be safely neglected. While the kinodynamic system model is kept generic to some extent, the lateral- and longitudinal transfer functions presented in this section introduce aspects specific to the V-Charge platform. However the considerations therein presumably apply to a broader range of vehicle platforms.

3.3.3.1 Kinodynamic Single-Track Model

The kinodynamic single-track model combines 2D position, (x/y), heading, θ, longitudinal speed, v, and steering wheel angle, φ, plus longitudinal speed and steering wheel angle derivatives up to desired order into its state vector, $\mathbf{x}$. Figure 3.4 gives a graphical explanation of the variables. The dynamics are then captured by

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \vdots \\ \dot{\varphi} \\ \vdots \end{bmatrix} = \begin{bmatrix} \cos(\theta) \cdot v \\ \sin(\theta) \cdot v \\ \frac{v}{L} \tan(\phi(\varphi)) \\ h_1(u_1, v, \dot{v}, \ddot{v}, \ldots) \\ \\ h_2(u_2, \varphi, \dot{\varphi}, \ddot{\varphi}, \ldots) \\ \\ \end{bmatrix}. \tag{3.3}$$

The kinodynamic single-track model translates into a system model as specified in Equation 3.1 via $\mathbf{f} = \mathbf{x}(t_0) + \int_{t_0}^{t+\Delta t} \dot{\mathbf{x}}(t)\, \mathrm{d}t$. The first three entries describe the non-holonomic "rolling-without-slipping" condition, and how the steering wheel angle φ relates to a change in heading via the inter-axle distance L of the vehicle and the mapping $\phi(\varphi)$ from steering wheel angle to single-track steering angle ϕ. The latter two entries relate acceleration $\dot{v}$ and steering wheel speed $\dot{\varphi}$ to the vehicle's control inputs

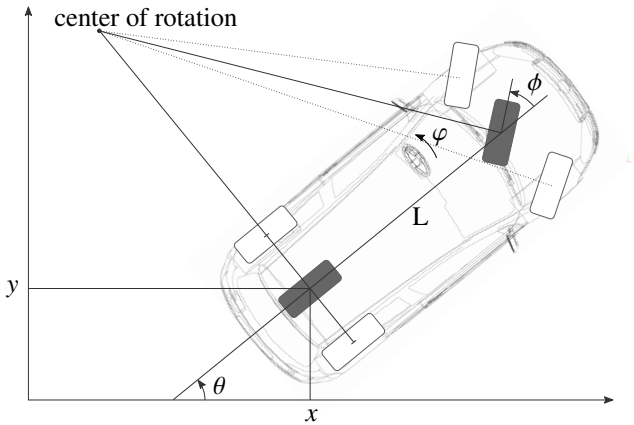


Figure 3.4: Single-track model for a car-like vehicle. The model describes the motion of the vehicle via two imaginary wheels in the vehicle's lateral center. The 2D pose (x, y, θ) is given in a suitable external coordinate frame. The relation between single-track steering angle ϕ and steering wheel angle φ is manufacturer- and model-dependent.

u_1 and u_2 via the mappings h_1 and h_2. The mappings might need higher order derivatives of v and φ, which have to be appended to the state vector in this case. On real vehicles, the longitudinal control input u_1 is often propagated through various (unfortunately black-box) underlying control modules and the vehicle's complex drivetrain. Likewise the control input u_2 related to the steering mechanism of the vehicle is not uniquely specified and differs between platforms. It might represent a commanded steering wheel angle as on the V-Charge platform, or, preferably, commanded steering wheel speed to give the motion planner more authority to control the smoothness and aggressiveness of lateral motions. Therefore we cannot provide a general formula for the mappings h_1 and h_2 at this point. In the following two sections, we elaborate on the mappings for the V-Charge platform.

The single-track model as presented here neglects effects like the side slip angle and tire slip. In case these effects matter, i.e. for high speed maneuvers or slippery roads, they can be easily incorporated into the system model and used within the planning framework. The reader is referred to Schramm et al. (2014) for more advanced vehicle models. State and actuator constraints are likewise easily incorporated during numerical integration.

3.3.3.2 Lateral System Model

For the V-Charge vehicle, we used a first-order model

$$h_2(u_2, \varphi) = \dot{\varphi} = \frac{1}{\alpha_1}(u_2 - \varphi)$$

for the dynamics of the steering wheel angle with lower and upper bounds on the steering speed $\dot{\varphi}$. The time constant α_1 was identified from step responses of the steering system. This yielded a very accurate, low-fidelity model as depicted in Figure 3.5.

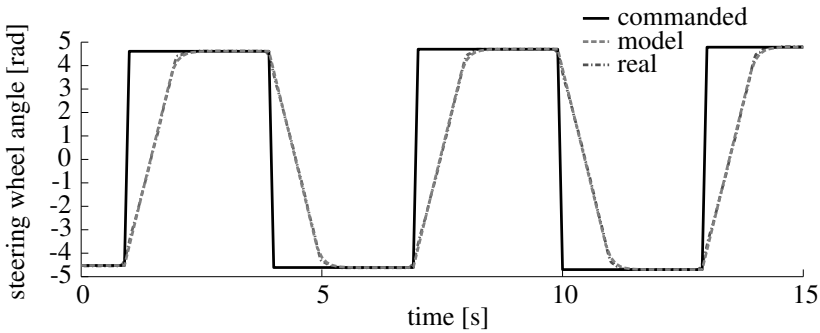


Figure 3.5: Steering wheel angle response of real system and identified first-order model (with bounded steering speed) to step inputs.

The mapping $\phi(\varphi)$ from steering wheel angle, φ, to single-track steering angle, ϕ, is in our case nonlinear. In practice, we identified a third-order polynomial function $\phi(\varphi, \mathbf{p}) = \mathbf{p}^\top \cdot \left(1, \varphi, \varphi^2, \varphi^3\right)^\top$ from driving data assuming the availability of an accurate localization system. In the V-Charge project, the visual localization system of Mühlfellner et al. (2015) provided centimeter-level accuracy and served as the ground-truth localization source for the identification of $\phi(\varphi)$. The optimal parameters $\mathbf{p}^\star$ were found via Least-Squares optimization

$$\mathbf{p}^\star = \underset{\mathbf{p}}{\operatorname{argmin}} \sum_t \|\phi_t - \phi(\varphi_t, \mathbf{p})\|^2 ,$$

where φ_t denote direct steering wheel angle measurements obtained from the attached sensor, and $\phi_t = \tan^{-1}\left(L\dot{\theta}_t / v_t\right)$ represent single-track steering angles computed from localization. Solving the optimization problem, the polynomial coefficients $\mathbf{p}^\star$ are obtained that minimize the sum of squared errors between observed single-track steering angles computed from localization and those computed from steering wheel angle

measurements. The polynomial mapping must intuitively be monotonic—otherwise an increase in steering wheel angle would turn the wheels in opposite direction at some point—which gives a unique inverse mapping $\varphi(\phi)$ that can be found in closed form. A dependency of the steering transfer function on the longitudinal speed was not considered due to the focus on low-speed operation.

3.3.3.3 Longitudinal System Model

Planning in *dynamic* environments mandates precise tracking of time-dependent trajectories. On our platform, the longitudinal response of the car to the input u_1 was modeled as a second-order system

$$h_1(u_1, \dot{a}, \ddot{a}) = a = u_1 + \beta_1 \dot{a} + \beta_2 \ddot{a}\,,$$

with acceleration $a = \dot{v}$. A second-order model provided a good compromise between accuracy and simplicity, favorable for fast forward propagation within the partial onlane planner. The parameters of the second-order system were identified based on step response measurements of the true system. Figure 3.6 shows a comparison of the real system and the model to the acceleration step commands.

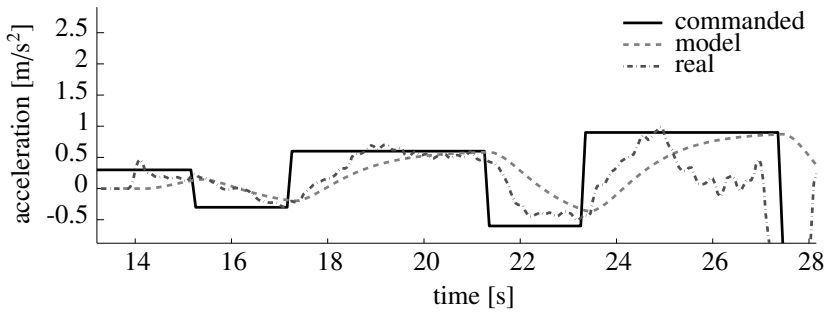


Figure 3.6: Acceleration response of combustion engine car and identified second-order model to step inputs. Acceleration of the real system is derived from wheel odometry. The mismatch at the end exists since the real system approached and eventually hit to maximum translational speed limit of the vehicle, triggering a harsh brake from the vehicle's low-level electronic control units (ECUs).

3.3.4 Controller Design

Control of the single-track vehicle model given in Equation 3.3 onto a set of reference trajectories following the road (and thus the generation of a set of locally expressive trajectory segments) is achieved via the nonlinear state-feedback control law

described in Samson et al. (1991). To decouple lateral from longitudinal control, the reference position is obtained via orthogonal projection of the rear vehicle axle onto the reference path. The projection can be efficiently pre-computed via a Voronoi decomposition of the discretized reference path (Schwesinger, Rufli, et al., 2013), such that the projection reduces to a single lookup in a 2D table. The error states for the lateral control are then given by the heading difference, $\Delta\theta$, and the lateral distance, Δy, to the reference position, both depicted in Figure 3.7. It is advisable to provide

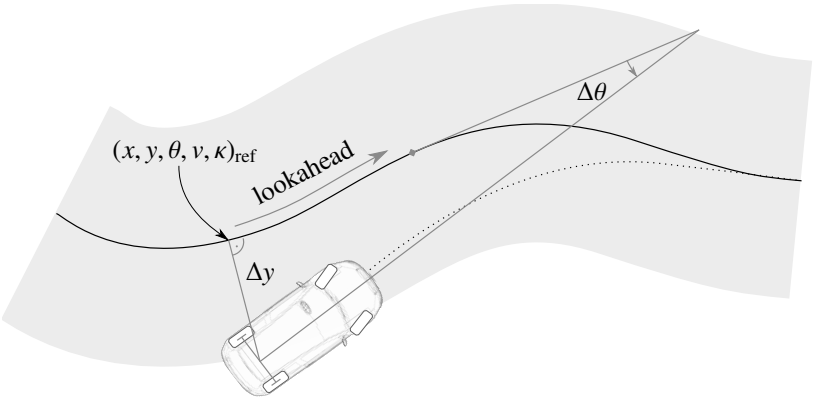


Figure 3.7: Control variables for lateral control of the vehicle onto a reference path. The nonlinear state feedback law regulates the error states $\Delta\theta$ and Δy to zero. A look-ahead for the heading offset $\Delta\theta$ may be required if no curvature information κ_{ref} is provided or maximum steering speed is small.

curvature information κ_{ref} of the reference path to improve on the tracking error. If no curvature information is available, a speed-dependent look-ahead along the reference path may be used to allow the vehicle to react to curvature changes in time. Following Samson et al. (1991), the lateral control output for the vehicle's turning rate ω then is

$$\omega = \omega_f + k_1 v_{\text{ref}} \frac{\sin(\Delta\theta)}{\Delta\theta} \Delta y - k_2 \Delta\theta \,, \tag{3.4}$$

with an optional feed-forward term for the turning rate, $\omega_f = v_{\text{ref}}\kappa_{\text{ref}}$, based on the curvature of the reference path. The single-track steering angle command is then derived from the turning rate via $\phi = \tan^{-1}\left(\frac{\omega L}{v_{\text{ref}}}\right)$, and mapped to a desired steering wheel angle $u_1 = \varphi(\phi)$. Using the reference velocity v_{ref} instead of the actual vehicle velocity was empirically found to work better as it yields a certain look-ahead for the velocity. In practice we found it necessary to use different lateral control gains k_1 and k_2 for different longitudinal speed ranges in order to achieve good convergence

towards the reference path without overshoots. This gain scheduling table was built and manually tuned in simulation.

Longitudinal control is performed via a cascaded acceleration and velocity control loop. With the open-loop longitudinal system model at hand, we tuned a proportional-integral (PI) acceleration controller to remove steady-state acceleration errors, for instance when driving up or down ramps. The integral term allowed us to maneuver the vehicle up ramps in multilevel parking garages as depicted in Figure 3.8. To

Figure 3.8: Autonomous operation in the multilevel parking garage at Shiphol airport, Amsterdam. An integral term in the vehicle's longitudinal control loop handles slopes.

prevent an unbounded increase of the integral term due to acceleration limits present, *anti-windup* mechanisms (Åström et al., 2006) were added.

Additionally, we employed the *deadband reset scheduling* approach by Ender (1997) to reduce oscillations induced by the integral part of the controller in combination with an acceleration deadband present in our system. A deadband, i.e. the interval around zero in which the system does not react to acceleration commands, causes the integral term to grow until the system eventually responds with an overshoot and the process repeats in opposite direction. If the control loop is tuned for fast response, this results in high-frequency oscillations affecting comfort and causing wear and tear of the vehicle's brakes. Deadband reset scheduling refers to de-tuning the integral setting of the controller when the error is small, such that the cycle time of the oscillation is reduced.

The setpoint for the inner acceleration control loop is computed with a standard proportional (P) controller $a_{\text{ref}} = k_v\,(v_{\text{ref}} - v)$. During the forward propagation step in

the planning cycle, the reference velocity, v_{ref}, is computed from the sample $\boldsymbol{\xi} = (d_{\text{ref}}, \Delta v_{\text{ref}}, \mathbf{a})$ drawn from the terminal manifold and the reference velocity profile. During execution of the plan on the real platform, the reference velocity is contained in the solution's state trajectory.

3.3.5 Collision Detection

Collision detection for candidate motions is performed separately for *static* and *dynamic* elements in the world. *Static* collision detection is carried out in 2D, using an occupancy grid representation of the world. The vehicle's rectangular shape is approximated with a collection of N disks as depicted in Figure 3.9. The disk approximation is optimized for a small overhang at the vehicle's side for navigation in tight environments while fully covering the original shape to ensure safety. In order allow the vehicle to pass very narrow passages in the world such as gates on a parking lot, a maximum lateral overhang of 5 cm—measured from the tip of the side view mirrors of the car—is allowed in our framework. The minimum distance of the vehicle to any obstacle in the world is given by $d_{\text{obs}} = \min_n \{d(x_n, y_n) \mid n = 1, \ldots, N\}$. The distance of disk n to any obstacle, $d(x_n, y_n)$, can be efficiently computed via a single lookup on a distance transform of the occupancy grid. A fast algorithm to compute a discretized distance field is given in Felzenszwalb et al. (2012).

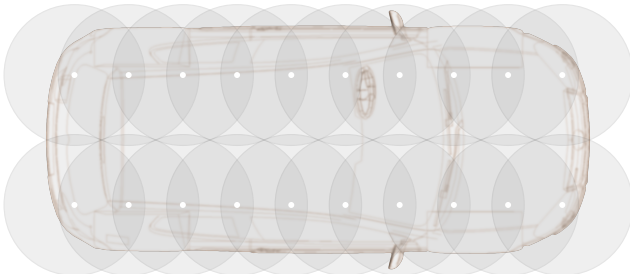


Figure 3.9: Disk approximation of the rectangular vehicle shape with $N = 20$ disks (10 in longitudinal direction, 2 in lateral). The disk approximation allows for fast minimum distance computation of the vehicle's collision geometry to any static obstacle in the world via N lookups on a distance transform of an occupancy grid representation of the world.

Collision detection with *dynamic* elements in the scene is considerably more difficult than static collision detection, as time must no longer be neglected. As one of the contributions of this thesis, a fast, time-aware collision detection approach is developed for this purpose based on partitioning schemes in space-time. The approach returns fast answers if and at which point in time in the future a trajectory is in collision

with any motion prediction of the dynamic objects in the world. This contribution is detailed in Chapter 4.

3.3.6 Objective Function

For each edge, a scalar cost value is computed comprised of a linear combination of different cost terms. Significant effort was spent to design an objective function that performs well in both static and dynamic environments. As static obstacles occupy their position in the world for infinite time, edges in collision with those are assigned infinite cost and not further considered within the search. Each remaining edge is scored with respect to its alignment with the reference path and the lane's reference speed profile. Furthermore, cost terms are introduced for soft penalization of close distances to static obstacles, integrated steering speed and acceleration, and anticipated collisions with dynamic objects. The cost terms and their motivations are summarized in Table 3.1. In the equations, $\mathbf{w}_{_}$ are user-supplied weight vectors, $s(t)$ denotes the curvilinear abscissa along the reference path associated with the vehicle pose at time t, and $\boldsymbol{\xi}_{l=1}(e^{\star}_{k-1})$ in the regularization cost term denotes the sample in the first relevant tree level corresponding to the optimal edge sequence from the previous planning cycle. The projection of any 2D position onto the reference path can be efficiently pre-computed via a Voronoi decomposition of the discretized reference path (Schwesinger, Rufli, et al., 2013). For a more intuitive tuning of the weights, the individual cost terms are normalized to lie within the range [0, 1], indicated by the operator *norm* in Equation 3.5. This requires the unnormalized cost of all edges to be available. The final cost of the edge corresponding to sample $\boldsymbol{\xi}_{\mathrm{n}}$ is computed as a weighted sum of the cost terms via

$$J(\boldsymbol{\xi}_{\mathrm{n}}) = \mathbf{w}_J{}^{\top} \begin{bmatrix} \mathrm{norm}\left(J_{\mathrm{sf}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \\ \mathrm{norm}\left(J_{\mathrm{id}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \\ \mathrm{norm}\left(J_{\mathrm{iv}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \\ \mathrm{norm}\left(J_{\mathrm{so}}(\boldsymbol{\xi}_{\mathrm{n}}) - J^{\star}_{\mathrm{so,d}}\right) \\ \mathrm{norm}\left(J_{\mathrm{reg}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \\ \mathrm{norm}\left(J_{\mathrm{u}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \\ G_d\left(J_{\mathrm{do}}(\boldsymbol{\xi}_{\mathrm{n}})\right) \end{bmatrix}. \tag{3.5}$$

While terminal offset penalization rewards path- and speed alignment of edges at their final state, the integral terms encourage smaller offsets along the complete trajectory. The integrated control effort cost term favors smoother edges. Adding a penalization with respect to the solution found in the previous planning cycle has a regularizing effect in case of several local minima in the cost function, and results in smoother behavior over consecutive planning cycles. Two cost terms related to static obstacle

Cost term	Effect	Formula
Terminal path/speed offset	Lane and target speed keeping; rewards alignment at the end of the solution	$J_{\mathrm{sf}} = \mathbf{w}_{\mathrm{f}}^{\top} \begin{bmatrix} d\left(T_{\mathrm{sim}}\right) \\ \left\lvert v\left(T_{\mathrm{sim}}\right) - v_{\mathrm{ref}}\left(s\left(T_{\mathrm{sim}}\right)\right)\right\rvert \end{bmatrix}$
Integrated path offset	Lane keeping; the integral avoids short-cutting in sharp turns	$J_{\mathrm{id}} = \int_{0}^{T_{\mathrm{sim}}} d\left(t\right)\, \mathrm{d}t$
Integrated speed offset	Target speed keeping along the reference speed profile	$J_{\mathrm{iv}} = \int_{0}^{T_{\mathrm{sim}}} \left\lvert v\left(t\right) - v_{\mathrm{ref}}\left(s\left(t\right)\right)\right\rvert \mathrm{d}t$
Static obstacle distance	Yields safety distance and resilience to perception noise	$J_{\mathrm{so}} = 1 - \min\left(\min_{t\in\left[0,T_{\mathrm{sim}}\right]}\left(d_{\mathrm{obs}}\left(t\right)\right), \epsilon\right)\Big/\epsilon$
Static obstacle collision	Prunes edges in collision with static elements	$J_{\mathrm{sc}} = \begin{cases} \infty\,, \text{ if } \min_{t\in\left[0,T_{\mathrm{sim}}\right]}\left(d_{\mathrm{obs}}\left(t\right)\right) \leq 0 \\ 0\,, \text{ otherwise} \end{cases}$
Deviation to last solution	Regularization term to disambiguate nearby minima; smooth behavior over consecutive planning cycles	$J_{\mathrm{reg}} = \begin{cases} \mathbf{w}_{\mathrm{r}}^{\top}\left(\boldsymbol{\xi} - \boldsymbol{\xi}_{l=1}(e_{k-1}^{\star})\right)\,, \text{ if } l = 1 \\ 0\,, \text{ otherwise} \end{cases}$
Integrated control effort	Lower energy consumption and smoother behavior	$J_{\mathrm{u}} = \int_{0}^{T_{\mathrm{sim}}} [\dot{\varphi}(t), a(t)]\mathbf{w}_{\mathrm{u}}^{\top}[\dot{\varphi}(t), a(t)]^{\top}\, \mathrm{d}t$
Dynamic collision	Collision avoidance w.r.t. dynamic objects, based on time to collision $T_{\mathrm{c,min}}$ with customizable cutoff δ.	$J_{\mathrm{do}} = \begin{cases} \beta \exp\left(-\gamma\left(T_{\mathrm{c,min}} - \delta\right)\right), \text{ if } T_{\mathrm{c,min}} \geq \delta \\ \infty\,, \text{ otherwise} \end{cases}$ with $T_{\mathrm{c,min}} = \min_{i\in\left[1,N_{\mathrm{dynobs}}\right]}\left(T_{\mathrm{c},i}\right)$

Table 3.1: Cost terms used in the partial onlane planning framework.

avoidance are responsible for hard collision avoidance and a soft distance preference respectively. The soft penalization of close distances is hereby essential to encourage a safety distance, and consequently yields better resilience to noise in the perception of the static world elements. Without soft penalization, a flip of a single cell in the occupancy grid may render the vehicle suddenly in collision. Hereby, the cost is modeled as a linear decreasing function of the minimum obstacle distance along the edge. Dynamic object avoidance is incorporated through a measure of collision risk, where risk is modeled as an exponentially decaying function over the estimated time-to-collision (TTC). The exponential decay intends to account for the fact, that TTC estimates are increasingly more uncertain the further the collision happens in the future.

In the following, we will discuss the cost terms related to static and dynamic collision avoidance in more detail.

Static obstacle distance cost term By introducing a *soft* penalization of close distances to static obstacles, the car is encouraged to keep a certain safety distance (cf. Figure 3.10). This safety distance is desirable to account for perception uncertainty, but also for uncertainties in the control chain. Sometimes however, the vehicle must

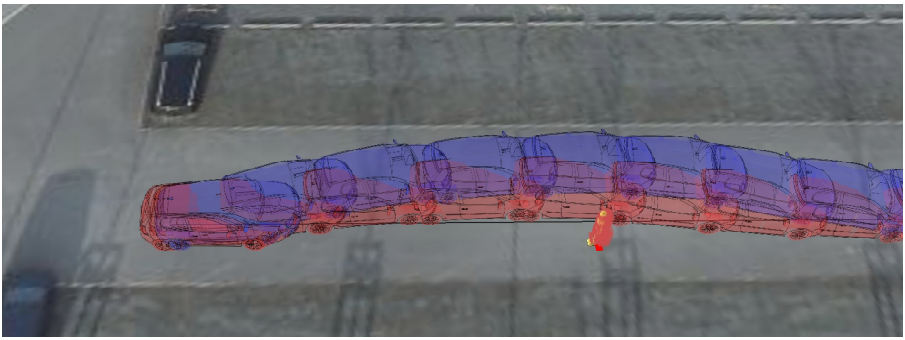


Figure 3.10: Influence of soft penalization of static obstacle distance. The red track depicts the navigation behavior without the cost term, where obstacles are passed without safety distance. The blue track shows navigation with active static obstacle distance penalization, resulting in a safe and comfortable avoidance of the object.

navigate through tight passageways, such as for instance gates at a parking lot entry. With a naive implementation, narrow passages appear as high cost regions as distance to static obstacles is high everywhere. Depending on the ratio between the static obstacle weight and the weights influencing the longitudinal progress, the cost function

might score candidate motions coming to a full standstill in front of the narrow passage better than motions passing the high cost region induced by the static obstacle cost term.

Dolgov et al. (2010) provide a principled solution to this problem based on the combination of the obstacle distance field with the minimum distances to the edges of the environment's Voronoi diagram. The resulting Voronoi field is designed in a way to reach its minimum on the edges of the Voronoi diagram. We wish to achieve a similar effect, however without having to compute the Voronoi tessellation. Therefore, we subtract from each edge the minimal static obstacle cost along the corresponding lateral offset dimension, $J^{\star}_{\mathrm{so,d}}$, in order to obtain a static obstacle cost equal to zero for the sample with the largest minimum distance to any obstacle. Information from the samples of the lateral offset dimension is used in order to sense the width of the driving corridor. The modification encodes the intention to keep a safety distance to static obstacles if possible, but does not penalize close distances if the navigation space gets tight. The resulting behavior is depicted in Figure 3.11.

Figure 3.11: Influence of the post-processing step applied to the static obstacle cost term to improve navigation performance in narrow passages. Without the post-processing step the vehicle denies to pass the narrow passage (red track).

Dynamic collision cost term In contrast to static obstacles, edges in collision with dynamic objects are not certainly pruned. Instead, an additional *soft* cost term based on the estimated TTC is introduced. This accounts for the fact that motion predictions of other traffic participants are inherently subject to uncertainty. Predictions become

more uncertain the farther they are in the future. Therefore, a cost term with exponential decay over time, J_{do}, is used (see Table 3.1). However, edges with TTC below a certain threshold δ—for instance determined through the maximum braking time—are assigned infinite costs to address safety aspects.

The dynamic collision detection framework, detailed in Chapter 4, returns binary answers to collision queries between candidate motions and predicted trajectories of other traffic participants. This results in an ego motion passing objects with the minimal realizable distance, rendering the planning stage vulnerable to measurement noise and resulting in maneuvers being perceived as risky. While the bounding volume hierarchy-based collision detection framework provides the possibility for accelerated distance queries between objects' trajectories, this comes with a significant computational overhead compared to binary collision tests (cf. Section 4.3). Inspired by the Dynamic Window Approach by Fox et al. (1997), we instead use a Gaussian smoothing filter G_d over the dynamic object cost term in the lateral offset dimension d_{ref}. The filter blends costs from collisions with dynamic objects into (laterally) neighboring candidate motions, which results in the desired side clearance without the need to perform expensive distance queries. The dynamic object cost term is excluded from the normalization procedure due to its safety-critical nature. The resulting navigation behavior is illustrated in Figure 3.12 in a simulated run. In the

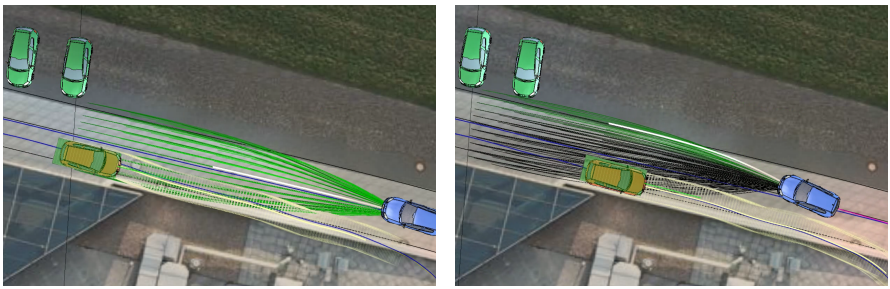


Figure 3.12: Simulated avoidance maneuver with an oncoming vehicle. Color scheme for candidate motions: green = low cost, black = high cost. *Left*: Time-to-collision (TTC) is still high and the ego vehicle delays the reaction. *Right*: As a collision comes closer in time, the ego vehicle initiates an avoidance maneuver. A Gaussian smoothing filter blends costs originating from the anticipated collision into nearby samples.

beginning (left figure), samples already collide with the anticipated future movement of the oncoming car, however the importance of the collision is rated low due to the exponential decay with respect to their TTC. The closer the oncoming car approaches

(right figure), the higher the cost of the colliding candidate motions becomes and the ego vehicle initiates the avoidance maneuver.

3.3.7 Situation-Specific Adaptations

The partial onlane motion planner operates almost completely without inputs from higher-level situational awareness modules. The primary goal is to encode suitable behaviors in the objective function of the local motion planner and only to fall back on higher-level behavior modifications if absolutely necessary.

For smooth platooning behavior, we found it necessary to decrease the reference speed for the ego vehicle to yield a specified time-gap and/or distance between the ego vehicle and the car in front. Otherwise, its behavior was perceived as slightly aggressive, attempting to overtake in case the vehicle in front gets too slow, in situations where (even-tempered) human drivers would show more patience. Modifying the reference speed solely shifts the minimum of the objective function, leaving the rest of the planning framework unchanged. To determine the relevant object in front, a scoring metric in the lane's Frenet frame is used based on the objects' longitudinal and lateral distances to the ego vehicle along the lane.

3.3.8 Movement Predictions for Traffic Participants

Reasoning about future movements of other traffic participants is important for well-behaved navigation in mixed-traffic scenarios. Our motion prediction scheme employed in the partial onlane planner is based on a forward simulation of different agent motion models. A suitable motion model is chosen online based on the associated object class label determined by the classification module. The classification module—and hence the motion prediction module—distinguishes between pedestrians and vehicles.

A constant velocity model is employed for pedestrians, due to missing annotations of pedestrian walkways and due to the fact that pedestrians often move in an unconventional way and do not respect their dedicated walkways rigorously. Especially on parking lots, traffic rules for pedestrians are usually fuzzy. The constant velocity model is clearly limited in its predictive capabilities (Lefèvre et al., 2014) and we will present an interaction-aware model in Chapter 5.

In contrast to pedestrians, the movement of vehicles is in general bound much tighter to the infrastructure. Therefore, we exploit the lane information stored in the road-graph data structure to improve upon the motion predictions of vehicles. Our prediction approach for vehicles resembles the one presented in Ferguson, Darms, et al.

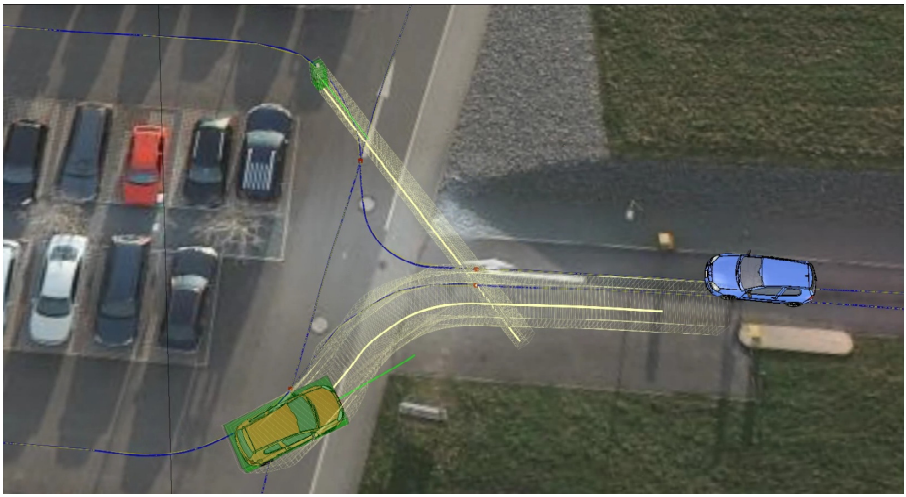


Figure 3.13: Movement predictions for two different agent types over a time horizon of 10 seconds. Green boxes represent the detected bounding boxes of the objects. The yellow wireframe objects depict the predictions.

(2008), in which vehicles are predicted along their associated lane. We extend it to account for common vehicle kinematics. In order to quickly associate the pose of an object with the road network, a *kd*-tree containing all three-dimensional (3D) roadgraph poses (2D position x, y and heading θ) is built in an offline step to accelerate nearest neighbor queries. The heading dimension is included in order to distinguish between nearby lanes with opposite directions. We use a custom distance metric between two poses $\mathbf{c_0}$ and $\mathbf{c_1}$ in SE(2)

$$d^2(\mathbf{c_0}, \mathbf{c_1}) = (\mathbf{c_0}_x - \mathbf{c_1}_x)^2 + \left(\mathbf{c_0}_y - \mathbf{c_1}_y\right)^2 + \alpha(\mathbf{c_0}_\theta - \mathbf{c_1}_\theta)^2 \,, \tag{3.6}$$

with α being a scaling factor for the angular heading dimension. For prediction, a forward simulation of a kinematic bicycle model with a discrete integration scheme is employed, performing three operations in each simulation timestep:

(1) The closest pose of the simulated vehicle position on the roadgraph is retrieved from the *kd*-tree.
(2) The lateral offset to the closest roadgraph pose is computed and a reference pose is created by laterally shifting the closest pose by this offset. This encodes the assumption that vehicles will keep the current lateral lane offset.

(3) The control law from Equation 3.4 is used to compute a desired steering angle. This steering angle regulates the simulated model towards the computed reference state.

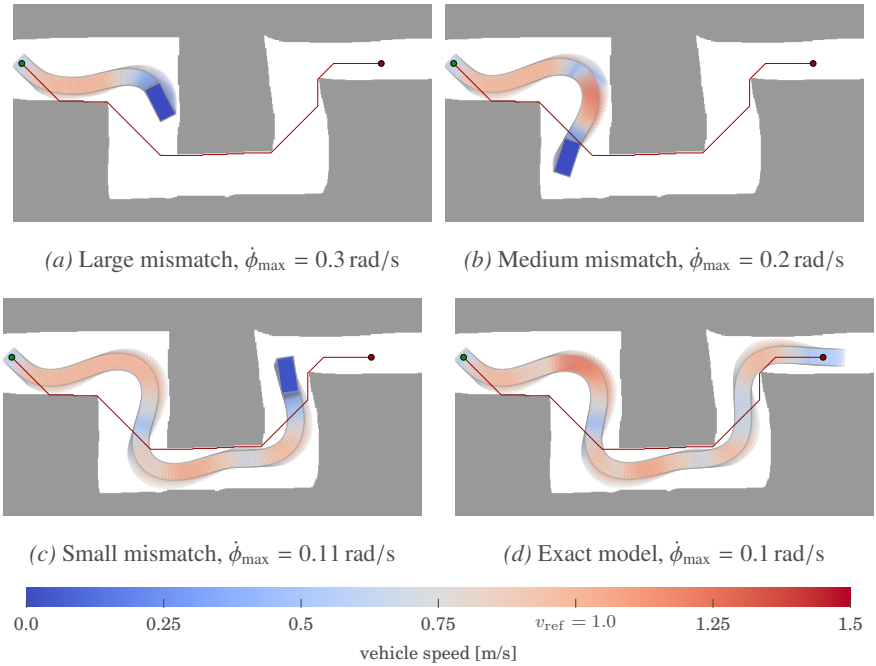
The speed of the vehicle is kept constant at its measured speed during the simulation time horizon. The result of this motion prediction scheme for pedestrians and vehicles is illustrated in Figure 3.13. Ferguson, Darms, et al. (2008) additionally create multiple hypotheses at intersections for every outgoing edge of the roadgraph. This enables the robot to react to different possible future decisions of other vehicles and could also be integrated in our prediction scheme, however computation time for collision detection would increase.

3.4 Experimental Results

In this section, we provide simulation results as well as real-world driving evaluations. In simulation, we highlight the need for system-compliant planning in tight environments and demonstrate the planner's capabilities to operate in urban traffic. Moving to real-world application, two dynamic, mixed-traffic scenarios that frequently appear during autonomous operation on a parking lot—platooning and mitigating crossing pedestrians—are analyzed. Finally, driving statistics are presented from long-term automated driving experiments.

3.4.1 System-Compliant Planning in Narrow Environment

In this first experiment, we want to underline the necessity for highly system-compliant planning when navigating in narrow environments. While it is intuitive, that a smaller divergence between planning and execution layer will lead to better navigation performance, it is not immediately clear to which extend the mismatch may be corrected via frequent replanning. Therefore, a simulated vehicle is placed in a narrow corridor with sharp turns as depicted in Figure 3.14. The vehicle is equipped with upper limits on steering angle, $\phi_{\max}$, and steering speed, $\dot{\phi}_{\max}$, that call for precise trajectory planning in order to successfully pass the turns. Maximum steering speed is harshly restricted such that a full turn of the wheels takes 14 s, to demonstrate the planner's capability to cope with challenging dynamic constraints. We performed four simulation runs at a replanning cycle time of 0.3 s; one run, in which the partial onlane planner is provided with the true system model (lower right figure), and three runs, in which the planner assumes a higher maximum steering speed than the vehicle is able to provide. The system inputs computed by the planner in each cycle are applied to the real system until replanning takes place and generates a new sequence of system commands. The terminal manifold was uniformly sampled into 21 reference



(a) Large mismatch, $\dot{\phi}_{\max} = 0.3\,\mathrm{rad/s}$ *(b)* Medium mismatch, $\dot{\phi}_{\max} = 0.2\,\mathrm{rad/s}$

(c) Small mismatch, $\dot{\phi}_{\max} = 0.11\,\mathrm{rad/s}$ *(d)* Exact model, $\dot{\phi}_{\max} = 0.1\,\mathrm{rad/s}$

Figure 3.14: Simulated runs with the partial onlane motion planner through a narrow environment. A significantly small maximum steering speed was imposed on the car-like vehicle. True vehicle parameters are $\phi_{\max} = 0.7\,\mathrm{rad}$ and $\dot{\phi}_{\max} = 0.1\,\mathrm{rad/s}$, the vehicle's minimum turning radius can be observed in the sharp turns. Varying mismatches in maximum steering speed between the planning and the execution layer are simulated. The C^0-continuous reference path (red line) is extracted from a simple 2D grid search. Only when using the accurate vehicle model in (d), the planner is able to navigate the car from its start (green dot) to its goal (red dot). Inaccurate models in the planning stage lead to deadlock situations (reverse driving disabled). The vehicle poses are color-coded by translational speed.

path offsets in the range $[-5, 5]$ and 31 speeds in the range $[0, 2.6]$. The static collision checker was configured to use a very fine disk approximation, to yield a minimal overhang to ensure the existence of a feasible path. The simulated controller was configured slightly different for each steering speed model, to obtain a well-behaved candidate motion set.

The vehicle's small upper steering speed calls for the early initiation of turns. In case the planner is not aware of this constraint, its plans are not aggressive enough to cope with the slow steering mechanism, and maneuver the vehicle right in front of the walls. Already a mismatch of 10 % is enough to let the planner fail to manage the last and sharpest turn. In contrast, when provided with an accurate model of the system, our partial onlane planner succeeds to maneuver the vehicle with its challenging dynamic constraint through the course (see Figure 3.14(d)). Once aware of the correct dynamic constraint, the motion planner slows down the vehicle early enough to deal with the limited steering speed. A C^0-continuous reference path—in this example not even collision-free—is sufficient as guidance through the corridor. While the upper steering speed limit used in this experiment is significantly smaller than on a real system, the results show that an inaccurate system model in the planning stage may have disastrous effects on the navigation performance in tight spaces. Our proposed planning framework uniquely allows to incorporate complex system models in a straightforward manner, while providing real-time capabilities required for fast replanning in partially unknown or only partially predictable (dynamic) environments. In this context, note that the exemplary steering speed constraint used in this experiment may also be satisfied with competing approaches such as Urmson et al. (2008); Bacha et al. (2008); Montemerlo et al. (2008); Bohren et al. (2008) and Werling et al. (2011) relying on a geometric construction of curves and model inversion techniques, yet these methods fall short for more complicated systems. Additionally, Werling et al. (2011) cannot deal with C^0-continuous reference paths.

3.4.2 Navigation in Simulated Traffic Scenario

In this experiment, as regulations for automated driving did not allow us to drive on public roads, we demonstrate our planning framework's capacity for complex decision-making in a simulated inner-city traffic scenario—including lane and intersection handling, static and dynamic obstacle avoidance, as well as platooning. Importantly, the handling of these situations *emerges* from the interplay between the framework (in particular space-time collision checking) and the environment and thus reduces the need for a separate situational awareness module.

Figure 3.15 shows an orthographic image of Zurich, Hegibachplatz, together with an artificially created reference path for the ego vehicle and three dynamic obstacle

tracks. Each dynamic obstacle track was populated with several circular dynamic objects of radius 1 m with a constant time gap of 4 s between them. The reference travel speeds for the ego vehicle and the dynamic obstacles on tracks 1 and 2 were set to 8.33 m/s = 30 km/h, while those on track 3 were set to 2 m/s to cause a congestion. We applied a binary mask to the static environment to mask impassable areas

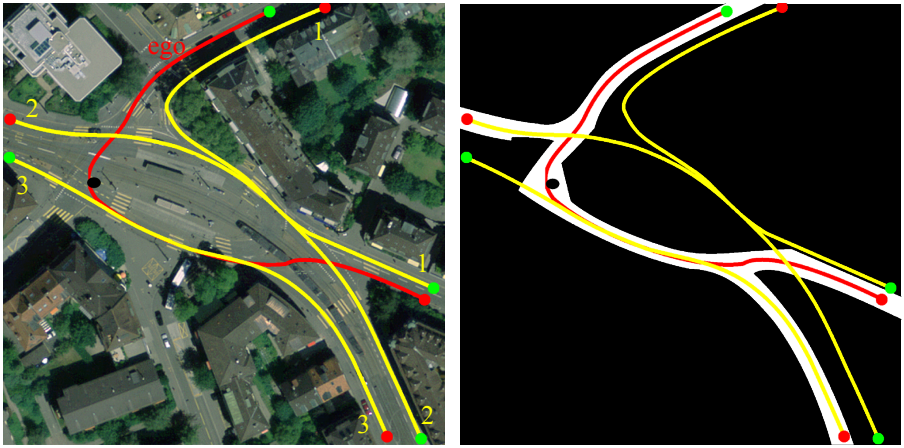


Figure 3.15: Setup for a simulated run at Zurich, Hegibachplatz. *Left*: Orthographic image of Zurich, Hegibachplatz. Reference paths for the ego vehicle (red) and three dynamic obstacle tracks (yellow) are displayed, with their start points (green) and end points (red) indicated. *Right*: Binary occupied(black) / free(white) space mask applied to the image, representing road information.

outside of drivable roads. In this regard, note that the tram displayed in the orthographic image was not considered in the scenario. The reference path for the ego vehicle was created based on a hand-drawn reference path in the image. Heading and curvature information was extracted by pixel-based finite differences, post-processed via a moving average filter for smoothing purposes. Despite the smoothing step, reference heading as well as steering angle information is rather noisy, which enables the demonstration of discontinuous reference information handling within our framework. After every planning cycle, the vehicle in simulation executes the initial part of the solution as feed-forward commands. In Figure 3.16 four snapshots of the simulated run are displayed, capturing four distinct situations that challenge the planner's capabilities: (a) intersection handling, (b) static obstacle avoidance, (c) platooning and (d) pure lane following. In the initial part of the run, from $t = 0$ s to $t = 10$ s, the ego vehicle faces oncoming vehicles on track 1. The knowledge that their future

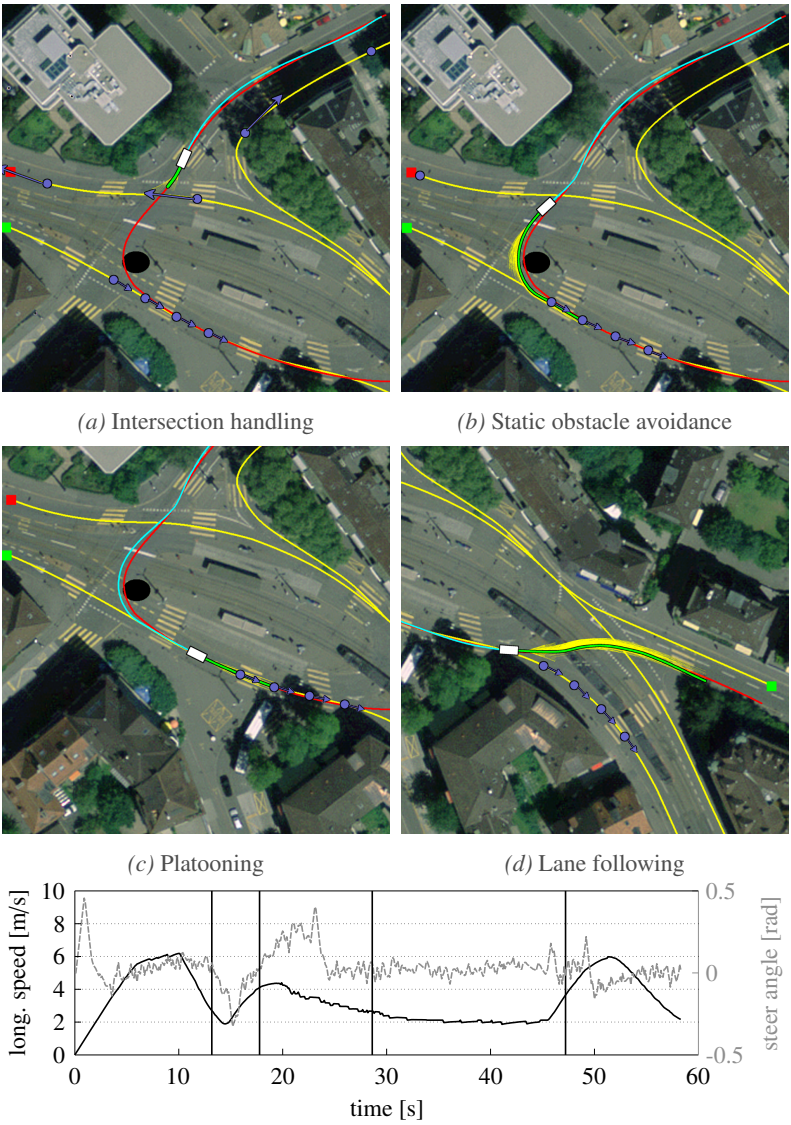


Figure 3.16: Top: Snapshots of a simulated run at Zurich, Hegibachplatz, with our partial onlane motion planner. The vehicle (odometry solution in cyan) successfully avoids dynamic obstacles (blue circles with velocity vectors), as well as static (black) obstacles while following the reference path (red line). From a set of explored trajectories (yellow lines), the best partial motion plan (green line) is chosen. *Bottom*: The graph shows longitudinal speed as well as steering angle of the ego vehicle. Vertical lines refer to the timestamps of the snapshots.

movement is constrained to the oncoming lane allows the ego vehicle to avoid improper evasive maneuvers. After 10 s, while approaching the first intersection, the vehicle needs to yield to dynamic obstacles on track 2. The ego vehicle slows down to let the obstacles pass, as time does not permit pulling out in front of them. At $t = 18$ s, the vehicle encounters a static obstacle (black ellipse) in the middle of the intersection and steers to the right to pass by. This illustrates that the reference path is readily departed if circumstances call for it. Thereafter, the vehicle faces a convoy of four slow traveling obstacles with 2 m/s. It decelerates and follows the convoy with a constant time gap of 3 s as space does not permit an overtaking maneuver. As soon as the road is clear, the ego vehicle accelerates again. In order to come to a standstill at the end of the reference path the ego vehicle brakes at the end of the run. All these maneuvers emerged from the same cost function without switching between different behavior or parameter sets.

3.4.3 Mitigation of Crossing Pedestrian

After having demonstrated the planner's capabilities in simulated environments, the remainder of this experimental result section is concerned with the evaluation on the real car. Hereby, the experiments can only evaluate the performance of the *whole online system*, consisting of visual localization (required for association with the reference path), vision-based environment perception, object detection, classification and tracking, as well as object prediction and planning.

Figure 3.17 depicts the navigation behavior in the presence of a pedestrian crossing the street in front of the car. Velocity information of the pedestrian in conjunction with space-time collision detection allows the vehicle to avoid a lateral evasive maneuver. The vehicle rather prefers a short reduction of its longitudinal speed in order to give way to the pedestrian, and afterwards accelerates again to meet its desired reference speed of 1.66 m/s.

Again, in order to achieve the behavior described, no behavior switching was required. However, we could observe that motion plans computed over consecutive planning cycles were quite sensitive to the velocity estimates of other traffic participants. Without any countermeasures, the vehicle would occasionally switch between passing left, waiting, and passing right, despite the temporal regularization term in the cost function. Therefore, an additional temporal smoothing approach was implemented, in which the cost of the best solution found in the current planning cycle is compared with the cost of the candidate motion corresponding to the best sample $(d_{\text{ref}}, \Delta v_{\text{ref}})^{\star}_{k-1}$ of the previous cycle. Only if a substantial improvement in the cost value was present, the planner would switch to the new solution and otherwise output

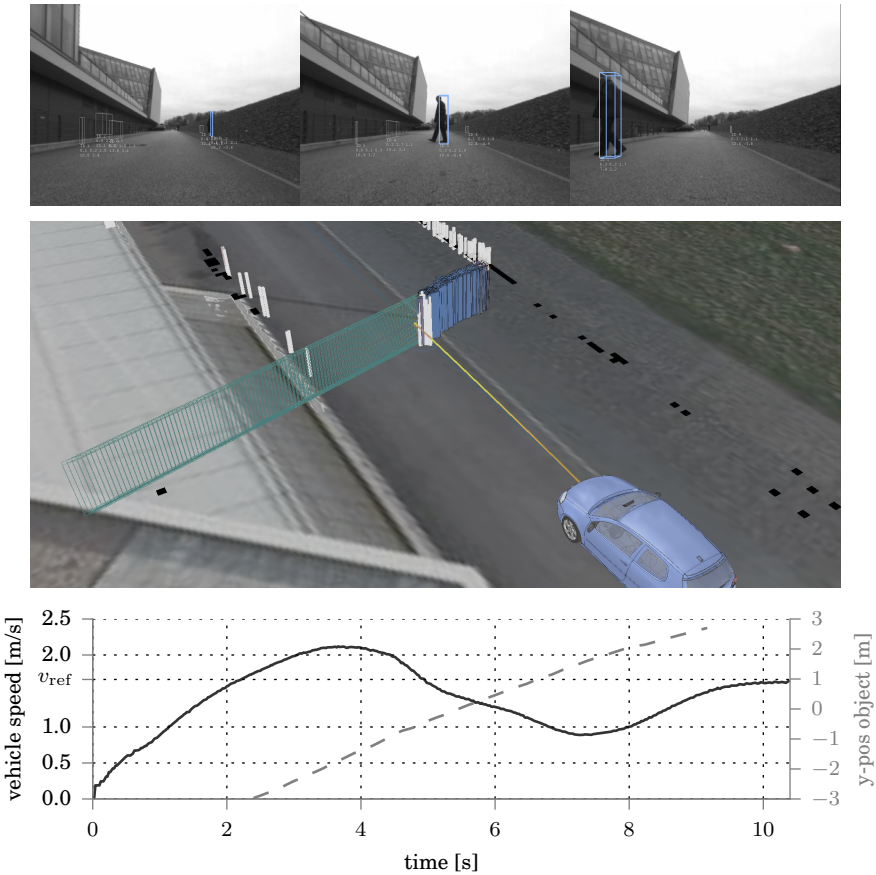


Figure 3.17: Real-world mitigation of a pedestrian crossing the street in front of the vehicle. *Top:* Rectified monocular camera images with annotated pedestrian detections for three time instances during the encounter (first detection left, last detection right). *Middle:* External 3D view with object history (blue bounding boxes), constant velocity prediction for pedestrian motion (green wireframe), stereo stixels (white bars), 2D occupancy grid (occupied black cells), and resulting motion plan (yellow line) for the ego vehicle. *Bottom:* Speed profile of the ego vehicle (left axis) and lateral pedestrian distance (right axis). The vehicle slows down to give way to the pedestrian without lateral adaptation of its motion plan.

the solution $(d_{\text{ref}}, \Delta v_{\text{ref}})^{\star}_{k-1}$. A similar smoothing approach is taken in Broggi et al. (2012), yet comparison is performed in input space rather than in cost space. To ensure that the algorithm eventually converges back to the optimal solution, we integrate the suboptimality, i.e. the cost difference between the optimal and the returned trajectory, and switch back to the current optimal trajectory once the integral hits a certain threshold. While this heuristic effectively suppressed nervous behavior of the vehicle, we clearly consider this an "engineering solution" without any theoretical foundation.

3.4.4 Platooning

Platooning is a situation appearing frequently in mixed traffic. Figure 3.18 visualizes the navigation performance of the V-Charge self-driving vehicle during such, involving multiple stop-and-go phases. In this run, the car in front is reliably tracked throughout the complete four minutes of the run. The speed profile of the ego vehicle and the target car match with a slight time offset induced by delays in the processing chain, but mainly due to the low-pass characteristics of the ego vehicle's drive-train.

The leftmost figure depicts the first approach to the stagnant car in front. The middle figure illustrates the local planner's ability to concurrently adapt its plan (orange trajectory) based on static obstacle information if required. The soft static obstacle distance cost term directs the ego vehicle slightly towards the left to keep a safety distance to the pillar on the right-hand side. In the rightmost figure, the roadgraph-based object prediction utilizing lane center information (blue path) is visible.

3.4.5 Long-Term Automated Driving Statistics

The real-world examples above demonstrate the motion planner's capabilities to cope with specific situations encountered in dynamic, mixed-traffic scenarios. Apart from flawless behavior in these situations, we were interested in potential problems only revealed during long-term operation of the vehicle. To detect those rare events, statistics were automatically created (in an unsupervised fashion) from log recordings over the course of 25 test days with more than seven hours of fully automated operation in which 26.63 km were covered. A total of 01:07:36 hours were spent in platooning mode. From the log files, we could recover the following operational statuses of the motion planner:

- *Success*: This status indicates that the vehicle is moving, operated by the partial onlane motion planner without indication of problems.

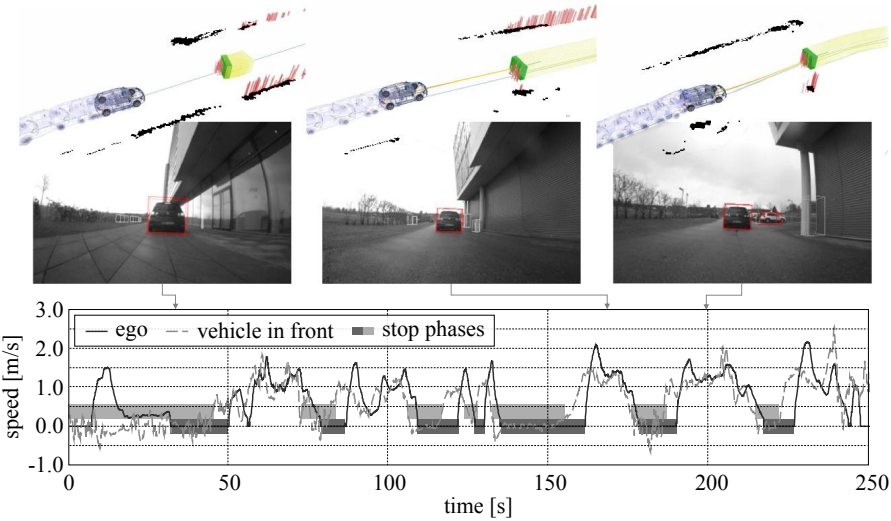


Figure 3.18: Real-world platooning performance in stop-and-go traffic. *Top left*: First approach to a stagnant car in the lane. The ego vehicle brakes up to standstill. *Top middle*: Soft distance preferences to static obstacles let the ego vehicle direct its route to its left in order to keep a safety distance to the pillar on the right-hand side. *Top right*: Object predictions (yellow wireframe) of other vehicles make use of lane information stored in a roadgraph data structure. *Bottom*: Velocity profiles of the ego vehicle and the car in front match up to a slight time offset induced by the processing chain and the low-pass characteristics of the ego vehicle's drive-train.

- *Standstill*: The motion planner is keeping the car stopped in a controlled way during normal operation. While these events may occur in normal situations—for instance in platooning mode when the car in front stops—they could also stem from a potential deadlock in which a global guidance would be necessary.
- *No solution found*: The vehicle is stopped with an emergency brake, triggered in case the motion planner is unable to compute a collision-free trajectory.

The statistics are shown in Table 3.2. In only 0.23 % of the time no collision-free trajectory could be computed. These events sporadically occurred when passing through very narrow gates on the parking lot with only a few tens of centimeters margin on each side of the vehicle—a highly challenging scenario for both vision-based perception and motion planning. More accurate sensing and an even finer resolution of the static collision checker can eliminate this problem. However more importantly, they were caused by pedestrians moving in very close proximity to the vehicle as depicted

Operational Status	Duration (hh:mm:ss)	Percentage
Success	07:17:53	99.34
Standstill	00:01:55	00.43
No solution found	00:01:00	00.23

Table 3.2: Driving statistics obtained from more than seven hours of automated driving, accumulated over a period of 25 test days on the parking lot at Mobile Life Campus, Wolfsburg, Germany. Our log parser automatically categorized the operational status of the vehicle into "success" (vehicle in motion, valid trajectory returned by the motion planner), "standstill" (vehicle stopped regularly by the motion planner) and "no solution found" (motion planner unable to find collision-free trajectory). Values in percentage are given with respect to operation time.

on the left-hand side of Figure 3.19. Given the considerable noise levels present in our vision-based object detection pipeline, pedestrians sporadically appeared to be on a direct collision course with the car, with no candidate motion able to avoid the collision.

Another similar incidence leading to an emergency brake is shown in Figure 3.20. The object detection and tracking system accidentally merged stereo stixels of a pedestrian with part of the static infrastructure (the fence), resulting in a temporary bias of its velocity estimate towards the road in front of the car. The motion planner was unable to compute a collision free trajectory mitigating the (wrong) constant velocity prediction and was abruptly stopped. This car's behavior might appear justified, if for a moment the reader tries to ignore the additional information provided by the orthographic image overlay and the raw camera images. In order to correctly cope with the situation encountered, additional semantic information is required, such as for instance the position of the pedestrian walkway, the pedestrian's posture, or its head facing direction. Unfortunately, this information is hard to extract algorithmically from sensory data available to the robot. Yet we believe that it are those rare events (depending on the quality of the perception system) that might lead to fatal traffic consequences—and triggering an emergency brake while driving can have fatal consequences—and have to be addressed before automated cars may safely join public mixed traffic.

In Chapter 5 we address the issue of over-conservative robot behavior raised within this experimental result section via a cooperative planning approach. As rich semantic information is still difficult for a robot to obtain in an online fashion, cooperative planning provides an excellent, autarkic mechanism to counteract over-conservative

Figure 3.19: In 0.23 % of the time no collision-free trajectory could be computed by the partial onlane planner. Noisy detections of pedestrians moving in close proximity to the ego vehicle (left, camera image captured by fisheye camera at vehicle front) and small perception noise when passing very narrow gates (right) were mostly the reason.

Figure 3.20: Incident in which the motion planner is unable to compute a collision-free trajectory, thus triggering an emergency brake (right). A pedestrian (yellow bounding boxes) is approaching on its walkway close to the road. Through a wrong stereo stixel (red bars) association, the pedestrian's velocity estimate is biased towards the road. *Left*: snapshot right before the incident, the vehicle follows its plan (yellow curve) approaching a left-turn. *Right*: emergency brake triggered as the constant velocity prediction (yellow wireframe) puts the pedestrian on a direct collision course with the car.

behavior through the assumption of other agents' participation in mutual collision avoidance.

3.5 Summary and Discussion

In this chapter, a novel sampling-based partial motion planning strategy was presented, designed to generate highly system-compliant motions for systems with pronounced kinodynamic constraints. The strategy utilizes a direct forward propagation of the (closed-loop) system model under consideration. In combination with a control law, regulating the system towards samples drawn from a manifold aligned with a reference path, the planning strategy produces inherently compliant candidate motions without the necessity for model inversion and post-pruning.

We described a system model and control law for the application of the planning framework in the context of self-driving cars. An objective function was designed to enable navigation in dynamic, mixed-traffic scenarios. Hereby, the widely applied time-to-collision metric was used to score candidate motions in the presence of moving objects. A simple physics-based motion prediction scheme was implemented, supported by road network information for the prediction of future vehicle motion.

The local motion planning framework, as presented in this section, was evaluated both in simulation and extensively on the V-Charge self-driving car while operating autonomously in parking lot environments. It has proven to be stable, reliable, and an appropriate choice for static environments. It produced promising results also for mixed-traffic scenarios, as demonstrated by its capability to navigate the real vehicle in stop-and-go traffic and in the presence of crossing pedestrians.

Long-term driving statistics however revealed a certain vulnerability of the planning approach to perception noise when navigating in close proximity to other traffic participants such as pedestrians. To our surprise, we could not find any references acknowledging this problem, which we account to the fact that (a) academic research on motion planning for self-driving vehicles is mainly performed in perfect simulation environments and results from research in industry are not accessible, (b) if real-world results are provided, accurate but expensive laser range finders are utilized, and (c) primarily application in highly structured environments is targeted. We see two remedies to this problem, both not mutually exclusive and addressing the need to obtain more accurate predictions of future movements of other traffic participants.

(a) Richer semantic environment information is beneficial to build more accurate prediction models. A simple annotation of pedestrian walkways for instance

constitutes a good prior for pedestrian motion and constrains their future movement to a large extend. However, given the current state of the art, the amount of semantic information is usually quite limited on real robotic platforms. Especially when needed to operate in unmapped environments, online extraction of such kind of semantic information is still a major challenge subject to active research efforts. Yet, with the latest achievements in deep network architectures to perform real-time capable semantic annotation, this kind of information might become available in the foreseeable future.

(b) Incorporating the assumption of cooperation in the motion planning stage will allow the robot to question the meaningfulness of apparently self-destructive behavior when other traffic participants appear to be on direct collision course (due to noisy observations). This comes without the need for additional semantic information. Yet it substantially increases the complexity of the motion planning problem as the problem can no longer be subdivided into prediction and planning, but now must be solved jointly.

Leveraging semantic information for object prediction is only useful in environments where agent motion is considerably driven by such. Challenging environments such as depicted in Figure 1.5 provide few useful semantic features to reason about agents' future whereabouts. Therefore, we focus on cooperative planning strategies in this thesis. In Chapter 5, we present an interaction-aware planning approach based on a maximum entropy model of joint navigation behavior and show its potential to reduce over-conservative robot behavior.

4

Time-Aware Collision Detection

The partial onlane planning framework, detailed in Chapter 3, needs to evaluate numerous candidate motions for collision with respect to static infrastructure and dynamic objects. While collision detection with static infrastructure can be performed efficiently as outlined in Section 3.3.5 since time may be safely neglected, collision detection with moving objects is substantially more time-consuming. The latter requires a collision detection framework that properly takes into account the time dependency of motion. Additionally, as we target motion planning for car-like vehicles in narrow environments, collision detection must be performed with sufficient accuracy. Neither an under-conservative approximation of the vehicle's rectangular collision shape raising safety issues, nor an over-conservative approximation eliminating too much of valuable space for navigation, is acceptable. In this section, we present a fast, time-aware collision detection framework, applicable for sampling-based motion planners in dynamic environments.

This chapter incorporates material from the following first author publication:

Schwesinger, U., Siegwart, R., and Furgale, P. (2015). "Fast Collision Detection Through Bounding Volume Hierarchies in Workspace-Time Space for Sampling-Based Motion Planners". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–68. DOI: 10.1109/ICRA.2015.7138981

4.1 Context and Related Work

For a large number of motion planners, collision detection is the most time-consuming operation in the planning process (LaValle, 2006). Both sampling-based planners like the Rapidly-Exploring Random Tree (RRT) algorithm (LaValle, 1998) and RRT* (Karaman et al., 2011), as well as state-lattice approaches (Pivtoraiko and Kelly, 2006) have to evaluate numerous explicitly constructed paths or trajectories for collision. The majority of implementations of sampling-based motion planners operate

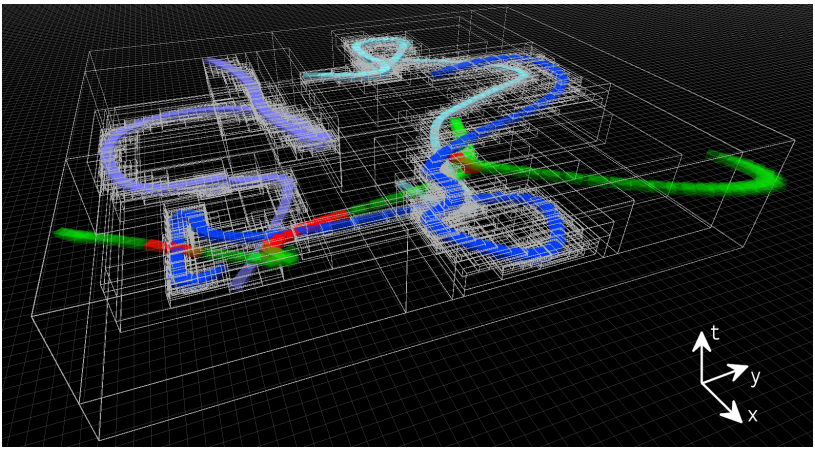


Figure 4.1: Visualization of an axis-aligned bounding box tree in workspace-time space for three obstacle trajectories generated by a random walk (blue shades). The ego trajectory (green/red) can be efficiently checked for collision with the help of the tree structure (wireframes).

in the time-agnostic configuration- or state-space of the robot (LaValle, 1998 and Pivtoraiko and Kelly, 2006). Operation in dynamic environments is tackled through frequent replanning. However, planning in dynamic environments eventually necessitates the detection of collisions of time-parametrized trajectories. Objects should not be considered in collision if they occupy the same space but at a different point in time. This problem can be transformed into the problem of static collision detection in configuration-time space $\mathcal{CT}$ (Erdmann et al., 1987 and Latombe, 1991) by introducing time as an additional dimension.

Valuable work was done applying sampling-based motion planning techniques in configuration-time space directly (Hsu et al., 2002 and Zucker et al., 2007). Hsu et al. (2002) adopt a Probabilistic Roadmap (PRM) planner in state-time space to plan for kinodynamic systems among moving obstacles. Though the authors mention the existence of more sophisticated collision detection techniques, collision detection is performed with a naive approach, in which every robot pose is tested for collision against every obstacle (*multiple interference test*). Berg and Overmars (2004) approach the planning problem in a dynamic environment via precomputation of a roadmap of motions avoiding static obstacles. Later, they utilize a regular grid for collision detection in a two-dimensional state-time space to find a collision-free trajectory on the roadmap. Lawitzky et al. (2012) presented an algebraic collision

detection algorithm for automotive applications in highway scenarios. It assumes trajectories for rectangular objects to be given as quintic polynomials and neglects the orientation of the vehicles. Given these simplifications, it provides runtimes around 5 µs for a collision test of two individual trajectories. As the authors mention, the negligence of the objects' orientation is acceptable for highspeed driving in certain environments such as highways, but intolerable for most other applications, which clearly limits the range of application of this method. Werling et al. (2011) target planning for a car-like vehicle along a reference path. A set of candidate trajectories is generated and tested for collision with moving objects with the method presented in Ziegler et al. (2010). This approach however necessitates a circular approximation of the shapes of all agents, which might result in poor navigation performance in tightly occupied workspaces (imagine a car navigating through a corridor framed by sideways parked cars). Closest to our approach comes the work of Ferguson, Darms, et al. (2008), conducted within the context of Carnegie Mellon's entry in the DARPA Urban Challenge (team "Boss"). Collisions with dynamic objects are detected via a three-staged, hierarchical sequence of checks. First, the authors perform a test with two axis-aligned bounding boxes fully enclosing the poses of both trajectories in a two-dimensional (2D) workspace. In case the axis-aligned bounding boxes do overlap, a *multiple interference test* with 2D bounding circles is performed. Afterwards, collision tests with oriented bounding boxes are performed at positions where bounding circles overlap. Though the initial AABB test speeds up computation for objects that are far apart, a time consuming *multiple interference test* has to be conducted in case objects are close in space. Our approach is a consistent continuation of the idea of hierarchical collision tests.

Within this thesis, we report on the implementation of an AABB data structure in workspace-time space for fast collision detection of candidate trajectories. This collision detection approach allows for fast collision queries with no constraints on the objects' shapes or motions.

The axis-aligned bounding box tree belongs to the class of bounding volume hierarchy (BVH) data structures. These data structures are commonly used in computer graphics and physics simulation engines in the so-called *broad phase* of the collision detection. They aim to reduce the number of exact collision computations that have to be performed between collision shapes in a so-called *narrow phase* by pruning most candidates through a hierarchically ordered sequence of simpler queries. Therefore, BVH approaches build a tree of bounding volumes on an object level. Bounding volumes proposed so far are bounding spheres, axis-aligned bounding boxes (AABB), oriented bounding boxes (Gottschalk et al., 1996) or discrete-orientation polytypes (Klosowski et al., 1998). For applications in three-dimensional (3D) workspaces,

BVH implementations are readily available in well-maintained software packages like the Open Dynamics Engine (Smith, n.d.), the Bullet Physics Library (Coumans, n.d.) or the recent Flexible Collision Library (FCL) by Pan et al. (2012). Spatial partitioning techniques like KD-Trees, Quadtrees, Octrees and voxel- or grid-based collision detection approaches are closely related to BVH approaches, but apply the partitioning in space rather than on the level of objects. The choice between spatial and object-oriented partitioning techniques strongly depends on the size of the environment under consideration and the amount of obstacles present therein. The partial onlane planning framework from Chapter 3 needs to consider a moderate amount of objects within the vicinity of the vehicle, rendering an object-based partitioning scheme superior.

In his pioneering work, S. Cameron introduced three distinct approaches to the collision detection problem for moving rigid objects (Cameron, 1984; Cameron, 1985 and Cameron, 1990). Firstly *multiple interference detection*, in which the shapes are tested for collision at consecutive fixed instances in time. Secondly the *swept volume* approach, that aims at computing volumes in the time-agnostic workspace that are swept by the objects over time and testing the volumes for intersection. However, swept volumes may intersect when no collision actually takes place due to different time instances at which the objects occupy the same space. And thirdly *extrusion*, which describes the process of creating and intersecting collision objects in workspace-time space. In our work, we pursue the latter approach.

4.2 Approach

We target the problem of detecting collisions between candidate trajectories of a single robot operating in a two-dimensional workspace, $\mathcal{W}_2 = \mathbb{R}^2$, with movement predictions of objects over time. Configurations of objects are only considered in collision in case they collide at a certain time. Collision queries are accelerated with the help of a bounding volume hierarchy data structure. At this point please note that the application of BVHs is not limited to a 2D workspace (yielding a 3D workspace-time space), as bounding volumes may as well be defined in higher dimensions.

First, we would like to recap the notion of a work- and configuration space and explain the space in which the AABB tree is constructed. A rigid body that translates and rotates within $\mathcal{W}_2$ has a three-dimensional configuration space $C = \mathbb{R}^2 \times \mathbb{S}^1$. Obstacles moving within this workspace create obstacle regions at certain time instances, imposing a configuration-time space obstacle, $\mathcal{CT}_{obs}$. Explicitly constructing $\mathcal{CT}_{obs}$ is difficult and time-consuming for arbitrarily shaped robots, since it requires a description of the obstacle region with respect to all possible rotations. Therefore we

explicitly construct the workspace-time obstacle region, $\mathcal{WT}_{obs}$, and incorporate the rotation at query time.

The volume occupied in $\mathcal{WT}$ by an agent moving along its trajectory forms a collision object. For general motions this collision object is usually non-convex. We define a trajectory for agent i, $\mathbf{T}^i$, as a time-ordered sequence of tuples of time, t_{k_i}, and configuration, $\mathbf{c}_{k_i}$: $\mathbf{T}^i = [(\mathbf{c}_{1_i}, t_{1_i}), (\mathbf{c}_{2_i}, t_{2_i}), \ldots, (\mathbf{c}_{K_i}, t_{K_i})]$. This restriction to discretized movements stems from the general difficulty to compute intersections for arbitrary analytic curves. The union of all volumes of the N agents (excluding the agent acquiring the plan),

$$\mathcal{WT}_{obs} = \bigcup_{i=1}^{N} \bigcup_{k=1}^{K_i} \mathbf{T}_k^i, \tag{4.1}$$

defines the obstacle region $\mathcal{WT}_{obs}$, where $\bigcup_{k=1}^{K_i} \mathbf{T}_k^i$ represents the workspace-time space collision object of agent i.

In a naive implementation (*multiple interference test*), the ego configuration of a candidate trajectory must be checked against the configuration of each agent at each discretized time, resulting in $O(KN)$ collision tests, where K is the number of configurations in the query trajectory, and N is the number of obstacle trajectories. By using a BVH data structure during broad phase collision detection, the number of required exact checks (narrow phase) between collision objects can be greatly reduced. Therefore, each pose of each collision object is inserted into the AABB tree. An exemplary three is depicted in Figure 4.1. The AABB tree is hereby usually constructed in a top-down manner, subsequently subdividing the set of collision shapes in the leaf nodes of the tree constructed so far. While for a binary tree the set of collision shapes is split into two subsets, different branching factors are possible. In the case of a binary tree, the splitting procedure first defines a single splitting plane which is then used to subdivide the set of collision shapes. The splitting plane is in general chosen to be orthogonal to the longest axis of the AABB. In Bergen (1997), the median of the AABB along the longest axis is taken as the position of the splitting plane. The splitting procedure is recursively executed until every leaf node contains only one collision shape. Figure 4.2 visualizes the construction of an exemplary AABB tree in two dimensions.

The root bounding box of the AABB tree fully contains all collision objects. Upon a query, the tree is traversed in depth, only for the branches where the axis-aligned bounding boxs of the query and the tree intersect. In case a leaf node is encountered, the collision detection goes into a *narrow phase* step in order to test the associated collision shapes for intersection. Narrow phase collision algorithms exist for various

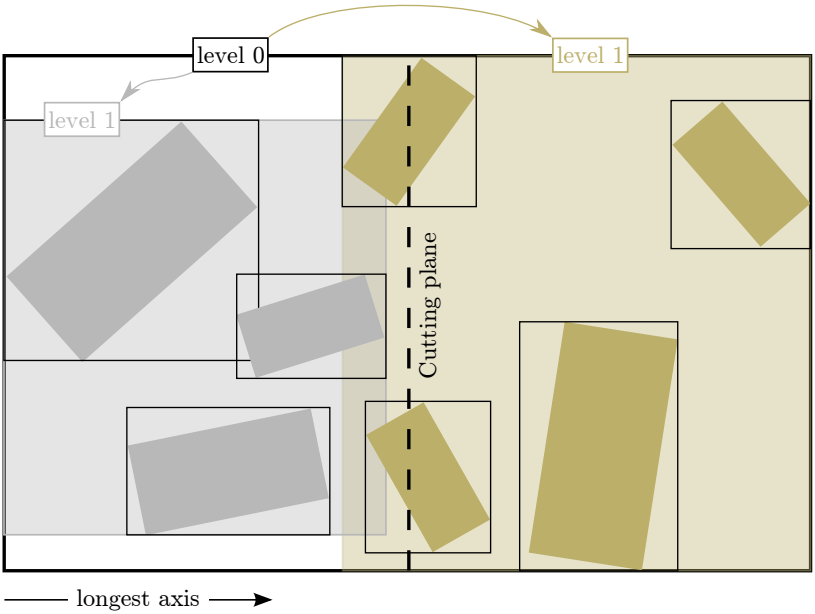


Figure 4.2: Visualization of the first subdivision step in the construction of a binary AABB tree in 2D. The root of the tree (black, level 0) possesses two children (gray/yellow, level 1). In order to fully construct the tree, the two sets of objects would be further subdivided until each leaf node contains only a single object.

collision shapes, however in the context of this thesis we exclusively restrict ourselves to oriented box shapes due to the focus on automotive application.

In order to integrate this collision detection technique into an online planning framework, there are various considerations within this approach that require further attention that we discuss in the following subsections.

4.2.1 Isolated Configuration Query

The simplest application is what we call the *isolated configuration query*. Each of the ego configurations, $\mathbf{T}_k^{\text{ego}}$, for each time instance t_k within a candidate motion is tested for collision against the union of collision objects over time, $\mathcal{WT}_{obs}$, individually. Though being inferior to the *tree-vs-tree* query presented in the next section in terms of computation time, this approach avoids the non-negligible overhead of computing a BVH data structure for each candidate motion. In a usual setting, the number of trajectories in the candidate motion set is much larger than the number of obstacles.

This fact may prevent the *tree-vs-tree* approach to be realizable with real-time constraints for a planner in which the candidate trajectories are generated on-the-fly. In fact, the *isolated configuration query* is used within our partial onlane planner.

Another fairly obvious choice is to abort the collision test for a trajectory, as soon as a collision is detected. Without prior knowledge of the index of the collision within the pose sequence, a linear traversal of the sequence is as good as any other. Applying a biased ordering (based on collision tests in previous planning cycles) can potentially turn out beneficial.

4.2.2 Tree-vs-Tree Query

Another option is to utilize a second BVH data structure for the trajectory in the query as displayed in Figure 4.3. In the broad phase of collision detection, two trees are tested against each other. This is done by traversing both trees top to bottom. This improves runtime performance with respect to the query, but comes at the additional cost of creating the BVH data structure for every candidate trajectory. For planners with predefined motion primitives such as for instance state-lattice planners or randomized trees with a fixed set of primitives, the tree structures for each candidate trajectory can be precomputed, further reducing the spare computation time for online operation. In Section 4.3 we provide profiling information to support the decision whether the *tree-vs-tree* query is beneficial in case precomputation is not possible. This is for example the case for motion planners presented in Chapter 3 or Werling et al. (2011), where the construction of motion primitives is dependent on the shape of a reference path and the relative position and orientation of the vehicle to it.

4.2.3 Time Discretization and Temporal Extent of Primitives

By default, a collision object does not have a temporal extent, i.e. a time-discretized trajectory of rectangular collision objects consists of a sequence of flat rectangular objects. Special attention has to be paid to apply the same time discretization for the obstacles' motion predictions and the candidate trajectories. Collisions will certainly be missed unless the time dimension is sampled at the same instances as illustrated in Figure 4.4.

Though applying an equal time discretization eliminates the danger of missed collisions for 2D objects, there are two arguments for introducing 3D primitives in $\mathcal{WT}$.

- Enforcing a minimum time-gap between trajectories becomes straight forward by specifying an extent of the collision shapes in the time dimension. Given a

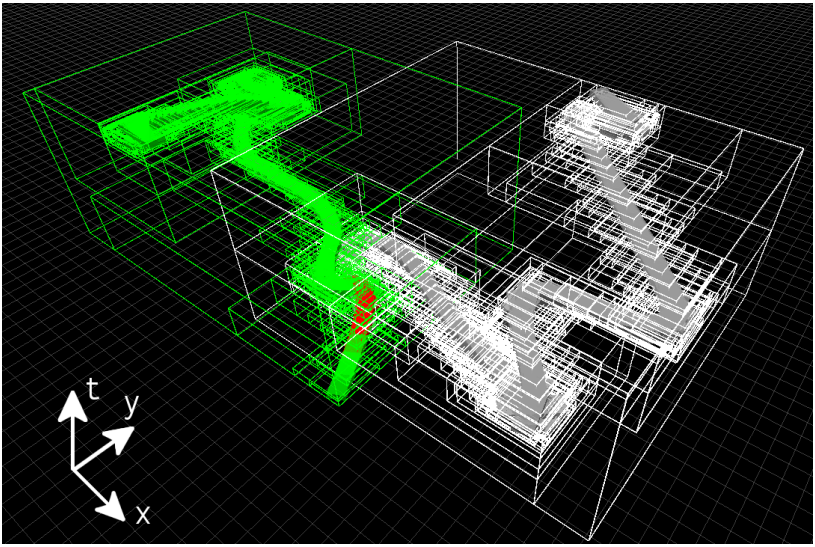
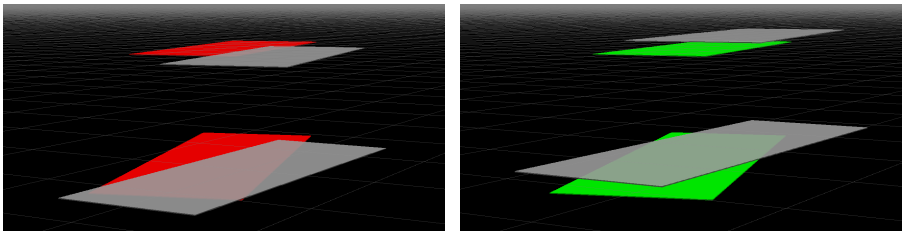


Figure 4.3: Collision test with two AABB tree data structures. The white tree contains all collision objects, the green tree all query configurations of one trajectory. Collisions can be detected faster through a more efficient resolution of the potential points of collision in workspace-time space.



(a) *(b)*

Figure 4.4: Problems arising from discretized trajectories. Grey boxes: obstacle shapes, red/green boxes: query shapes detected as colliding/not colliding, (a) Equal time discretization for ego and obstacle trajectories. Collisions are detected correctly. (b) Different time-discretization for ego and obstacle trajectories results in certainly missed collision detections.

time extent, the framework labels configurations as colliding if they occupy the same space *close in time*. This is a desirable behavior for many motion planning tasks—above all in automotive applications—as it results in less aggressive behavior of the robot.

- It allows for the use of highly optimized software libraries such as Smith, n.d., Coumans, n.d., or Pan et al. (2012) without modification, as these libraries are designed for collision detection between 3D objects.

4.2.4 Convex Hull Representation of Trajectories

The use of discretized motions gives rise to potentially missed collisions depending on the coarseness of the time-resolution. A way to overcome this issue is to compute the convex hull of every subsequent 2-tuple of configurations within a trajectory. The convex hull computation should be performed for both the collision objects and the query trajectories. The workspace-time obstacle of agent i is represented as a sequence of $N-1$ convex hulls

$$\mathcal{WT}_{obs,i} = \bigcup_{j=2}^{N} \text{convex_hull}\left(\mathbf{T}^i_{j-1}, \mathbf{T}^i_j\right), \tag{4.2}$$

Again, each of the convex hull shapes can be inserted into the AABB tree for accelerated collision queries. Figure 4.5 depicts the convex hull representations and the resulting AABB tree for three obstacles and one candidate ego trajectory, each coarsely discretized.

4.3 Results

Within this section we provide runtime results in order to give implementation guidance for the interested reader. The computations were performed on a single core of an Intel Core i7@2.67 GHz processor with 32 KiB / 256 KiB / 6 MiB of L1 / L2 / L3 cache. Profiling was done measuring CPU cycles with code compiled with gcc 4.8.4. For the sake of a more intuitive interpretation, CPU cycles are converted to timings via $t_{\text{comp}} = {}^{N_{\text{cycles}}}/_{2.67\,\text{GHz}}$. To account for disturbances such as multitask handling during the profiling step, each timing measurement is generated by the minimum CPU cycle count of 10 measurements. We utilized the AABB tree data structure from Willow Garage's Flexible Collision Library (FCL) (status May 21st, 2014). Note that a slightly more efficient implementation would be possible via a specialized narrow phase collision detection algorithm accounting for the fact that roll and pitch angles of the workspace-time space objects are always equal to zero for the 2D workspace

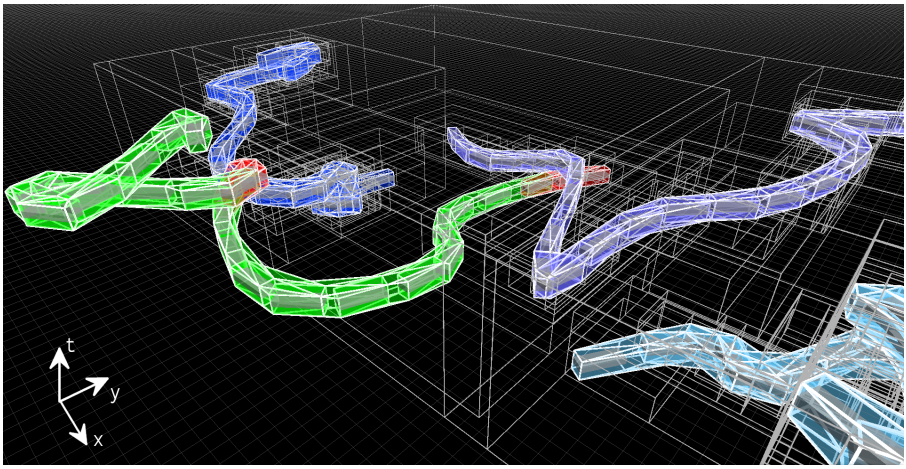


Figure 4.5: A convex hull representation in workspace-time space eliminates the problem of missed collisions due to coarse time-discretization and aliasing effects. This figure illustrates three obstacle trajectories (blue shades) and one ego trajectory (green/red). Collision queries between convex hull shapes can be accelerated with the AABB tree data structure (wireframe boxes).

at hand. In our experiments we used the standard implementation of the FCL library without the specialized narrow phase routine though. As runtimes are highly dependent on the proximity and collision state of the trajectories, trajectories are generated via a random walk.

Figure 4.6 shows the dependency between the number of obstacle trajectories and the runtime for the *isolated configuration query* (cf. Section 4.2.1) and *tree-vs-tree* approach (cf. Section 4.2.2) compared to the naive *multiple interference test*. In order to provide a fair comparison, the same narrow phase collision detection implementation and shape primitives were used. Due to our focus on automotive applications, we use 3D boxes (2D collision shapes with extent in time), which approximate the shape of most cars fairly well. For a varying amount of obstacle trajectories from 1 to 30, random scenes comprised of the ego trajectory and all obstacles trajectories were generated and tested for collision with each approach. The timing information is separated into two cases (a) in which the ego trajectory is in collision with at least one obstacle and (b) in which the ego trajectory is collision-free . As the probability of collision increases with the number of obstacles, random scenes were drawn until at least 1000 colliding and 1000 collision-free scenes were generated for each obstacle count.

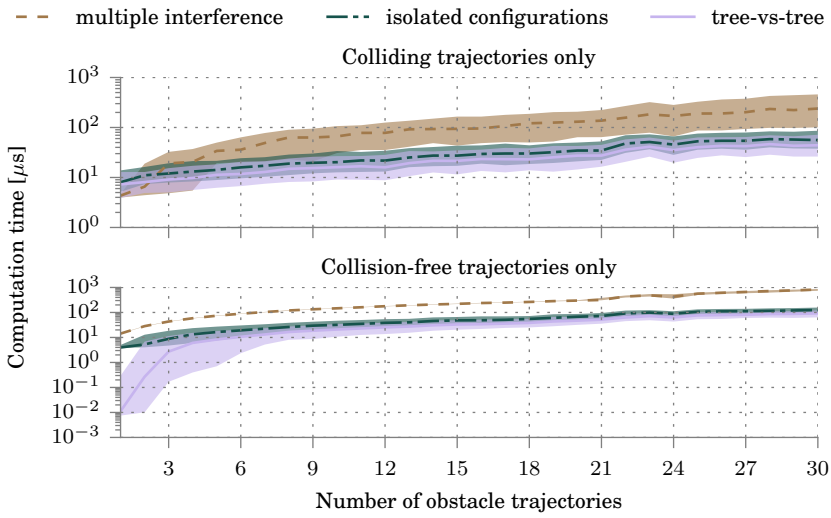


Figure 4.6: Median (lines) and 0.25/0.75-percentiles (filled areas) of computation times given in µs for collision test of one query trajectory with 150 poses with respect to number of obstacle trajectories for three different approaches. Both the *isolated configuration query* and *tree-vs-tree* approaches, utilizing a bounding volume hierarchy data structure in workspace-time space, are superior to the naive *multiple interference test*.

The case of a collision-free constellation represents the worst-case for the *multiple interference test* and the *isolated configuration query*. All configurations have to be tested in order to rule out a potential collision. As expected, the *multiple interference test* shows a linear dependency with respect to the number of obstacles (which is not obvious from the plot due to the logarithmic scale). Runtimes for the approaches utilizing the accelerating AABB tree data structure likewise grow due to the increasing depth of the tree, yet outperform the naive approach more and more as the number of obstacle trajectories increases. The *tree-vs-tree* approach—more effectively determining the potential points of collisions in workspace-time space—is hereby slightly superior to the *isolated configuration query*.

The performance difference shrinks when collisions start to occur due to the random time of occurrence within the ego trajectory, upon which further tests may be skipped. In case the collision occurs early, there is only a small number of narrow phase checks to be performed. Depending on the depth of the AABB tree, directly performing the narrow phase checks in the *multiple interference test* approach is sometimes faster

than the AABB tree accelerated queries, as the latter ones exhibit some overhead introduced by the tree traversal.

Figure 4.7 illustrates the computation time needed for construction of the AABB tree data structures for an increasing amount of obstacle trajectories. This operation has to be performed once per planning cycle, in case of fixed motion predictions within a cycle. For a total of 30 obstacle trajectories, the computation time for constructing the accelerating tree data structure is around 7.8 ms. We argue that 30 obstacles represents a fairly large number for a broad spectrum of applications and that the computation time for this one-time operation within a planning cycle is well suitable for online operation.

The runtime for the construction of the AABB tree data structure for one trajectory amounts to 260 µs. This clearly limits the number of ego trajectories that can be tested for collision within the *tree-vs-tree* approach in a real-time scenario, assuming a planning rate of approximately 100 ms is targeted. Sampling-based planners such as those presented in Werling et al. (2011) and Schwesinger, Rufli, et al. (2013) explore up to thousands of candidate trajectories, hence rendering the *tree-vs-tree* approach infeasible for planners relying on an online generation of the candidate trajectories. Yet it constitutes a valuable option if the tree data structures for the candidate trajectories can be precomputed in an offline step.

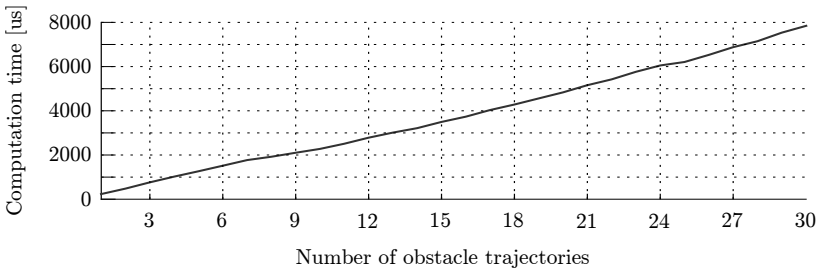


Figure 4.7: Mean runtimes given in microseconds for the construction of an AABB tree data structure for varying number of obstacle trajectories.

In our experiments, the computation time for the creation of a convex hull representation of one trajectory is 740 ± 38 (683 – 968) µs (`mean` ± `std` (`min` - `max`)). Here, we used the Bullet Physics Library (Coumans, n.d.) due to the immature support of convex hull shapes within FCL (Pan et al., 2012). Similar to the construction times of the accelerating AABB tree for the candidate trajectories, this exceeds the computation time available for online operation, but again constitutes a valuable option given the possibility for offline preprocessing. Although the computation time for a narrow

phase collision query between two convex hull shapes is 2 to 3 times slower than for 3D boxes, the computation time for complete trajectory collision queries with convex hulls is only around 20 % higher on average. Narrow phase collision checks only have to be performed in cases where the AABBs intersect at a leaf node of the tree, thus yielding computation times far below the sum of narrow phase checks.

The collision detection framework as described returns binary answers, i.e. whether a trajectory is in collision or not. Computing minimum distances instead would provide further useful information to motion planners in order to determine how risky a certain maneuver is. Distance computation likewise benefits from the accelerating tree data structures, as the distance between bounding volumes provides an upper bound for the distance of the collision shapes, and can be used to prune further distance computations. Minimum distance queries within our framework would naturally return a distance in workspace-time space. While this may seem unintuitive at first glance, the time dimension may be warped (stretched or compressed), such that these anisotropic distances in workspace-time space correspond to a notion of risk. As an example, passing another agent with a distance of one meter at the same time instance could be perceived as equally risky as occupying the same position one second apart. Imagine two agents traveling at high speed to realize that the two examples describe indeed different situations. Unfortunately, in our experiments, minimum distance computations with the *isolated configuration query* approach were around two orders of magnitude slower compared to binary collision queries. With computation times in the millisecond range for already a small number of objects, this renders the approach unsuitable for real-time operation. The *tree-vs-tree* approach did accelerate minimum distance queries in the range of an order of magnitude compared to distance computations with the *isolated configuration query* method. Given not too densely populated workspaces, these computation times would be in the acceptable range for real-time operation with a small set of candidate ego motions.

4.4 Summary and Discussion

In this section we presented AABB tree data structures in workspace-time space for accelerated collision queries between motion predictions of objects and computation of time-to-collision metrics. The collision detection framework handles arbitrary shapes and motions of moving obstacles. It is well suited for motion planning problems in narrow, dynamic environments, where approximations of collision shapes of objects are unacceptable.

We discussed and compared different settings within the framework. Testing single configurations against an AABB tree capturing the union of workspace-time obsta-

cles (*isolated configuration query*) results in substantial improvements in computation time over the naive *multiple interference test*, while keeping the computational overhead that comes with the construction of the accelerating AABB tree data structures low. The computation times obtained in the experiments underline the real-time applicability of this approach. Computing an accelerating AABB tree data structure for every candidate trajectory provides potential for further speed-up of collision queries, however comes with a significant additional computational cost for common sizes of the candidate trajectory set, rendering this approach only feasible if precomputation of the tree data structures is possible.

In the context of this thesis, the exploitation of temporal consistency was not investigated yet. Assuming object predictions change only slightly in between consecutive planning cycles, dynamic BVH trees, avoiding to build the tree from scratch in each cycle, could provide an additional speedup regarding the construction time of the tree data structures.

5

Cooperative Motion Planning via Maximum Entropy Models

In Chapter 3, a non-cooperative motion planning approach was presented, focusing on highly system-compliant planning and precise collision detection. As highlighted in Section 3.5, the approach revealed one major shortcoming. Over-conservative behavior could be observed when navigating in close proximity to other traffic participants, leading to sporadic emergency braking maneuvers as no collision-free trajectory could be found. This behavior is a consequence of too simplistic prediction models for other agents, as well as noise in the observations of their velocities. In this chapter, a cooperative motion planning approach building upon the work of Kuderer, Kretzschmar, Sprunk, et al. (2012) is pursued in order to address this shortcoming.

This chapter incorporates material from the following publication[1], awarded with the *IROS KROS Best Paper Award on Cognitive Robotics*, 2016:

Pfeiffer, M., Schwesinger, U., Sommer, H., Galceran, E., and Siegwart, R. (2016). "Predicting Actions to Act Predictably: Cooperative Partial Motion Planning with Maximum Entropy Models". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329

5.1 Introduction

Cooperative behavior is studied in a broad range of interdisciplinary fields, reaching from biology and evolutionary research to social sciences, finances and robotics. It

[1] This thesis extends the content of the aforementioned publication in many aspects, but especially with a more detailed description of the approach, new concepts in the model training stage and more extensive experimental results. This document tries to restrict itself to the parts of the collaborative work that the author of this thesis contributed.

describes the joint effort of two or more individuals to reach a common goal and constitutes the counterpart to competitive, selfish behavior. Cooperation is a deeply rooted concept in nature, including our society and everyday life, and it significantly contributes to the success of a species. Cooperative behavior of an individual may stem from several sources of motivation. It may find its origin in pure goodness of heart, helping other individuals to achieve their goals. In turn, one can argue that this selfless behavior could be explained with the selfless individual trying to maximize a more "spiritual" personal reward. The concept of maximizing a personal reward can also be found in game theory and was studied in the context of evolution. In groups of evolving species, cooperative behavior often dominates in the long-term over extortion. Cooperation may help individuals to maximize a purely personal reward through the return of favor by others. So, may it be out of moral implications, personal maximization of a reward or implied by externally imposed rules, cooperative behavior can be motivated by each of them.

In this thesis, we explore cooperative behavior in the context of robot motion planning on a trajectory level. The problem of robotic navigation in static, known environments is nowadays well-studied and understood. Yet navigating in dynamic environments with agents moving in a varied set of patterns is still an open research topic. In contrast to planning in static environments, where optimization of e.g. the path length or travel time is intuitive, the optimization objective in shared environments is often not well-defined. If the task is to integrate the robot in a smooth fashion into a workspace shared with other agents, optimization of e.g. travel time will result in aggressive behavior, perceived as unnatural by other individuals. Eventually we wish to realize a robot navigation style which is close (in some sense) to human behavior, commonly referred to as *social compliance* (Kretzschmar et al., 2014). Otherwise, a navigation behavior perceived as artificial will always stand out and will cause suboptimal interaction diminishing the navigation performance of the robot.

Joint navigation behavior is influenced by a multitude of factors, such as preferred avoidance distance of static obstacles and other agents, preferred velocities, goal-driven behavior and group formation aspects (Helbing and Molnár, 1995). Perhaps the hardest challenge to achieve social compliance is modeling the reaction of each individual agent to its neighbors. Neglecting these reactions and operating the robot in a two-staged "predict-then-react" fashion often leads to over-conservative actions of the robot as depicted in Figure 5.1 (left), and consequently to behavior not expected by other agents. Therefore, an effective solution requires a holistic approach, where motion of interacting agents is jointly treated.

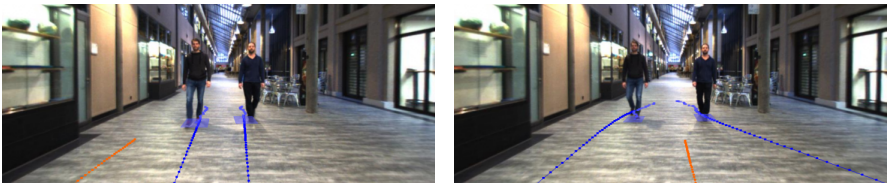


Figure 5.1: Predicted trajectories (blue) and resulting plan of robot (orange) with two pedestrians approaching the robot. *Left*: Neglecting the reaction of other agents to the robot's future motion through constant velocity predictions results in over-conservative actions of the robot. *Right*: In our maximum entropy model, we build a probability distribution over joint navigation behavior. The cooperative planning approach anticipates other agents' participation in mutual collision avoidance and effectively reduces over-conservative behavior.

5.2 Related Work

The task of modeling interactions between agents has been approached in different fields like mobile robotics, object tracking, behavioral science and character animation for video games, to name a few. Though the applications in these fields may be different, they share a common motivation to model interactions between agents: interaction-aware models help to forecast the actions of individual agents in the collective more accurately. In mobile robotics, more accurate prediction models help to improve the navigation behavior of an agent navigating in a workspace shared with other decision-making agents—this constitutes the application targeted in this thesis. Object tracking benefits from an improved prediction step to handle occlusions when multiple agents shall be tracked. In behavioral sciences, crowd formation aspects are studied inter alia to improve on emergency evacuation procedures of buildings. And realistic animation of characters is of major concern for the video game industry. In this section, the most relevant work in the respective fields is summarized.

Helbing and Molnár (1995) presented the social forces model, building upon the early work of Henderson (1971) who approached the problem of modeling crowd behavior using an analogy with fluid dynamics. The reactive social forces model treats agents as particles subject to social and physical forces. Attractive social forces model for instance goal-driven behavior, whereas repulsive social forces model interactions between agents. Physical forces originating from actual contacts between pedestrians were introduced to simulate high density crowds. Later, Helbing and Tilch (1998) applied the social forces model to simulate one-dimensional velocity scheduling of interacting vehicles in traffic. In the end however, the social forces model falls short on accurately describing individual agent motion within crowds. Lakoba et al. (2005)

later refined the model to better predict the motion of individual agents, while retaining the macroscopic realism achieved.

Conceptually quite similar to the social forces model, Reynolds (1999) suggested to use behavioral-driven steering functions that generate a force acting on agents. The author states that complex behaviors can be assembled from a set of simpler behavioral steering functions, modeling for instance obstacle avoidance, goal-directed behavior or escaping from other agents. However, the author acknowledges that interpolation between behaviors may be problematic as forces might cancel each other out, resulting in rather meaningless behaviors.

The Optimal Reciprocal Collision Avoidance (ORCA) approach by Berg, Guy, et al. (2011) is a popular method accounting for interactions between agents. ORCA computes joint collision-free velocity commands, assuming constant linear future motion, known preferred velocities and shared avoidance policies of agents. The approach originated from the fundamental work of Fiorini et al. (1998) on Velocity Obstacles (VO) and Berg, Lin, et al. (2008) on Reciprocal Velocity Obstacles (RVO), and proved to be a computationally efficient method for interaction aware motion planning, suitable for the simulation of large crowds of agents. The efficiency stems from the possibility to formulate the problem as a linear program in relative velocity space, such that the problem can be solved in real-time for thousands of agents. With the RVO framework, Berg, Lin, et al. (2008) addressed the issue of oscillating behavior ("reciprocal dance") inherent to the non-cooperative VO approach, caused by multiple agents taking over-conservative actions. ORCA introduced a suitable linearization procedure of the constraints in relative velocity space to yield a formulation which can be efficiently solved. A drawback of RVO and ORCA is their inability to handle probabilistic behavior and to learn from observed real-world behavior. Also, the linear extrapolation employed may be too simplistic for many situations. ORCA was extended by Alonso-Mora et al. (2013) to account for nonholonomic platform constraints. However, the nonholonomic extension is ill-suited for operation of highly constrained platforms in tight spaces, as it enlarges collision shapes of objects by their maximum tracking error with respect to linear trajectories. The assumption of known preferred velocities of agents is only valid for crowd simulations and multi-robot systems, where all agents are controlled through a centralized instance. For real-world deployment of a robot in a shared environment, preferred velocities of agents are not given per se. Recently, Kim et al. (2014) addressed this problem via recursive estimation of preferred velocities. The authors jointly estimate the positions, velocities and preferred velocities of agents in a Bayesian filter. This renders the approach more suitable for application in decentralized settings. Yet, it does not model "soft"

preferences, such as for instance comfortable accelerations and socially-compliant avoidance distances.

Guy et al. (2012) compared the capabilities of the three aforementioned approaches to generate realistic microscopic crowd behavior. He found that the RVO approach clearly outperforms the social forces model and Reynolds's steering method.

Targeting character animation, K. H. Lee et al. (2007) use a data-driven approach to synthesize microscopic agent behavior models. The authors use a locally weighted linear regression model to interpolate between a set of actions observed in similar real-world situations. Situations are described via features projected onto a lower dimensional space via principal component analysis. Different agent behaviors are extracted from real-world examples with a clustering technique, and selected during behavior generation from a high-level state machine. While crowd simulation techniques for character animation typically generate visually appealing motion patterns perceived as realistic in terms of macroscopic behavior, they are not effective for motion prediction in real-world applications. Usually global knowledge of crowd parameters such as crowd density (Treuille et al., 2006) or preferred motion direction/target locations (Berg, Patil, et al., 2008) for every agent is required. This information is in general difficult to obtain online.

In the context of object tracking, Pellegrini, Ess, Schindler, et al. (2009) use a handcrafted energy function to be minimized, sharing commonalities with the social forces model and the VO method. Parameters of the energy function are learned from real-world data. Though not constituting a truly interaction-aware model as every agent is forecasted assuming fixed trajectories of others, the authors show that incorporating basic collision avoidance techniques improves tracking performance. In Pellegrini, Ess, and Van Gool (2010), the authors demonstrate the benefit of modeling group formation aspects with respect to the quality of data association in a tracking application. More recently, Karasev et al. (2016) analyze the benefit of joint estimation of trajectories and goal locations on the quality of long-term predictions, additionally taking into account static obstacle information. The authors show that estimates of agents' goal locations can greatly stabilize long-term predictions of their future positions, given that the observation history is sufficiently long to yield reliable estimates.

Trautman and Krause (2010) introduced Interacting Gaussian Processes (IGP), where each agent's trajectory is modeled as a stochastic process in form of a Gaussian process (GP). Individual GPs are coupled through an interaction potential, modeling pairwise agent interactions. This work was later extended in Trautman, J. Ma, et al. (2013) through a GP mixture model to account for the multimodality originating from different possible goal locations of agents. The latter approach was extensively

tested in a public cafeteria. The results of the experiment suggested that modeling cooperation becomes increasingly more important with higher crowd densities. The cooperative IGP approach outperformed non-cooperative approaches in terms of efficiency (travel time) and the authors "suggest that modeling cooperation between agents is more important than modeling individual agent behavior". While the GP framework provides an exact inference method in general, the interaction potential coupling multiple GPs destroys the closed-form inference machinery. This forces Trautman and Krause (2010) to fall back on approximate inference via importance sampling. The authors claim that only a relatively low amount of samples was "sufficient for navigation" and can be computed in a time budget suitable for online operation. Yet we argue that random sampling in the planning stage is undesirable as it (a) usually comes with an overhead in computation time compared to analytic or iterative gradient-based optimization schemes, and (b) produces erratic behavior to some extend . The IGP approach provides the standard GP toolkit to learn the kernel hyperparameters of a single agent GP via marginal likelihood maximization. However, the IGP framework does not provide a solution how to tune or learn the potential terms related to agent interaction. Also, static obstacle avoidance has not been addressed.

A significant amount of work has been conducted on learning policies from demonstrations. Learning from demonstration describes a supervised learning technique, in which demonstrated expert behavior is used to derive a model capable of reproducing the demonstrated behavior. Learning such models from expert demonstration is appealing in many senses, as it constitutes an intuitive knowledge transfer from an expert of the task to the robot (Argall et al., 2009). The model might come in the form of a learned direct mapping from states to actions (policy), or as a reward function (Argall et al., 2009). Reinforcement learning (RL) assumes the reward function to be known and may exploit human demonstration to seed or direct the search for an optimal policy, whereas in inverse reinforcement learning (IRL) the learner tries to infer the expert's reward function from demonstrations. In the IRL setting, the learner can utilize the reward function to compute a policy for an encountered situation online by deriving the action sequence optimal with respect to the learned reward function. Knowledge representation in form of a direct mapping from states to actions as in RL is often criticized for not generalizing well to unseen situations. The expert's reward function inferred by IRL may represent a more general, compact and better transferable representation of the expert's knowledge (Ng and Russell, 2000 and Abbeel and Ng, 2004). However, the IRL concept is in general ill-posed, as potentially many reward functions explain the expert's behavior.

In the context of IRL, Abbeel and Ng (2004) learn a reward function represented as a linear combination of predefined feature functions. The feature weights indicate

the expert's trade-off between different desiderata. The authors derive the concept of matching feature expectations between an observed and the learner's policy to ensure that the learner performs comparable to the expert on its unknown linear reward function. The authors apply the approach to learn simple driving styles with a discrete state- and action space in a highway simulator. In Abbeel, Dolgov, et al. (2008), the approach is applied to generate human-like driving behavior on a parking lot.
The approach of Abbeel and Ng (2004) still suffers from the ambiguity that many reward functions may lead to the same feature expectations of the expert and the learner. Clearly the degenerate case where zero reward is assigned to all states is a solution, producing equal feature expectations independent of the policy. Ziebart, Maas, Bagnell, et al. (2008) resolve this ambiguity via the principle of maximum entropy. By maximizing the entropy of the distribution over resulting behavior subject to matching feature expectations, the distribution with minimal commitment to anything not imposed by the constraints is preferred. The authors apply the maximum entropy IRL formulation to learn driver route preferences from demonstrations of human preferences on a road network (Ziebart, Maas, Dey, et al., 2008). The maximum entropy IRL framework is extended by Henry et al. (2010) to operate in partially observable dynamic environments, and features are introduced to capture aspects of crowd navigation. However, the work focuses more on modeling macroscopic crowd phenomena rather than the effects of cooperative behavior on individual agent motion. In Kuderer, Gulati, et al. (2015), maximum entropy IRL is applied in the context of learning driving styles for autonomous highway driving, yet without incorporating other traffic participants.

Kuderer, Kretzschmar, et al. leverage the maximum entropy IRL approach of Ziebart, Maas, Bagnell, et al. (2008) to continuous state-spaces and introduce features to capture socially-compliant navigation behavior (Kuderer, Kretzschmar, Sprunk, et al., 2012; Kuderer, Kretzschmar, and Burgard, 2013; Kretzschmar et al., 2014; Kretzschmar, 2014; Kuderer, 2015 and Kretzschmar, Spies, et al., 2016). In their work, the authors build a maximum entropy distribution over a set of simultaneous agent trajectories. To cope with difficulties in the training and planning stage arising from the multimodality of this distribution, the authors resort to a mixture distribution over distinct homotopies. The approach provably outperforms the social forces and RVO models in terms of its predictive qualities for pedestrian motion. It is evaluated both on test datasets as well as on a mobile robotic platform, yet the latter evaluation involves only few pedestrians in a lab environment—most probably aware of the ongoing experiment. For the latter evaluation, the authors also assume known final destination points for each agent and perform a two-dimensional (2D) grid search to create a set of intermediate waypoints for the local interaction-aware planner. The local planner then computes the most probable trajectory set (including the ego agent),

given a fixed start point and a set of destination points for each agent. The strong assumption about known final destination points hinders experimental validation in real-world scenarios. It also imposes a strong commitment towards the presumably suboptimal high-level solution from the global waypoint planner.

In general, the aforementioned maximum entropy IRL methods work well if the expert's true reward function can be represented sufficiently well as a linear combination of feature functions. If this is not the case, the linear combination might not be expressive enough to model the true reward function. Additionally, one is burdened with the design of appropriate feature functions, which poses a challenge for more evolved tasks. To cope with nonlinear reward functions, Levine et al. (2011) suggest Gaussian processes (GPs) to model the expert's reward function over the feature space. They show superior performance for nonlinear reward structures, yet the method suffers from the growing complexity of GPs proportional to the amount of training data. Wulfmeier et al. (2016) apply a deep artificial neural network in a maximum entropy IRL framework. Deep neural networks alleviate the problem of the data-dependent model complexity of GPs found in Levine et al. (2011). Furthermore, convolutional layers in the network can provide a mechanism for automatic feature extraction, eliminating the need for manual feature design.

We observe a clear tendency from purely reactive prediction and planning methods such as the social forces model from Helbing and Molnár (1995) and the steering method by Reynolds (1999), towards data-driven, predictive approaches. Except the work of Kuderer, Kretzschmar, et al., the majority of the related work on IRL presented in this section was not applied to model cooperative navigation behavior. Their approach stands out as the only framework modeling cooperative behavior in a continuous state- and action space with a principled way to learn relevant parameters of the model from demonstrations. It shares similarities with Trautman and Krause (2010) and Trautman, J. Ma, et al. (2013) as both approaches formulate trajectories of agents as stochastic processes and couple those either via an "interaction potential" (Trautman and Krause, 2010) or via feature functions acting on pairwise trajectories (Kuderer, Kretzschmar, Sprunk, et al., 2012). In this thesis, we further explore the approach of Kuderer, Kretzschmar, et al.—particularly with the goal to leverage cooperation in order to better cope with noisy observations of other agents. In the following Section 5.3, we review the principle of maximum entropy.

5.3 The Principle of Maximum Entropy

Statistical entropy was first introduced by Shannon (1948) in the context of information transmission in order to quantify the information content of messages. Entropy

can be thought of as a measure of randomness or indifference, where large entropy corresponds to high uncertainty about the outcome of a random variable. Consequently, the maximum entropy distribution is the least-informative or most uniform distribution respecting certain testable information. The principle of maximum entropy states that the maximum entropy distribution is the only natural choice to correctly encode the absence of knowledge, since any other distribution would exhibit a bias not imposed by the known facts. Shannon first developed a discrete version of the entropy definition

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) \ , \tag{5.1}$$

usually referred to as *Shannon entropy*. Later, Shannon extended the discrete entropy to continuous domains by simply replacing the summation with an integral, yielding his definition of the *differential entropy*

$$H(p) = -\int_{\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) \ \mathrm{d}\mu(\mathbf{x}) \ , \tag{5.2}$$

with μ being a base measure on $\mathcal{X}$. The discrete Shannon entropy from Equation 5.1 may be derived from this definition by taking $\mathcal{X}$ to be a discrete set and $\mu(\mathbf{x})$ the counting measure. With μ taken as the Lebesgue measure on $\mathcal{X}$, implying $\mathrm{d}\mu(\mathbf{x}) = \mathrm{d}\mathbf{x}$, one can derive the more commonly found definition of differential entropy $H(p) = -\int_{\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x}) \ \mathrm{d}\mathbf{x}$. In contrast to Shannon entropy, the differential entropy depends on the definition of a base measure and is not invariant under coordinate transformations. For instance, a distribution over variables expressed in meters will have a different entropy than one expressed in centimeters, which was often criticized as they should have the same information content. In order to use differential entropy, we must choose a base measure against which to integrate. To alleviate this deficiency, Jaynes (1957) introduced an invariant measure $m(\mathbf{x})$ against which the entropy of a distribution is compared. As both the distribution under question and the invariant measure are equally affected by the effects of coordinate transformations, Jayne's definition of the continuous entropy

$$H(p) = -\int_{\mathcal{X}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{m(\mathbf{x})} \ \mathrm{d}\mathbf{x} \, , \tag{5.3}$$

becomes independent of a base measure. In the following we will assume the Lebesgue measure.

Assuming some known moment statistics, $\mathrm{E}_{p(\mathbf{x})}[\mathbf{f}(\mathbf{x})] = \int_{\mathcal{X}} p(\mathbf{x})\,\mathbf{f}(\mathbf{x})\,\mathrm{d}\mathbf{x} \overset{!}{=} \tilde{\mathbf{f}}$, of a set of measurable functions $\mathbf{f}(\mathbf{x})$, one can find the distribution with maximum entropy via the constrained functional optimization problem

$$\begin{aligned} p^{\star}(\mathbf{x}) &= \underset{p(\mathbf{x})}{\operatorname{argmax}}\, H(p) = \underset{p(\mathbf{x})}{\operatorname{argmax}}\; -\int_{\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x})\,\mathrm{d}\mathbf{x} \\ &\text{subject to} \\ &\int_{\mathcal{X}} p(\mathbf{x})\,\mathrm{d}\mathbf{x} = 1 \\ &\int_{\mathcal{X}} p(\mathbf{x})\,\mathbf{f}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \tilde{\mathbf{f}}\,. \end{aligned} \tag{5.4}$$

In addition to the moment constraints, we have plugged in the knowledge at this point that—in order to be a proper probability distribution—the probability mass must sum to one. The constraint that the density shall be non-negative everywhere, $p(\mathbf{x}) \geq 0\ \forall\ \mathbf{x} \in \mathcal{X}$, is omitted at this point for simplicity as we will see that this constraint is indeed non-binding and hence not required in the first place. Following the standard procedure for constrained optimization and introducing the Lagrange multipliers λ and $\boldsymbol{\theta}$, we can build the Lagrangian

$$\begin{aligned} \mathcal{L}(p, \lambda, \boldsymbol{\theta}) = &-\int_{\mathcal{X}} p(\mathbf{x}) \log p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \lambda\left(1 - \int_{\mathcal{X}} p(\mathbf{x})\,\mathrm{d}\mathbf{x}\right) + \\ &\boldsymbol{\theta}^{\top}\left(\tilde{\mathbf{f}} - \int_{\mathcal{X}} p(\mathbf{x})\,\mathbf{f}(\mathbf{x})\,\mathrm{d}\mathbf{x}\right). \end{aligned} \tag{5.5}$$

Taking the functional derivative of the Lagrangian with respect to the distribution under consideration and setting it to zero yields

$$\frac{\partial\,\mathcal{L}(p, \lambda, \boldsymbol{\theta})}{\partial p(\mathbf{x})} = -\log p(\mathbf{x}) - 1 - \lambda - \boldsymbol{\theta}^{\top} \cdot \mathbf{f}(\mathbf{x}) \overset{!}{=} 0\,. \tag{5.6}$$

It follows that the maximum entropy distribution has an exponential structure

$$p^{\star}(\mathbf{x}) = \exp\left(-\boldsymbol{\theta}^{\top} \cdot \mathbf{f}(\mathbf{x}) - \lambda - 1\right) = \frac{1}{\exp(\lambda + 1)} \exp\left(-\boldsymbol{\theta}^{\top} \cdot \mathbf{f}(\mathbf{x})\right) \tag{5.7}$$

and that the normalizing constant $z(\boldsymbol{\theta})$ must be chosen as

$$z(\boldsymbol{\theta}) := \exp(\lambda + 1) = \int_{\mathcal{X}} \exp\left(-\boldsymbol{\theta}^{\top} \cdot \mathbf{f}(\mathbf{x})\right) \mathrm{d}\mathbf{x} \tag{5.8}$$

in order to yield a properly normalized distribution. The exponential function is strictly non-negative for all $\boldsymbol{\theta}$ and $\mathbf{x}$.

The moment constraints are obviously satisfied if

$$\frac{\partial \mathcal{L}(p, \lambda, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \tilde{\mathbf{f}} - \int_{\mathcal{X}} p(\mathbf{x})\, \mathbf{f}(\mathbf{x})\, \mathrm{d}\mathbf{x} \stackrel{!}{=} 0\,, \tag{5.9}$$

which simply yields the original constraints. For the majority of cases it is not possible to obtain a solution for $\boldsymbol{\theta}$ in closed form, in which case one may fall back to sample approximations (see Section 5.7.2).

In the following chapters we will refer to the measurable function $\mathbf{f}(\mathbf{x})$ as *feature functions*, and to the Lagrange multipliers $\boldsymbol{\theta}$ as *feature weights*. As an exercise, we can derive the maximum entropy distribution for $x \in [a, b]$ with no constraints from Equations (5.6) to (5.8) as

$$p^\star(\mathbf{x}) = \frac{1}{\int_a^b 1 \mathrm{d}x} = \frac{1}{b-a}$$

Hence the maximum entropy distribution on a bounded interval without any implied knowledge about the distribution is the uniform distribution. This intuitively makes sense, as we inserted no knowledge why to prefer any outcome in $[a, b]$ over any other, and the maximum entropy distribution is most indifferent to anything beyond the encoded knowledge.

It is important to notice that the distribution may indeed include higher order statistics of the data, even though it is derived from first-order moment constraints. Imagine the single moment constraint $\mathrm{E}_{p(x)}\left[(x-\mu)^2\right] \stackrel{!}{=} \sigma^2$ for $x \in \mathbb{R}$ on the real line, which we recognize as a constraint on the variance. The structure of the maximum entropy distribution for this case is then

$$p^\star(\mathbf{x}) = \frac{\exp\left(-\theta_0 \cdot (x-\mu)^2\right)}{\int_{-\infty}^{+\infty} \exp\left(-\theta_0 \cdot (x-\mu)^2\right) \mathrm{d}x}$$

We can use the moment constraint from Equation 5.9 to determine $\theta_0 = \frac{1}{2\sigma^2}$, which yields the normal distribution as the maximum entropy distribution given a known variance.

It is also of importance to note that the maximum entropy distribution does not necessarily exist for arbitrary constraints, depending on the space $\mathcal{X}$. This is the case, when the normalizing constant $z(\boldsymbol{\theta})$ is not finite, i.e. the integral of the distribution over $\mathcal{X}$ is divergent. A simple example is $\mathrm{E}_{p(x)}[x] \stackrel{!}{=} \mu$ for $x \in \mathbb{R}$, which means we seek the maximum entropy distribution with known mean defined on the interval $[-\infty, +\infty]$. From Equation 5.8 we get

$$z(\theta_0) = \int_{-\infty}^{+\infty} \exp\left(-\theta_0 \cdot x\right) \mathrm{d}x = \infty \qquad \forall\, \theta_0 \in \mathbb{R}$$

for the normalizing constant and thus the maximum entropy distribution for this setting does not exist.

The *relative entropy*, also called the Kullback-Leibler divergence (Kullback et al., 1951), is a measure of the difference between two distributions $q(\mathbf{x})$ and $p(\mathbf{x})$ and is defined via

$$D_{\mathrm{KL}}(q(\mathbf{x}) \mid p(\mathbf{x})) = \int_{\mathcal{X}} q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \, \mathrm{d}\mathbf{x} \,. \tag{5.10}$$

It becomes handy when one desires to incorporate newly discovered information about a distribution at hand. Say we have a baseline model $p(\mathbf{x})$ and discovered the new moment constraint(s) $\mathrm{E}_{q(\mathbf{x})}[\mathbf{f}(\mathbf{x})] \overset{!}{=} \mathbf{f}_{\mathrm{new}}$. We wish to augment the baseline model with this new information. We can do so by minimizing the relative entropy, which boils down to finding the distribution $q(\mathbf{x})$ closest to $p(\mathbf{x})$ while obeying the new constraint. We will skip the derivation of the minimum relative entropy update at this point, as it closely follows Equations 5.4, 5.5, 5.7 and 5.8 with the definition of the relative entropy. The result for the update is

$$q^{\star}(\mathbf{x}) = \frac{1}{z(\boldsymbol{\theta})} p(\mathbf{x}) \exp\left(-\boldsymbol{\theta}^{\top} \cdot \mathbf{f}_{\mathrm{new}}(\mathbf{x})\right) , \tag{5.11}$$

where the normalizing constant $z(\boldsymbol{\theta})$ must be chosen such that $\int_{\mathcal{X}} q^{\star}(\mathbf{x}) = 1$.

This is consistent with the maximum entropy framework outlined above; with an (improper) uniform prior $p(\mathbf{x}) = 1$ for $\mathbf{x} \in \mathcal{X}$, Equation 5.11 is equivalent to Equation 5.7. Instead of maximizing the entropy of a distribution, we can think of it as minimizing the relative entropy to the uniform distribution.

5.4 Distribution over Composite Trajectories

In the following, we seek a maximum entropy distribution over the navigation behavior of multiple interacting and cooperating agents. Interaction is captured with a distribution defined over multiple agent trajectories. Each individual agent trajectory is hereby mathematically represented as a curve

$$\mathbf{x}_i : t \in [t_0, t_H] \mapsto \mathbf{x}_i(t) \in X = \mathbb{R}^d \tag{5.12}$$

defined over the time interval $[t_0, t_H]$ with d being the number of states. X is the extrinsic state space of a single agent, e.g. $\mathbb{R}^2$ to represent a 2D position. We aggregate the individual agent trajectories in a *composite trajectory*

$$\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_R, \ldots, \mathbf{x}_n) \,, \tag{5.13}$$

defined as the tuple of all n (relevant) agent trajectories, including the ego agent (robot) trajectory $\mathbf{x}_R$. The composite trajectory thus represents a curve in nd-dimensional space.

In our formulation, the end time of a composite trajectory is fixed and the end position is free, whereas in Kuderer, Kretzschmar, Sprunk, et al. (2012) the opposite is the case. We consider fixed end positions as problematic for real-world applications since (a) end positions for other agents are usually unknown and if estimated cannot be located at a single point, and (b) using end positions from a low fidelity high-level planner constrains us to potentially suboptimal solutions. Utilizing variable end positions eliminates the need to find a set of fixed waypoints for agents with an unknown destination and is especially helpful when deploying the robot in unknown (unmapped) environments. The release of the endpoints comes with another interesting property: With fixed end positions, different modes of the distribution such as those induced by static obstacles (passing left/right) are isolated, as trajectories from different modes cannot be continuously transformed into each other. With variable end positions, there always exists a continuous transformation through the composite "standstill" trajectory $\mathbf{x}(t) = \mathbf{x}(t_0)\ \forall\ t \in [t_0, t_H]$, assuming that the two composite trajectories are collision-free. We will exploit this fact in Section 5.7.2, when computing a sample approximation of the feature expectations needed for model training.

As the number of agents n may vary in between demonstrations and consecutive planning cycles, we condition the distribution on a *configuration* $C(\mathbf{x})$, describing e.g. the number of agents in the composite trajectory $\mathbf{x}$.

$$p(\mathbf{x} \mid \boldsymbol{\theta}, C(\mathbf{x})) = \frac{1}{z(\boldsymbol{\theta}, C(\mathbf{x}))} \cdot \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big), \tag{5.14}$$

This conditional distribution constitutes the one needed for the online planning task later, where all agents need to be predicted given their currently observed configuration. It also clarifies the sample space in Section 5.7. The maximum entropy distribution inherently encodes cooperative behavior due to its definition over a composite trajectory $\mathbf{x}$ in the time interval given by the planning horizon.

In practice we represent each trajectory as a time parametrized 2D B-spline (Boor, 1972). A composite trajectory is then defined as the concatenation of spline control vertices of individual agent trajectories, and the distribution from Equation 5.14 is in fact a distribution over the composite spline control vertices.

Though at this point not all details have been rolled out yet and the feature functions employed still have to be defined, we want to give an illustrative example of the maximum entropy distribution over composite trajectories. Figure 5.2 visualizes

the maximum entropy density via composite trajectory samples from the model for a configuration of two agents in a medium-sized office building room. Figure 5.3 provides a different view on the density in terms of the distribution of agent positions at varying time instances. The robot (blue) encounters an oncoming pedestrian with an unknown destination while attempting to reach its goal on the right. In the beginning

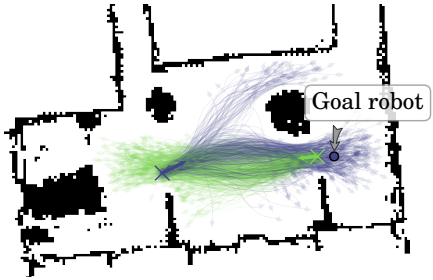
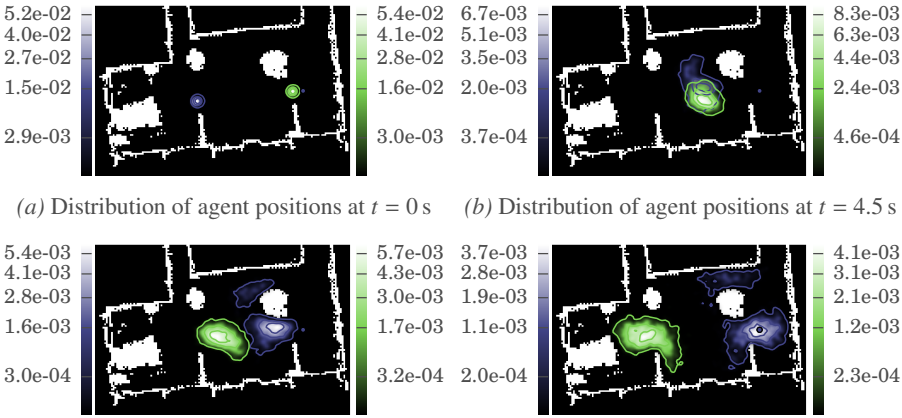


Figure 5.2: Samples from the maximum entropy distribution over the composite agent trajectory of two agents. The robot is located on the left (blue, known target location) at $t_0 = 0\,\mathrm{s}$ (blue cross) with a velocity vector pointing to the right. The density predicts a cooperative avoidance maneuver with the agent moving from right to left (green, unknown target). At the same time, the density rules out a composite trajectory in which either of the agents would collide with the static infrastructure (occupied cells in black). Figure 5.3 depicts the same density in the form of composite position probabilities over time.

at $t_0 = 0\,\mathrm{s}$, the density is highly concentrated around the measured positions of the agents (cf. Figure 5.3(a)). The density spreads out over time, accounting for static obstacles as well as interactions between the two agents. In the middle of the time interval, the density captures the interactions of the two agents, where the largest portion of the probability mass is assigned to a left-hand side avoidance maneuver (cf. Figure 5.3(b)). Note that the overlapping densities do not stem from collisions of the agents, but rather from two modes originating from passing either left or right. Towards the end of the time interval, the density is mainly influenced by the static obstacles, whereby the density captures several topological navigation options (cf. Figures 5.3(c) and 5.3(d)).

5.5 Feature Functions

The selection of meaningful features is essential in order to provide the model the chance to capture relevant statistics of the data. In this section we shine a light on the design of the specific feature functions employed in our framework.



(a) Distribution of agent positions at $t = 0$ s *(b)* Distribution of agent positions at $t = 4.5$ s

(c) Distribution of agent positions at $t = 7$ s *(d)* Distribution of agent positions at $t = 10$ s

Figure 5.3: Visualization of the maximum entropy distribution from Figure 5.2 in terms of the probabilities of agent positions at varying time instances. Colors encode the square root of the negative log-density in order to raise low probability regions. For visualization purposes, the density is smoothed with a Gaussian filter to attenuate sampling artifacts. Isocontours are drawn for 5 %, 33 % and 67 % of the maximum density per agent.

We consider three distinct kinds of features: (i) motion features, (ii) interaction features and (iii) observation features. The former two encode information about independent agent motion and agent-to-agent interaction, respectively. The third kind encodes information gained via observations of the agents' states. Each feature function is associated with a specific agent type or pair of agent types in the case of interaction features. This way, different motion and interaction models can be learned and used for planning depending on the types of agents encountered. For instance a vehicle and a pedestrian will move in a distinct fashion, and the interaction between pedestrians will differ from e.g. pedestrian-vehicle interaction.

The features employed in our framework are designed as C^1-continuous functions, i.e. functions with continuous first-order derivatives. This is motivated by the practical fact that we wish to utilize them within a fast Quasi-Newton optimization scheme for motion planning as explained in Section 5.8. Non-smooth functions pose some challenges for this optimization scheme (Lewis et al., 2013), that we intend to circumvent via the feature design in the first place.

In the following, we will give the mathematical formulation of each feature, and briefly discuss the underlying motivation.

5.5.1 Motion Features

Motion features are intended to capture the typical motion style of individual agents. Each motion feature maps a trajectory $\mathbf{x}_i$ of agent i to a scalar real value. The final feature value is obtained as the sum over agents in the composite trajectory, $f(\mathbf{x}) = \sum_{i=1}^{n} f(\mathbf{x}_i)$.

5.5.1.1 Velocity Feature

In line with the social force model from Helbing and Molnár (1995), we like to capture agents' typical navigation speeds. Therefore, for each agent i, we introduce two features

$$f_{\text{vel}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \|\dot{\mathbf{x}}_i(t)\|^2 \, \mathrm{d}t \tag{5.15}$$

and

$$f_{\text{vel,diff}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \left(\sqrt{\|\dot{\mathbf{x}}_i(t)\|^2 + \epsilon} - v_{\max} \right)^2 \mathrm{d}t \,, \tag{5.16}$$

where $v_{\max}$ is a sensible guess for the scalar upper bound of the translational velocity, and ϵ a small scalar constant to avoid the singularity of the derivative of the square root at velocity zero. The feature $f_{\text{vel,diff}}$, capturing the difference from the upper bound $v_{\max}$ of the translational velocity, is not required to describe the typical navigation speed (expectation), but may pull the location of the mode, used for online planning (cf. Section 5.8), away from zero translational velocity.

For vehicles in traffic it is presumably not sensible to learn typical driving speeds, as the driving speed is mostly determined by the imposed speed limits. In that case, one could learn to which extend drivers diverge from the speed limit. However, this was not pursued in this work.

5.5.1.2 Acceleration Feature

We expect agents to exhibit a tendency to minimize their accelerations throughout their movement. The feature

$$f_{\text{acc}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \|\ddot{\mathbf{x}}_i(t)\|^2 \, \mathrm{d}t \tag{5.17}$$

captures this aspect and is of major importance to prevent erratic trajectory solutions during online planning.

5.5.1.3 Direction of Motion Feature

We assume that agents have a preference to keep their current velocity and continue to move in the most recent direction. The feature

$$f_{\text{dir}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \|\dot{\mathbf{x}}_i(t) - \mathbf{v}_{i,t_0}\|^2 \, \mathrm{d}t \tag{5.18}$$

models this preference and shapes the distribution in the direction of the currently observed movement $\mathbf{v}_{i,t_0}$ of the agent.

5.5.1.4 Static Obstacle Feature

The motion of every agent is constrained and influenced by static obstacles in the world. On the one hand, agents physically cannot penetrate obstacles. On the other hand, agents will keep a safety distance, dependent on their current speed. By taking into account the reactions of all agents to static obstacles, the robot can act more courteously in situations such as depicted in Figure 5.4.

Figure 5.4: Predicting the reaction of other agents to static obstacles in the environment affects the robot's plan (orange) too. Based on the typical avoidance distance learned from demonstrations, the robot leaves the pedestrian more room for navigation by swerving to the left.

We model this intuition with three feature functions, one accounting for the *hard* no-collision constraint, $f_{\text{static,hard}}$, and two for the *soft* safety distance preference, $f_{\text{static,soft}}$ and $f_{\text{static,soft,vel}}$. All three features use a logistic function over the minimum static obstacle distance, mirrored at the ordinate, such that negative distances (inside obstacles) yield high costs and positive distances yield lower costs. We approximate collision shapes of agents with conservative bounding circles and incorporate the

radii r_i via a shift of the logistic function on the abscissa. The static obstacle feature responsible for hard collision avoidance,

$$f_{\text{static,hard}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \underbrace{\frac{h}{2} \cdot \left(1 + \tanh\left(-\frac{\eta}{2} \cdot (d_{\text{obs,min}}(\mathbf{x}_i(t)) - r_i)\right)\right)}_{F_{\text{static}}(h,\eta,r_i,\mathbf{x}_i(t))} \mathrm{d}t\,, \tag{5.19}$$

uses large values for both the height h and the steepness η of the logistic function in order to yield a C^∞-continuous approximation (with respect to $d_{\text{obs,min}}$) of a high step function. Figure 5.5 visualizes the function logistic function for varying steepness parameters η.

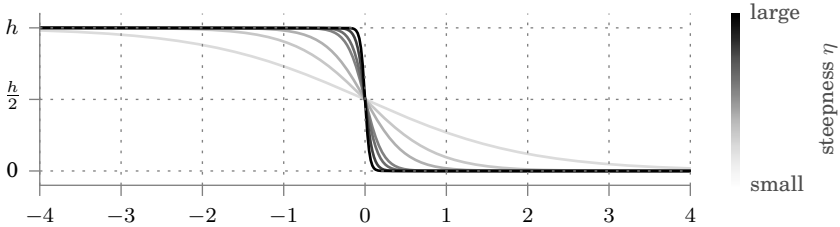


Figure 5.5: Logistic function mirrored at the ordinate for varying steepness parameters η. A larger steepness induces a sharper transition.

To model the soft distance preference, we introduce two features

$$f_{\text{static,soft}}(\mathbf{x}_i) = \int_{t_0}^{t_H} F_{\text{static}}(h, \eta, r_i, \mathbf{x}_i(t))\, \mathrm{d}t \tag{5.20}$$

and

$$f_{\text{static,soft,vel}}(\mathbf{x}_i) = \int_{t_0}^{t_H} F_{\text{static}}(h, \eta, r_i, \mathbf{x}_i(t)) \cdot \|\dot{\mathbf{x}}_i(t)\|^2\, \mathrm{d}t \tag{5.21}$$

that use moderate values for h and η. The feature $f_{\text{static,soft,vel}}$ additionally scales the logistic function with the squared velocity norm, which effectively reduces the impact of low probability regions arising in narrow corridors and lets the robot pass those with a low speed. It can be interpreted as a type of cross-product feature (Elith et al., 2011), capturing inter-dependencies between features—in this case obstacle distance and translational velocity.

To compute the closest distance to any obstacle, $d_{\text{obs,min}}(\mathbf{x}_i(t))$, we use a signed distance transform of a 2D binary occupancy grid representation of the static environment. The signed distance transform is computed by applying the method described

in Felzenszwalb et al. (2012) twice, once on the original grid, and also on the inverted version in order to obtain the negative distance field inside the obstacles. The method provides a fast discrete approximation of the continuous distance field within a few milliseconds for a reasonable grid size (in our case usually 200×200). Figure 5.6 depicts a distance transform of an exemplary occupancy grid, together with its isocontours and gradients. Additionally, we employ a bicubic interpolation with Catmull-Rom splines (Catmull et al., 1974) on the discrete distance field to ensure continuous gradients of the minimum obstacle distance with respect to the positions of the spline(s).

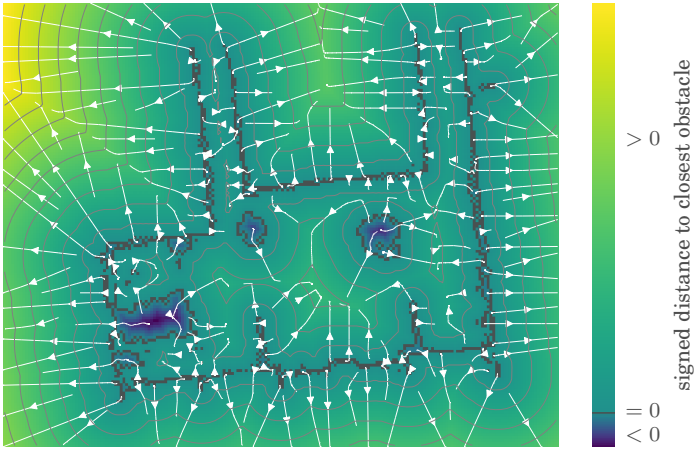


Figure 5.6: Signed distance transform of the occupancy grid representation of an exemplary static environment. Blue regions lie within occupied areas. Dark gray pixels highlight the boundary cells of obstacles. The signed distance transform provides gradient information (white lines) inside obstacles and is therefore especially suited for gradient-based optimization techniques. Additionally iso-contour lines are drawn in gray.

The approximation of collision shapes with conservative bounding circles is justified as long as the true geometry can be described with sufficient accuracy. The advantage of the approximation lies in the rotational invariance of the circle, allowing for minimum distance computation via a single lookup on the discrete distance transform, and straightforward computation of the derivative of the distance with respect to the configuration of the agent. For pedestrians the approximation is in general valid, whereas the quality of the approximation is inferior for cars. For rotation-variant collision shapes, the closest distance to any obstacle can be obtained via the minimum

value of the distance field inside the footprint of the shape. The gradient with respect to the spline control vertices can be approximated via finite differences. In this work however, we restrict ourselves to the bounding circle approximation.

5.5.1.5 Goal Feature

Agents are assumed to pursue a well-defined navigation goal such as reaching a certain position in the world. In the partial onlane planning approach presented in Chapter 3, goal-directed navigation behavior was achieved by penalizing offset to a reference path and speed profile. This way, motion at a certain speed along the lane center was rewarded. While this would likewise be possible to be encoded as a feature function in the maximum entropy framework, unstructured environments usually do not provide an obvious choice for a reference path such as lanes for automotive applications. If a high-level planner is employed to find a reference path in an unstructured environment, a suboptimal separation of the motion planning problem is introduced, and the usually richer information from the high-level planner is reduced to a single solution.

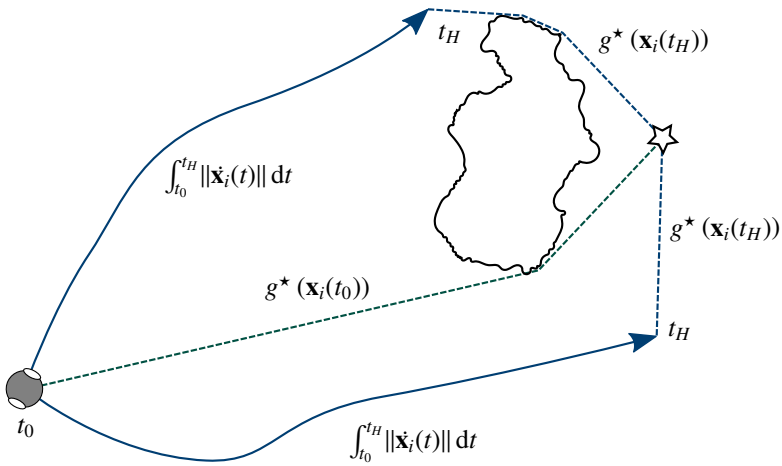


Figure 5.7: The goal feature computes the deviation of a trajectory from the shortest path to the goal location (star), taking into account static obstacles (region with black outline). $g^\star(\mathbf{x}_i(t_0))$ denotes the length of the shortest path (green) with its origin at the start position $\mathbf{x}_i(t_0)$ of the trajectory. The lengths of the blue paths (solid + dashed), $\int_{t_0}^{t_H} \|\dot{\mathbf{x}}_i(t)\| \, \mathrm{d}t + g^\star(\mathbf{x}_i(t_H))$, are the minimum path lengths achievable after executing $\mathbf{x}_i$. In the example in this figure, the feature values of both trajectories (blue) score equally.

In our maximum entropy framework we remove the dependency on a high-level planner by introducing a feature to capture the agents' preferences to follow short paths the goal

$$f_{\text{goal}}(\mathbf{x}_i) = \int_{t_0}^{t_H} \|\dot{\mathbf{x}}_i(t)\| \, \mathrm{d}t + g^\star\left(\mathbf{x}_i(t_H)\right) - g^\star\left(\mathbf{x}_i(t_0)\right) \,, \tag{5.22}$$

where $g^\star(\cdot)$ denotes the length of the shortest path to the goal location starting from the position in its argument. The shortest path length is taken as the geodesic distance, taking into account static obstacles. As visualized in Figure 5.7, the first two terms of the integrand denote the minimum possible path length after following trajectory $\mathbf{x}_i$, and the third term represents the length of the shortest path starting from the current location of the robot. The feature concentrates the probability mass around paths with short distances to the goal location. Hereby, the feature yields similar values for topologically different routes in case their path length to the goal is comparable, and with that avoids the loss of information when using a single reference path.

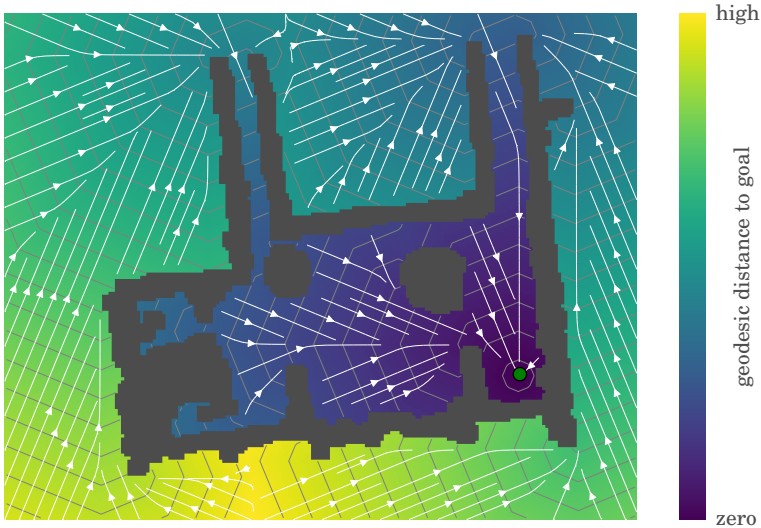
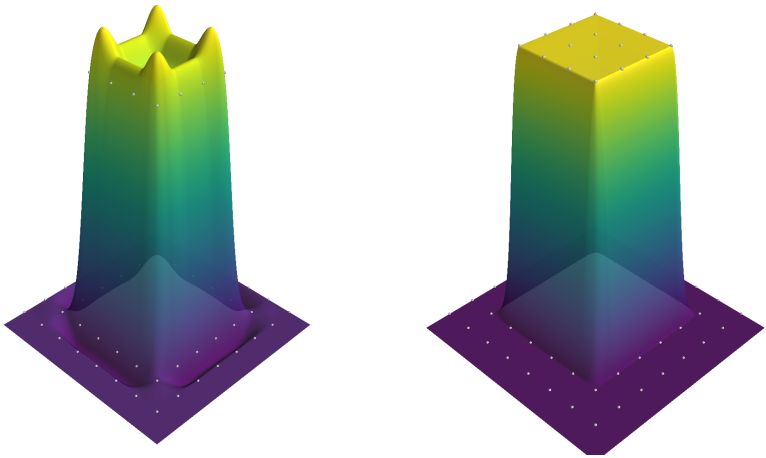


Figure 5.8: Geodesic distance to goal (green dot) computed via Dijkstra's algorithm with an 8-connected neighborhood. The obstacle map is dilated by the robot's radius. The gradient of the geodesic distance to the goal (white lines) provides valuable information in cul-de-sac situations (e.g. top, middle) and serves as global guidance.

The geodesic distances can be efficiently computed on a grid with Dijkstra's algorithm (Dijkstra, 1959), allowing to incorporate obstacle information on-the-fly. An

exemplary geodesic distance field is depicted in Figure 5.8. The goal distance field provides an important global guidance to the otherwise short-sighted receding horizon approach and effectively deals with cul-de-sac situations.

As in the case of the static obstacle feature, the discretized geodesic distance field is interpolated with a bicubic interpolation scheme to ensure smooth gradients. However, in contrast to the signed obstacle distance, the geodesic distance field exhibits large jumps at obstacle boundaries, causing substantial over- and undershoots of the Catmull-Rom interpolation method (Catmull et al., 1974) used above. This behavior is depicted in Figure 5.9(a). Especially the undershoots close to obstacles pose



(a) Catmull-Rom splines *(b)* PCHIP method

Figure 5.9: Comparison of two bicubic interpolation methods on data with a plateau due to a step change. The Catmull-Rom method produces both over- and undershoot at the cells adjacent to the step which is fatal for gradient-based optimization techniques. The PCHIP (Piecewise Cubic Hermite Interpolating Polynomials) method is more expensive to compute, but the shape-preserving nature prevents over- and undershoots at step changes.

a problem, as they wrongly suggest that the distance to the goal can be significantly reduced when moving very close to obstacles. To avoid this problem, we utilize a shape-preserving interpolation scheme for the goal feature based on Piecewise Cubic Hermite Interpolating Polynomials (PCHIP) presented in Fritsch et al. (1980). This interpolation scheme is more costly to compute, however preserves the monotonicity of the underlying data (cf. Figure 5.9(b)).

5.5.2 Interaction Features

Interaction features finally introduce the cooperative aspect into the framework. By allowing the model to capture statistics of the data that depend on the simultaneous motion of multiple agents, the maximum entropy distribution becomes sensitive to joint navigation behavior. In our framework, interaction features model the interaction between each pair of agent trajectories within a composite trajectory $\mathbf{x}$. Each interaction feature maps a 2-tuple of trajectories $(\mathbf{x}_i, \mathbf{x}_j)$ of agent i and agent j to a scalar real value. Finally, the sum over agent tuples in the composite trajectory, $f(\mathbf{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f(\mathbf{x}_i, \mathbf{x}_j)$, comprises the feature value.

As stated in Helbing and Molnár (1995), agents tend to dislike other agents entering their *private spheres*. By using the integral of a monotonically decreasing function of the Euclidean distance of two agents as in Kuderer, Kretzschmar, Sprunk, et al. (2012) and Kretzschmar et al. (2014), the model can capture typical avoidance distances. Again, we model the physical impossibility of two agents occupying the same position in the world at the same time and the proximity discomfort of the agents as separate feature functions. A similar formulation as for the static obstacle features in Equations 5.19 and 5.20 is used with

$$f_{\text{dyn,hard/soft}}(\mathbf{x}_i, \mathbf{x}_j) = \int_{t_0}^{t_H} \frac{h}{2} \cdot \left(1 + \tanh\left(-\frac{\eta}{2} \cdot \left(\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|^2 - (r_i + r_j)^2\right)\right)\right) \mathrm{d}t\,. \tag{5.23}$$

Two parameter sets for h and η yield one steep logistic function that models the hard no-collision constraint and one flatter logistic function that models the discomfort.

Figure 5.10 depicts a challenging joint navigation problem in which both static obstacle avoidance and agent-to-agent interaction play a crucial role to accurately forecast the agents' trajectories.

More expressive interaction feature functions might incorporate anisotropic distances, accounting for the fact that the agents' discomfort is presumably higher if another agent is located in front rather than behind of them. We leave this for future work at this point.

5.5.3 Observation Features

Observation features comprise a probabilistic measurement model that copes with the noisy observations of the observed agents. We employ an isotropic Gaussian noise model for each position measurement, $\mathbf{x}_{i,k}$, and velocity measurement, $\dot{\mathbf{x}}_{i,k}$, we obtained for an agent i in the world.

$$f_{\text{obs,pos}}(\mathbf{x}_i) = \sum_k \left(\mathbf{x}_{i,k} - \mathbf{x}_i(t_k)\right)^\top \left(\mathbf{x}_{i,k} - \mathbf{x}_i(t_k)\right) \tag{5.24}$$

Figure 5.10: Interaction features capture typical aspects of pairwise agent motion such as agent's comfort distances. In conjunction with static obstacle information, the model can predict complex scenarios as for instance an agent (*left*) giving room to another (*right*) due to a predicted conflict with a static obstacle.

$$f_{\text{obs,vel}}(\mathbf{x}_i) = \sum_k (\dot{\mathbf{x}}_{i,k} - \dot{\mathbf{x}}_i(t_k))^\top (\dot{\mathbf{x}}_{i,k} - \dot{\mathbf{x}}_i(t_k)) \tag{5.25}$$

Note that Equations 5.24 and 5.25 describe Gaussian densities when combined with the exponential maximum entropy structure. The corresponding feature weights are related to the uncertainty of the measurements. To this point, we do not adapt the weights of observation features in the training stage due to a lack of ground-truth information.

5.6 Baseline Model

Attempting to learn every navigation aspect from data is questionable. The fact that agents cannot penetrate static obstacles and each other is given by laws of (classical) physics, and learning such an aspect from data would solely complicate the training procedure. Therefore we incorporate some of the feature functions outlined above in a baseline model, and use the principle of minimum relative entropy from Section 5.3 to train the remaining feature weights. According to Equation 5.11, the model utilized in the training stage thus becomes

$$p(\mathbf{x}) = \frac{\exp\Big(- [\boldsymbol{\theta}_{\text{fixed}}, \boldsymbol{\theta}_{\text{learn}}]^\top \cdot [\mathbf{f}_{\text{fixed}}(\mathbf{x}), \mathbf{f}_{\text{learn}}(\mathbf{x})] \Big)}{\int_{\mathcal{X}} \exp\Big(- [\boldsymbol{\theta}_{\text{fixed}}, \boldsymbol{\theta}_{\text{learn}}]^\top \cdot [\mathbf{f}_{\text{fixed}}(\mathbf{x}), \mathbf{f}_{\text{learn}}(\mathbf{x})] \Big) \mathrm{d}\mathbf{x}}, \tag{5.26}$$

with $\boldsymbol{\theta}_{\text{fixed}} / \mathbf{f}_{\text{fixed}}$ being the fixed features/weights not adapted during training, and $\boldsymbol{\theta}_{\text{learn}} / \mathbf{f}_{\text{learn}}$ the features/weights to be learned. Our baseline model contains the fea-

tures for hard static and dynamic obstacle avoidance, as well as the observation features. The remaining features capturing *soft preferences* are learned from real-world demonstrations, as detailed in the next Section 5.7. For the sake of brevity, we will omit the subscript $(\cdot)_{\text{learn}}$ for the trainable features in the following.

5.7 Model Training

In the training stage, we seek the feature weights $\boldsymbol{\theta}^\star$ best explaining a set $\mathcal{X}_D$ of observed composite trajectories. If the model is trained with observations of cooperative behavior, the robot will be able to reproduce such behavior when utilizing the model for online navigation.

5.7.1 The Loss Function

We formulate a loss function as the negative sum of log-likelihoods of the demonstrations

$$\ell\left(\mathcal{X}_D;\boldsymbol{\theta}\right) := -\sum_{\mathbf{x}\in\mathcal{X}_D} \ln p\left(\mathbf{x}|\boldsymbol{\theta}, C(\mathbf{x})\right) \tag{5.27a}$$

$$= \sum_{\mathbf{x}\in\mathcal{X}_D} \Big(\boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) + \ln z(\boldsymbol{\theta}, C(\mathbf{x}))\Big), \tag{5.27b}$$

assuming that all $\mathbf{x} \in \mathcal{X}_D$ are independent. This requires that, for any two composite trajectories, either the time intervals are disjoint, or the trajectories involve different agents. $\ln z(\boldsymbol{\theta}, C(\mathbf{x}))$ represents the log normalizing constant of $p\left(\mathbf{x}|\boldsymbol{\theta}, C(\mathbf{x})\right)$. We then obtain the maximum likelihood parameter estimates $\boldsymbol{\theta}^\star$ from

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \ell\left(\mathcal{X}_D;\boldsymbol{\theta}\right) . \tag{5.28}$$

To efficiently search for $\boldsymbol{\theta}^\star$ with gradient-based optimization techniques, one needs to evaluate the gradient of the loss function given in Equation 5.27 with respect to the feature weights. For each demonstration $\mathbf{x} \in \mathcal{X}_D$ we get

$$\frac{\partial}{\partial \boldsymbol{\theta}^\top} - \ln p\left(\mathbf{x};\boldsymbol{\theta}, C(\mathbf{x})\right) = \mathbf{f}(\mathbf{x}) + \frac{\partial}{\partial \boldsymbol{\theta}^\top} \ln z(\boldsymbol{\theta}, C(\mathbf{x})) . \tag{5.29}$$

We adopted the notation here, that $\frac{\partial}{\partial \boldsymbol{\theta}}$ and $\frac{\partial}{\partial \boldsymbol{\theta}^\top}$ yield a row- and column vector, respectively. The second term on the right-hand side of Equation 5.29 is given by the negative feature expectations of the distribution

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}^\top} \ln z\left(\boldsymbol{\theta}, C(\mathbf{x})\right) &= \frac{\partial}{\partial \boldsymbol{\theta}^\top} \ln \int_{\mathcal{X}} \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x} && (5.30a)\\
&= \frac{1}{\int_{\mathcal{X}} \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x}} \cdot \int_{\mathcal{X}} \frac{\partial}{\partial \boldsymbol{\theta}^\top} \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x} && (5.30b)\\
&= \int_{\mathcal{X}} \frac{1}{z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)} \cdot \frac{\partial}{\partial \boldsymbol{\theta}^\top} \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x} && (5.30c)\\
&= \int_{\mathcal{X}} \frac{1}{z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)} \cdot \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \cdot \Big(- \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x} && (5.30d)\\
&= - \int_{\mathcal{X}} p(\mathbf{x}|\boldsymbol{\theta}, C(\mathbf{x})) \cdot \mathbf{f}(\mathbf{x}) \; \mathrm{d}\mathbf{x} && (5.30e)\\
&= - \mathrm{E}_{p(\cdot|\boldsymbol{\theta}, C(\mathbf{x}))} \left[\mathbf{f}\right] , && (5.30f)
\end{aligned}$$

where $\mathrm{E}_{p(\cdot|\boldsymbol{\theta}, C(\mathbf{x}))} \left[\mathbf{f}\right]$ denotes the feature expectations with respect to the maximum-entropy distribution $p(\mathbf{x}|\boldsymbol{\theta}, C(\mathbf{x}))$.

This results in an intuitive equation for the gradient of the loss function

$$\frac{\partial}{\partial \boldsymbol{\theta}^\top} \ell\left(\mathcal{X}_D; \boldsymbol{\theta}\right) = \sum_{\mathbf{x} \in \mathcal{X}_D} \Big(\mathbf{f}(\mathbf{x}) - \mathrm{E}_{p(\cdot|\boldsymbol{\theta}, C(\mathbf{x}))} \left[\mathbf{f}\right] \Big), \tag{5.31}$$

where we have to match the feature expectations of the maximum entropy model with the ones from demonstrated composite trajectories.

The Hessian of the log normalizing constant is given by the covariance matrix of the features. This can be derived with the help of

$$\frac{\partial}{\partial \boldsymbol{\theta}^\top} \Big(z\left(\boldsymbol{\theta}, C(\mathbf{x})\right) \Big) = \int_{\mathcal{X}} -\mathbf{f}(\mathbf{x}) \exp\Big(- \boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x}) \Big) \mathrm{d}\mathbf{x} = -z\left(\boldsymbol{\theta}, C(\mathbf{x})\right) \cdot \mathrm{E}_{p(\cdot|\boldsymbol{\theta}, C(\mathbf{x}))} \left[\mathbf{f}\right] \tag{5.32}$$

starting from Equation 5.30b via

$$\frac{\partial^2}{\partial\boldsymbol{\theta}^\top\partial\boldsymbol{\theta}} \ln z\left(\boldsymbol{\theta}, C(\mathbf{x})\right) = \frac{\partial}{\partial\boldsymbol{\theta}}\left(\frac{1}{z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)} \cdot \int_{\mathcal{X}} \frac{\partial}{\partial\boldsymbol{\theta}} \exp\Big(-\boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x})\Big)\mathrm{d}\mathbf{x}\right) \tag{5.33a}$$

$$= -\frac{\frac{\partial}{\partial\boldsymbol{\theta}^\top}\Big(z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)\Big)}{z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)^2} \cdot \int_{\mathcal{X}} -\mathbf{f}^\top(\mathbf{x}) \exp\Big(-\boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x})\Big)\mathrm{d}\mathbf{x}$$

$$+ \frac{1}{z\left(\boldsymbol{\theta}, C(\mathbf{x})\right)} \cdot \int_{\mathcal{X}} \frac{\partial^2}{\partial\boldsymbol{\theta}^\top\partial\boldsymbol{\theta}} \exp\Big(-\boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x})\Big)\mathrm{d}\mathbf{x} \tag{5.33b}$$

$$= -\mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}\right] \cdot \mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}^\top\right]$$

$$+ \int_{\mathcal{X}} \mathbf{f}(\mathbf{x})\,\mathbf{f}^\top(\mathbf{x}) \exp\Big(-\boldsymbol{\theta}^\top \cdot \mathbf{f}(\mathbf{x})\Big)\mathrm{d}\mathbf{x} \tag{5.33c}$$

$$= \mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f} \cdot \mathbf{f}^\top\right] - \mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}\right] \cdot \mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}^\top\right] \tag{5.33d}$$

$$= \mathrm{Cov}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}\right] . \tag{5.33e}$$

The Hessian of the loss function is easily derived with Equation 5.27 as

$$\nabla^2 \ell\left(\mathcal{X}_D; \boldsymbol{\theta}\right) = |\mathcal{X}_D| \cdot \mathrm{Cov}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}\right] . \tag{5.34}$$

As the covariance matrix is guaranteed to be positive semidefinite, the loss function is always non-strictly convex, guaranteeing a unique minimum value of the loss function. If the feature covariance matrix has full rank, i.e. there exists no feature(s) which may be represented as a linear combination of others, the loss function is even strictly convex, which additionally yields a unique feature weight vector at the minimum. However, in rare cases, finite sample approximations (cf. Section 5.7.2) may destroy positive (semi-)definiteness due to sample fluctuations (Anderson et al., 1984).

5.7.2 Computation of Feature Expectations

Analytical evaluation of the feature expectations, required for the gradient of the loss function with respect to the feature weights in Equation 5.31, is impossible for all but the simplest feature functions. Aggravating this situation, the integral involved in the feature expectations needs to be computed over the infinite-dimensional function space of composite trajectories. Even with the parametric trajectory representation as B-spline curves, the dimensionality of the sample space is still very high-dimensional and challenging for numerical integration routines. In practice, we use a Markov chain Monte Carlo (MCMC) sample approximation

$$\mathrm{E}_{p(\cdot|\boldsymbol{\theta},C(\mathbf{x}))}\left[\mathbf{f}\right] = \int_{\mathbf{x}} p(\mathbf{x} \mid \boldsymbol{\theta}, C(\mathbf{x})) \cdot \mathbf{f}(\mathbf{x})\ \mathrm{d}\mathbf{x} \approx \frac{1}{|\mathfrak{X}|} \sum_{\mathbf{x}\in\mathfrak{X}} \mathbf{f}(\mathbf{x}) \tag{5.35}$$

of the feature expectations. After computation of the feature expectations, the gradient given in Equation 5.31 can be used in conjunction with an iterative, gradient-based optimization technique to obtain the maximum likelihood estimate of the feature weights. Therefore, computation of the feature expectations has to be repeated for each new iteration and estimate of the parameters $\boldsymbol{\theta}$.

MCMC methods are nowadays widely used to sample from complex distributions where direct sampling is not possible. MCMC methods operate by building a Markov chain in the corresponding state space which has the desired distribution as equilibrium distribution. First, the Markov chain is simulated until convergence to its stationary distribution. This convergence phase, commonly called *burn-in phase*, can by drastically shortened with a good initialization point of the chain. Oftentimes, the mode of the distribution, obtained via gradient-based optimization as detailed in Section 5.8, serves as a good initialization point. After burn-in, the states of the chain represent samples from the target distribution, though subject to correlation induced by the Markov chain itself. Optimally, the chain exhibits no correlation in which case all states can be used as independent samples from the target distribution. If correlation is high, the information about the target distribution carried by the samples is reduced. More precisely, correlation reduces the effective sample size by $n_{\text{eff}} = {}^{n}/_{1+\tau_f}$ with n being the original number of samples and τ_f being the integrated autocorrelation time. The integrated autocorrelation time

$$\tau_f = 1 + 2 \sum_{k=1}^{\infty} \rho(k) \,, \tag{5.36}$$

computed via the summation of the chain's autocorrelation coefficients ρ at lag k, is therefore often taken as a performance measure to compare different MCMC algorithms.

5.7.2.1 The Metropolis-Hastings Algorithm

The fundamental problem in MCMC is how to construct a Markov chain with the desired properties. Hastings (1970) developed an astonishing simple way of constructing a Markov chain with a desired stationary target distribution. Utilizing a proposal distribution $q(\tilde{\mathbf{x}}|\mathbf{x})$, the Metropolis-Hastings algorithm generates a new candidate successor $\tilde{\mathbf{x}}$ of the current state $\mathbf{x}$. The candidate state is accepted with probability

$$p\left(\tilde{\mathbf{x}} \mid \mathbf{x}\right) = \min\left\{1, \frac{p\left(\tilde{\mathbf{x}}\right) \cdot q(\mathbf{x}|\tilde{\mathbf{x}})}{p\left(\mathbf{x}\right) \cdot q(\tilde{\mathbf{x}}|\mathbf{x})}\right\} . \tag{5.37}$$

In case of a symmetric proposal distribution with $q(\mathbf{x}|\tilde{\mathbf{x}}) = q(\tilde{\mathbf{x}}|\mathbf{x})$, the acceptance criterion further simplifies to

$$p(\tilde{\mathbf{x}} \mid \mathbf{x}) = \min\left\{1, \frac{p(\tilde{\mathbf{x}})}{p(\mathbf{x})}\right\}. \tag{5.38}$$

A new state is always accepted if the probability increased, and otherwise accepted with a probability proportional to the fraction of the density values at the current and candidate state. Due to the Metropolis acceptance criterion from Equation 5.37 or Equation 5.38, only a density proportional to the target density has to be provided, mitigating the problem of computing the normalizing constant $z(\boldsymbol{\theta}, C(\mathbf{x}))$ from Equation 5.14. The proposal distribution q has to yield an ergodic Markov chain, i.e. the probability of reaching any state from any other state must be positive, to ensure a unique stationary distribution. A widely used proposal distribution is the normal distribution $q(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}\left(\mathbf{x}, \sigma^2\mathbf{I}\right)$ centered at the current state of the system $\mathbf{x}$ with standard deviation σ.

In general, the Metropolis-Hastings sampler suffers from the curse of dimensionality. As the dimension of the state space increases, the probability of rejection rises as every dimension provides an opportunity to reject a proposal state. The width of the proposal distribution should be comparable to the standard deviation of the target distribution in the most constrained direction (Brooks et al., 2011). When the target distribution is highly correlated, one is forced to choose "small" proposal distributions in order to maintain acceptable rejection rates. This comes at the cost of slow exploration of the state space in the less constrained directions, and therefore a high autocorrelation of samples in the chain. It renders the Metropolis-Hastings algorithm unsuitable for the problem at hand, in which samples from the space of composite spline control points for the configuration $C(\mathbf{x})$ have to be drawn. This space is high-dimensional and—depending on the the feature design—we expect large correlations among the spline control points. Velocity and acceleration features, for instance, prevent consecutive spline control points to move arbitrarily far apart.

5.7.2.2 The Hybrid Monte Carlo Sampler

The Hybrid Monte Carlo (HMC) algorithm by Duane et al. (1987), also known as Hamiltonian Monte Carlo, is more suitable for high-dimensional, correlated densities as it avoids the random walk behavior of the Metropolis-Hastings sampler through a guidance of the sampling process with gradient information of the target density. It replaces the uninformed proposal distribution in Metropolis-Hastings with a simulation of Hamiltonian dynamics. In this setting, the random variable $\mathbf{x}$ is interpreted as a position with the associated potential energy $U(\mathbf{x}) = -\log p(\mathbf{x}) + A$, where A

is an arbitrary constant usually chosen to cancel the normalizing constant of p. The space is extended by an auxiliary momentum variable $\mathbf{v}$ with associated kinetic energy $K(\mathbf{v}) = -\log q(\mathbf{v})$, yielding the Hamiltonian H of the system as the sum of potential and kinetic energy

$$H(\mathbf{x}, \mathbf{v}) = U(\mathbf{x}) + K(\mathbf{v}) \,. \tag{5.39}$$

The letter $\mathbf{v}$ is chosen to underline the interpretation as a velocity. A common choice for the distribution of the momentum $q(\mathbf{v})$ is the standard normal distribution, in which case the kinetic energy (neglecting the normalizing constant) becomes $K(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top\mathbf{v}$. More advanced approaches, as for instance presented in Zhang et al. (2011), use an anisotropic normal distribution where the covariance matrix is related to the Hessian of the target distribution at $\mathbf{x}$. The evolution of the joint configuration of position and momentum in phase space can then be determined through Hamilton's differential equations

$$\frac{\partial \mathbf{x}}{\partial t} = +\frac{\partial H}{\partial \mathbf{v}} = +\frac{\partial K}{\partial \mathbf{v}} \tag{5.40a}$$

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial U}{\partial \mathbf{x}} \,. \tag{5.40b}$$

In a software implementation, the differential equations are numerically integrated to yield a trajectory through phase space. This is typically done with the *leapfrog* integration scheme

$$\mathbf{v}\left(t + \tfrac{\epsilon}{2}\right) = \mathbf{v}(t) - \frac{\epsilon}{2}\,\frac{\partial U}{\partial \mathbf{x}}\,(\mathbf{x}(t)) \tag{5.41a}$$

$$\mathbf{x}\,(t + \epsilon) \;= \mathbf{x}(t) + \epsilon\,\frac{\partial K}{\partial \mathbf{v}}\left(\mathbf{v}\left(t + \tfrac{\epsilon}{2}\right)\right) \tag{5.41b}$$

$$\mathbf{v}\,(t + \epsilon) \;= \mathbf{v}\left(t + \tfrac{\epsilon}{2}\right) - \frac{\epsilon}{2}\,\frac{\partial U}{\partial \mathbf{x}}\,(\mathbf{x}\,(t + \epsilon))\;, \tag{5.41c}$$

that shows superior accuracy compared to Euler integration methods (Brooks et al., 2011). De facto, the choice of integration scheme in HMC is more delicate than pure accuracy considerations, but we refer the reader to Brooks et al., 2011 at this point for considerations regarding volume preservation in phase space. Equation 5.41 can easily be derived by applying finite differences on Equation 5.40. To advance the system from initial time t by a time step ϵ to $t + \epsilon$, first the momentum is advanced by half a time step. Afterwards the position is updated using the updated momentum from Equation 5.41a. Finally, the momentum is again simulated for half a step length, this time with the updated position from Equation 5.41b. This is repeated until a trajectory of desired length is obtained. In this case, Equation 5.41a and Equation 5.41c can be combined for intermediate steps.

The step size ϵ and the length of the trajectory represent tuning parameters. If the step size is chosen too large, the numerical simulation of the Hamiltonian dynamics may become unstable and diverge. If chosen properly, the numerical simulation is usually stable, independent of the length of the trajectory. If ϵ is selected too small, computation time is wasted to generate a trajectory of same length. In practice we implemented a heuristic tuning routine during the burn-in phase of the sampler, in which the step length is subsequently increased or decreased to find a large but stable value.

Due to the discretization of Hamilton's equations, the total energy of the system $H(\mathbf{x}, \mathbf{v})$ may slightly change. In the HMC algorithm, this is accounted for via a Metropolis update after each leapfrog simulation. The Metropolis acceptance criterion from Equations 5.37 and 5.38 is hereby replaced by the ratio of the joint probabilities of position and momentum at the end and the start of the trajectory

$$p\left(\tilde{\mathbf{x}}, \tilde{\mathbf{v}} \mid \mathbf{x}, \mathbf{v}\right) = \min\left\{1, \frac{p\left(\tilde{\mathbf{x}}, \tilde{\mathbf{v}}\right)}{p\left(\mathbf{x}, \mathbf{v}\right)}\right\} = \min\left\{1, e^{-H(\tilde{\mathbf{x}}, \tilde{\mathbf{v}}) + H(\mathbf{x}, \mathbf{v})}\right\} . \tag{5.42}$$

The simulation of the Hamiltonian dynamics alone does not yield an ergodic Markov chain in phase space, i.e. not every state is reachable from every other state, as the phase particle moves on a hyperplane in phase space. Therefore the HMC algorithm iteratively draws a random momentum from $q(\mathbf{v})$, simulates the Hamiltonian dynamics and finally applies a Metropolis update to correct for numerical errors introduced by the discrete-time simulation.

5.7.2.3 Parallel Tempering

Common to both MCMC methods discussed above is their problem to sample from multimodal distributions, i.e. distributions with multiple peaks separated by valleys of low or zero probability. In order to traverse the state space to reach another mode, the samplers first have to move towards regions of low probability. These moves are rejected with a high probability though. To address this problem, various approaches have been studied in the past. Among the most popular approaches is *parallel tempering*, first introduced by Swendsen et al. (1986) and later extended by Geyer (1991). It is often referred to as temperature replica exchange.

The idea of parallel tempering is to run multiple Markov chains in parallel, each one with a scaled version $p_T(\mathbf{x}) = p(\mathbf{x})^{1/T}$ of the target distribution. In analogy with physical systems, the parameter T may be interpreted as a temperature of the distribution. In the high temperature replicas of the chain, low probability regions are raised, enabling the chains to traverse these regions more easily. The information from the high temperature chains is propagated to the low temperature chains via swap move

proposals. At fixed or random iterations, swaps between temperature-adjacent chains k and $k+1$ are proposed, and accepted with probability

$$p((\mathbf{x}_k, T_{k+1}),(\mathbf{x}_{k+1}, T_k) \mid (\mathbf{x}_k, T_k), (\mathbf{x}_{k+1}, T_{k+1}))$$
$$= \min\left\{1, \frac{p_{T_{k+1}}(\mathbf{x}_k) \cdot p_{T_k}(\mathbf{x}_{k+1})}{p_{T_k}(\mathbf{x}_k) \cdot p_{T_{k+1}}(\mathbf{x}_{k+1})}\right\} \tag{5.43a}$$
$$= \min\left\{1, \exp\left(\left(\frac{1}{T_{k+1}} - \frac{1}{T_k}\right) \cdot \left(E(\mathbf{x}_{k+1}) - E(\mathbf{x}_k)\right)\right)\right\} . \tag{5.43b}$$

The fraction in Equation 5.43a hereby constitutes the ratio of the probabilities of the ensemble before and after the swap. With $p_T(\mathbf{x}) = e^{-1/T \cdot E(\mathbf{x})}$ one easily arrives at Equation 5.43b. The choice of the number of temperatures and their spacing is highly dependent on the landscape of the distribution. The highest temperature should be chosen that efficient traversal of the whole state space is possible. The spacing of the temperatures should be designed in a way that temperature adjacent distributions sufficiently "overlap" in order to yield reasonable acceptance rates of swap proposals. After simulating the ensemble of chains for a certain amount of time in which they exchange information, the states of the cold chain at temperature $T = 1.0$ constitute samples from the target distribution.

The combination of parallel tempering and HMC (Fukunishi et al., 2002) provides a powerful toolset to sample from high-dimensional, correlated, and multimodal distributions. In Figures 5.11–5.13, a comparison between regular HMC and Parallel Tempered Hybrid Monte Carlo (PTHMC) to sample from a multimodal distribution arising from a static obstacle is shown. While HMC gets stuck in one mode of

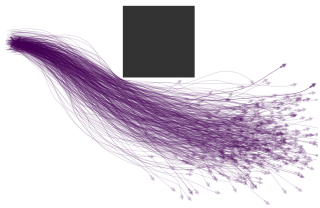


Figure 5.11: 2000 draws obtained with the Hybrid Monte Carlo (HMC) sampler from a distribution with a region of low probability induced by a static obstacle in the center (black obstacle). The sampler explores one mode well (passing obstacle on the right), yet is unable to reach the other mode (passing obstacle on the left). Both modes have equal probability due to the symmetric design of the configuration.

the distribution, PTHMC succeeds in sampling from both modes, illustrated in Figure 5.11 and Figure 5.12. At a temperature of approximately $T = 100$, the chains start

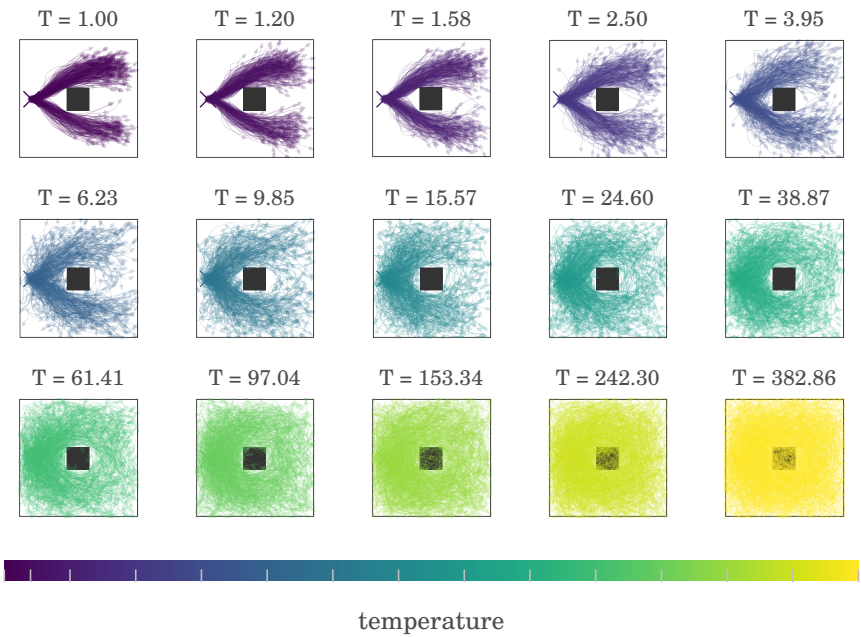


Figure 5.12: 2000 draws from the same distribution and configuration as in Figure 5.11 with a Parallel Tempered Hybrid Monte Carlo (PTHMC) sampler. The parallel tempering algorithm runs multiple Markov chains at different temperatures of the distribution. At high temperatures low probability regions are raised, allowing the sampler to better traverse these regions and reach other modes of the distribution. The information from higher temperature chains is propagated to the low temperature chains (ultimately to the chain at $T = 1.0$ which represents the target distribution) via swap moves. Colors encode the log-temperatures of the chains.

to traverse the the low-probability region induced by the static obstacle. Presumably long before that, the chains are able to traverse between the two modes by exploiting the trajectory's variable end position and move it in front of the obstacle from top to bottom and vice versa. The barriers here are rather in velocity space and much lower compared to the ones induced by static obstacles. However, PTHMC not always explored all modes equally in our experiments, apparent from the trace plot depicted in Figure 5.13—presumably due to a not perfectly tuned temperature ladder.

Other approaches to sample from multimodal distributions exist. In Laskey et al. (2003), the authors borrow ideas from genetic algorithms to exchange information between multiple Metropolis-Hastings samplers in order to improve on the proposal

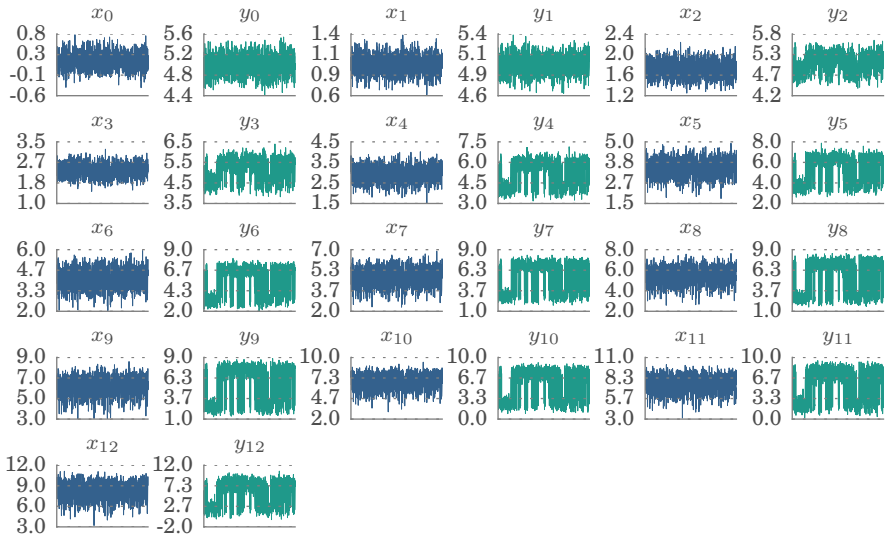


Figure 5.13: Trace plot of the thirteen 2D spline control points of the Parallel Tempered Hybrid Monte Carlo (PTHMC) sampler at temperature $T = 1.0$ for the setup given in Figure 5.12. The chain is mixing well in horizontal direction (x-axis) of the world with low autocorrelation. In vertical direction (y-axis), where the static obstacle creates two modes, the sampler successfully explores both modes of the distribution, yet mixing is not perfect.

distribution. Sminchisescu et al. (2011) build on the concept of *Darting Monte Carlo*, in which jump regions are defined around modes of the target distribution. Jumps between regions are proposed once the Markov chain enters one of the regions. Lan et al. (2014) propose a similar approach, in which a network of modes is established and a "wormhole" metric is used to shorten the distance between modes, allowing the sampler to tunnel to otherwise isolated modes. Modes are discovered and incorporated on-the-fly with local optimization schemes. These sampling procedures are especially attractive compared to parallel tempering as they eliminate the computational overhead of multiple parallel MCMC chains, but we have to leave the application of these to the problem at hand for future work.

5.7.3 Stochastic Optimization and Incremental Learning

Incremental learning is beneficial in long-term operation scenarios. In order to obtain the batch gradient descent solution for the maximum likelihood parameters $\boldsymbol{\theta}^\star$ from

Equation 5.28, all demonstrations $\mathcal{X}_D$ need to be stored, and processed in every iteration. Incremental learning procedures, such as stochastic gradient descent (SGD), enable online incorporation of new demonstrations. The idea is to use a (randomly) subsampled gradient of the full objective in each iteration to update the model via

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{\Lambda}_k \cdot \mathbf{g}_k \,, \tag{5.44}$$

where $\boldsymbol{\theta}_k$ represents the feature weights, $\boldsymbol{\Lambda}_k$ a learning rate matrix to control the step sizes, and $\mathbf{g}_k = \frac{\partial}{\partial \boldsymbol{\theta}^\top} \ell\left(\mathcal{X}_{B,k}; \boldsymbol{\theta}\right)$ the gradient of the loss function at a randomly selected subset $\mathcal{X}_{B,k} \subseteq \mathcal{X}_D$ of the training data, at the k-th update step respectively. When $\mathcal{X}_{B,k} = \mathcal{X}_D$ the original batch gradient descent formulation is attained. In the extreme case $\mathcal{X}_{B,k} = \{\mathbf{x}\}$ only a single training example is used. Hence each demonstration may be processed only once, and there is no need to keep observed trajectories in memory. Most importantly, SGD often converges much faster than batch gradient descent given a fixed amount of computation time, as batch optimization in many cases performs partially redundant computations for similar training examples (LeCun et al., 1998).

$\boldsymbol{\Lambda}_k$ is often assumed to be diagonal due to the main application of SGD for large-scale optimization problems with a huge number of parameters, where the explicit construction of the dense matrix exceeds memory limitations. The simplest choice for $\boldsymbol{\Lambda}_k$ is a constant learning rate for all parameters $\boldsymbol{\Lambda}_k = \lambda_0 \mathbf{I}$. With a constant learning rate however, SGD will eventually fluctuate around the minimum due to the noisy estimates of the gradient. Fluctuations can be suppressed if the learning rate is slowly decreased through an appropriate annealing schedule. In fact, convergence results for SGD usually demand decreasing learning rates with $\sum_k \lambda_k^2 < \infty$ and $\sum_k \lambda_k = \infty$ (Bottou, 2010).

SGD with common learning rates for all parameters has difficulties when parameters are correlated and the loss function exhibits elongated valleys. In that case, similar to steepest descent in the batch optimization case, the parameters will oscillate in the direction of the steepest descent and only make very slow progress in the shallow direction towards the minimum. Also, the ignorance of curvature information can lead to slow progress in flat directions of the parameter space. Various algorithms have been proposed to address these problems, such as AdaGrad (Adaptive Gradient) by Duchi et al. (2011), AdaDelta by Zeiler (2012) and Adam (Adaptive Moment Estimation) from Kingma et al. (2015). Duchi et al. (2011) derived the AdaGrad learning rate schedule

$$\boldsymbol{\Lambda}_k = \lambda_0 \left(\mathbf{G}_k + \epsilon \mathbf{I}\right)^{-1/2} \text{, with } \mathbf{G}_k = \left(\sum_{i=1}^{k} \mathbf{g}_i \mathbf{g}_i^\top\right) , \tag{5.45}$$

as well as a diagonal variant for large-scale problems. AdaDelta is derived from the diagonal variant of AdaGrad and addresses the practical problem of aggressively diminishing learning rates due to the accumulation of gradients in $\mathbf{G}$. Therefore, AdaDelta replaces the sum with an exponential moving average of squared gradients. Adam as well employs a diagonal version and corrects for the bias in the exponential moving average in AdaDelta due to initialization at zero. It proposes the update rule

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k \tag{5.46a}$$

$$\mathbf{v}_k = \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\left(\mathbf{g}_k \circ \mathbf{g}_k\right) \tag{5.46b}$$

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{1-\beta_1^k} \tag{5.46c}$$

$$\hat{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1-\beta_2^k} \tag{5.46d}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\lambda_0}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon} \circ \hat{\mathbf{m}}_k\,, \tag{5.46e}$$

where $\circ$ denotes the element-wise Schur product. The algorithm keeps an element-wise exponential moving average of both the gradient and the squared gradient, serving as an estimate of the gradient's first and second raw moment, respectively. The smoothing factors β_1 and β_2 used in the exponential moving averages, as well as the general learning rate λ_0 constitute tuning parameters. Equations 5.46c and 5.46d correct the bias induced by initialization of the estimates at zero. We refer the reader to the related work section in Kingma et al. (2015) for a brief comparison with the other SGD versions mentioned in this section. Adam (as well as AdaDelta) is especially attractive in optimization problems with non-stationary objectives as found in life-long learning settings as the learning rate can increase if the data changes.

5.7.4 Model Selection

The task of feature design is not trivial and one might introduce features with no or little correlation with the data. This increases the complexity of the model without improving the predictive qualities of it and raises the issue of model/feature selection. Additionally, overfitting to the training data can occur, in which the model explains the training dataset well, yet fails to generalize to unseen situations. In the context of learning maximum entropy distributions, overfitting may come in the form of fitting expected feature values too closely to inaccurate empirical expectations given a limited amount of observations.

A common way to deal with overfitting and model selection is to incorporate priors on the model parameters via regularization in the form of

$$\ell\left(\mathcal{X}_D;\boldsymbol{\theta}\right) = -\sum_{\mathbf{x}\in\mathcal{X}_D} \ln p\left(\mathbf{x} \mid \boldsymbol{\theta}, C(\mathbf{x})\right) + \beta \left\|\boldsymbol{\theta}\right\|_d , \tag{5.47}$$

where $\|\boldsymbol{\theta}\|_d$ is a suitable vector norm of the model parameters, and β a weight for the regularizer term. The most commonly used norms for regularization are L1 and L2. Ng (2004) showed that the L1-norm $\sum |\boldsymbol{\theta}_i|$ tends to produce sparse solutions in contrast to the L2-norm $\sum \boldsymbol{\theta}_i^2$. Sparsity in this context means, that irrelevant entries of the feature weight vector will be driven exactly to zero and hence these features may be removed from the model. This is equivalent to eliminating the moment constraint in the maximum entropy derivation imposed by the feature. With that in mind we can see how L1 regularization can serve as a means to implement Occam's razor to automatically select simpler models (i.e. with fewer features) if possible. Choosing the regularization weight β is non-trivial and often a trade-off between bias introduced by the penalization of large parameter values and overfitting. In the context of maximum entropy models, L1 regularization can be interpreted as a relaxation of equality constraints to box constraints (Kazama et al., 2003), i.e. an upper bound on the difference between feature expectations.

Within stochastic optimization, L1 regularization has to be implemented carefully, as noise in the stochastic gradient usually prevents feature weights to be driven to zero exactly. Tsuruoka et al. (2009) proposed an adapted L1 regularization procedure applicable for stochastic optimization. The method aggregates the cumulative penalty a certain weight has received, and by that smoothes the effects of noisy gradients.

5.8 Online Motion Planning

The planning framework hinges on the assumption that all agents plan their trajectories by maximizing the probability of the composite trajectory from Equation 5.14, given their current observed configuration C. This is equivalent to minimizing the negative log-density of the maximum-entropy distribution. The optimization problem for motion planning yields a minimization of the weighted sum of feature functions

$$\begin{aligned}
\mathbf{x}^\star &= \operatorname*{argmax}_{\mathbf{x}, C(\mathbf{x})=C} p\left(\mathbf{x} \middle| \boldsymbol{\theta}^\star, C(\mathbf{x})\right) \\
&= \operatorname*{argmin}_{\mathbf{x}, C(\mathbf{x})=C} -\ln p\left(\mathbf{x} \middle| \boldsymbol{\theta}^\star, C(\mathbf{x})\right) \\
&= \operatorname*{argmin}_{\mathbf{x}, C(\mathbf{x})=C} \boldsymbol{\theta}^{\star\top} \cdot \mathbf{f}(\mathbf{x}) ,
\end{aligned} \tag{5.48}$$

since the normalizing constant $z(\boldsymbol{\theta}^\star, C(\mathbf{x}))$ of the distribution may be neglected as it only depends on the maximum likelihood parameter estimates of the model $\boldsymbol{\theta}^\star$, and the configuration $C(\mathbf{x}) = C$. The weighted sum of feature functions can also be interpreted as weighted sum of cost terms. This interestingly establishes a link to former motion planning techniques such as the dynamic window approach from Fox et al. (1997), where multiobjective optimization criteria are encoded in the same fashion. With $\mathbf{x}^\star$ at hand, the part of the composite trajectory corresponding to the ego robot can be used for its control—or in a multi-robot scenario one might use the solution for multiple controllable agents.

The composite trajectory is optimized in the time interval $[t_0, t_H]$. In practice, we choose t_0 with a certain duration in the past, in which we collected observations of the agents' positions and velocities, as well as observations of the static environment. The history primarily provides additional information in order to constrain the derivatives of the spline(s) in the future. Figure 5.14 sketches the operation principle of the framework for online motion planning.

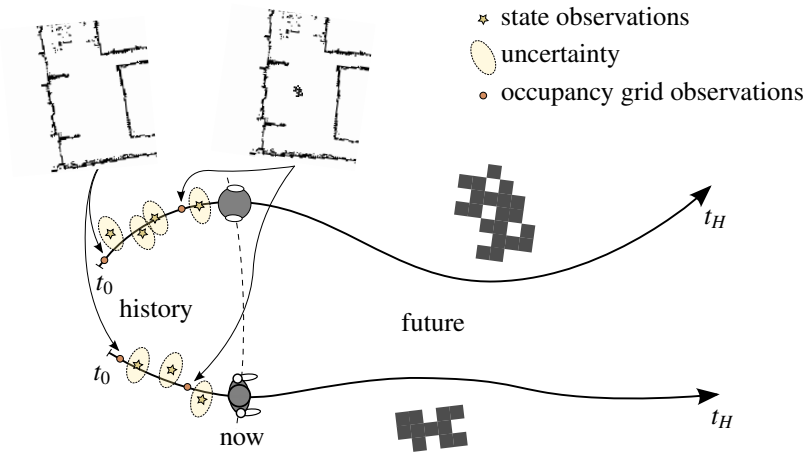


Figure 5.14: Sketch of the operation principle of the maximum entropy framework for online motion planning. The most probable composite trajectory $\mathbf{x}^\star$ is found over the time interval $[t_0, t_H]$, where t_0 is chosen with a certain history duration in the past. In the history interval, both position and velocity observations with Gaussian measurement noise, and occupancy grid measurements are incorporated.

For the optimization problem given in Equation 5.48, it is difficult to derive from a theoretical point of view which nonlinear optimization routines will perform best. Therefore, we compared the performance of the Rprop (Resilient Back-Propagation)

algorithm (Riedmiller and Braun, 1992), advocated for online planning by Kuderer, Kretzschmar, Sprunk, et al. (2012), with the well-known BFGS (Broyden-Fletcher-Goldfarb-Shanno) and CG (Conjugate Gradient) method. These optimization routines are all Hessian-free, i.e. they solely require objective and gradient information. They differ, however, in the way gradient information is incorporated. The Rprop optimization scheme disregards gradient norm information and solely uses its direction. Step lengths for parameter updates are computed with a heuristic schedule per parameter, increasing the step length in case the gradient points in the same direction as in the last step, and decreasing step length in the opposite case. This especially yields robustness in the presence of noisy gradients or rough objective landscapes—both not the case for the online optimization problem at hand where analytical gradients are available, and a smooth objective function is guaranteed through C^1-continuous feature design. Four variants of the Rprop method proposed by Igel et al. (2000) exist, from which we evaluate the iRprop$^+$ version with an advanced back-tracking scheme based on objective values. BFGS is a Quasi-Newton method, which approximates the inverse Hessian of the objective function $\mathbf{f} = \boldsymbol{\theta}^{\star\top} \cdot \mathbf{f}(\mathbf{x})$ from gradients of subsequent iterations. An update is performed via a Newton step $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \hat{\mathbf{B}}_k \nabla \mathbf{f}(\mathbf{x}_k)$, where $\hat{\mathbf{B}}_k$ represents the approximate inverse Hessian, and the step length α_k is obtained via a line search procedure $\alpha_k = \operatorname{argmin}_s \left\{\mathbf{f}(\mathbf{x}_k + s\hat{\mathbf{B}}_k \nabla \mathbf{f}(\mathbf{x}_k))\right\}$. The Hessian matrix encodes (multidimensional) curvature information and provides valuable information during optimization about the correlation of the design variables, leading to improved convergence speeds especially when the objective is close to quadratic. While from a theoretical point of view exact Hessian information is desirable, approximating the Hessian matrix from subsequent gradients is sometimes appealing as (a) the computation of the inverse can be implemented more efficiently for high-dimensional problems, and (b) additional computations for the second-order derivatives are avoided. The (nonlinear) CG method is a first order method, in which an update to the design variables of the problem is acquired via $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$. The conjugate direction of the update, $\mathbf{s}_k$, is computed by combining gradients of subsequent iterations $\mathbf{s}_k = -\nabla \mathbf{f}(\mathbf{x}_k) + \beta_k \mathbf{s}_{k-1}$. The step length, α_k, is again determined via line search along the direction of the update. For further information the reader is referred to Nocedal et al. (2006).

Figure 5.15 shows a comparison between the aforementioned optimization strategies for randomly created artificial online motion planning problems. As computation time is highly dependent on the implementation, we rather evaluated the progress of the objective, i.e. the negative log density, with respect to the number of gradient and objective evaluations. Hereby, the negative log density is normalized via

$$\frac{-\log p(\mathbf{x}) + \log p(\mathbf{x}^\star)}{-\log p(\mathbf{x}_0) + \log p(\mathbf{x}^\star)},$$

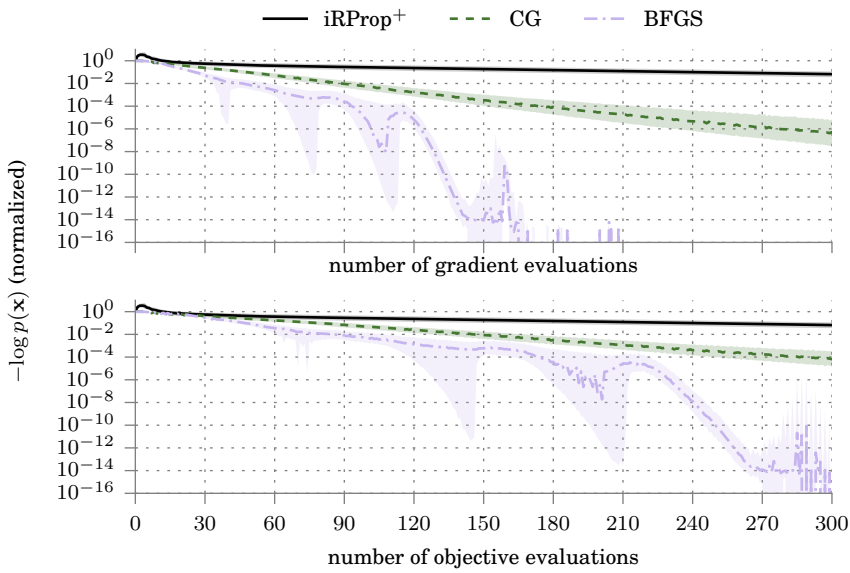


Figure 5.15: Comparison of gradient-based optimization strategies for randomly created artificial online motion planning problems. The graph depicts the 25 %, 50 % and 75 % percentiles of the normalized negative log likelihood of 1000 random configurations with respect to number of gradient evaluations (*top*) and number of objective evaluations (*bottom*). In our implementation, gradient evaluation is approximately three times slower than objective evaluation. The negligence of gradient norm information in the Rprop optimization routine leads to slow convergence in comparison to the BFGS (Broyden-Fletcher-Goldfarb-Shanno) and CG (Conjugate Gradient) optimization methods, exploiting the full gradient information.

where the minimum negative log density in each trial, $\log p(\mathbf{x}^\star)$, is taken as the minimum value found by any of the optimizers. The normalized negative log density values are averaged over 1000 randomized trials with 1 to 5 agents. Trajectories are initialized with a constant velocity model. The results show that BFGS compares favorably to CG and Rprop in terms of convergence speed. Especially Rprop exhibits painfully slow convergence, rendering it unsuitable for online optimization.

5.9 Implementation Details

In our implementation, the integrals involved in the feature definitions are computed via simple Euler integration as in $\int_{t_0}^{t_H} f(\mathbf{x}(t))\,\mathrm{d}t \approx \sum_{t=t_0}^{t_H} f(\mathbf{x}(t))\,\Delta t$. We typically

choose the planning horizon $t_H - t_0$ between 7 s and 10 s, and a numeric integration time Δt of 0.1 s. The cubic B-splines employed use a uniform knot spacing of 1 s.

Our implementation uses a custom B-spline library, most importantly providing an expression and auto-differentiation framework. This allows for convenient design of new features as the derivative computation is automatically handled by the library. On the downside, the auto-differentiation framework is not the ideal choice in terms of performance. We optimized for speed by caching the results of some identical sub-expressions of features in the expression tree, and minimized dynamic memory allocations to a large extent. Additionally, gradient computation on the CPU (central processing unit) is multithreaded, which benefits both the gradient-based BFGS optimizer for online planning as well as the leapfrog integration scheme used as part of the PTHMC sampler for model training.

Despite these optimization efforts, we could achieve real-time capability only for a fairly small number of agents. The computational complexity of the approach is quadratic in the number of agents, due to the $n(n-1)/2$ agent pairs in the interaction features. The computation time for motion- and observation features is linear in the number of agents. Geodesic distance computation with Dijkstra's algorithm, and 2D signed distance transform computation on a 200×200 grid, amount to 12 ms and 1 ms, respectively, measured on an Intel Core i7@2.50GHz. Those have to be performed once upon a new observation of the static environment, which typically arrives at least once per planning cycle. In our implementation, one (multithreaded) gradient evaluation for all motion- and observation features of a single agent takes around 0.9 ms. Interaction features for a single agent pair add approximately 0.25 ms. Given that around 80 iterations are necessary for sufficient convergence of the BFGS optimizer (cf. Section 5.8), and accounting for some additional computation time for linesearch and approximate Hessian computation, we could only consider up to two additional agents besides the ego agent during online planning. For this setting, gradient computation alone amounts to $80 \cdot (3 \cdot 0.9\,\text{ms} + 3 \cdot 0.25\,\text{ms}) = 276\,\text{ms}$. Compelled by this computational constraints, we ignore all agents located behind the robot, and select the two closest other agents in front.

During model training with the PTHMC sampler, we use 500 leapfrog steps, a temperature ladder with 15 temperatures, and 200 samples to approximate the feature expectations. The computation time to obtain a single leapfrog trajectory amounts to approximately 1 s for a configuration with 2 agents observed over a time horizon of 10 s. This results in a computation time of 50 min for one iteration within stochastic gradient descent optimization. A similar computation time was reported in Kretzschmar et al. (2013) for batch optimization, yet without specifying the num-

ber of demonstrations, so we assume that the computation time has to be multiplied by the number of demonstrations in their batch approach. Non-tempered sampling consequently yields a per-iteration computation time of 3.3 min.

Besides consequently eliminating redundant computations, especially a better numerical integration scheme and dynamic adaptation of the integration time constant offers potential to reduce computation time. Together with general code optimization, we believe that a speedup of factor 5 to 10 is realistic with a pure CPU implementation, which would enable a larger number of agents to be considered during online planning, as well as very reasonable training times. This is, however, an educated guess only that we cannot verify at the moment.

5.10 Experimental Results

In this section, results of the maximum entropy navigation framework are presented, both for the training stage and during online operation. They complement the already presented results, incorporated into the sections above. Most importantly, we determine whether the assumption of cooperation indeed facilitates motion planning under observation noise in Section 5.10.3.2. Due to legal issues preventing the usage of the V-Charge (Automated Valet Parking and Charging for e-Mobility) self-driving vehicle prototype on public ground at the given time, and a lack of representative pedestrian activities at the private test ground, results are obtained with the EUROPA2 (European Robotic Pedestrian Assistant 2)[1] platform described in the following Section 5.10.1.

5.10.1 Experimental Platform Setup

The cooperative maximum entropy framework was implemented and extensively tested on the mobile robotic platform of the European project EUROPA2, depicted in Figure 5.16. The project targets the development of a pedestrian assistant robot for urban environments. This section briefly describes the robotic platform in Section 5.10.1.1, and the object detection, classification, and tracking pipeline employed in Section 5.10.1.2.

5.10.1.1 Robotic Platform

The robot is equipped with a differential drive locomotion system, and has a diameter of 0.8 m, slightly larger than a human. It is controlled through translational and rotational velocity commands. The maximum entropy navigation software is running on a standard laptop (Intel Core i7@2.80 GHz) mounted in the robot body.

[1] `http://europa2.informatik.uni-freiburg.de/`

Figure 5.16: The EUROPA2 platform in its testing environment, surrounded by various pedestrians interacting with the robot. Dynamic objects are detected from the robot's laser range finders. The 3D laser range finder mounted on the robot's head is additionally used for localization within a pre-built map.

A Velodyne HDL-32E LiDAR sensor mounted on the robot's head provides accurate 3D distance measurements and is complemented by two Sick LMS151 2D laser scanners mounted at the robot's front and back at a height of approximately 55 cm in order to cover blind spots in the Velodyne's field of view (FOV). Additionally, a Hokuyo UTM-30LX is mounted on a pan-tilt unit in front (visible for instance in Figure 5.17) to detect the presence or disappearance of small objects in very close proximity. A PointGrey Bumblebee2 stereo camera delivers color images with a horizontal FOV of 100°, however only monocular information is used.

5.10.1.2 Pedestrian Detection, Classification, and Tracking

Both model training and online motion planning require the detection of other agents in the world. During online motion planning, knowledge of the presence of other agents is crucial in order to treat them as entities the robot can interact with, as opposed to static infrastructure. Optimally, knowledge about the type of agent (classification) encountered—such as pedestrian, bicyclist or vehicle—is extracted to apply distinct motion- and interaction models. During model training, additional knowl-

edge about the unique identity of an agent over time is required, such that integral trajectory features can be computed. The latter requirement necessitates a tracking procedure in order to uniquely associate agents in the world over time. As we employ a small history of the agents' states also for online motion planning as detailed in Section 5.8, tracking is likewise required for this stage, but not absolutely mandatory.

Our object detection, classification, and tracking pipeline developed combines information from the robot's 3D Velodyne laser range finder and its camera. With their accurate distance measurements, laser range finders are perfectly suited to extract spatial information. In contrast, cameras provide much richer information concerning the semantics of objects in the world. Therefore, we perform object detection in the laser range finder's 3D point cloud, and use the camera exclusively for classification of objects found. This additionally allows us to leverage the 360° FOV of the laser scanner to track objects beyond the limited FOV of the camera employed on our platform. Figure 5.17 visualizes the internals of the pipeline, that are explained in the following.

First, object hypotheses from the laser data are computed by clustering adjacent cells of a discretized, binary 2D representation of the 3D point cloud. The 2D representation is obtained via a projection of the laser data onto a 2D grid, counting the number of laser points that fall into each grid cell, followed by a binary threshold operation. Clusters are extracted with a floodfill algorithm, finding spatially connected components in the grid. The 2D projection proved to be necessary to meet real-time computation constraints, that could not be satisfied with full 3D Euclidean clustering. Next, the object hypotheses are run through a series of handcrafted tests—for instance regarding the size of the 2D footprint, height and width/length-ratio—to filter out clusters certainly not originating from pedestrians in the world. The centroids of remaining clusters then form the input to a standard object tracking loop, comprised of track prediction, gating, measurement-to-track association, track update and track maintenance.

Prediction and track update are performed by means of a standard Kalman filter (Kalman, 1960), maintaining a multivariate Gaussian distribution over the 2D position and linear velocity of each registered track. The estimate of objects' velocities in conjunction with the Kalman filter prediction step greatly improves the robustness of measurement-to-track association in case of temporary observation outages due to occlusions. Measurement-to-track association is performed through a global

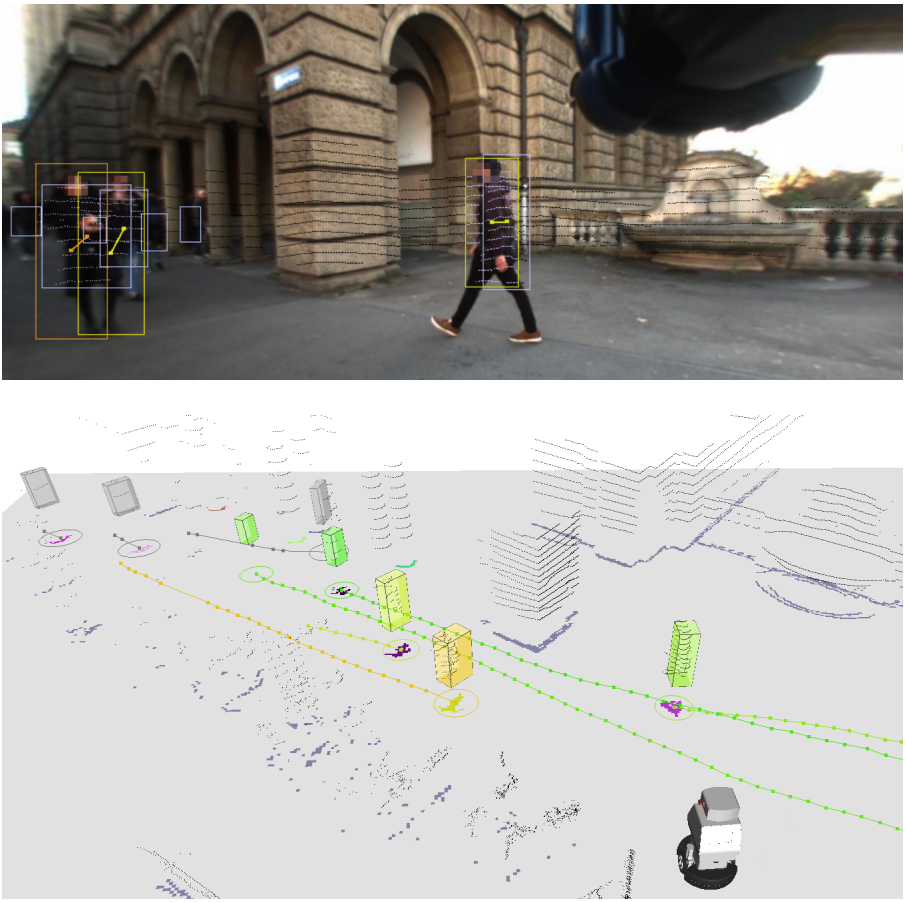


Figure 5.17: Object detection, classification, and tracking pipeline. *Top:* Camera-based pedestrian classifications (colored red to green by increasing classification score), and laser clusters (light blue) projected to the image plane. Colored lines indicate associations between pedestrian classifications and laser clusters. *Bottom:* 3D laser point cloud (black dots), 2D grid representation (gray / blue cells indicate free / occupied space) of laser data, grid clusters (colored grid cells) and 3D object tracks. Object tracks classified as pedestrians are colored by classification score of the aggregate channel features (ACF) classifier employed. Confirmed, but unclassified tracks are visualized in gray. Ellipses denote 95 % confidence intervals. The pipeline keeps track of object classifications beyond the camera's FOV and in case of occlusions (leftmost two pedestrians not classified in current image).

nearest neighbor (GNN) approach, determining the optimal measurement-to-track assignment matrix $\mathbf{A}^\star$ with elements $a_{ij} \in \{0, 1\}$ via

$$\mathbf{A}^\star = \underset{\mathbf{A}}{\operatorname{argmin}} \sum_{a_{ij} \in \mathbf{A}} a_{ij} \cdot d_{ij}^2$$
$$\text{subject to} \sum_i a_{ij} = 1 \text{ and } \sum_j a_{ij} = 1\,,$$

where d_{ij} denotes the Mahalanobis distance between measurement i and track j. The two constraints ensure that each observation is uniquely assigned to either an existing or new track. Measurements outside the validation gate of any track, determined from the cumulative inverse χ^2 distribution at significance level α, instantiate new tentative tracks. The validation gate can be understood as the acceptance region, such that $1-\alpha$ true measurements are rejected. The optimal assignment can be found efficiently in polynomial time through the Kuhn-Munkres algorithm (Munkres, 1957).

After updating tracks with their associated measurements, track maintenance takes care of updating the track status from tentative to confirmed if a track proves to be stable, and deleting tracks for objects not measured for a certain duration.

In a final step, confirmed object tracks are associated with pedestrian classifications obtained through an image-based aggregate channel features (ACF) classifier, provided by KU Leuven in the context of the EUROPA2 project. In order to establish associations between the pedestrian classifications, given as 2D bounding boxes in the image plane, and objects in the world, the 3D bounding boxes of objects extracted from the laser point cloud are first projected into the image. With a custom dissimilarity metric between the 2D bounding boxes of the camera classifications, B_C, and the laser objects, B_L, in the image plane,

$$s = 1 - \min\left(\frac{V(B_C \cap B_L)}{V(B_C)}, \frac{V(B_C \cap B_L)}{V(B_L)}\right),$$

where V denotes the volume/size operator, the optimal assignment is again determined through GNN. The dissimilarity is zero in case the two bounding boxes are identical, and highest in case no overlap is present. Otherwise, the metric captures the relative overlap with respect to the larger bounding box.

5.10.2 Model Training

In this section, the training stage of the maximum entropy model is discussed. During model training, the maximum likelihood estimate of the feature weights is obtained by minimization of the loss function from Equation 5.27. The calculation of the

gradient of the loss function with respect to the feature weights in Equation 5.29 involves the computation of the expected feature values for all configurations in the N demonstrations. Obtaining the gradient for a single point $\boldsymbol{\theta}$ in parameter space thus requires the computation of N sample approximations of the expected feature values. Batch optimization quickly becomes infeasible for training datasets with even moderate size regarding computation time, as all demonstrations are used to compute the gradient for a single parameter update. Therefore, the focus in this chapter is on stochastic optimization, that uses only a subset of the training data—or in the extreme case a single demonstration—in each iteration as an approximation of the true gradient.

In the following two sections, we evaluate the training stage (a) with synthetic demonstrations created from a known maximum entropy distribution in Section 5.10.2.1, and (b) real-world training examples in Section 5.10.2.2.

5.10.2.1 Synthetic Demonstrations

By generating synthetic demonstrations from a known ground-truth distribution, $p_{\boldsymbol{\theta}^\star}$, we are able to derive additional information to facilitate monitoring the progress of the learning stage. With synthetic demonstrations, also comparison of stochastic optimization with a batch gradient descent becomes feasible as it allows to generate demonstrations with a unique configuration $C(\mathbf{x}) = C \ \forall \mathbf{x} \in \mathcal{X}_D$ and observations. A unique configuration is usually not given for real-world data, as for instance the agent count, time intervals of agents and occupancy grid observations vary across demonstrations. Without unique configurations in a large demonstration set, required in order to faithfully represent the ground-truth distribution, batch optimization is computationally infeasible. In the case of a single unique configuration, batch optimization needs to evaluate the feature expectations $\mathrm{E}_{p(\cdot|\boldsymbol{\theta}_i,C)}[\mathbf{f}]$ only once per iteration. The gradient from Equation 5.31 then becomes

$$\frac{\partial}{\partial \boldsymbol{\theta}^\top} \ell\left(\mathcal{X}_D; \boldsymbol{\theta}\right) = \left(\sum_{\mathbf{x} \in \mathcal{X}_D} \mathbf{f}(\mathbf{x})\right) - |\mathcal{X}_D| \cdot \mathrm{E}_{p(\cdot|\boldsymbol{\theta}_i,C)}[\mathbf{f}] \ .$$

If the loss function, and with that the gradient, is divided by the number of demonstrations $|\mathcal{X}_D|$ which does not change the minimizer, the gradient is identical to the difference of the feature expectations of the empirical distribution and the model.

In order to monitor the progress during model training in this section we use:

(a) A sample estimate of the Kullback-Leibler (KL) divergence $\hat{D}_{\mathrm{KL}}(p_{\boldsymbol{\theta}^\star} \parallel p_{\boldsymbol{\theta}})$ as a measure of the distance of the estimated distribution from the true one. Obtaining a reliable estimate of the KL divergence given in Equation 5.10 is far from

being trivial, as the computation involves an integration over the space of composite trajectories, as well as the two unknown normalizing constants of the densities. We utilized the divergence estimator from Wang et al. (2009), which uses k-NN density estimates obtained from random independent and identically distributed (i.i.d.) draws of the two distributions. The estimator is designed to yield more accurate results for high-dimensional spaces with a limited number of samples, however only yields an asymptotically unbiased estimator. In our experiments, the estimates appeared highly unreliable when computed over the high-dimensional space of spline control vertices with a rather low sample size. We instead resort to computing the KL divergence over the feature values, effectively comparing the similarity of the models in terms of the feature distributions they induce.

(b) The parameter-wise differences of the feature expectations of the ground-truth distribution and the model. The entries are proportional to the true (full batch) gradient of the loss function. This metric enables us to compare the quality of the model in each iteration without the noise induced by the subsampling procedure of the gradient in SGD.

(c) The parameter-wise differences of the true and estimated feature weights. In this context, the magnitude of the difference must not be taken as a measure how close the estimated model is to the true one. Especially correlated features introduce shallow directions in parameter space, and hence feature weights along these directions yield very similar objective values.

Unimodal Example First, a fairly simple problem with a single agent in the absence of static obstacles and goal information is examined. Consequently, the distribution exhibits a single mode only and effects arising from difficulties when sampling from a multimodal density are circumvented. A total of 2000 demonstrations are sampled from a ground-truth maximum entropy distribution with all feature weights set to arbitrary values. In the optimizers, the feature weights are intentionally initialized far away from the true weights to determine the capability of the training algorithms to cope with bad initial estimates. In Figure 5.18, the comparison between several SGD optimizers, and batch gradient descent with Rprop is shown. The optimizer parameters employed are summarized in Table 5.1. Please refer to Section 5.7.3 and Section 5.8 for a brief description of the SGD optimization strategies and Rprop, respectively. All SGD optimization strategies converge within a fraction of epochs (full passes through the training set) with respect to all aforementioned performance metrics. Rprop reveals a certain problem under the default parameters given in Riedmiller and Braun (1992) and Igel et al. (2000) for the increase and decrease factor of the parameter-wise learning rates, η^+ and η^-. As soon as the noise level in the

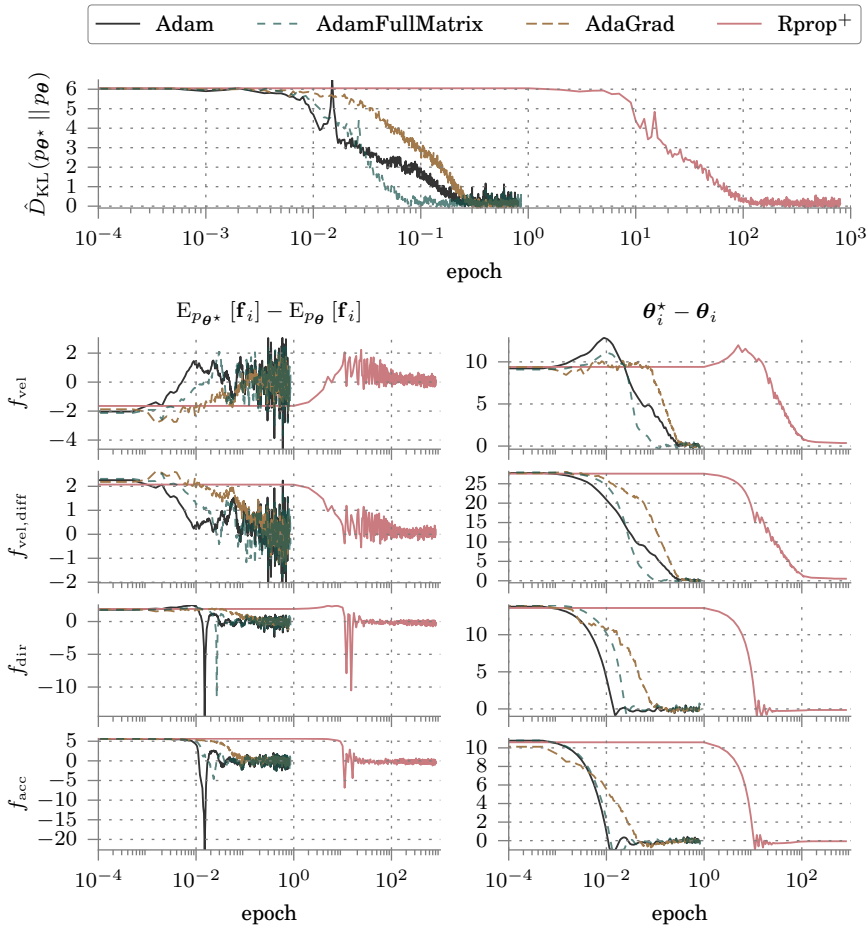


Figure 5.18: Comparison of stochastic gradient descent (Adam with diagonal and AdamFullMatrix with dense squared gradients, AdaGrad using full matrix variant) and batch (Rprop) optimization strategies for model training. An epoch of 1.0 corresponds to one full SGD pass through the 2000 demonstrations sampled from $p_{\boldsymbol{\theta}^\star}$, rendering the ordinate proportional to the computation time. *Top:* Kullback-Leibler divergence (relative entropy) between the estimated distribution $p_{\boldsymbol{\theta}}$ and the ground-truth distribution. The relative entropy is computed with respect to the distribution over feature values induced by the model parameters, where small relative entropy corresponds to similar distributions. *Bottom left:* Difference between empirical and estimated feature expectations. This metric is closely related to the gradient of the model with respect to the feature weights. *Bottom right:* Difference between true and estimated feature weights.

Adam/ AdamFullMatrix		
λ_k	$\max(1.0 - 0.009 \cdot k, 0.1)$	general learning rate, reduced from 1.0 to 0.1 in between iteration (k) 0 to 100
β_1	0.95	smoothing factor for gradient / momentum
β_2	0.999	smoothing factor for squared gradient
ϵ	1e−8	fuzz factor to ensure positive definiteness of squared gradients
AdaGrad (full matrix variant)		
λ_k	1.0	constant general learning rate
ϵ	1e−6	fuzz factor to ensure positive definiteness of squared gradient matrix
Rprop$^+$		
λ_0	0.4	initial learning rate
$\lambda_{\min}$	1e−20	minimum learning rate
$\lambda_{\max}$	3.0	maximum learning rate
η^-	0.5	learning rate decrease factor
η^+	1.2	learning rate increase factor

Table 5.1: Optimizer parameters for model training employed in the training example with unimodal synthetic demonstrations depicted in Figure 5.18.

computation of the feature expectations causes the gradient to frequently switch its sign, the learning rate is drastically decreased which lets Rprop fail to exactly recover the ground-truth parameters for f_{vel} and $f_{\text{vel,diff}}$ in the example. In this respect, SGD may be even better suitable due to the inherent design to cope with a noisy gradient. AdamFullMatrix denotes a homebrew Adam variant of ours, that we want to motivate in the following.

In both Adam and Rprop optimization strategies the two feature weights f_{vel} and $f_{\text{vel,diff}}$ converge significantly slower than the remaining features (note the logarithmic scale on the abscissa of the graph). This is caused by a strong negative correlation of these two features in the example at hand, visualized through the feature cross-correlation matrix computed from the 2000 demonstrations in Figure 5.19. We recall at this point that according to Equation 5.33 the feature covariance matrix is proportional to the Hessian of the loss function. The cross-correlation matrix depicted is thus a sample approximation of the Hessian at the optimum (normalized by the standard deviations). As a result of the strong negative correlation, only the difference between the two feature weights is well observable. Both Adam (and most other

Figure 5.19: Sample-approximation of the correlation matrix of motion features from demonstrations. Especially f_{vel} and $f_{\text{vel,diff}}$ exhibit a large negative correlation, resulting in a bad observability of the scale of these features. Correlations may slow down convergence in optimization strategies with parameter-wise learning rates, as parameter updates have negative side-effects on other dimensions.

SGD optimization techniques) and Rprop employ per-parameter learning rates without considering cross-correlations between parameters. This leads to the problem that an update of a feature weight affects other dimensions as well. While a decorrelation of the feature space cannot eliminate the problem of badly observable directions, it may still improve convergence speed. To the best of our knowledge, the full matrix version of AdaGrad and stochastic Quasi-Newton methods are the only SGD methods able to take into account cross-correlations between parameters. Stochastic Quasi-Newton methods, such as presented in Mokhtari et al. (2014), usually double the number of gradient evaluations, which is inappropriate in our case due to computational demands. AdaGrad is rather unattractive for life-long learning scenarios due to its monotonically decreasing learning rate. Additionally, we found convergence of the full matrix version of AdaGrad slow in the above experiment (cf. Figure 5.18). Therefore, we modified Adam to compute an exponential moving average over the full matrix of squared gradients, instead of the diagonal terms only. The results are included in Figure 5.18 under the label AdamFullMatrix and underline the usefulness of considering cross-correlations between parameters. Please note that we derived AdamFullMatrix in a late stage; results obtained in the remainder of this document employ the diagonal Adam version.

In summary, this section gave experimental validation of the applicability of SGD for the problem at hand. We highlighted the benefit of approximate full Hessian informa-

tion to cope with correlations between features. In contrast to large-scale optimization problems as found in deep neural network training, the explicit construction of the full Hessian matrix does not pose memory issues for the fairly small feature space in our setting.

Multimodal Example In a second experiment, a more complex configuration depicted in Figure 5.21 is constructed, comprised of two agents and a wall-like static obstacle, giving rise to multiple modes in the distribution over the composite agent trajectory. As one agent (green) clearly has two options of passing the wall on the left or right side, at least two modes are present. Other options, such as coming to a standstill in front of the wall, have insignificant probability given the artificial feature weights we chose. In case the agent passes on the left, the other agent takes a bigger detour to the top in order to avoid close distances. 500 samples from the ground-truth distribution are presented to the Adam optimizer, and separate motion feature sets for both agents are assumed. In order to speed up the training process, feature expectations are computed via the standard HMC sampler in the beginning, whereas parallel tempering is activated later at a predefined iteration count. Figure 5.20 depicts the same performance metrics used in the previous experiment. The epoch at which parallel tempering is activated is indicated with a vertical line. In addition, sample approximations of the estimated density for six selected snapshots during optimization are shown in Figure 5.21 for visual inspection. One can observe from the difference of the true and estimated feature weights, that standard HMC moves the feature weight vector already into a good region. The peaks in the KL divergence in Figure 5.20, and the sample approximations in Figures 5.21(c)–5.21(e) however indicate, that HMC fails in equally sampling from different modes, resulting in biased estimates of the feature expectations. At the epoch right before the switch to parallel tempering, the feature weights have settled with HMC. Activating parallel tempering further improves the estimates and recovers the true feature weights up to small differences induced by the finite sample approximation of the ground-truth distribution.

5.10.2.2 Real-World Demonstrations

After having demonstrated the capability of the training stage to recover the true feature weights in a setup with synthetic ground-truth data in the previous section, we now present training results with real-world data collected with the mobile robotic platform described in Section 5.10.1. The goal is to show—in conjunction with the results on online planning in Section 5.10.3—autarkic robot training and planning using onboard sensors only. This paves the way for self-contained autonomous agents

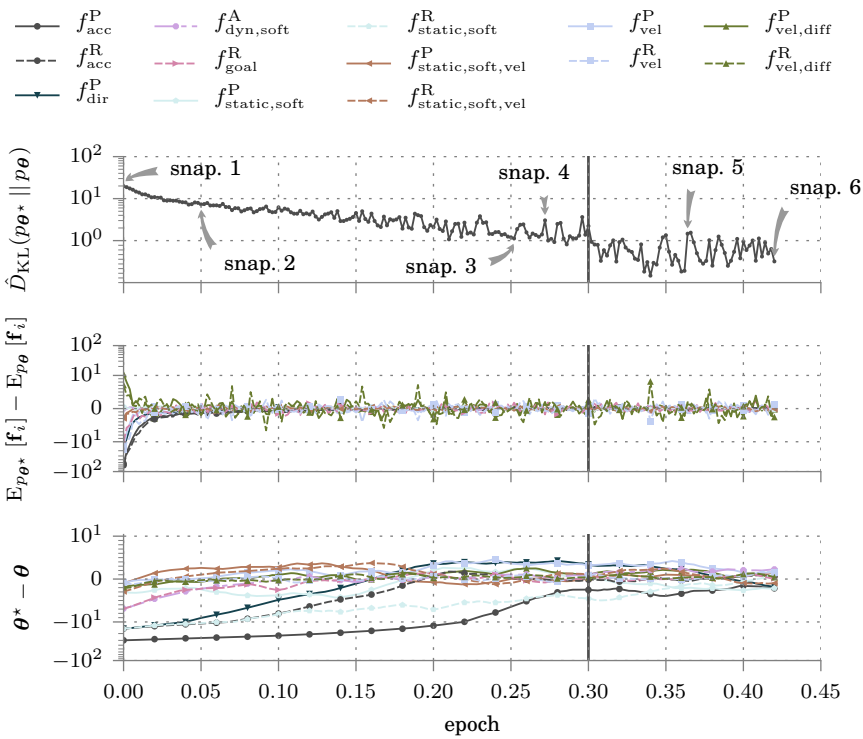
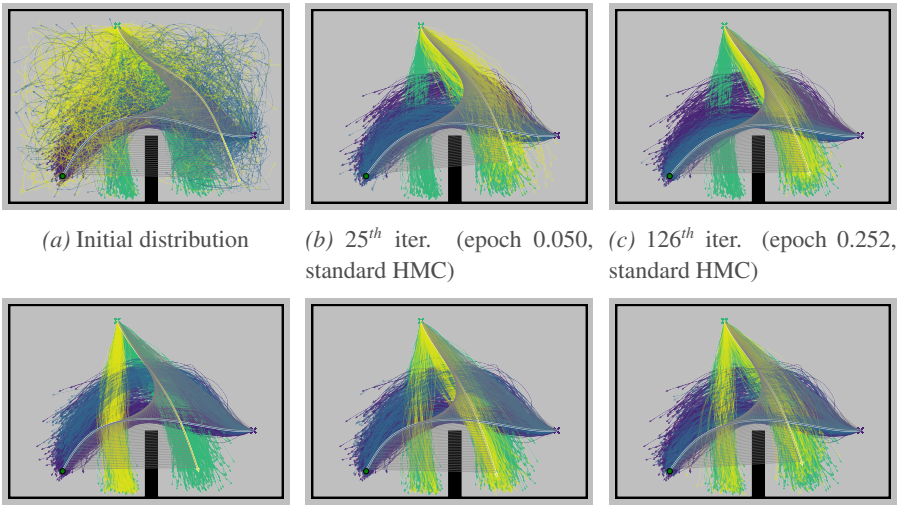


Figure 5.20: Progress during model training with demonstrations sampled from a multimodal ground-truth distribution $p_{\theta^\star}$ with two agents and a static obstacle. Superscripts in the legend denote the agent type: ^{R} = robot, ^{P} = pedestrian, ^{A} = pedestrian / robot interaction. An epoch of 1.0 corresponds to one full SGD pass through the 500 demonstrations sampled from $p_{\theta^\star}$. The training stage uses standard Hybrid Monte Carlo sampling in the beginning; parallel tempering is activated at a predefined iteration marked with a vertical line. A sample-based visualization is given in Figure 5.21 for six instances during the progress, marked as *snap. x* in the uppermost graph. Note that the ordinates of all graphs employ a symmetric logarithmic scale, in which certain ranges around zero use a linear scaling (see minor ticks). *Top:* Kullback-Leibler divergence (relative entropy) between the estimated distribution p_θ and the ground-truth distribution. A relative entropy of zero corresponds to identical distributions. *Middle:* Difference between empirical and estimated feature expectations. *Bottom:* Difference between true and estimated feature weights.



(a) Initial distribution

(b) 25th iter. (epoch 0.050, standard HMC)

(c) 126th iter. (epoch 0.252, standard HMC)

(d) 136th iter. (epoch 0.272, standard HMC)

(e) 182th iter. (epoch 0.364, parallel tempering)

(f) 195th iter. (epoch 0.420, parallel tempering)

Figure 5.21: Sample-based visualization of the maximum entropy distribution during model training via stochastic gradient descent (SGD) with the Adam optimizer. Trajectories in green/violet represent 500 artificially generated demonstrations with two agents (green: pedestrian, unknown target, violet: robot: target at green circle on the lower left), sampled from a known underlying distribution. Blue/yellow trajectories depict samples from the model during training, where blue/violet and yellow/green trajectories refer to the same agent. Trajectories highlighted in white constitute the global mode of the distribution (obtained via multistart optimization) given the feature weights in the corresponding optimizer iteration, where gray lines indicate equal time instances. In (a) the initial distribution is depicted, intentionally initialized with poor estimates of the feature weights. Up to including (d) the model is trained with standard Hybrid Monte Carlo sampling, yielding quick training progress despite being unable to explore multiple modes simultaneously. The densities from (e) and (f) are obtained after the activation of parallel tempering, correctly sampling from the multimodal distribution.

that can quickly adapt to new environments and situations, without being reliant on external monitoring equipment such as overhead cameras.

Dataset We collected demonstrations of pedestrian motion from eight locations on the ETH Zurich campus, primarily during busy hours. Figure 5.22 depicts the locations and snapshots of the recordings. At each location, we recorded around 2 to 4 minutes of data. Pedestrians were uninformed about the experiment to avoid artificial behavior, and to test the algorithm with fully realistic data as expected during unsupervised, life-long operation. The dataset at 12 o'clock in Figure 5.22 constitutes an

Figure 5.22: Training datasets for the maximum entropy model recorded at seven locations on the campus of ETH Zurich. Pedestrians were uninformed about the experiment to prevent artificial behavior. Source of 3D model: Imagery © 2016 Google, Map data © 2016 Google.

exception, as it contains pedestrian motion designed in a controlled way to teach the robot static and dynamic obstacle avoidance. In all datasets, the recorded occupancy

gridmaps had to be manually cleaned up from clutter occasionally left behind from moving objects due to an imperfect automatic separation of static and dynamic elements in the world. To obtain the demonstration trajectories, continuous B-splines are fitted through the discrete pedestrian motion tracks obtained from the robot's onboard sensors, and the time interval containing all tracks is split into N sub-intervals of equal length, yielding $|\mathcal{X}_D| = N$ demonstrations.

Figure 5.23 depicts the tracks obtained from the dataset at 12 o'clock in Figure 5.22 as an example. Note that the start positions of different agents in the same training

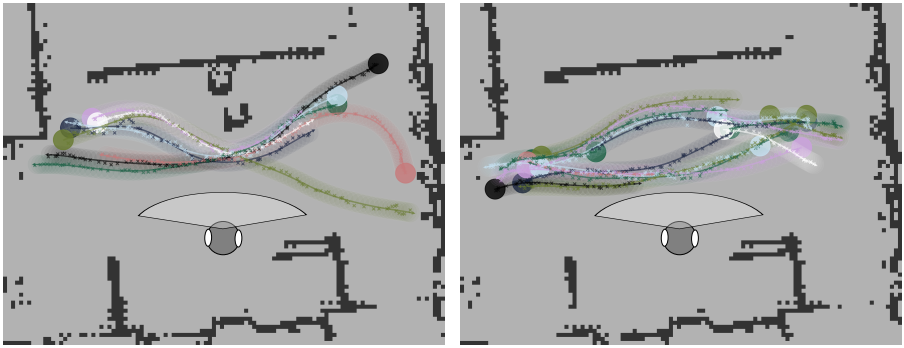


Figure 5.23: Real-world pedestrian tracks for the dataset at 12 o'clock in Figure 5.22. *Left:* Evasive maneuvers for static obstacle avoidance. *Right*: Agent-to-agent interactions with two pedestrians. Tracks are colored depending on their association to different training examples $\mathbf{x} \in \mathcal{X}_D$. Solid lines represent spline fits through the discrete measurements (crosses) of the onboard 2D laser-based object tracker. Circles indicate the geometry of the agents, the measured start positions are highlighted with stronger colors.

example, highlighted by stronger colors, do not necessarily coincide in time. They indicate rather the first observation of an agent within the time interval specified by the training example $\mathbf{x} \in \mathcal{X}_D$. When agents passed each other, the tracker sporadically lost the track due to the occlusion occurring (cf. Figure 5.23, right; cyan, dark green and olive green training examples). In that case, the same agent in the real world is treated as two separate agents.

Results The full demonstration set was randomly split into a training dataset comprised of 70 % of the data, whereas 30 % of the data was held back for validation in order to quantify the predictive capabilities of the model on unseen data. As performance metric we used the root-mean-square error (RMSE) between the positions of the demonstrated trajectories and the modes of the maximum entropy distribution.

This metric does not capture the uncertainty of the model about its predictions but is intuitive to interpret, and as the mode constitutes the property of the distribution used in the online planning stage, the validation error is linked to the online planning capabilities. For orientation, the RMSE for four alternative prediction types is added: (a) constant velocity predictions, (b) shortest path predictions computed on a 2D grid, taking into account static obstacle information for all agents, and (c) predictions obtained with the ORCA approach (Berg, Guy, et al., 2011). For a brief description of ORCA, please refer to Sections 1.2.6 and 5.2. In order to obtain the shortest path predictions, we used an unpublished any-angle path planner[1] similar to Field D* (Ferguson and Stentz, 2007) and Theta* (Daniel et al., 2010). The path planner computes an interpolated distance-to-goal field, eliminating the problem of suboptimal paths due to discretization effects arising from search on a grid graph. Note that interpolated search was introduced to reduce the prediction error of the respective alternative methods for the sake of the fairest comparison. The same any-angle planner is used to compute the direction of preferred velocity in the ORCA approach in order to add a global guidance, rendering it equivalent to the shortest path predictions in case no inter-agent avoidance maneuvers are required. All comparison approaches assume constant agent speeds throughout the agents' movement, obtained from the object tracker's measurement.

To assess the impact of the size and the diversity of the training dataset, results are included in which the model was trained with only a subset of the full demonstrations, recorded in a single environment. The small dataset is comprised of the demonstrations depicted in Figure 5.22 at 12 o'clock, and Figure 5.23. The validation dataset is identical for both cases.

As in the previous training examples with synthetic data, we applied regular HMC sampling in the beginning and activated parallel tempering at a later stage in order to speed up the training process in the beginning. Additionally, a small model regularization with the cumulative L1-regularization procedure described in Tsuruoka et al. (2009) was employed to reduce overfitting due to the after all limited size of even the big demonstration dataset.

Figure 5.24 depicts the progress of the training- and validation error throughout model training. One can observe a rapid decrease of the RMSE validation error for the maximum entropy model(s) within the first 100 iterations. Already after 50 iterations, the maximum entropy model trained on the big dataset clearly outperforms all alternative prediction types on unseen data. This underlines that pedestrian motion is not solely explainable by collision free motion with respect to other agents along the shortest

[1] The any-angle planner is intellectual property of Dr. Francis Colas and used in this work with his kind permission.

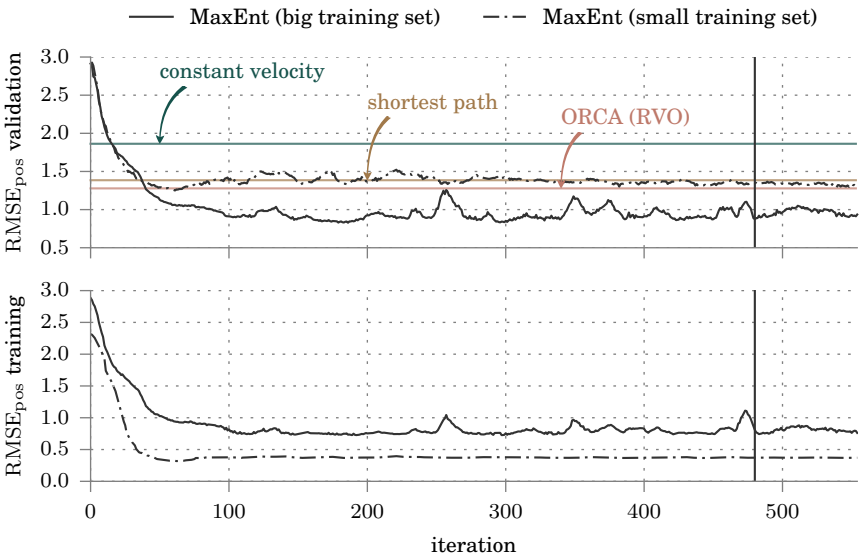


Figure 5.24: Root-mean-square error (RMSE) between the positions of the demonstrated trajectories and the modes of the maximum entropy distribution for the validation dataset (*top*) and training dataset (*bottom*). The figure depicts the prediction error of the maximum entropy model when trained with a small and big dataset using SGD (Adam). The training stage uses standard Hybrid Monte Carlo sampling in the beginning, parallel tempering is activated at a predefined iteration marked with a vertical line. For comparison, the RMSE of three alternative prediction methods is shown. Constant velocity predictions ignore interactions, static obstacle information, and agents' goal locations. Shortest path predictions incorporate static obstacles and goal locations only, whereas ORCA predictions consider all three.

path as encoded in the ORCA model, but additional soft preferences play a role that the maximum entropy approach is able to learn from demonstrations. However, the size and diversity of the training data influence the generalization capabilities of the model to predict motions in other environments. As apparent from Figure 5.24 (top), the maximum entropy model trained on the small, low-diversity dataset exhibits a larger error on unseen data. The bottom graph is included to prove that in both cases a small error on the training dataset is obtained. The training error achieved with the big dataset is slightly higher since the bigger training set contains a larger amount of longer tracks, consequently yielding higher prediction errors in between start and goal locations of agents. It can be seen that training and evaluation error for the model trained on the big dataset is approximately equal, whereas one observes a significantly

lower training than validation error for the model trained on the small dataset. In conclusion, the model should be trained with sufficiently diverse data, and if done so, the maximum entropy model yields superior prediction of joint pedestrian motion.

Figure 5.25 depicts four snapshots illustrating the training progress via composite trajectory samples from the model. The deficiencies of the untrained model, and the higher similarity of the trained model and the demonstrations, are apparent from visual inspection.

Influence of Agents' Intentions/Goals In order to obtain the above results, we assumed known destinations of agents, taken as their last measured position in the demonstrations, in order to compute the goal feature, f_{goal}, for every agent. By doing so, we effectively learn a distribution $p(\mathbf{x} \mid \boldsymbol{\theta}, \mathbf{g})$ conditioned on the composite goal location, $\mathbf{g}$. This is in line with the approach taken in Ziebart, Ratliff, et al. (2009) and Kuderer, Kretzschmar, Sprunk, et al. (2012). We also attempted to train the model using the dataset in Figure 5.23 without incorporating information about the agents' goal locations. However this attempt was not successful—for instance, the model assumed an attraction between pedestrians in order to explain the fact that none of them walked into the corridors to the top (cf. Figure 5.23), but rather stayed within the single room. Likewise, attractive terms for static obstacles were learned to explain the prevalent motion towards walls. In the end, an agent's goal is the most important explanation for a spatial preference and cannot be captured through features rather designed to describe motion styles.

Obviously, destinations of agents are not readily available during online navigation as opposed to the training stage where fully observed trajectories exist. The posterior distribution over destinations may however be inferred online from partial observations of the agents' movement, given a prior distribution obtained for instance from points of interest or goal statistics in a fixed environment (Ziebart, Ratliff, et al., 2009).

Opportunities for Improvement of the Model Despite the good results, we like to point out a few shortcomings of the model and problems encountered, that might have influenced the results when applying the training algorithm on real-world data—or at least prevented us to train the model in a completely unsupervised fashion.
The fact that people behave differently towards friends and strangers has probably influenced the weight of the feature capturing agent interactions, $f_{\text{dyn,soft}}$. From manual inspection of the dataset, we found a considerable percentage of pedestrians traveling in groups of two. None of the feature functions in our model is dedicated to explain

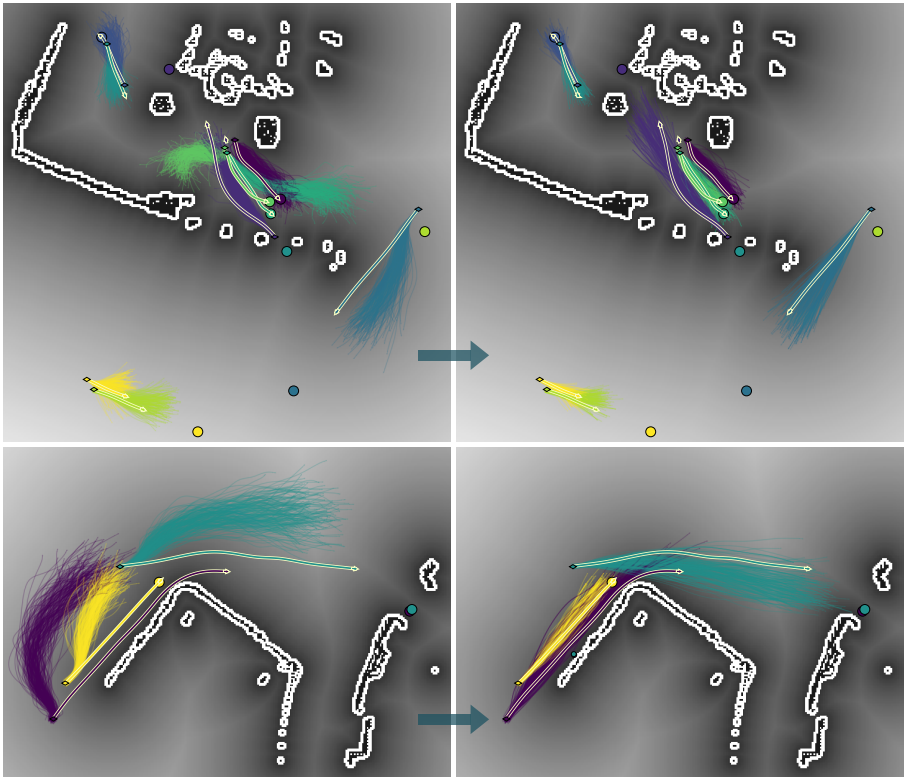


Figure 5.25: Visualization of learning progress with real-world data during the early / late (*left* / *right*) stage of the training progress. Demonstrations correspond to Figure 5.22 at 3 o'clock (*top*) and 6 o'clock (*bottom*). Obstacles (black) are inflated (white) by pedestrian radius; obstacle distances are encoded from black (near) to white (far). Colors distinguish different agents and destinations are marked by circles; demonstrated trajectories are highlighted through brighter outlines. In the beginning, amongst other deficiencies, the model's static obstacle distances and accelerations are too high, apparent from nervous trajectories with high obstacle clearance. In contrast, the trained model induces composite trajectory samples with higher resemblance to demonstrations.

grouping behavior, presumably resulting in smaller learned avoidance distances than with only strangers encountering each other.
The second issue is related to missing semantic information. The training dataset contained one location with a stair leading to an elevated entrance of a tram stop (cf. Figure 5.22, snapshot at 9 o'clock). For pedestrians without the intention to reach the tram stop, the stair represented an object in the world to circumvent, whereas pedestrians targeting the tram stop would pass those unaffectedly. Our laser-based perception pipeline labeled the stairs as impassable, braking the assumption in the baseline model, that agents cannot physically penetrate obstacles. The binary 2D occupancy grid representation of the world employed in our framework forced us to manually remove the stairs from the grid. This might have affected the training result, since the reason for the detours of pedestrians circumventing the stairs was no longer apparent to the training stage. Additionally, the same location included a pedestrian crosswalk that the robot was incapable of detecting—and no dedicated feature to incorporate such semantic information is present. It is possible that the training stage slightly "misuses" the features available in the model in order to explain the detours that pedestrians take to move on the crosswalk and not on the street. In this context, we see the incorporation of the environment's semantics as one of the important next steps. A straightforward way would be through a separate feature per semantic category, encoding the agents' preferences to move on floors with different semantic meanings.
Probably the biggest practical issue encountered was the attention the robot attracted from passers-by. A significant amount of persons suddenly changed their intended destination out of curiosity once they spotted the robot. Such a change in the intention (target) of pedestrians is not modeled, and might have slightly affected the learned feature weight of the goal feature, f_{goal}.

5.10.3 Online Planning

This section is concerned with the evaluation of the online planning stage, where the mode of the maximum entropy distribution over the composite trajectory is used to control the robot. To the best of our knowledge, this work is the first to demonstrate the maximum entropy planner outside of a controlled lab environment in a real-world setting with onboard sensing only. We are particularly interested in the two questions whether (a) interaction-aware motion planning indeed provides an added value in realistic settings, and whether (b) interaction-aware motion planning can soften the negative effects of perception noise on the robot behavior.

To answer the first question, extensive real-world, autonomous driving experiments were conducted in a public space, comparing the cooperative maximum entropy planner with a non-interaction-aware counterpart using a constant velocity model for other agents. The results are presented in Section 5.10.3.1. In order to quantify the benefit of interaction-aware motion planning with respect to the perception noise level in the system, the number of anticipated *inevitable collisions* between cooperative and non-interaction-aware models is compared. Therefor, in an offline processing stage, we simulate additive velocity noise for other agents with varying amplitudes. Afterwards, optimized composite trajectories that exhibit a collision of the ego robot with another object for each approach are counted. This experiments provides an intuition to which extend the assumption of cooperative behavior lets us find a collision-free *joint* solution in cases where non-cooperative planning approaches fail. Section 5.10.3.2 describes the results.

In the following, the cooperative maximum entropy planning approach is compared to two different non-interaction-aware counterparts. In order to isolate the evaluation of particular aspects, the non-interaction-aware variants are likewise based on the cooperative maximum entropy framework. Both variants assume fixed object predictions throughout optimization of the robot's trajectory. The ego agent is thus seeking a collision-free trajectory through a set of fixed obstacles in state-time space. Here, the robot maximizes the probability from Equation 5.14 only for the ego motion as in

$$\mathbf{x}^\star = \left(\mathbf{x}_1^{\text{init}}, \ldots, \underset{\mathbf{x}_R, C(\mathbf{x})=C}{\operatorname{argmin}} \; \boldsymbol{\theta}^{\star\top} \cdot \mathbf{f}(\mathbf{x}), \ldots, \mathbf{x}_n^{\text{init}} \right), \tag{5.49}$$

and hence does not account for the reaction of the other agents to its actions. The first variant, in the following labeled as *non-cooperative*, uses a constant velocity model for other agents. The constant velocity model does not account for static infrastructure, that constrains the motion of other agents. The second variant is labeled as *non-cooperative*$^+$, and performs an independent optimization of other agents based on their motion- and observation features. This variant thus considers static obstacles when forecasting the motion of other agents.

5.10.3.1 Real-World Driving Experiments

The real-world driving experiments were conducted in a public indoor space at ETH Zurich. Besides a 3D laser map used for localization, no environment-specific information was used and the robot operated with onboard sensor data only. Please note that at the time of the experiments the object detection, tracking, and classification pipeline described in Section 5.10.1.2 was not in place yet. The experiments in this section are performed with the approach described in Kümmerle et al. (2015), clustering points labeled as dynamic in the robot's 2D laser point cloud.

As a performance metric, we chose the prediction error of the ego motion of the robot. Establishing a clear evaluation metric is difficult for motion planners with fuzzy goals such as behaving in a socially compliant manner. With our model, we wish to predict joint navigation behavior as accurate as possible. With a sufficiently large planning time horizon, noise-free observations, a perfect model and every agent indeed choosing its trajectory from the most likely composite trajectory, the robot would have to plan its trajectory only once. Plans acquired in subsequent re-planning cycles would not deviate from the initial plan. The less accurate the model can predict the future, the larger the re-planning effort for the robot. Using a more accurate prediction model for all participating agents therefore requires less re-planning effort and results in a smoother and more predictable behavior of the robot. Hence we can use the prediction error of the ego motion as a score for how well the model describes joint navigation behavior. A higher prediction error of joint behavior is directly linked to a higher re-planning effort for the robot. We chose to exclude prediction error statistics for other agents as those would include substantial measurement noise of the robot's object tracker. For the error statistics shown in the experiments, the error is only evaluated during intervals where at least one other pedestrian is in an area of interest in front of the robot, in order to filter out uninteresting time intervals with no interactions.

Experiments with uninformed pedestrians To test the real-world applicability of the approach in a realistic scenario, persons crossing the public space were uninformed about the ongoing experiments. We conducted two experiments under this premise, one with the fully cooperative approach, and one with the *non-cooperative* constant velocity variant. In both experiments, the robot performed nonstop autonomous navigation for approximately one hour, where it traveled around 2 km each by alternating between two fixed waypoints. The robot path and tracks of pedestrians encountered are shown in Figure 5.26 for the experiment utilizing the interaction-aware model. During this run, more than 300 pedestrians were encountered (only observations longer than 2 s considered, double-counting possible), yielding a statistically relevant amount of interactions.

Figure 5.27 depicts the distribution of the Euclidean position error of the ego motion for both experiments at varying time horizons in the future. Surprisingly, no significant difference between the interaction-aware and non-interaction-aware approach can be noticed. We observed that pedestrians uninformed of the ongoing experiment interact with the robot in a significantly different manner than among other humans. Either they try to keep a large security distance to the robot since they do not want to disturb a potential experiment, or they are curious about the robot's behavior when

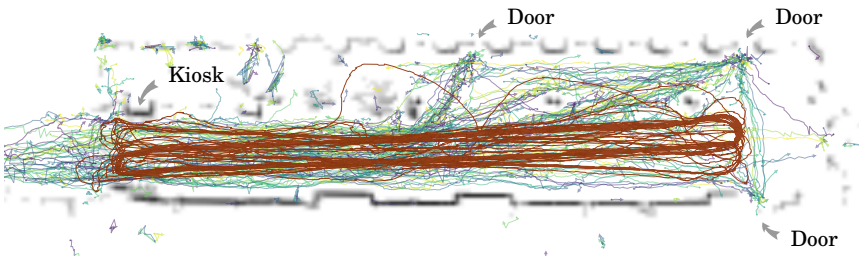


Figure 5.26: Path driven by the robot (red) and pedestrian tracks (cool colors) for one of our experiment runs. The overall traveled distance by the robot in this run was more than 2 km. Throughout this experiment, more than 300 pedestrians were detected (multiple counting possible) by the detection system of the robot. The distance between the two landmarks of the alternating trajectories amounts to roughly 50 m.

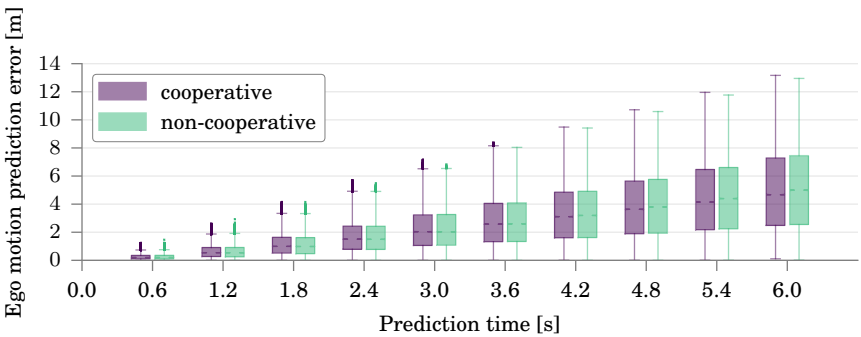


Figure 5.27: Prediction error statistics for the ego motion at varying time horizons with *uninformed* pedestrians. The encountered pedestrians where not informed about the experiment and either avoided the robot at large security distances in order not to disturb it, or they tried to aggressively block its path. The latter behavior results in large prediction errors, as the robot has to be stopped abruptly in order to avoid collisions.

they aggressively block its path. Especially the latter case produces large prediction errors if the robot is stopped abruptly. In the former case where pedestrians intentionally avoid close interactions with the robot, the two models become equal as they exhibit the same optimization problem for the robot.

The outcome of this experiment is unfortunate for the point we want to make. However it raises the interesting question of how robots should behave in situations where people obviously act rude towards them. This ethical issue is beyond the scope of this thesis though. In this context, self-driving vehicles offer a great benefit compared to other robotic platforms, as they can easily hide their robotic nature. At least from a distance, or potentially with tinted windows, the fact that the vehicle is not controlled by a human operator may be covered. This may be necessary to suppress intentionally non-cooperative behavior of other traffic participants towards self-driving cars.

Experiments with informed pedestrians As the previous experiments showed, human-human interaction may differ from human-robot interaction, depending on the familiarity of the individuals with fellow robots. In another experiment, we explicitly asked the participants to interact with the robot in a similar fashion as they would interact with other people. As before, one run for the interaction-aware and non-interaction-aware model was conducted with the robot.

Figure 5.28 shows the evaluation of the prediction error for interaction with informed pedestrians. The interaction-aware approach now outperforms the non-interaction-aware planner in terms of prediction error and consequently re-planning effort. Both the absolute value of the error quantiles and the number and extent of the outliers is significantly reduced. The improvement over the CV model is based on the fact that informed pedestrians interact similarly with the robot as with other pedestrians. The behavior of informed pedestrians can be better explained with the interaction-aware motion model, trained with data from human-human interaction.

5.10.3.2 Resilience To Observation Noise

Our research efforts for the cooperative planning approach were predominantly motivated by the difficulty of the non-interaction-aware partial onlane planner outlined in Chapter 3 to navigate in close proximity to other traffic participants. Especially noise in the observations of agents' velocity vectors sporadically led to emergency brakes, as pedestrians were anticipated to pursue a direct collision course towards the robot. By incorporating the assumption of non-hostile, cooperating agents into a joint cost function, cooperative planning can use this information to rule out such unreasonable behavior.

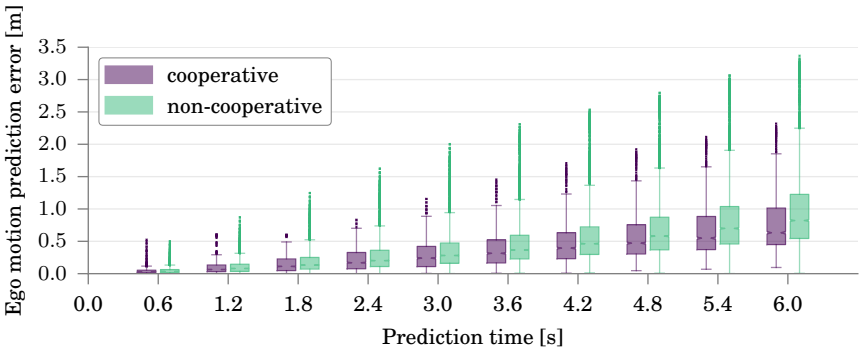


Figure 5.28: Prediction error statistics for the ego motion at varying time horizons with *informed* pedestrians. Only if pedestrians were instructed to behave normally when encountering the robot, the interaction-aware model outperforms the simple constant velocity approach (cf. Figure 5.27).

In the experiment presented in this section, the assumption shall be verified that the cooperative planning approach indeed provides superior resilience to noisy observations. The cooperative planning approach is compared to its non-cooperative counterparts, evaluated with varying artificial noise levels on velocity observations of other agents. Therefore, one of the autonomous runs described in Section 5.10.3.1, consisting of approximately one hour of operation in an area with frequent pedestrian activities, is processed offline. For the sake of a most meaningful comparison, additive velocity noise for other agents is generated once per configuration, such that both non-interaction-aware and interaction-aware motion planners are confronted with identical situations. The fact that the platform is not actually controlled by the produced motion plans is of minor importance for this evaluation. All planners are triggered at a maximum frequency of 5 Hz, resulting in around 18'000 motion plans per planning approach and noise level. As offline processing eliminates the stringent limits on computation time, all dynamic objects detected are considered in the planning query. Occasions are counted in which the robot's trajectory is in collision with any other agent within the planning time horizon after optimization, i.e. the optimization step fails to find a collision-free trajectory. Anticipated collisions with dynamic objects may occur as the framework employs a barrier function with finite height. Hence collisions still possess a non-zero probability, and are weighted against the probabilities induced by motion- and observation features. Intuitively, we expect the number of collisions to be lower for the cooperative approach, possessing more degrees of

freedom to find a collision-free solution via the adaption of other agents' trajectories throughout optimization.

Figure 5.29 depicts the collision rates accumulated throughout the experiment described above. The collision rate is hereby defined as the fraction of queries with

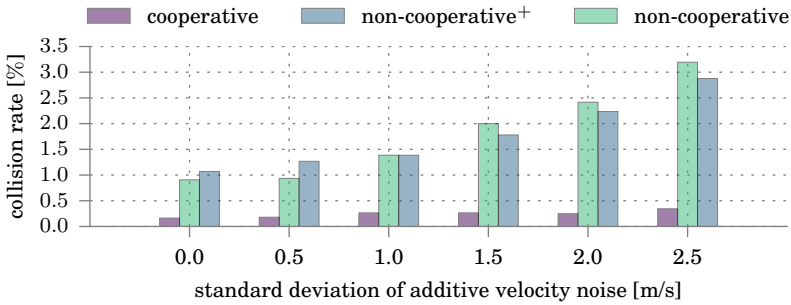


Figure 5.29: Ratio of anticipated collisions with respect to total planning cycles between two non-cooperative variants and the cooperative version of the maximum entropy planner. *Non-cooperative*: constant velocity predictions for other agents. *Non-cooperative*$^+$: independent optimization of other agents and robot. *Cooperative*: fully cooperative version through joint optimization. Data was recorded during one hour of automated operation in a public space, and later corrupted with artificial noise of varying amplitude on the velocity measurements of other agents. The cooperative motion planning yields lower anticipated collision rates for all noise levels.

colliding robot trajectories after optimization, and total planning queries. We observe that the collision rate is increasing with the noise level for both *cooperative* and non-cooperative approaches. The *cooperative* planning approach hereby clearly outperforms the non-interaction-aware approaches for all noise levels and exhibits only a small increase in collision rates. With the default noise level of the object tracker (additive noise equal to zero), the *cooperative* motion planning approach reduces the collision rate by a factor of 6.5. The small differences between the *non-cooperative* and the *non-cooperative*$^+$ variant at low and high noise levels is not immediately intuitive. We suspect that the *non-cooperative*$^+$ shows slightly worse results at low noise levels because static obstacles constrain the solution space, rendering other agents' anticipated future movements more likely to interfere with the robot. At higher noise levels, the fact that constant velocity predictions for other agents ignore static obstacles in the *non-cooperative* variant might yield a higher anticipated collision rate as static obstacles do not "shield" the robot from agents approaching the robot with high speed from behind obstacles.

We conclude that cooperative planning is indeed able to significantly soften the effects of noise in sensor observations. One may however argue that cooperative planning achieves this at the cost of safety, as the assumption of cooperative behavior may not hold in all cases. For instance, other traffic participants may be unaware of the robot due to occlusions or distraction, or simply not willing to cooperate. We therefore advise to treat the results in this section with care.

5.11 Summary & Discussion

In this chapter, a cooperative motion planning approach based on maximum entropy models was presented in order to address the over-conservative behavior of non-cooperative planning techniques. In the approach, a maximum entropy probability distribution over joint navigation behavior is built. Soft preferences of agents are learned from real-world data in order to mimic a human-like navigation style. In order to control the robot, the most likely joint solution is sought, equipping the robot with some form of social etiquette. The approach was implemented and demonstrated on a mobile robotic platform with onboard sensing only, operating continuously for several hours in an uncontrolled public space. We furthermore investigated the potential of the approach in the context of motion planning amongst other decision making agents with substantial measurement noise.

The work presented in this thesis builds upon the work of Kretzschmar et al. (2014), and extends and modifies the approach in various ways. On the side of the problem formulation, we suggest to adopt a *receding horizon approach* with a fixed time horizon and variable end positions of trajectories instead of the formulation in Kuderer, Kretzschmar, Sprunk, et al. (2012) with fixed end positions and a variable time horizon. By doing so, one alleviates the problem of inheriting most likely suboptimal waypoints from a global planner. We also showed that variable end positions open up new possibilities in the training stage, when sample approximations of feature expectations have to be computed. With Parallel Tempered Hybrid Monte Carlo sampling and variable end positions, we could reasonably well sample directly from the high-dimensional, multimodal density.

We identified the computation time of batch model training as one of the main limitations of the previous approach. By utilizing *stochastic optimization*, the training time to find reasonably good model parameters could be drastically improved. The incremental learning aspect of stochastic optimization furthermore provides a life-long learning mechanism, in which the robot may constantly gather new demonstrations and incorporate them into its model.

We trained the model with observations of pedestrian motion recorded with onboard sensors of the robot only. The data was recorded in a variety of public places without informing pedestrians about the ongoing data collection, in order to obtain realistic data equal to the one expected in unsupervised, life-long operation scenarios. We showed experimentally the need for a sufficient diversity in the training data. We could achieve superior performance compared to simpler prediction models such as ORCA only when training the model with data recorded over a broader range of conditions. Reducing computation time of the training stage will be important, such that a sufficient amount of data can be incorporated to better separate the true driving factors of human motion from random aspects. Finally, we highlighted several possibilities for improvement of the model. Incorporating group behaviors and semantics of the environment such as roads, pedestrian walkways and zebra crossing will yield more powerful predictive capabilities.

In Section 5.8 we evaluated the performance of different optimization strategies for the online planning stage. In contrast to Kuderer, Kretzschmar, Sprunk, et al. (2012), we found that *approximate Hessian information* as utilized in Quasi-Newton optimization techniques significantly benefits convergence time.

Uniquely, we demonstrated *real-world navigation* with the cooperative maximum entropy planner in uncontrolled, public spaces. The results obtained during real-world autonomous driving experiments showed that the assumption of cooperative behavior improves joint navigation, as long as other agents behave similar to the learned model. In our experiments, pedestrians not informed about the ongoing experiment often attempted to challenge the robot by actively blocking its path. This behavior was not encoded in the maximum entropy model and resulted in deteriorated performance. Yet, in the context of motion planning for automotive applications, other traffic participants will presumably act less brisk towards the car as consequences may be more fatal and the appearance of autonomous and manual vehicles is similar.

Finally, we demonstrated that interaction-aware motion planning is *less vulnerable to observation noise*. The assumption of cooperative behavior partially rules out anticipated "inevitable collisions", as the other agents' future trajectories are no longer assumed to be fix which provides more degrees of freedom to find a collision-free *joint solution*.

In our opinion, the main shortcomings of the approach presented in this section lie in the quadratic runtime dependency with respect to the number of agents considered, as well as the need for manual feature design. The quadratic runtime stems from the definition of pairwise interaction features and renders the approach inappropriate for larger, dense crowds. For automotive applications the selection of a small

set of relevant traffic participants is more realistic. The computation time for model training is likewise substantial, which results from the challenging task to compute feature expectations of the high-dimensional, multimodal density. For both online planning and model training, moving computations onto the graphics processing unit (GPU) could drastically speed up computation times. Excessive computation times are likewise observed for CPU implementations of deep neural networks. Here, GPU implementations commonly provide a speedup of up to a factor of 50. In our implementation, the discrete Euler integration utilized when computing integral features (or derivatives of those) yields a highly parallelizable structure, suitable for GPU implementation. Whether and to which extent computation times can be reduced through a GPU implementation is however hard to estimate for our case. Also, more sophisticated techniques to focus computational effort on relevant surrounding agents are necessary.

Manual feature design is both a curse and a blessing at the same time. On the one hand it preserves the transparency of the individual features; their meaning is clear as they were designed to capture a certain physical aspect, and the possibility for manual tuning and debugging of the model is retained. On the other hand, substantial expertise is required for the manual feature design. The model's capabilities are in the end limited by the feature designer, and dependencies in the data that are not foreseen by him or her will remain uncovered. When it comes to such complex tasks as autonomous navigation, it is as least questionable whether the engineer's expertise is sufficient to reveal all aspects relevant to the task, or whether mechanisms for automatic feature extraction are required. In the end, computational constraints in the training stage prevented us to explore larger and more powerful feature sets combined with automatic model selection.

6

Conclusion and Future Directions

In this thesis, the problem of motion planning in semi-structured and unstructured environments shared with other decision-making agents, with a focus on automotive application, was addressed. First, a sampling-based, non-cooperative motion planner for navigation along a reference path was developed. The motion planner leveraged the state of the art to provide a high level of system compliance and fast and accurate collision detection. It is applicable for a wide range of mobile systems with constrained kinodynamics and was applied both on three combustion engine and electric autonomous Golf platforms in the context of automated valet parking, as well as on the rough terrain navigation robot ARTOR within an adaptive Teach & Repeat (T&R) framework. The motion planner proved excellent performance in static environments. However, it showed limitations in close proximity to other traffic participants observed only through a noisy monocular vision pipeline, in which observation noise sporadically triggered (unnecessary) emergency brakes.

These difficulties led us to investigate in a cooperative motion planning approach based on a maximum entropy model of joint navigation behavior. As online extraction of rich semantic information of the robot's environment in order to obtain more accurate object predictions is still challenging, cooperation provides an important, autarkic assumption to rule out unreasonable joint behavior. Agents' soft navigation preferences are learned from demonstrated behavior by minimizing the relative entropy to a base model to yield socially compliant robot behavior. Uniquely, we could demonstrate cooperative motion planning with this maximum entropy model in extensive autonomous real-world navigation tests in an uncontrolled public area. The approach proved its potential to address the problem of over-conservative behavior for navigation in close proximity to other decision-making agents.

Nevertheless, there exist areas with room for improvement or issues that have not been addressed in this thesis. In the following, we give a summary of items that need further investigation.

In our implementation, the cooperative maximum entropy planner utilizes a B-spline representation of trajectories, encoding the two-dimensional (2D) positions of agents (and derivatives of such). This representation bears singularities for some physical quantities that might be interesting to include in feature functions. For instance the rotation rate is undefined in case the 2D linear velocity is equal to zero. Holonomic system models were assumed for all agents which eventually could result in trajectories that cannot be faithfully followed by a non-holonomic robot. This raises the question of a suitable continuous *state representation* for different system models. For car-like platforms, kinodynamic constraints are even more pronounced than for differential drive platforms and neglecting them might lead to a large deviation between the planning and the execution layer. One solution to this problem would be to let the splines represent linear velocity v and orientation θ for differential drive systems, and linear velocity v and steering angle ϕ for car-like robots. Yet in this case there exists no analytic solution in general to obtain the 2D position for a given time instance. For a differential drive system for instance, the integral $x(t) = x(t_0) + \int_{t_0}^{t} v(t) \cdot \cos(\theta(t))\,\mathrm{d}t$ has to be evaluated in order to obtain the x-coordinate for time t. An analytic solution exists only for the the simplest functions $v(t)$ and $\theta(t)$. However, employing a numeric integration scheme as in Kelly et al. (2003) is still possible, and analytic gradients of the numeric integration scheme can be computed.

Furthermore, in the cooperative maximum entropy framework, we employed a simplifying outer bounding circle approximation for collision shapes of agents in order to obtain derivatives of inter-agent distances and static obstacle distances independent of the agents' orientation. For the European Robotic Pedestrian Assistant 2 (EUROPA2) platform used in our experiments and for pedestrians, this approximation is reasonably good. However, for platforms with elongated shapes such as cars, the outer bounding circle approximation might eliminate too much space for navigation. Depending on the narrowness of the environment, this may be unacceptable. Incorporating *exact collision shapes* into the framework is in general possible, for instance using finite differences for the derivatives of distances between objects and objects and the static world. The runtime overhead of computing finite differences is presumably acceptable, as the splines' design variables have only local influence on the trajectory. Yet in combination with an alternative state representation as outlined in the paragraph before, the validity of this concept needs clarification.

In the online planning stage of the cooperative maximum entropy framework, the most probable composite trajectory (mode) is determined by minimizing the negative log density of the maximum entropy distribution. The negative log density bares multiple *local minima*, induced by static obstacles and dynamic objects. Finding the *global optimum* given the stringent runtime constraints present in the online plan-

ning stage is very challenging, and at the present we have no guarantees that even a good local minimum is found. Kuderer, Kretzschmar, and Burgard (2013) and Kretzschmar, Spies, et al. (2016) therefore maintain a set of homotopically different composite trajectories. The authors use winding angles to compute a homotopy fingerprint of a composite trajectory, and with that enumerate homotopy classes. In order to compute homotopically distinct paths, the authors employ a Voronoi diagram for the static environment, and a forward simulation technique to generate homotopically distinct inter-agent avoidance maneuvers. Unfortunately the concept of winding angles is not applicable any more if the end positions of trajectories are free, as it is the case in the receding horizon formulation adopted in this thesis. In fact, one can always find a continuous transformation of two trajectories into each other (assuming the two trajectories are collision free). The problem of vanishing homotopies with variable end positions is also recognized by Bender et al. (2015) in the context of onlane driving applications. However, the techniques presented by Kuderer, Kretzschmar, and Burgard (2013) and Bender et al. (2015) to generate homotopically distinct composite trajectories are to some extent also useful in our case as we expect the homotopic regions often to coincide with the convex regions of the distribution. These techniques could then produce as a set of initialization points for gradient-based optimization. In this work at hand we did not further pursue this direction, as performing multi-start optimization was hardly online applicable in our implementation.

The issues encountered in the gradient-based maximum entropy planner—namely *state representation*, *exact collision shapes*, and *global optimization*—are readily addressed in the partial onlane motion planner presented in Chapter 3 and Chapter 4. We outlined ideas of how to approach these issues in the cooperative framework, yet this is subject to future research effort. Apart from the approach-specific enhancements described in this section, we see still major work to be done in order to create truly intelligent navigation systems in environments with a richer set of rules and semantics, such as the environments encountered in public traffic. It is questionable whether a single cost function yields appropriate behavior over all possible situations encountered, such as driving on highways, approaching traffic lights or navigating on parking lots. Training different situation-specific models and relying on a higher-level situational-awareness module is a common approach. The maximum entropy model may even provide opportunities to generate sensible behavior for situations never encountered so far, as interpolation between cost functions (or feature weights) will presumably retain meaningful behavior. Yet we believe that situational-awareness modules should ultimately be avoided, as they do not pave the way towards a holistic, powerful, and low-maintenance motion planning framework. We are therefore exited about the recent progress in deep artificial neural network architectures and their

potential for feature extraction and as powerful approximators of reward functions (Wulfmeier et al., 2016).

This thesis furthermore raised a more ethical issue concerning the introduction of autonomous agents into everyday life. Mobile robots in public will presumably attract the attention and curiosity of fellow humans up to the point that execution of their tasks becomes impossible—at least until humans become familiar with their new companions. Robots will need to be equipped with intuitive means to communicate their intentions and goals, and mechanisms to enforce those to an acceptable extent. The assumption of cooperation pursued in this thesis constitutes a step towards less conservative mobile robots, yet the definition of acceptable behavior is a delicate matter that requires interdisciplinary discussion.

Bibliography

Abbeel, P., Dolgov, D., Ng, A. Y., and Thrun, S. (2008). "Apprenticeship Learning for Motion Planning with Application to Parking Lot Navigation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1083–1090. DOI: 10.1109/IROS.2008.4651222
(Cited on page 85)

Abbeel, P. and Ng, A. Y. (2004). "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the 21st International Conference on Machine Learning ICML)*. ACM Press, p. 1. DOI: 10.1145/1015330.1015430
(Cited on pages 19, 84, 85)

Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., and Siegwart, R. (2013). "Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots". In: *Distributed Autonomous Robotic Systems*. Springer Tracts in Advanced Robotics 83. Springer Berlin Heidelberg, pp. 203–216. DOI: 10.1007/978-3-642-32723-0_15
(Cited on pages 11, 82)

Anderson, J. C. and Gerbing, D. W. (1984). "The effect of sampling error on convergence, improper solutions, and goodness-of-fit indices for maximum likelihood confirmatory factor analysis". *Psychometrika* 49.2, pp. 155–173. DOI: 10.1007/BF02294170
(Cited on page 105)

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). "A Survey of Robot Learning from Demonstration". *Robotics and Autonomous Systems* 57.5, pp. 469–483. DOI: 10.1016/j.robot.2008.10.024
(Cited on page 84)

Åström, K. J. and Hägglund, T. (2006). *Advanced PID Control*. The Instrumentation, Systems, and Automation Society, p. 460
(Cited on page 42)

Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., Cacciola, S., Currier, P., Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Covern, D. V., and Webster, M. (2008). "Odin: Team VictorTango's entry in the DARPA Urban Challenge". *Journal of Field Robotics* 25.8, pp. 467–492. DOI: 10.1002/rob.20248
(Cited on pages 15, 30, 53)

Bender, P., Tas, O. S., Ziegler, J., and Stiller, C. (2015). "The combinatorial aspect of motion planning: Maneuver variants in structured environments". In: *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, pp. 1386–1392. DOI: 10.1109/IVS.2015.7225909
(Cited on page 151)

Berg, J. van den, Guy, S. J., Lin, M., and Manocha, D. (2011). "Reciprocal n-Body Collision Avoidance". In: *Robotics Research: The 14th International Symposium ISRR*. Ed. by C. Pradalier, R. Siegwart, and G. Hirzinger. Vol. 70. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, pp. 3–19. DOI: 10.1007/978-3-642-19457-3_1
(Cited on pages 11, 82, 135)

Berg, J. van den, Lin, M., and Manocha, D. (2008). "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928–1935. DOI: 10.1109/ROBOT.2008.4543489
(Cited on pages 11, 82)

Berg, J. van den and Overmars, M. H. (2004). "Roadmap-Based Motion Planning in Dynamic Environments". In: *Intelligent Robots and Systems, IEEE/RSJ International Conference on*. Vol. 2, pp. 1598–1605. DOI: 10.1109/IROS.2004.1389624
(Cited on page 66)

Berg, J. van den, Patil, S., Sewall, J., Manocha, D., and Lin, M. (2008). "Interactive Navigation of Multiple Agents in Crowded Environments". In: *Proceedings of the Symposium on Interactive 3D Graphics and Games*. Vol. 1. ACM Press, pp. 139–148. DOI: 10.1145/1342250.1342272
(Cited on pages 10, 83)

Bergen, G. v. d. (1997). "Efficient collision detection of complex deformable models using AABB trees". *Journal of Graphics Tools* 2.4, pp. 1–13
(Cited on page 69)

Bircher, A., Alexis, K., Schwesinger, U., Omari, S., Burri, M., and Siegwart, R. (2017). "An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees". *Robotica* 35.6, pp. 1327–1340. DOI: 10.1017/S0263574716000084
(Cited on page 27)

Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J., and Satterfield, B. (2008). "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge". *Journal of Field Robotics* 25.9, pp. 598–614. DOI: 10.1002/rob.20260
(Cited on pages 9, 15, 30, 53)

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). *End to End Learning for Self-Driving Cars*. Version 1. arXiv: 1604.07316 [cs.CV]
(Cited on page 12)

Boor, C. de (1972). "On Calculating with B-splines". *Journal of Approximation Theory* 6.1, pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9
(Cited on page 91)

Borenstein, J. and Koren, Y. (1991). "The vector field histogram - fast obstacle avoidance for mobile robots". *IEEE Transactions on Robotics and Automation* 7.3, pp. 278–288. DOI: 10.1109/70.88137
(Cited on page 29)

Bottou, L. (2010). "Large-Scale Machine Learning with Stochastic Gradient Descent". In: *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, pp. 177–186. DOI: 10.1007/978-3-7908-2604-3_16
(Cited on page 113)

Broggi, A., Medici, P., Zani, P., Coati, A., and Panciroli, M. (2012). "Autonomous vehicles control in the VisLab Intercontinental Autonomous Challenge". *Annual Reviews in Control* 36.1, pp. 161–171. DOI: 10.1016/j.arcontrol.2012.03.012
(Cited on pages 9, 58)

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC. DOI: 10.1201/b10905
(Cited on pages 107, 108)

Cameron, S. (1984). “Modelling Solids in Motion”. PhD thesis. University of Edinburgh
(Cited on pages 23, 68)

Cameron, S. (1985). “A study of the clash detection problem in robotics”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2, pp. 488–493. DOI: 10.1109/ROBOT.1985.1087245
(Cited on pages 23, 68)

Cameron, S. (1990). “Collision Detection by Four-Dimensional Intersection Testing”. *IEEE Transactions on Robotics and Automation* 6.3, pp. 291–302. DOI: 10.1109/70.56661
(Cited on pages 23, 68)

Catmull, E. and Rom, R. (1974). “A Class of Local Interpolating Splines”. *Computer aided geometric design* 74, pp. 317–326
(Cited on pages 97, 100)

Coumans, E. *Bullet Physics Library*. www.bulletphysics.org, accessed September 2016
(Cited on pages 68, 73, 76)

Daniel, K., Nash, A., Koenig, S., and Felner, A. (2010). “Theta*: Any-Angle Path Planning on Grids”. *Journal of Artificial Intelligence* 39, pp. 533–579. DOI: 10.1613/jair.2994
(Cited on page 135)

Derendarz, W., Iser, R., Mühlfellner, P., Wonneberger, S., Nguyen, T. N., Brosig, S., Effertz, J., Mielenz, H., Mimura, K., Beermann, M., Schwesinger, U., Rufli, M., Furgale, P., Timpner, J., Rottmann, S., Wolf, L., Häne, C., Heng, L., Lee, G., Pollefeys, M., Grimmet, H., Triebel, R., Posner, I., Newman, P., Cardarelli, E., Cattani, S., Bertozzi, M., Brüning, S., Horstmann, S., Stellmacher, M., and Siegwart, R. (2013). “Vollautomatisches Laden und Fahren auf öffentlichen Parkplätzen”. In: *Internationaler Kongress Elektronik im Fahrzeug*. VDI-Berichte 2188. VDI Wissensforum GmbH
(Cited on page 26)

Dijkstra, E. W. (1959). “A Note on Two Problems in Connexion with Graphs”. *Numerische Mathematik* 1.1, pp. 269–271. DOI: 10.1007/BF01386390
(Cited on pages 6, 99)

Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. (2010). "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments". *The International Journal of Robotics Research* 29.5, pp. 485–501. DOI: 10.1177/0278364909359210
(Cited on pages 7, 47)

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). "Hybrid Monte Carlo". *Physics Letters B* 195.2, pp. 216–222. DOI: 10.1016/0370-2693(87)91197-X
(Cited on page 107)

Dubins, L. E. (1957). "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, pp. 497–516. DOI: 10.2307/2372560
(Cited on page 6)

Duchi, J., Hazan, E., and Singer, Y. (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". *Journal of Machine Learning Research* 12, pp. 2121–2159
(Cited on page 113)

Elith, J., Phillips, S. J., Hastie, T., Dudík, M., Chee, Y. E., and Yates, C. J. (2011). "A statistical explanation of MaxEnt for ecologists". *Diversity and Distributions* 17.1, pp. 43–57. DOI: 10.1111/j.1472-4642.2010.00725.x
(Cited on page 96)

Ender, D. (1997). "Implementation of deadband reset scheduling for the elimination of stick-slip cycling in control valves: Feature issue controller tuning". *Journal A* 38.1, pp. 11–15
(Cited on page 42)

Erdmann, M. and Lozano-Pérez, T. (1987). "On Multiple Moving Objects". English. *Algorithmica* 2.1, pp. 477–521. DOI: 10.1007/BF01840371
(Cited on page 66)

Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). "Distance Transforms of Sampled Functions". *Theory of Computing* 8, pp. 415–428. DOI: 10.4086/toc.2012.v008a019
(Cited on pages 43, 97)

Ferguson, D., Darms, M., Urmson, C., and Kolski, S. (2008). "Detection, Prediction, and Avoidance of Dynamic Obstacles in Urban Environments". In: *Proceedings of*

the IEEE Intelligent Vehicles Symposium (IV), pp. 1149–1154. DOI: 10.1109/IVS.2008.4621214
(Cited on pages 9, 49, 51, 67)

Ferguson, D., Howard, T. M., and Likhachev, M. (2008). "Motion Planning in Urban Environments". *Journal of Field Robotics* 25.11-12, pp. 939–960. DOI: 10.1002/rob.20265
(Cited on pages 6, 9)

Ferguson, D. and Stentz, A. (2007). "Field D*: An Interpolation-Based Path Planner and Replanner". *International Symposium on Robotics Research* 28, pp. 239–253. DOI: 10.1007/978-3-540-48113-3_22
(Cited on page 135)

Fiorini, P. and Shiller, Z. (1998). "Motion Planning in Dynamic Environments using Velocity Obstacles". *The International Journal of Robotics Research* 17, pp. 760–772. DOI: 10.1177/027836499801700706
(Cited on pages 3, 11, 82)

Fox, D., Burgard, W., and Thrun, S. (1997). "The Dynamic Window Approach to Collision Avoidance". *IEEE Robotics & Automation Magazine* 4.1, pp. 23–33. DOI: 10.1109/100.580977
(Cited on pages 9, 29, 48, 116)

Fritsch, F. N. and Carlson, R. E. (1980). "Monotone Piecewise Cubic Interpolation". *SIAM Journal on Numerical Analysis* 17.2, pp. 238–246. DOI: 10.1137/0717021
(Cited on page 100)

Fukunishi, H., Watanabe, O., and Takada, S. (2002). "On the Hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction". *The Journal of Chemical Physics* 116.20, pp. 9058–9067. DOI: 10.1063/1.1472510
(Cited on page 110)

Furgale, P., Krüsi, P., Pomerleau, F., Schwesinger, U., Colas, F., and Siegwart, R. (2014). "There and Back Again—Dealing with highly-dynamic scenes and long-term change during topological/metric route following". *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*
(Cited on page 26)

Furgale, P., Schwesinger, U., Rufli, M., Derendarz, W., Grimmett, H., Mühlfellner, P., Wonneberger, S., Rottmann, J. T. S., Li, B., Schmidt, B., Nguyen, T. N., Cardarelli,

E., Cattani, S., Brüning, S., Horstmann, S., Stellmacher, M., Mielenz, H., Köser, K., Beermann, M., Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Iser, R., Triebel, R., Posner, I., Newman, P., Wolf, L., Pollefeys, M., Brosig, S., Effertz, J., Pradalier, C., and Siegwart, R. (2013). "Toward Automated Driving in Cities using Close-to-Market Sensors: An Overview of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 809–816. DOI: 10.1109/IVS.2013.6629566
(Cited on page 25)

Garrote, L., Premebida, C., Silva, M., and Nunes, U. (2014). "An RRT-based Navigation Approach for Mobile Robots and Automated Vehicles". In: *12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 326–331. DOI: 10.1109/INDIN.2014.6945533
(Cited on pages 8, 16)

Geyer, C. J. (1991). "Markov Chain Monte Carlo Maximum Likelihood". *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*. Ed. by E. M. Keramidas, pp. 156–163
(Cited on page 109)

Gottschalk, S., Lin, M. C., and Manocha, D. (1996). "OBBTree: A Hierarchical Structure for Rapid Interference Detection". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. ACM, pp. 171–180. DOI: 10.1145/237170.237244
(Cited on page 67)

Green, M. (2000). ""How Long Does It Take to Stop?" Methodological Analysis of Driver Perception-Brake Times". *Transportation Human Factors* 2.3, pp. 195–216. DOI: 10.1207/STHF0203_1
(Cited on page 13)

Grimmett, H., Buerki, M., Paz, L., Piniés, P., Furgale, P., Posner, I., and Newman, P. (2015). "Integrating Metric and Semantic Maps for Vision-Only Automated Parking". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2159–2166. DOI: 10.1109/ICRA.2015.7139484
(Cited on page 33)

Guy, S. J., Berg, J. van den, Liu, W., Lau, R., Lin, M. C., and Manocha, D. (2012). "A Statistical Similarity Measure For Aggregate Crowd Dynamics". *ACM Transactions on Graphics (TOG)* 31.6. DOI: 10.1145/2366145.2366209
(Cited on page 83)

Häne, C., Sattler, T., and Pollefeys, M. (2015). "Obstacle Detection for Self-Driving Cars Using Only Monocular Cameras and Wheel Odometry". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5101–5108. DOI: 10.1109/IROS.2015.7354095
(Cited on page 32)

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107. DOI: 10.1109/TSSC.1968.300136
(Cited on page 6)

Hastings, W. K. (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika* 57.1, pp. 97–109. DOI: 10.2307/2334940
(Cited on page 106)

Helbing, D. and Molnár, P. (1995). "Social Force Model for Pedestrian Dynamics". *Physical Review E* 51.5, pp. 4282–4286. DOI: 10.1103/PhysRevE.51.4282
(Cited on pages 80, 81, 86, 94, 101)

Helbing, D. and Tilch, B. (1998). "Generalized Force Model of Traffic Dynamics". *Physical Review E* 58.1, pp. 133–138. DOI: doi.org/10.1103/PhysRevE.58.133
(Cited on page 81)

Henderson, L. F. (1971). "The Statistics of Crowd Fluids". *Nature* 229.5284, pp. 381–383. DOI: 10.1038/229381a0
(Cited on page 81)

Henry, P., Vollmer, C., Ferris, B., and Fox, D. (2010). "Learning to Navigate Through Crowded Environments". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 981–986. DOI: 10.1109/ROBOT.2010.5509772
(Cited on page 85)

Howard, T. M. and Kelly, A. (2007). "Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots". *The International Journal of Robotics Research* 26.2, pp. 141–166. DOI: 10.1177/0278364906075328
(Cited on page 30)

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). "Randomized Kinodynamic Motion Planning with Moving Obstacles". *The International Journal of Robotics Research* 21.3, pp. 233–255. DOI: 10.1177/027836402320556421
(Cited on page 66)

Hundelshausen, F. von, Himmelsbach, M., Hecker, F., Mueller, A., Wuensche, H.-J., and Hundelshausen, F. V. (2008). "Driving with Tentacles: Integral Structures for Sensing and Motion". *Journal of Field Robotics* 25.9, pp. 640–673. DOI: 10.1002/rob.20256
(Cited on page 9)

Igel, C. and Hüsken, M. (2000). "Improving the Rprop Learning Algorithm". *Proceedings of the 2nd International Symposium on Neural Computation*, pp. 115–121
(Cited on pages 117, 126)

Jaynes, E. T. (1957). "Information Theory and Statistical Mechanics". *Physical Review* 106.4, pp. 620–630. DOI: 10.1103/PhysRev.106.620
(Cited on page 87)

Jeon, J., Cowlagi, R. V., Peters, S. C., Karaman, S., Frazzoli, E., Tsiotras, P., and Iagnemma, K. (2013). "Optimal Motion Planning with the Half-Car Dynamical Model for Autonomous High-Speed Driving". *American Control Conference*, pp. 188–193. DOI: 10.1109/ACC.2013.6579835
(Cited on page 8)

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems". *Transactions of the ASME–Journal of Basic Engineering* 82.Series D, pp. 35–45
(Cited on page 122)

Karaman, S. and Frazzoli, E. (2009). "Sampling-based Motion Planning With Deterministic μ-Calculus Specifications". *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 2222–2229. DOI: 10.1109/CDC.2009.5400278
(Cited on page 8)

Karaman, S. and Frazzoli, E. (2011). "Sampling-based Algorithms for Optimal Motion Planning". *The International Journal of Robotics Research* 30.7, pp. 846–894. DOI: 10.1177/0278364911406761
(Cited on pages 8, 65)

Karasev, V., Ayvaci, A., Heisele, B., and Soatto, S. (2016). "Intent-Aware Long-Term Prediction of Pedestrian Motion". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2543–2549. DOI: 10.1109/ICRA.2016.7487409
(Cited on pages 10, 83)

Katrakazas, C., Quddus, M., Chen, W.-H., and Deka, L. (2015). "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". *Transportation Research Part C: Emerging Technologies* 60, pp. 416–442. DOI: 10.1016/j.trc.2015.09.011
(Cited on page 12)

Kazama, J. and Tsujii, J. (2003). "Evaluation and Extension of Maximum Entropy Models with Inequality Constraints". In: *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. Ed. by M. Collins and M. Steedman. Vol. 10. 2. Association for Computational Linguistics, pp. 137–144. DOI: 10.3115/1119355.1119373
(Cited on page 115)

Kelly, A. and Nagy, B. (2003). "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control". *The International Journal of Robotics Research* 22.7-8, pp. 583–601. DOI: 10.1177/02783649030227008
(Cited on pages 6, 9, 150)

Kim, S., Guy, S. J., Liu, W., Wilkie, D., Lau, R. W. H., Lin, M. C., and Manocha, D. (2014). "BRVO: Predicting Pedestrian Trajectories Using Velocity-Space Reasoning". *The International Journal of Robotics Research* 34.2, pp. 201–217. DOI: 10.1177/0278364914555543
(Cited on pages 11, 82)

Kingma, D. and Ba, J. (2015). "Adam: A Method for Stochastic Optimization". *The 3rd International Conference on Learning Representations*. arXiv: 1412.6980 [cs.LG]
(Cited on pages 113, 114)

Klosowski, J., Held, M., Mitchell, J., Sowizral, H., and Zikan, K. (1998). "Efficient Collision Detection Using Bounding Volume Hierarchies of K-DOPs". *IEEE Transactions on Visualization and Computer Graphics* 4.1, pp. 21–36. DOI: 10.1109/2945.675649
(Cited on page 67)

Kretzschmar, H. (2014). "Learning Probabilistic Models for Mobile Robot Navigation". PhD thesis. University of Freiburg
(Cited on page 85)

Kretzschmar, H., Kuderer, M., and Burgard, W. (2013). "Inferring Navigation Policies for Mobile Robots From Demonstrations". *Proceedings of the Autonomous*

Learning Workshop at the IEEE International Conference on Robotics and Automation (ICRA) 132
(Cited on page 119)

Kretzschmar, H., Kuderer, M., and Burgard, W. (2014). "Learning to Predict Trajectories of Cooperatively Navigating Agents". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4015–4020. DOI: 10.1109/ICRA.2014.6907442
(Cited on pages 11, 12, 24, 80, 85, 101, 146)

Kretzschmar, H., Spies, M., Sprunk, C., and Burgard, W. (2016). "Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning". *The International Journal of Robotics Research* 35.11, pp. 1289–1307. DOI: 10.1177/0278364915619772
(Cited on pages 85, 151)

Krüsi, P., Bücheler, B., Pomerleau, F., Schwesinger, U., Siegwart, R., and Furgale, P. (2015). "Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching". *Journal of Field Robotics* 32.4, pp. 534–564. DOI: 10.1002/rob.21524
(Cited on pages 14, 15, 22, 26)

Kuderer, M. (2015). "Socially compliant mobile robot navigation". PhD thesis. University of Freiburg
(Cited on page 85)

Kuderer, M., Gulati, S., and Burgard, W. (2015). "Learning driving styles for autonomous vehicles from demonstration". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2641–2646. DOI: 10.1109/ICRA.2015.7139555
(Cited on pages 12, 85)

Kuderer, M., Kretzschmar, H., and Burgard, W. (2013). "Teaching Mobile Robots to Cooperatively Navigate in Populated Environments". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3138–3143. DOI: 10.1109/IROS.2013.6696802
(Cited on pages 12, 85, 151)

Kuderer, M., Kretzschmar, H., Sprunk, C., and Burgard, W. (2012). "Feature-Based Prediction of Trajectories for Socially Compliant Navigation". In: *Proceedings of*

Robotics: Science and Systems (RSS). DOI: 10.15607/RSS.2012.VIII.025
(Cited on pages 11, 12, 19, 23, 79, 85, 86, 91, 101, 117, 137, 146, 147)

Kullback, S. and Leibler, R. A. (1951). "On Information and Sufficiency". *The Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: 10.1214/aoms/1177729694
(Cited on page 90)

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., and Burgard, W. (2015). "Autonomous Robot Navigation in Highly Populated Pedestrian Zones". *Journal of Field Robotics* 32.4, pp. 565–589. DOI: 10.1002/rob.21534
(Cited on page 140)

Lakoba, T. I., Kaup, D. J., and Finkelstein, N. M. (2005). "Modifications of the Helbing-Molnar-Farkas-Vicsek Social Force Model for Pedestrian Evolution". *Simulation* 81.5, pp. 339–352. DOI: 10.1177/0037549705052772
(Cited on page 81)

Lan, S., Streets, J., and Shahbaba, B. (2014). "Wormhole Hamiltonian Monte Carlo". In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pp. 1953–1959
(Cited on page 112)

Laskey, K. B. and Myers, J. W. (2003). "Population Markov Chain Monte Carlo". *Machine Learning* 50.1, pp. 175–196. DOI: 10.1023/A:1020206129842
(Cited on page 111)

Latombe, J.-C. (1991). *Robot Motion Planning*. Springer US. DOI: 10.1007/978-1-4615-4022-9
(Cited on page 66)

LaValle, S. M. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. 98-11. Deptartement of Computer Science, Iowa State University, pp. 11–98
(Cited on pages 8, 14, 29, 65, 66)

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press, p. 842
(Cited on pages 2, 16, 65)

Lawitzky, A., Wollherr, D., and Buss, M. (2012). "Maneuver-based Risk Assessment for High-speed Automotive Scenarios". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1186–1191. DOI: 10.1109/IROS.201

2.6385825
(Cited on pages 16, 17, 23, 66)

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). "Efficient BackProp". In: *Neural Networks: Tricks of the Trade*. Ed. by G. B. Orr and K.-R. Müller. Springer Berlin Heidelberg, pp. 9–50. DOI: 10.1007/3-540-49430-8_2
(Cited on page 113)

Lee, D. N. (1976). "A Theory of Visual Control of Braking Based on Information About Time-to-collision". *Perception* 5.4, pp. 437–459. DOI: 10.1068/p050437
(Cited on page 16)

Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). "Group Behavior from Video: A Data-Driven Approach to Crowd Simulation". In: *Eurographics/SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, pp. 109–118. DOI: 10.2312/SCA/SCA07/109-118
(Cited on pages 10, 83)

Lefèvre, S., Vasquez, D., and Laugier, C. (2014). "A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles". *ROBOMECH Journal* 1.1, pp. 1–14. DOI: 10.1186/s40648-014-0001-z
(Cited on pages 9, 10, 16, 49)

Levine, S., Popović, Z., and Koltun, V. (2011). "Nonlinear Inverse Reinforcement Learning with Gaussian Processes". In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. Curran Associates, Inc., pp. 19–27
(Cited on page 86)

Lewis, A. S. and Overton, M. L. (2013). "Nonsmooth Optimization via Quasi-Newton Methods". *Mathematical Programming* 141.1, pp. 135–163. DOI: 10.1007/s10107-012-0514-2
(Cited on page 93)

Likhachev, M. and Ferguson, D. (2008). "Planning Long Dynamically-Feasible Maneuvers For Autonomous Vehicles". *The International Journal of Robotics Research* 28.8, pp. 933–945. DOI: 10.1177/0278364909340445
(Cited on page 6)

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005). "Anytime Dynamic A*: An Anytime, Replanning Algorithm". *Proceedings of the Interna-*

tional Conference on Automated Planning and Scheduling (ICAPS), pp. 262–271
(Cited on page 6)

Ma, L., Yang, J., and Zhang, M. (2012). "A Two-level Path Planning Method for On-road Autonomous Driving". In: *Second International Conference on Intelligent System Design and Engineering Application*. IEEE, pp. 661–664. DOI: 10.1109/ISdea.2012.607
(Cited on page 9)

Macek, K., Becker, M., and Siegwart, R. (2006). "Motion Planning for Car-Like Vehicles in Dynamic Urban Scenarios". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4375–4380. DOI: 10.1109/IROS.2006.282013
(Cited on pages 8, 16)

McNaughton, M., Urmson, C., Dolan, J. M., and Lee, J.-w. (2011). "Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1, pp. 4889–4895. DOI: 10.1109/ICRA.2011.5980223
(Cited on pages 8, 16)

Mokhtari, A. and Ribeiro, A. (2014). "RES: Regularized Stochastic BFGS Algorithm". *IEEE Transactions on Signal Processing* 62.23, pp. 6089–6104. DOI: 10.1109/TSP.2014.2357775
(Cited on page 129)

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. (2008). "Junior: The Stanford entry in the Urban Challenge". *Journal of Field Robotics* 25.9, pp. 569–597. DOI: 10.1002/rob.20258
(Cited on pages 9, 15, 30, 53)

Mühlfellner, P., Bürki, M., Bosse, M., Derendarz, W., Philippsen, R., and Furgale, P. (2015). "Summary Maps for Lifelong Visual Localization". *Journal of Field Robotics* 33.5, pp. 561–590. DOI: 10.1002/rob.21595
(Cited on pages 33, 39)

Munkres, J. (1957). "Algorithms for the Assignment and Transportation Problems". *Journal of the Society for Industrial and Applied Mathematics* 5.1, pp. 32–38. DOI:

10.1137/0105003
(Cited on page 124)

Ng, A. Y. (2004). "Feature selection, L1 vs. L2 regularization, and rotational invariance". In: *21st International Conference on Machine learning (ICML)*. ACM Press, p. 78. DOI: 10.1145/1015330.1015435
(Cited on page 115)

Ng, A. Y. and Russell, S. J. (2000). "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the 17th International Conference on Machine Learning*. ICML '00. Morgan Kaufmann Publishers Inc., pp. 663–670
(Cited on page 84)

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. 2nd edition. Springer Series in Operations Research and Financial Engineering. Springer New York. DOI: 10.1007/978-0-387-40065-5
(Cited on page 117)

Pan, J., Chitta, S., and Manocha, D. (2012). "FCL: A general purpose library for collision and proximity queries". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3859–3866. DOI: 10.1109/ICRA.2012.6225337
(Cited on pages 68, 73, 76)

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009). "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260
(Cited on pages 10, 83)

Pellegrini, S., Ess, A., and Van Gool, L. (2010). "Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings". In: *Proceedings of the 11th European Conference on Computer Vision: Part I*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 452–465. DOI: 10.1007/978-3-642-15549-9_33
(Cited on pages 10, 83)

Pfeiffer, M., Schwesinger, U., Sommer, H., Galceran, E., and Siegwart, R. (2016). "Predicting Actions to Act Predictably: Cooperative Partial Motion Planning with Maximum Entropy Models". In: *IEEE/RSJ International Conference on Intelligent*

Robots and Systems (IROS), pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329
(Cited on pages 21, 25, 79)

Pivtoraiko, M. and Kelly, A. (2006). "Constrained Motion Planning in Discrete State Spaces". In: *Field and Service Robotics: Results of the 5th International Conference*. Vol. 25. Springer Tracts in Advanced Robotics, pp. 269–280. DOI: 10.1007/978-3-540-33453-8_23
(Cited on pages 6, 14, 29, 65, 66)

Pivtoraiko, M. and Kelly, A. (2011). "Kinodynamic Motion Planning with State Lattice Motion Primitives". *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2172–2179. DOI: 10.1109/IROS.2011.6094900
(Cited on page 6)

Pivtoraiko, M., Knepper, R. A., and Kelly, A. (2009). "Differentially Constrained Mobile Robot Motion Planning in State Lattices". *Journal of Field Robotics* 26.3, pp. 308–333. DOI: 10.1002/rob.20285
(Cited on page 6)

Reeds, J. A. and Shepp, L. A. (1990). "Optimal paths for a car that goes both forwards and backwards". *Pacific Journal of Mathematics* 145.2, pp. 367–393. DOI: 10.2140/pjm.1990.145.367
(Cited on page 6)

Reynolds, C. W. (1999). "Steering Behaviors For Autonomous Characters". *Game Developers Conference*, pp. 763–782
(Cited on pages 82, 83, 86)

Riedmiller, M. and Braun, H. (1992). "RPROP - A Fast Adaptive Learning Algorithm". In: *Proceedings of the 7th International Symposium on Computer and Information Sciences*
(Cited on pages 117, 126)

Riedmiller, M., Montemerlo, M., and Dahlkamp, H. (2007). "Learning to Drive a Real Car in 20 Minutes". In: *Frontiers in the Convergence of Bioscience and Information Technologies*. IEEE, pp. 645–650. DOI: 10.1109/FBIT.2007.37
(Cited on page 12)

Rufli, M. and Siegwart, R. (2010). "On the Design Of Deformable Input- / State-Lattice Graphs". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3071–3077. DOI: 10.1109/ROBOT.2010.5509350
(Cited on pages 7, 29)

Sadigh, D., Sastry, S., Seshia, S. A., and Dragan, A. D. (2016). "Planning for Autonomous Cars that Leverages Effects on Human Actions". In: *Proceedings of Robotics: Science and Systems Conference (RSS)*, pp. 239–248. DOI: 10.15607/RSS.2016.XII.029
(Cited on page 12)

Samson, C. and Ait-Abderrahim, K. (1991). "Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2, pp. 1136–1141. DOI: 10.1109/ROBOT.1991.131748
(Cited on page 41)

Santana, E. and Hotz, G. (2016). *Learning a Driving Simulator*. Version 1. arXiv: 1608.01230 [cs.LG]
(Cited on page 12)

Schramm, D., Hiller, M., and Bardini, R. (2014). "Single Track Models". In: *Vehicle Dynamics: Modeling and Simulation*. Springer Berlin Heidelberg, pp. 223–253. DOI: 10.1007/978-3-540-36045-2_10
(Cited on page 38)

Schwarting, W. and Pascheka, P. (2014). "Recursive Conflict Resolution for Cooperative Motion Planning in Dynamic Highway Traffic". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1039–1044. DOI: 10.1109/ITSC.2014.6957825
(Cited on page 17)

Schwesinger, U., Bürki, M., Timpner, J., Rottmann, S., Wolf, L., Paz, L. M., Grimmett, H., Posner, I., Newman, P., Häne, C., Heng, L., Lee, G. H., Sattler, T., Pollefeys, M., Allodi, M., Valenti, F., Mimura, K., Goebelsmann, B., Derendarz, W., Mühlfellner, P., Wonneberger, S., Waldmann, R., Grysczyk, S., Last, C., Brüning, S., Horstmann, S., Bartholomäus, M., Brummer, C., Stellmacher, M., Pucks, F., Nicklas, M., and Siegwart, R. (2016). "Automated Valet Parking and Charging for e-Mobility: Results of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 157–164. DOI: 10.1109/IVS.2016.7535380
(Cited on page 26)

Schwesinger, U., Rufli, M., Furgale, P., and Siegwart, R. (2013). "A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 391–396. DOI:

10.1109/IVS.2013.6629500
(Cited on pages 21, 25, 29, 41, 44, 76)

Schwesinger, U., Siegwart, R., and Furgale, P. (2015). "Fast Collision Detection Through Bounding Volume Hierarchies in Workspace-Time Space for Sampling-Based Motion Planners". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–68. DOI: 10.1109/ICRA.2015.7138981
(Cited on pages 21, 25, 65)

Schwesinger, U., Versari, P., Broggi, A., and Siegwart, R. (2015). "Vision-only Fully Automated Driving in Dynamic Mixed-traffic Scenarios". *it - Information Technology* 57.4, pp. 231–242. DOI: 10.1515/itit-2015-0005
(Cited on pages 14, 21, 25, 29, 33)

Shannon, C. E. (1948). "A Mathematical Theory of Communication". *Bell System Technical Journal* 27.3, pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x
(Cited on page 86)

Silver, D., Bagnell, J. A., and Stentz, A. (2013). "Learning Autonomous Driving Styles and Maneuvers from Expert Demonstration". In: *International Symposium on Experimental Robotics*. Springer International Publishing, pp. 371–386. DOI: 10.1007/978-3-319-00065-7_26
(Cited on pages 11, 12)

Sminchisescu, C. and Welling, M. (2011). "Generalized darting Monte Carlo". *Pattern Recognition* 44.10-11, pp. 2738–2748. DOI: 10.1016/j.patcog.2011.02.006
(Cited on page 112)

Smith, R. *Open Dynamics Engine*. www.ode.org, accessed September 2016
(Cited on pages 68, 73)

Swendsen, R. H. and Wang, J.-S. (1986). "Replica Monte Carlo Simulation of Spin-Glasses". *Physical Review Letters* 57 (21), pp. 2607–2609. DOI: 10.1103/PhysRevLett.57.2607
(Cited on page 109)

Trautman, P. and Krause, A. (2010). "Unfreezing the Robot: Navigation in Dense, Interacting Crowds". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 797–803. DOI: 10.1109/IROS.2010.5654369
(Cited on pages 3, 11, 83, 84, 86)

Trautman, P., Ma, J., Murray, R. M., and Krause, A. (2013). "Robot Navigation in Dense Human Crowds: The Case for Cooperation". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2153–2160. DOI: 10.1109/ICRA.2013.6630866
(Cited on pages 83, 86)

Treuille, A., Cooper, S., and Popović, Z. (2006). "Continuum Crowds". In: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM Press, pp. 1160–1168. DOI: 10.1145/1141911.1142008
(Cited on pages 10, 83)

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). "Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Vol. 1. Association for Computational Linguistics, p. 477. DOI: 10.3115/1687878.1687946
(Cited on pages 115, 135)

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". *Journal of Field Robotics* 25.8, pp. 425–466. DOI: 10.1002/rob.20255
(Cited on pages 6, 15, 30, 53)

Wang, Q., Kulkarni, S. R., and Verdu, S. (2009). "Divergence Estimation for Multidimensional Densities Via *k*-Nearest-Neighbor Distances". *IEEE Transactions on Information Theory* 55.5, pp. 2392–2405. DOI: 10.1109/TIT.2009.2016060
(Cited on page 126)

Werling, M., Kammel, S., Ziegler, J., and Groll, L. (2011). "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds". *The International Journal of Robotics Research* 31.3, pp. 346–359. DOI: 10.1177/0278364911423042
(Cited on pages 9, 15, 16, 22, 23, 30, 34, 53, 67, 71, 76)

Wulfmeier, M. and Ondruska, P. (2016). *Maximum Entropy Deep Inverse Reinforcement Learning*. Version 3. arXiv: 1507.04888 [cs.LG]
(Cited on pages 86, 152)

Zeiler, M. D. (2012). *ADADELTA: An Adaptive Learning Rate Method*. Version 1. arXiv: 1212.5701 [cs.LG]. URL: http://arxiv.org/abs/1212.5701
(Cited on page 113)

Zhang, Y. and Sutton, C. A. (2011). "Quasi-Newton Methods for Markov Chain Monte Carlo". In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., pp. 2393–2401
(Cited on page 108)

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence*. Vol. 3. AAAI Press, pp. 1433–1438
(Cited on pages 12, 19, 85)

Ziebart, B. D., Maas, A. L., Dey, A. K., and Bagnell, J. A. (2008). "Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior". In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. ACM, pp. 322–331. DOI: 10.1145/1409635.1409678
(Cited on page 85)

Ziebart, B. D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J. A., Hebert, M., Dey, A. K., and Srinivasa, S. (2009). "Planning-based Prediction for Pedestrians". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3931–3936. DOI: 10.1109/IROS.2009.5354147
(Cited on page 137)

Ziegler, J. and Stiller, C. (2010). "Fast Collision Checking for Intelligent Vehicle Motion Planning". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 518–522. DOI: 10.1109/IVS.2010.5547976
(Cited on page 67)

Ziegler, J. and Stiller, C. (2009). "Spatiotemporal State Lattices for Fast Trajectory Planning in Dynamic On-Road Driving Scenarios". In: *International Conference on Intelligent Robots and Systems (IROS)*, pp. 1879–1884. DOI: 10.1109/IROS.2009.5354448
(Cited on pages 7, 30)

Zucker, M., Kuffner, J., and Branicky, M. (2007). "Multipartite RRTs for Rapid Replanning in Dynamic Environments". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1603–1609. DOI: 10.1109/ROBOT.2007.363553 (Cited on page 66)

Curriculum Vitae

Ulrich Schwesinger

born September 12, 1984 in Germany

10/2011 – 12/2016	*ETH Zurich, Switzerland* Doctoral studies at the Autonomous Systems Lab of Prof. Roland Siegwart at the Institute of Robotics and Intelligent Systems, Department of Mechanical and Process Engineering; graduated with Dr. sc. ETH Zurich.
10/2010 – 09/2011	*ETH Zurich, Switzerland* Engineer at the Autonomous Systems Lab, ETH Zurich. Participation in the design of the locomotion system of ESA's mars rover for the *ExoMars* mission.
03/2009 – 09/2009	*Robert Bosch GmbH, Leonberg, Germany* Internship in the area of advanced driver assistance systems.
10/2004 – 05/2010	*Karlsruhe Institute of Technology, Germany* Undergraduate and graduate studies. Graduated with diploma in electrical engineering & information technology.
09/1995 – 06/2004	*Theodor-Heuss Gymnasium, Freiburg im Breisgau, Germany* High school education. Graduated with Matura (high school diploma for admission to higher education).

List of Publications

Peer-Reviewed Journals

Bircher, A., Alexis, K., **Schwesinger, U.**, Omari, S., Burri, M., Siegwart, R., (2017). "An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees". *Robotica* 35.6, pp. 1327–1340. DOI: 10.1017/S0263574716000084

Krüsi, P., Bücheler, B., Pomerleau, F., **Schwesinger, U.**, Siegwart, R., Furgale, P., (2015). "Lighting-invariant Adaptive Route Following Using Iterative Closest Point Matching". *Journal of Field Robotics* 32.4, pp. 534–564. DOI: 10.1002/rob.21524

Schwesinger, U., Versari, P., Broggi, A., Siegwart, R., (2015). "Vision-only Fully Automated Driving in Dynamic Mixed-traffic Scenarios". *it - Information Technology* 57.4, pp. 231–242. DOI: 10.1515/itit-2015-0005

Peer-Reviewed Conference Proceedings

Pfeiffer, M., **Schwesinger, U.**, Sommer, H., Galceran, E., Siegwart, R., (2016). "Predicting Actions to Act Predictably: Cooperative Partial Motion Planning with Maximum Entropy Models". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2096–2101. DOI: 10.1109/IROS.2016.7759329

Schwesinger, U., Bürki, M., Timpner, J., Rottmann, S., Wolf, L., Paz, L. M., Grimmett, H., Posner, I., Newman, P., Häne, C., Heng, L., Lee, G. H., Sattler, T., Pollefeys, M., Allodi, M., Valenti, F., Mimura, K., Goebelsmann, B., Derendarz, W., Mühlfellner, P., Wonneberger, S., Waldmann, R., Grysczyk, S., Last, C., Brüning, S., Horstmann, S., Bartholomäus, M., Brummer, C., Stellmacher, M., Pucks, F., Nicklas, M., Siegwart, R., (2016). "Automated Valet Parking and Charging for e-Mobility: Results of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 157–164. DOI: 10.1109/IVS.2016.7535380

Schwesinger, U., Siegwart, R., Furgale, P., (2015). "Fast Collision Detection Through Bounding Volume Hierarchies in Workspace-Time Space for Sampling-Based Motion Planners". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–68. DOI: 10.1109/ICRA.2015.7138981

Furgale, P., **Schwesinger, U.**, Rufli, M., Derendarz, W., Grimmett, H., Mühlfellner, P., Wonneberger, S., Rottmann, J. T. S., Li, B., Schmidt, B., Nguyen, T. N., Cardarelli, E., Cattani, S., Brüning, S., Horstmann, S., Stellmacher, M., Mielenz, H., Köser, K., Beermann, M., Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Iser, R., Triebel, R., Posner, I., Newman, P., Wolf, L., Pollefeys, M., Brosig, S., Effertz, J., Pradalier, C., Siegwart, R., (2013). "Toward Automated Driving in Cities using Close-to-Market Sensors: An Overview of the V-Charge Project". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 809–816. DOI: 10.1109/IVS.2013.6629566

Schwesinger, U., Rufli, M., Furgale, P., Siegwart, R., (2013). "A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 391–396. DOI: 10.1109/IVS.2013.6629500

Schwesinger, U., Pradalier, C., Siegwart, R., (2012). "A Novel Approach for Steeringwheel Synchronization with Velocity/Acceleration Limits and Mechanical Constraints". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5360–5366. DOI: 10.1109/IROS.2012.6385644

Zwirello, L., Janson, M., Ascher, C., **Schwesinger, U.**, Trommer, G. F., Zwick, T., (2010a). "Accuracy Considerations of UWB Localization Systems Dedicated to Large-Scale Applications". In: *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, pp. 1–5. DOI: 10.1109/IPIN.2010.5647481

Zwirello, L., Janson, M., Ascher, C., **Schwesinger, U.**, Trommer, G. F., Zwick, T., (2010b). "Localization in Industrial Halls via Ultra-wideband Signals". In: *Proceedings of the 7th Workshop on Positioning, Navigation and Communication, WPNC'10*. IEEE, pp. 144–149. DOI: 10.1109/WPNC.2010.5649369

Other Publications

Furgale, P., Krüsi, P., Pomerleau, F., **Schwesinger, U.**, Colas, F., Siegwart, R., (2014). "There and Back Again—Dealing with highly-dynamic scenes and long-term change during topological/metric route following". *ICRA14 Workshop on Modelling, Estimation, Perception, and Control of All Terrain Mobile Robots*

Derendarz, W., Iser, R., Mühlfellner, P., Wonneberger, S., Nguyen, T. N., Brosig, S., Effertz, J., Mielenz, H., Mimura, K., Beermann, M., **Schwesinger, U.**, Rufli, M., Furgale, P., Timpner, J., Rottmann, S., Wolf, L., Häne, C., Heng, L., Lee, G., Pollefeys, M., Grimmet, H., Triebel, R., Posner, I., Newman, P., Cardarelli, E., Cattani, S., Bertozzi, M., Brüning, S., Horstmann, S., Stellmacher, M., Siegwart, R., (2013). "Vollautomatisches Laden und Fahren auf öffentlichen Parkplätzen". In: *Internationaler Kongress Elektronik im Fahrzeug*. VDI-Berichte 2188. VDI Wissensforum GmbH

Theses

Schwesinger, U. (2010). "Modellierung eines UWB-Trackingsystems zur Stützung von inertialsensorbasierten Navigationssystemen in Gebäuden". German. (English translation: Modelling of a UWB tracking system to support inertial navigation systems in buildings). Diploma thesis, supervisors: Christian Ascher and Lukasz Zwirello. Karlsruhe Intitute of Technology (KIT), Institute of Systems Optimization and Institute of Radio Frequency Engineering and Electronics

Schwesinger, U. (2009). "Entwicklung und Implementierung neuartiger Konzepte zur ungestützten Fußgängernavigation". German. (English translation: Development and implementation of novel concepts for inertial pedestrian navigation systems). Semester project, supervisor: Christian Ascher. Karlsruhe Intitute of Technology (KIT), Institute of Systems Optimization