Nonholonomic Motion Planning for Automated Vehicles in Dense Scenarios

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik

des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Holger Banzhaf

aus Laichingen

Tag der mündl. Prüfung: 11. Dezember 2019
Erster Gutachter: Prof. Dr.-Ing. J. Marius Zöllner, KIT
Zweiter Gutachter: Prof. Dr.-Ing. Rüdiger Dillmann, KIT

Document template provided by:

Institute of Industrial Information Technology (IIIT)
Karlsruhe Institute of Technology (KIT)
Hertzstraße 16
76187 Karlsruhe

Abstract

Motion planning is one of the crucial components in the software stack of an automated vehicle. It is responsible for the computation of a safe and preferably optimal trajectory from a given start state to a desired goal. While a local solution to this problem is sufficient for highway driving, this thesis focuses on the computation of a global solution, which is typically required to handle unstructured environments or complex maneuvers. Relevant scenarios include dead ends, blocked lanes, or various parking problems that have proven difficult for automated vehicles to solve, particularly when space is tight.

The contributions of this thesis can be grouped into three parts. The first part focuses on steering functions for car-like robots, which play a major role in both search-based and sampling-based motion planning. Within this context, the novel steering function hybrid curvature (HC) steer is introduced that computes smoother paths than the well-known Reeds-Shepp steering function [152] and shorter paths than continuous curvature (CC) steer [53]. Especially in tight environments, HC steer proves to be a powerful tool for the computation of directly executable motion plans with continuous curvature between direction switches. In addition to that, the two novel steering functions continuous curvature rate (CCR) and hybrid curvature rate (HCR) steer are presented that extend the smoothness of both CC and HC steer from curvature to curvature rate continuity. This allows to increase the comfort for passengers as well as the tracking performance of the low-level motion controller.

The second part of this thesis focuses on motion planning under uncertainty aiming to improve the robustness of the motion plans by explicitly considering the localization and control errors of the system. For this purpose, the previously mentioned steering functions are extended to belief space in which every vehicle state is associated with its respective uncertainty. Furthermore, two novel algorithms for probabilistic collision checking are introduced in order to bound the collision probability of the computed vehicle motion.

The third part addresses the problem of slow convergence in sampling-based motion planning if samples are only drawn from a uniform distribution. To overcome this problem, a data-driven approach is presented that utilizes a convolutional neural network to predict a distribution over future vehicle poses given the current environment and the boundary conditions of the planning problem. Samples from this distribution can then be used to bias the motion planner towards promising regions in the state space allowing to improve the planning performance in complex scenarios.

Finally, the proposed methods from all three parts are integrated into the sampling-based motion planner RRT* [93] and its bidirectional extension BiRRT* [86] to demonstrate their benefits in a broad set of challenging environments. The motion planner is not only tested in simulation, but also integrated into a research vehicle proving its effectiveness in real-world applications.

Acknowledgment

This thesis evolved during my time at the automated driving research department of the Robert Bosch GmbH. The following lines are dedicated to the people who made this work possible.

First of all, I would like to thank Professor Zöllner, director of the Technical Cognitive Systems (TKS) group at the Research Center for Information Technology (FZI) in Karlsruhe, for the scientific supervision of this thesis. Thank you for the fruitful discussions, for pointing me in the right directions, and the freedom and trust you gave me to conduct this research. Also, I would like to thank Professor Dillmann, head of the Humanoids and Intelligence Systems Lab at the KIT, for the interest in my work and for serving as the second examiner on the thesis committee.

This work would not have been possible without the support of the Robert Bosch GmbH, my department manager, Arno Schaumann, my group leader, Axel Stamm, and my strategic program leader, Frank Niewels. Many thanks for providing me with the resources and such an inspiring environment. Special thanks also to my Bosch colleagues, in particular to Dennis Nienhüser who supervised me from the company's side. I always appreciated your valuable and sincere feedback, our great conversations on so many different topics, and your advice that helped me to successfully finish this thesis. My thanks also go to my project leader, Steffen Knoop, and team leader, Thomas Schamm, for discussing and supporting my ideas already from the very beginning. Extra thanks to Luigi Palmieri for the inspiring motion planning exchanges and to Maxim Dolgov for valuable discussions and support in the development of the probabilistic collision checking approaches. Special thanks also to Ulrich Baumann and Jan Stellet for reviewing my publications and the insightful conversations far beyond the topic of automated driving. Also, I would like to thank Frank Quedenfeld, my students, the entire TKS group, and my former PhD colleagues, Benjamin Völz, Jan Rhode, Di Feng, and Stefan Jesenski, for the excellent collaboration.

I have been extremely fortunate to be mentored by Franz Fehrenbach, chairman of the supervisory board of Robert Bosch GmbH, and his assistants,

Jörg Klingler and Cornelius Surkamp. It was a real pleasure exchanging ideas with you, looking at problems from novel perspectives, and listening to your advice. Thank you very much for your time and effort.

Finally, I would like to express my greatest thanks to my parents and family for their unconditional support, their open feedback, and the courage and energy they have given me to realize this thesis. It is a privilege to build on such a great foundation.

Karlsruhe, December 2019 Holger Banzhaf

Contents

Nomenclature

Abbreviations

Abbreviaton	Description
BiRRT*	Bidirectional RRT*
BIT*	Batch informed tree
CC	Continuous curvature
CCR	Continuous curvature rate
CNN	Convolutional neural network
CP	Collision probability
CVAE	Conditional variational autoencoder
D	Dubins
DOF	Degree of freedom
EKF	Extended Kalman filter
EKF-MP	Extended Kalman filter for motion planning
FMT*	Fast marching tree
GPS	Global positioning system
HC	Hybrid curvature
HCR	Hybrid curvature rate
KC	Key contribution
LiDAR	Light detection and ranging
LQG	Linear quadratic Gaussian
MC	Monte Carlo
OSE	Orientation-aware space exploration
PDF	Probability density function
PMF	Probability mass function
QP	Quadratic program
ROS	Robot operating system
RQ	Research question
RRBT	Rapidly-exploring random belief tree
RRT	Rapidly-exploring random tree
RRT*	Asymptotically optimal rapidly-exploring random tree
RS	Reeds-Shepp
SH	Sigma hull

Abbreviaton	Description
SLAM	Simultaneous localization and mapping
SST	Stable sparse RRT
TTFS	Time-to-first-solution
UKF	Unscented Kalman filter

Mathematical Notations

Notation	Description
x, y, ...	Scalars
$\mathbf{x}$, $\mathbf{y}$, ...	Vectors
${}_i\mathbf{x}$, ${}_i\mathbf{y}$, ...	Vectors given in coordinate frame i
$\mathbf{A}$, $\mathbf{B}$	Matrices
$\mathbf{I}_n$	Identity matrix of dimension $n \times n$
$f(\bullet)$	Function that returns a scalar
$\mathbf{f}(\bullet)$	Function that returns a vector
$\nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x})$	Jacobian of $\mathbf{f}(\mathbf{x})$
$\arg\min_x \ f(x)$	Argument that minimizes $f(x)$
$\mathrm{diag}(a_1, \ldots, a_n)$	Diagonal matrix with entries $a_1, \ldots, a_n$
$i : j$	Sequence of integers $(i, i+1, \ldots, j)$
$bel(\bullet)$	Belief distribution
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal distribution of random vector $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathbf{x} \sim (\bullet)$	Random vector $\mathbf{x}$ is distributed as
$p(\bullet)$	Probability density function
$\Pr(\bullet)$	Probability measure
$\mathrm{Cov}[\bullet]$	Covariance
$E[\bullet]$	Expected value
$\lvert \bullet \rvert$	Absolute value
$\lVert \bullet \rVert_2$	Euclidean norm
$\mathrm{erf}(\bullet)$	Error function
$\exp(\bullet)$	Natural exponential function
$\mathbb{1}_{(\bullet)}$	Indicator function
$\mathrm{sgn}(\bullet)$	Sign of a variable
$\mathrm{tr}(\bullet)$	Trace of a matrix
$(\bullet)^\mathsf{T}$	Transpose of a variable

1 Introduction

With the significant progress in the field of automated driving over the last decades, the dream of self-driving cars seems to be closer than ever before. While the first experiments with driverless cars were already conducted in 1921 [41] and 1925 [169], the development of this technology has gained massive traction since the 1980s [19, 88]. Especially the DARPA Urban Challenge in 2007 [24] marks the start of numerous research projects [18, 103, 168, 203] that envision level 4/5 autonomy according to the SAE Standard J3016 [185].

One of the main drivers for the development of automated vehicles is Vision Zero [190], namely the reduction of road traffic fatalities from 1.35 million in 2016 [64] to zero. Self-driving cars are expected to contribute to this vision by replacing the human driver with a robotic system. In contrast to humans, robots typically have a shorter reaction time, can keep a 360° view of the environment, do not get tired while driving, and can be programmed to drive carefully. Hence, it is expected that many accidents that were formerly caused by humans can be prevented with automated vehicles in the future.

Besides increasing safety, the goal of this technology is also to raise the comfort of daily commutes and to reduce urban traffic with shared self-driving cars. Furthermore, automated vehicles allow a larger fraction of our society, e.g. children, disabled or elderly people, the access to personal mobility.

Leveraging these advantages requires to solve the problem of automated driving, which is often divided into three parts: (1) the vehicle needs to sense its environment and perceive all relevant objects in the scene, (2) it has to plan a safe and preferably optimal sequence of actions from the current state to a desired goal, and (3) it must execute these actions by controlling its actuators accordingly. This thesis focuses on the second part of this sense-plan-act paradigm [174] and more specifically on the motion planning problem for self-driving cars. Further details are provided in the remainder of this chapter, which is organized as follows: Sec. 1.1 states the problem addressed in this thesis, Sec. 1.2 highlights the state of the art in motion planning, and Sec. 1.3 presents the research questions and the scientific key contributions. The structure of this thesis is finally outlined in Sec. 1.4.

1.1 Problem Statement

Automating the driving task is currently one of the most challenging problems in engineering. Although some companies [18, 103] are already working on level 4 solutions (autonomy in restricted areas [185]), it remains an open question whether automated driving can ever be solved up to level 5 (autonomy without restrictions [185]).

The complexity of the underlying problem mostly stems from two facts: (1) automated vehicles are required to operate in an open world and thus have to make appropriate decisions in all possible situations, and (2) solving the driving task requires the interaction of many software and hardware components, where already small failures in the system can lead to fatal consequences. The second aspect can be observed in Fig. 1.1 that visualizes the dependencies between the different components of an automated vehicle.

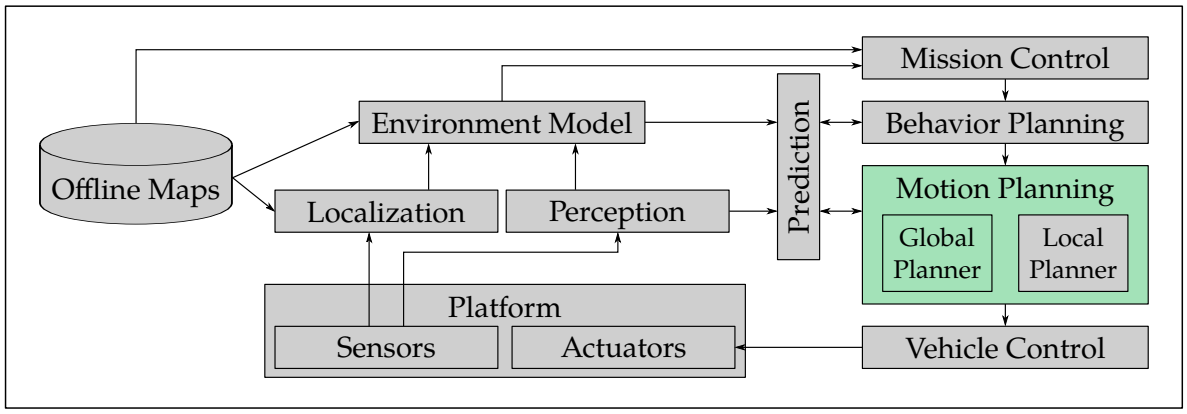


Figure 1.1 Currently dominating system architecture[1] in automated driving. The focus of this thesis is highlighted in green. Based on [16] and adapted from [212], © 2017 IEEE.

Starting with the platform itself, it can be seen in Fig. 1.1 that the information from the on-board sensors is directly forwarded to the localization and the perception module. While the former computes the current vehicle pose using an offline generated map, the latter perceives the world and tries to detect all surrounding obstacles. The output from both modules as well as the map itself is then utilized to generate a model of the environment that includes the static and dynamic obstacles in the scene. This information

[1] With the advances in machine learning, it is possible that some components in Fig. 1.1 will be combined and jointly learned in the future (see e.g. [5]).

along with a prediction of the dynamic obstacles is then forwarded to the navigation stack, which describes all components on the right side in Fig. 1.1. While the mission control computes the route to the desired destination, the behavior planner outputs a sequence of high-level actions (e.g. keep/switch lane, make a turn, park the car) that guide the vehicle on that previously computed route to the goal. On this basis, a collision-free trajectory is optimized by the motion planner and sent to the low-level vehicle controller. The latter finally actuates the vehicle based on the computed trajectory and thereby closes the loop in the signal chain.

As highlighted in Fig. 1.1, the focus of this thesis lies on the motion planning component of an automated vehicle. Here, the **underlying problem** is to find an optimal collision-free trajectory between a start and a goal state that takes into account the vehicle model, the obstacles in the environment, and the comfort requirements of the passengers on board. Due to the complexity of this problem, current automated vehicles are typically equipped with two different motion planners. The **local** planner is generally used for corridor driving on highways and urban streets. In these scenarios, the availability of a lane centerline (obtained e.g. from a precomputed map) allows to solve the motion planning problem locally, i.e. to only compute a trajectory close to the center of the lane. In contrast to that, complex maneuvers or unstructured environments often require a **global** solution to the underlying problem as local approaches typically fail. As visualized in Fig. 1.2 and Fig. 1.3, possible scenarios within this context include dead ends, blocked lanes, or various parking problems.

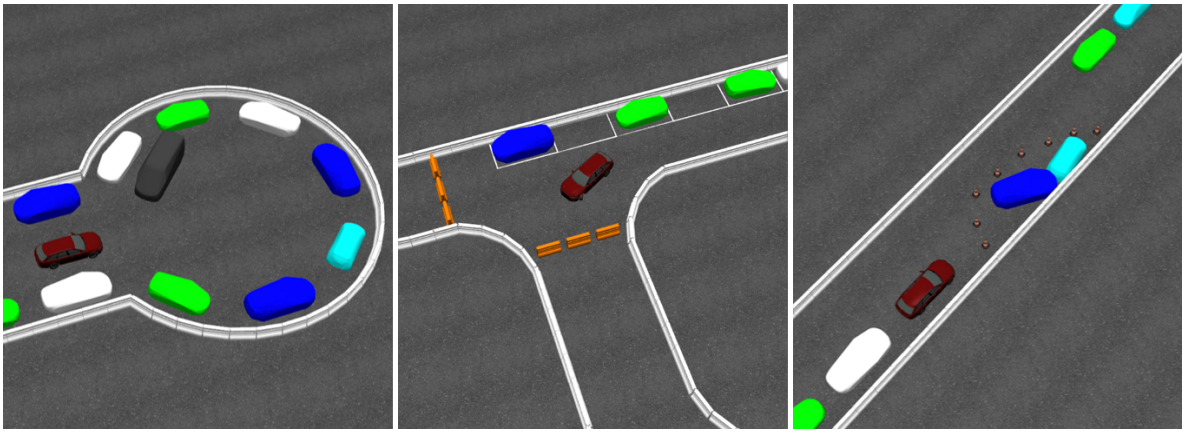


Figure 1.2 Dense scenarios in automated driving (part 1). From left to right, a dead end, a blocked intersection, and a blocked lane. The ego-vehicle is highlighted in red in all three scenarios.

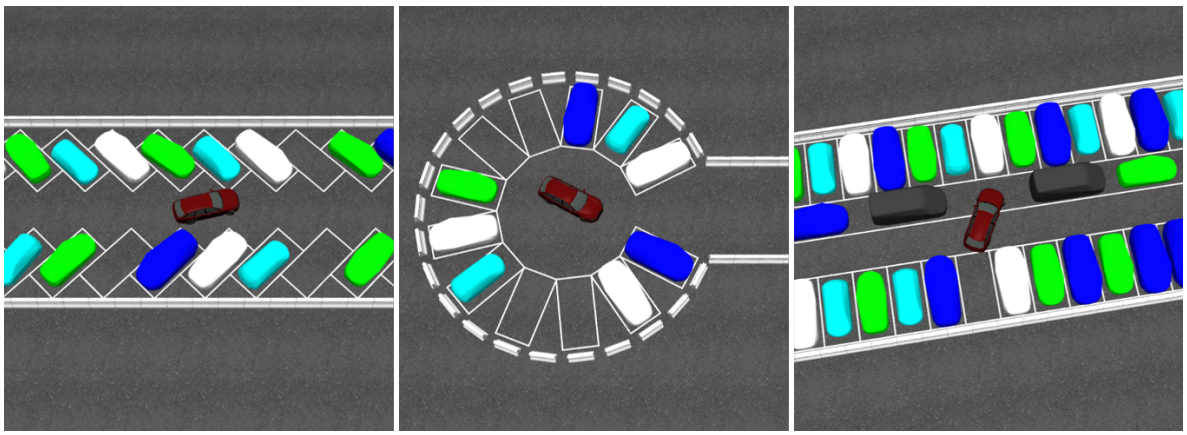


Figure 1.3 Dense scenarios in automated driving (part 2). The images show various parking problems including 45° parking on the left, a circular parking lot in the middle, and a high density parking layout [210] on the right.

While promising approaches already exist for the local motion planning problem [193, 204], computing globally optimal solutions is still an actively researched topic [28, 139, 163]. Especially in dense scenarios with a highly non-convex environment (see Fig. 1.2 and Fig. 1.3), various (complicating) aspects have to be considered as further detailed below on the basis of Fig. 1.4.

Here, the red ego-vehicle is required to make a turn at the illustrated dead end, however, the centerline used for local motion planning is blocked by a parked vehicle. In order to resolve this situation, a multi-point turn must be computed using a global motion planner. Thereby, the following three aspects have to be considered:

(1) The kinematic of the vehicle (front steering with maximum steering angle $< \pi/2$) as well as the nonholonomic constraints (lateral velocity at tires (appox.) zero) prevent the vehicle from making a turn on the spot [112]. For the computation of a kinematically feasible solution, it is therefore required to take into account the constraint on the minimum turning radius of the vehicle.

(2) The computed motion plan must be smooth in order to guarantee a high degree of comfort for passengers on board. This can be achieved by minimizing the jerk (change in acceleration with respect to time) along the trajectory.

(3) Noisy measurements and imperfect models result in uncertainty that must be considered in the motion plan in order to keep the collision probability below a given threshold.

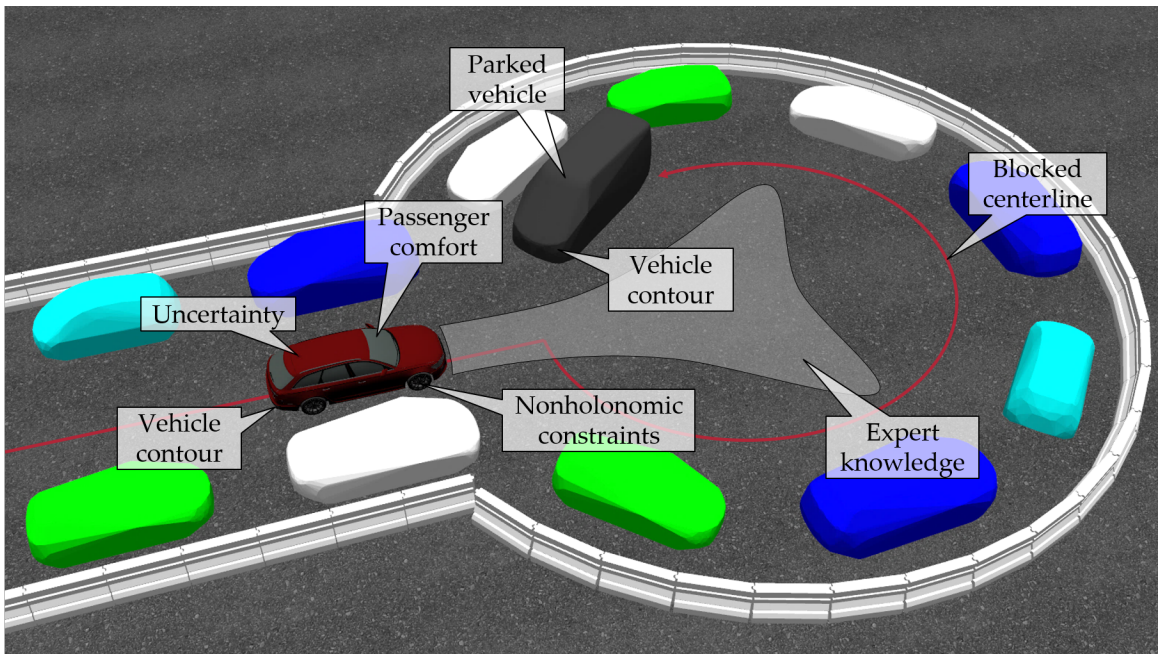


Figure 1.4 Detailed view of the dead end scenario from Fig. 1.2. The red ego-vehicle is required to make a turn, however, the parked black transporter is blocking the lane centerline. As a result, a multi-point turn must be executed to resolve this situation. The annotations highlight the various aspects that either have to be considered or are beneficial to consider in the computation of the motion plan. Adapted from [209].

In addition to those three aspects, it might also be beneficial in certain scenarios to take into account the following two points:

(4) To avoid overly conservative solutions, it might be required to consider the actual contour (and not just a rough bounding box) of both the ego-vehicle and the surrounding obstacles.

(5) Biasing the planner towards promising regions in the state space (expert knowledge in Fig. 1.4) is often necessary to increase the efficiency of the underlying algorithm or sometimes even to find a solution within the limited computation time.

Combining these aspects along with the ones given in [163] allows to derive the requirements listed in Tab. 1.1 for global motion planning. Briefly summarized, the motion planner should find the optimal solution in arbitrary environments with minimal computing effort, and failure should only be reported if no solution exists. The output of the motion planner, namely the motion plan, should be collision-free and smooth in oder to ensure a high

Table 1.1 Requirements for global motion planning based on [163] and on the previously discussed aspects (1)–(5). A short description of the adjectives used below can be found under this table.

The motion **planner** should be:	The motion **plan** should be:
▪ generic[1]	▪ collision-free[5]
▪ complete[2]	▪ smooth[6]
▪ optimal[3]	▪ robust[7]
▪ efficient[4]	▪ true-to-contour[8]

[1] solve arbitrary scenarios
[2] return a solution, if one exists
[3] return minimal cost solution
[4] require little computing power
[5] do not collide with obstacles
[6] guarantee a high degree of comfort
[7] take into account uncertainty
[8] consider actual shape of objects

degree of comfort. Furthermore, it should be robust against uncertainties and also take into account the actual shape of all objects in the scene.

As there is currently no approach that satisfies all of these requirements, the **aim** of this thesis is to advance the state of the art in **global motion planning** for automated vehicles. The focus lies on challenging **dense scenarios** that are particularly difficult to solve for existing generic approaches. Instead of building one monolithic algorithm, several modular components are developed, analyzed, and benchmarked within the presented work in order to (better) satisfy the previously discussed requirements. As a result, this thesis provides the community with several novel concepts that allow to increase the effectiveness of different existing motion planners including sampling-based and search-based approaches.

1.2 State of the Art

Solving the global motion planning problem both in robotics and automated driving has attracted many researchers over the years. Due to the complexity of the underlying problem, there is currently no dominant solution, but rather a large and diverse pool of available methods. Therefore, the aim of this section is to review the state of the art in this field as well as to highlight the capabilities and limitations of the available approaches. In comparison to the state of the art in Chapters 2–4, this one gives a broad overview while the other ones focus on specific (sub)problems arising in several of the algorithms

presented below. Additionally, the reader is referred to the recent surveys [66, 95, 133, 167] that also provide a summary of the local methods for corridor driving (not detailed here).

Based on the taxonomy from [66], the remainder of this section is organized as follows: Sec. 1.2.1 presents the related work in search-based motion planning, Sec. 1.2.2 reviews the available algorithms in sampling-based motion planning, and Sec. 1.2.3 summarizes three other approaches that can often be found in the literature. On this basis, Sec. 1.2.4 finally derives the open problems that are consequently tackled in the following chapters.

1.2.1 Search-Based Planners

The general idea in search-based motion planning is to discretize the state-action space and to solve the resulting problem using classic graph search techniques [48]. For instance, the state lattice approach from [143] discretizes the state space and uses offline computed paths[2] to locally connect the discrete states. These connections must comply with the motion model of the vehicle and can, for instance, be obtained using a steering function (see Ch. 2). To generate a feasible path from start to goal, the constructed graph must then be searched with an appropriate algorithm such as A* [72] or D* Lite [101]. The obtained solution is both resolution-complete and resolution-optimal in the sense that both properties are fulfilled up to the underlying discretization resolution. Therefore, the discretization must be selected carefully (adapted to the problem) in order to avoid a failure of the planner although a solution exists.

The previously described state lattice approach was, e.g., used by the winning team of the DARPA Urban Challenge in unstructured environments [120, 188]. Here, the state lattice was constructed on the basis of the approach described in [73, 144] and then searched with AD* [119]. Another application of motion planning with state lattices in the context of automated parking can be found in [175]. Both applications reveal the major disadvantage of lattice-based approaches: unnatural (jagged) paths that are snapped to the discrete states of the lattice. In order to still guarantee a high degree of comfort, a post-smoothing step is typically required. An advantage of this method, however, is that it can be extended from pure path planning to

[2] In contrast to a trajectory, a path is defined as a sequence of vehicle states without temporal information.

spatiotemporal motion planning. The latter allows to also consider dynamic obstacles (up to a given horizon) as shown in [106, 139, 157].

The search-based planners in [28, 40, 206] follow a different paradigm than the previous approach. Here, the state of the vehicle is kept continuous and not discretized to a lattice. This allows to make node expansions by forward propagation of the current state using (1) the underlying motion model and (2) a discrete set of actions. Then, a feasible path can be computed from start to goal by applying a search algorithm such as Hybrid A* [40]. Although the latter has proven its effectiveness in the DARPA Urban Challenge [40, 127], Hybrid A* has several drawbacks including the fact that it cannot provide any theoretical guarantees with respect to completeness and optimality. In addition to that, a post-smoothing step might be required if only a simple motion model and a coarse set of actions is selected [40].

In summary, search-based techniques are an effective tool for global motion planning in automated driving whenever completeness is a minor problem (e.g. in large open spaces) and suboptimality can be overcome with a post-smoothing algorithm. However, if completeness is essential, as e.g. in dense scenarios, search-based techniques require expert knowledge to come up with a proper discretization of the state-action space. Otherwise, the planner might fail although a feasible solution exists.

1.2.2 Sampling-Based Planners

In addition to search-based approaches, the motion planning problem can also be solved with sampling-based techniques. Here, the idea is to generate a solution incrementally using samples from the underlying state-action space. One of the simplest algorithms in this field is the well-known rapidly-exploring random tree (RRT) [114, 116] that computes a motion plan by growing a tree from start to goal. During this process, RRT only requires a motion model of the system in order to propagate the tree as close as possible to the randomly generated states. The advantage of RRT is its simplicity and its probabilistic completeness, which means that an existing solution is guaranteed to be found in the limit of infinite samples. However, as there is no optimization step involved in the algorithm, the quality of the computed solutions is typically poor making this approach less suitable for automated driving.

The previously mentioned suboptimality can be overcome with the asymptotically optimal rapidly-exploring random tree (RRT*) [93, 94]. This algo-

rithm not only grows a tree from start to goal, but also rewires that tree locally in order to ensure asymptotic convergence to the optimal solution. However, RRT* requires in contrast to RRT a steering function that outputs an optimal connection between two arbitrary states while neglecting obstacles in the environment. Fortunately, such a steering function exists for automated vehicles at low speeds as further detailed in Ch. 2. An extension to RRT* is bidirectional RRT* (BiRRT*) [86, 100] that constructs a tree from start to goal and vice versa at the same time. Compared to the single-tree version, BiRRT* typically improves the overall planning performance leading to, e.g., a higher success rate and a lower cost than RRT* itself. An application of both RRT* and BiRRT* to global motion planning for automated vehicles can be found in [92, 100]. Both approaches, however, only compute discrete curvature paths that do not satisfy a high degree of comfort as typically required in automated driving.

Recent advances in sampling-based motion planning combine both sampling and search techniques to improve the performance of the planner. For instance, the asymptotically optimal fast marching tree (FMT*) [82, 164] first generates a set of samples and then uses graph search combined with a steering function to derive a motion plan from start to goal. Instead of generating a single set of samples at the start of the algorithm, it is proposed in [55] to iteratively execute sampling and graph search on only a batch of samples. The so-called batch informed trees (BIT*) are, however, currently limited to a Euclidean state space, where the output of the steering function reduces to a straight line. Therefore, it remains an open question whether this approach can be efficiently extended to systems with nonholonomic constraints (see e.g. [104, 199]).

While all of the previously described asymptotically optimal motion planners require a steering function, the stable sparse RRT (SST) [118] only relies on forward propagation to compute an asymptotically near-optimal solution. To provide this guarantee, however, SST depends on a Monte Carlo (MC) sampling of the underlying action space resulting in a potentially slow convergence rate.

In conclusion, sampling-based motion planners provide important theoretical guarantees with respect to completeness and optimality that allow to generate high quality solutions in challenging environments. Most of the approaches require a steering function, which is in general hard to derive, but exists for automated vehicles at low velocities (see Ch. 2). Furthermore, a problem-specific sampling distribution is often required to ensure a fast

convergence to the optimal solution [77].

In the context of automated driving, the previously described approaches have seen only few applications. This is mainly due to (1) the randomization in the underlying sampling process, and (2) the potentially slow convergence rate. Both aspects are, however, subject to current research [77, 83, 123, 149] and might result in future motion planners that are more powerful than the currently often preferred search-based approaches from Sec. 1.2.1.

1.2.3 Other Approaches

In addition to the previous approaches, three other classes of motion planners can often be found in the literature: interpolating curve planners, optimization-based planners, and a fairly new direction denoted here as learning-based approaches.

The idea of **interpolating curve planners** in the context of global motion planning is to concatenate a set of predefined geometric primitives, such as straight lines, circular arcs, or splines, in order to solve a specific planning problem. One of the most prominent use-cases for these planners are parallel and perpendicular parking [85, 192], where simple strategies can be formulated to move in/out of a parking bay. Although such techniques are computationally efficient, they can only handle a very limited number of predefined scenarios making them impractical for generic motion planning.

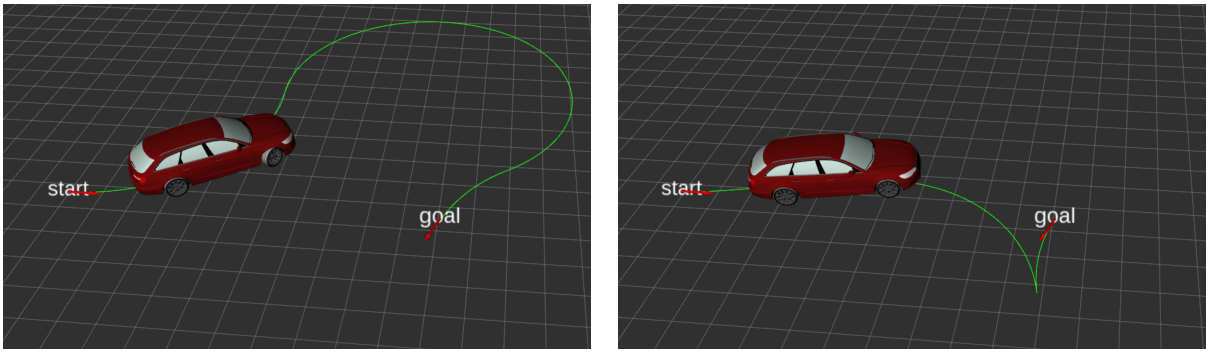
Optimization-based planners solve the motion planning problem typically along a given reference line using numerical optimization [70, 204]. Due to the local nature of these approaches, they are less suitable for global motion planning, but can be used for post-smoothing of the generated path [201].

Recently, a new class of **learning-based approaches** emerged that either learn the driving task end-to-end [5, 15] or replace single building blocks of a classic motion planner with a learned component [34, 77, 183]. As Ch. 4 deals exactly with the latter, an in-depth review of the related work in this field can be found in Sec. 4.1.

1.2.4 Open Problems

On the basis of the presented state of the art, three open problems can be identified:

(1) **Steering functions**: The main idea of a steering function is to solve a simplified motion planning problem by explicitly considering the kine-



(a) Dubins path. **(b)** RS path.

Figure 1.5 Visualization of two exemplary paths computed with the steering functions Dubins [43] and Reeds-Shepp (RS) [152]. While the Dubins path only allows the vehicle to move either forwards or backwards, the RS path also contains direction switches to further optimize the path length. Both solutions are discrete in curvature and consist of straight lines and circular arcs of minimum turning radius.

matic and dynamic constraints of the system while neglecting obstacles in the environment [113]. The output is typically a path or a trajectory connecting two states with a preferably optimal solution (see Fig. 1.5). As discussed above, steering functions are crucial for both search-based [40, 143] and sampling-based [94, 164] motion planners. While many of these approaches rely on the well-known steering functions Dubins [43] and Reeds-Shepp (RS) [152] (see Fig. 1.5), the resulting output is only discrete in curvature requiring often a post-smoothing step. An alternative to RS steer is continuous curvature (CC) steer [53] that outputs a curvature-continuous path. At direction switches, however, CC steer always enforces zero curvature resulting in unnatural solutions especially in tight environments such as encountered in parallel parking. Therefore, a novel steering function is required that closes the gap between RS and CC steer. Additionally, CC steer's bang-bang steering inputs require an infinite steering acceleration, which might violate the physical limits of the actuator. As a result, new approaches must be found that enforce even higher degrees of continuity.

(2) **Planning under uncertainty**: Due to the complexity of planning in belief space, many motion planners neglect the uncertainty of the vehicle motion and the environment model. Especially in safety-critical applications, such as automated driving, it is however essential to compute robust motion plans that keep the underlying collision probability

bounded. Therefore, novel techniques must be found that allow to efficiently take into account the uncertainties of the system in the planner.

(3) **Sampling distributions**: Many sampling-based motion planners guarantee asymptotic optimality, however, the convergence rate often depends strongly on the underlying sampling distribution. Especially in real-time motion planning, fast convergence is essential to derive high quality solutions within the limited cycle time. As a consequence, non-uniform sampling distributions are required that consider the environment, the boundary conditions, and the vehicle model in order to guide the motion planner quickly towards the optimal solution.

All three open problems play a central role in this thesis and are consequently tackled in the following chapters.

1.3 Research Questions and Key Contributions

Based on the requirements from Tab. 1.1 and the previously discussed open problems, this section highlights the research questions (RQs) and the scientific key contributions (KCs) of this thesis[3].

The first RQ addresses the two crucial requirements completeness and smoothness (see Tab. 1.1) and is given as

RQ1 How can smoothness be integrated into global motion planning without losing completeness?

This RQ is motivated by the fact that the motion planner should always find a solution, if one exists, and additionally, guarantee a possibly high degree of comfort (smoothness) along the computed trajectory. RQ1 is addressed on the level of the previously discussed steering functions that not only determine the smoothness of the motion plans, but also the completeness of the planner [170]. The work on this RQ led to three KCs that are given as

KC1.1 Introduction of the novel steering function hybrid curvature (HC) steer that enforces curvature continuity while the vehicle is moving

[3]The KCs have been previously published in [210, 211, 213, 214, 216, 218] out of which [213] received the best paper award at the 2018 IEEE Intelligent Transportation Systems Conference (ITSC) and [214] the third prize best application paper award at the 2018 IEEE Intelligent Vehicles Symposium (IV). Both ITSC and IV are the annual flagship conferences of the IEEE Intelligent Transportation Systems Society.

either forwards or backwards, but allows curvature discontinuities at direction switches [211] (see Sec. 2.3.2). Compared to the state of the art, HC steer computes smoother paths than RS steer [152] and outperforms CC steer [53] with respect to path length (see Sec. 2.3.4). These properties allow to compute motion plans in dense scenarios with less direction switches than CC steer while providing a higher degree of comfort for passengers than RS steer. As HC steer also allows to preserve the completeness of the planner (see Sec. 2.3.2.4), it proves to be an effective tool for motion planning in tight environments (see Sec. 5.3.1 and Sec. 5.3.2). Note that the implementation of HC steer is provided as open source and can be found along with the other state-of-the-art steering functions in [208].

KC1.2 Extension of CC steer [53] to arbitrary start and goal curvatures (see Sec. 2.3.3). This allows now to compute a curvature-continuous shortest path approximation between any two four-dimensional vehicle states given with position, orientation, and curvature. Such a property is, for instance, crucial if the motion plan needs to be replanned while the vehicle is in an arbitrary state with non-zero velocity and curvature.

KC1.3 Introduction of the two novel steering functions continuous curvature rate (CCR) and hybrid curvature rate (HCR) steer that extend both CC and HC steer to curvature rate continuity [214] (see Sec. 2.4.1 and Sec. 2.4.2). Compared to the latter two steering functions, CCR and HCR steer not only allow to limit the maximum curvature and the maximum curvature rate, but also the maximum curvature acceleration along the computed path. Due to this additional constraint, CCR and HCR steer enforce the highest degree of smoothness compared to all other previously mentioned approaches (see Sec. 2.4.3). Thus, more comfort for passengers as well as a better tracking performance of the low-level motion controller can be achieved with those two steering functions.

While KC1.1–KC1.3 assume that the computed vehicle motion can be precisely executed by a low-level controller, this is typically not the case in reality due to e.g. localization and control uncertainty. Taking into account these uncertainties leads to the fact that the vehicle state can no longer be represented by a deterministic value, but must rather be described by a

belief distribution over possible states. Combined with the desire to compute motion plans that also consider the actual shape of the vehicle (see Tab. 1.1), this leads to the second RQ given as

RQ2 How can the full geometric information of the vehicle contour be leveraged in the motion planner while also taking into account the localization and control uncertainty?

The KCs related to RQ2 can be summarized as

KC2.1 Extension of the previously discussed steering functions to belief space by computing a belief distribution along the nominal path of a steering procedure [213] (see Sec. 2.5). This can either be achieved with the MC simulation presented in Sec. 2.5.1 or the extended Kalman filter for motion planning (EKF-MP) [23] described in Sec. 2.5.2. Such a consideration of uncertainties in the steering function lifts the strong assumption of knowing the exact vehicle state at all times.

KC2.2 Introduction of two novel algorithms for probabilistic collision checking. They allow to determine whether the collision probability of a vehicle state exceeds a user-defined threshold considering the actual contour of the vehicle [213, 218] (see Sec. 3.2). This can be used to increase the robustness of the motion plans by bounding the collision probability of the computed vehicle motion (see Sec. 5.3.4).

Due to the complexity of the motion planning problem [27, 153], it is well known [28, 40, 77, 207] that high quality solutions can often only be computed using problem-specific heuristics. Especially in nonholonomic motion planning, the design of such heuristics is still an open problem as they must adapt to the various different environments and take into account the constraints of the system. Therefore, the next RQ is given as

RQ3 How can all information about the planning problem, such as the environment, the boundary conditions, and the system dynamics, be used to bias the planner towards promising regions in the state space?

This RQ is addressed for sampling-based motion planning, where a problem-specific sampling distribution is required to ensure a fast convergence to the optimal solution. The corresponding KC is given as

KC3 Introduction of a convolutional neural network (CNN) that computes a sampling distribution over future vehicle poses given observations of

the environment as well as the vehicle's start and goal state [215, 216] (see Ch. 4). The CNN is trained end-to-end using a dataset recorded in a simulator. Compared to uniform sampling and an A*-based approach [30], the network predicts future vehicle poses more accurately (see Sec. 4.4) leading to an overall better planning performance when integrated into a sampling-based motion planner (see Sec. 5.3.5).

All of the previously mentioned KCs are first benchmarked standalone against their corresponding state of the art and then integrated into RRT* [93] and its bidirectional extension BiRRT* [86]. The performance of the resulting motion planner is evaluated in Ch. 5 using an automated vehicle simulator. The latter has been built up from scratch for this thesis and allows to execute a motion plan in closed loop while perceiving the environment with a simulated light detection and ranging (LiDAR) system. Additionally, the developed motion planner has been successfully integrated into a research vehicle proving its effectiveness in real-world applications.

With the capability to maneuver automated vehicles in extremely tight environments (see Sec. 5.3), the question arises whether future vehicles can be parked more efficiently. The corresponding RQ is given as

RQ4 How can the potential of the developed motion planner be leveraged to park automated vehicles more efficiently?

In fact, parking is one of the key problems in urban areas. It not only requires massive land to store the vehicles [33], but also increases the traffic due to a lack of parking space [173]. The KC related to this problem is given as

KC4 Introduction of a novel parking layout that not only stores vehicles perpendicular to the driveway, but also in double-ended queues on one side of it [210] (see Fig. 1.3 on the right). This parking layout can be integrated into existing parking lots and increases their capacity by up to 25 %. Furthermore, an algorithm is introduced for the coordination of the vehicles that keeps the per-car shunting operations bounded during parking. Combining both aspects allows a more efficient use of parking space while limiting unnecessary vehicle movement.

This KC is not further detailed in the following chapters as they focus exclusively on the motion planning problem. However, additional information on KC4 can be found in the corresponding publication [210].

1.4 Outline

This thesis consists of six chapters that are organized according to Fig. 1.6.

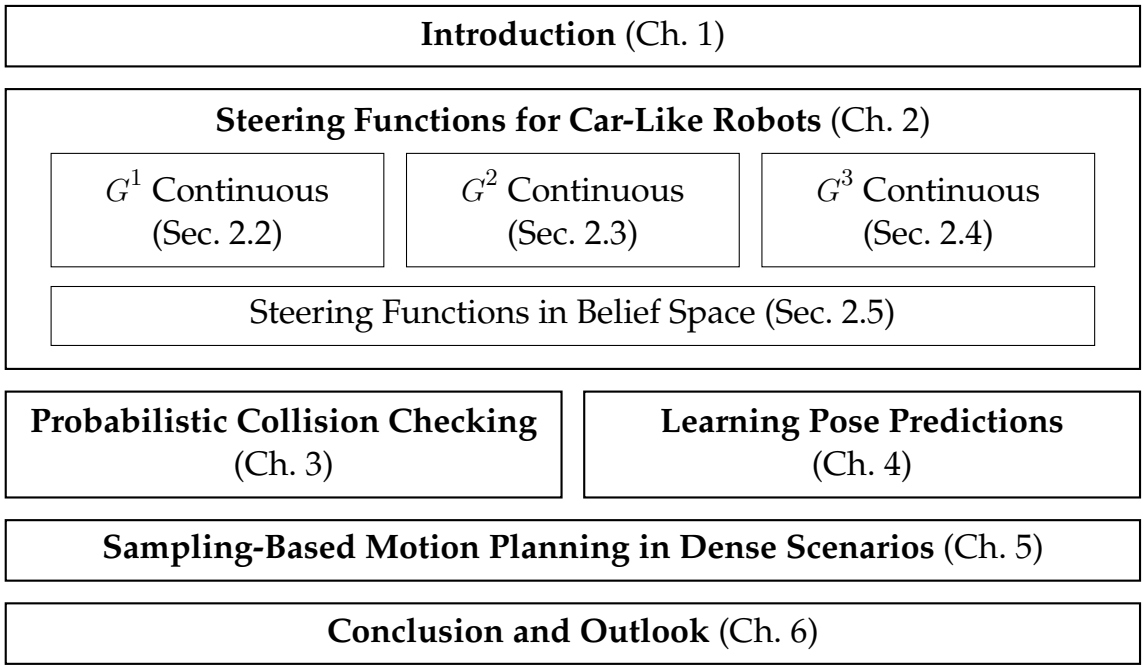


Figure 1.6 Outline of this thesis.

After the introduction, Ch. 2 addresses the previously described steering problem for car-like robots. The presented steering functions in Sections 2.2–2.4 increase successively the complexity of the underlying motion model resulting in solutions with a higher degree of smoothness. These steering functions are then extended to belief space in Sec. 2.5, where every state of the vehicle is associated with its uncertainty originating from localization and control errors.

Given such a belief of the vehicle state, the natural question arises of how to assess whether the corresponding collision probability with the environment remains below a user-defined threshold. This question is addressed in Ch. 3 with the novel concept of beliefprints that can be used for probabilistic collision checking. Furthermore, two algorithms are described that compute the vehicle's beliefprint under the assumption of a Gaussian belief distribution.

Ch. 4 introduces a data-driven approach for the computation of a non-uniform sampling distribution. Here, a CNN is presented that predicts a distribution over future vehicle poses given a start and goal state as well as observations of the environment. Interfacing the CNN with a sampling-based motion planner allows to bias the planner towards promising regions

in the state space and thus improve its overall performance in challenging scenarios.

Building on the results of the previous chapters, the goal in Ch. 5 is to solve the motion planning problem in various dense scenarios. This is accomplished using the sampling-based motion planner RRT* and its bidirectional extension BiRRT* along with the algorithms from Chapters 2–4. The presented experimental results highlight the effectiveness of the developed methods in a broad set of challenging environments.

Ch. 6 finally concludes this thesis and gives an outlook on possible future research directions.

2 Steering Functions for Car-Like Robots

Steering functions for car-like robots are one of the essential concepts in nonholonomic motion planning. The basic idea is to solve a simplified version of the planning problem by explicitly considering the kinematic and dynamic constraints of the robot while neglecting obstacles in the environment [113]. The resulting output is a feasible and preferably optimal connection between a start and a goal state as visualized in Fig. 2.1.

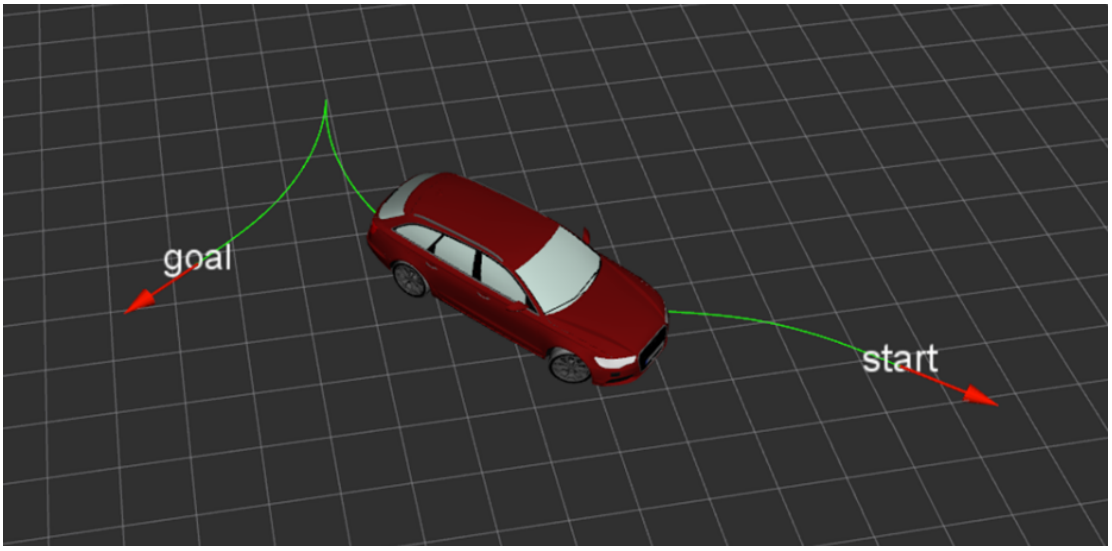
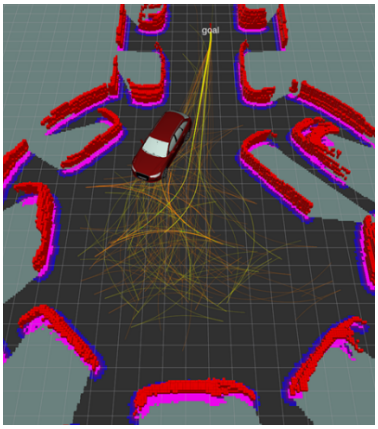


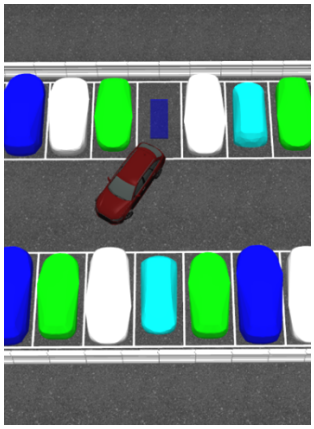
Figure 2.1 Exemplary solution of a steering procedure connecting a start and goal state with a feasible path. Reprinted from [214], © 2018 IEEE.

Over the years, four different use cases have emerged in the literature that either rely or benefit from the solution of a steering function. One of the most prominent applications are sampling-based motion planners such as the asymptotically optimal rapidly-exploring random tree (RRT*) [93]. The latter, for example, does not only rely on a steering function to build a tree from start to goal, as illustrated in Fig. 2.2(a), but also to rewire the tree locally for asymptotic convergence to the optimal solution. The second

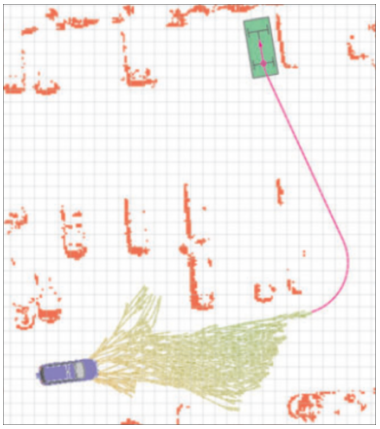
application uses the steering function to generate point-to-point motion toward a goal state, i.e. to derive the inputs from the current state to a desired terminal state. As visualized in Fig. 2.2(b), this can be used in parking and inductive charging, where a precise positioning at a given target pose is required. Equally important are steering functions in the context of search-based planners. Here, the steering function can be used to reduce the search effort by directly connecting the best vertex in the graph with the goal [40] (see Fig. 2.2(c)) and also to offline generate the directed graph in state lattice planning [143]. Lastly, numerical optimization-based planners [202, 204]



(a) Sampling-based motion planning.



(b) Goal controller for exact positioning.



(c) Goal extension in search algorithms [40].

Figure 2.2 Use cases for steering functions in the context of automated driving. (a)–(b) Reprinted from [209]. (c) Reprinted with permission from [40], © 2010 Dmitri Dolgov et al.

require a good initial guess for a fast convergence to the optimal solution. Such a guess can be provided by a steering function, which then has to be modified by the optimization algorithm to also take into account the surrounding obstacles.

As previously described, a steering function solves a relaxed motion planning problem. The corresponding mathematical formulation for a general dynamic system is given as

$$\underset{\check{\mathcal{X}},\check{\mathcal{U}}}{\arg\min} \quad J(\check{\mathcal{X}}, \check{\mathcal{U}}) \tag{2.1a}$$

$$\text{s.t.} \quad \check{\mathbf{x}}_0 = \check{\mathbf{x}}_s, \tag{2.1b}$$

$$\check{\mathbf{x}}_{\bar{N}} = \check{\mathbf{x}}_g, \tag{2.1c}$$

$$\check{\mathbf{x}}_{k+1} = \check{\mathbf{f}}(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k), \tag{2.1d}$$

$$\check{\mathbf{x}}_{\text{min}} \leq \check{\mathbf{x}}_{k+1} \leq \check{\mathbf{x}}_{\text{max}}, \tag{2.1e}$$
$$\check{\mathbf{u}}_{\text{min}} \leq \check{\mathbf{u}}_{k} \leq \check{\mathbf{u}}_{\text{max}}, \quad k = 0 : \bar{N} - 1, \tag{2.1f}$$

where $\check{\mathbf{x}}_k \in \mathbb{R}^n$ describes the state of the system, $\check{\mathbf{u}}_k \in \mathbb{R}^m$ the input, and $\check{\mathbf{f}}(\bullet)$ the respective motion model. The start state is denoted by $\check{\mathbf{x}}_s$, the goal state by $\check{\mathbf{x}}_g$, and the state and input constraints of the vehicle by $\check{\mathbf{x}}_{\text{min}}$, $\check{\mathbf{x}}_{\text{max}}$, $\check{\mathbf{u}}_{\text{min}}$, $\check{\mathbf{u}}_{\text{max}}$. The user-defined objective function is given by $J(\bullet)$ and maps the $\bar{N}+1$ states $\check{\mathcal{X}} = \langle \check{\mathbf{x}}_0, \ldots, \check{\mathbf{x}}_{\bar{N}} \rangle$ and the corresponding inputs $\check{\mathcal{U}} = \langle \check{\mathbf{u}}_0, \ldots, \check{\mathbf{u}}_{\bar{N}-1} \rangle$ to a scalar cost.

Due to the complexity of solving (2.1) for arbitrary objective functions and vehicle models, three simplifications are often made in automated driving. First, the problem can be simplified by decoupling path planning (spatial dimension) from velocity profiling (temporal dimension), which is also known as path-velocity decomposition [91]. The major challenge of this decoupled problem lies in the computation of a smooth path that takes into account the nonlinearity as well as the constraints of the vehicle.

Second, the complexity of the vehicle dynamics in (2.1d) can be reduced to a kinematic motion model if the vehicle only moves at low longitudinal velocities (e.g. less than $5\,\text{m}\,\text{s}^{-1}$) [146, 151]. This is often fulfilled in challenging tight scenarios, where possible maneuvers might also contain switches in the driving direction.

Third, the minimization objective in (2.1a) is commonly restricted to functions that facilitate the optimization. For instance, a common approach in low-speed maneuvering is to minimize path length (shortest path problem) as comfort requirements are typically dominated by the desire for a low execution time. To still guarantee a certain level of smoothness, the state and input constraints in (2.1e)–(2.1f) can be adjusted accordingly. Within this context, the definition of smoothness is adapted from [6, 10, 22], where geometric continuity of planar curves is expressed as

- G^1 continuity: a path with continuous orientation.
- G^2 continuity: a path with continuous curvature.
- G^3 continuity: a path with continuous curvature rate.
- G^4 continuity: a path with continuous curvature acceleration.
- G^k continuity: a path with geometric continuity of order k.

Here, the terms curvature rate and curvature acceleration denote the first and second derivative of curvature with respect to arc length. Note that the provided definitions of geometric continuity can be directly linked to the smoothness of the steering input using the flatness property of car-like robots [22, 50]. For instance, it is shown in [22] that a G^2 continuous path has a C^0 continuous steering profile.

In general, higher degrees of smoothness positively influence the driving comfort and the tracking performance of the motion controller [132, 172, 201] while decreasing the mechanical stress on the steering system. However, computing smoother paths might also increase the computation time, which is typically limited by the real-time constraint of the motion planner.

In this regard, the following sections focus on fast evaluations of (2.1) for car-like robots, where the vehicle is not only allowed to drive forwards, but also forwards and backwards. Sec. 2.1 details the state of the art, and Sec. 2.2, Sec. 2.3, and Sec. 2.4 present three classes of steering functions ranging from G^1 to G^3 continuity. An extension to belief space as required for probabilistic motion planning is presented in Sec. 2.5, and a concluding summary is finally given in Sec. 2.6. Note that parts of this chapter have previously been published in [208, 209, 211, 213, 214].

2.1 State of the Art

Solving the steering problem in (2.1) has attracted many researchers since the 1950s [43] and still remains an active field of research [29, 57, 132]. Among the various different approaches in the robotics and control literature, the four most prominent ones are highlighted in this section including non-parametric as well as parametric optimization, geometric primitives, and sinusoidal inputs.

Non-parametric optimization Numerical optimization with no assumptions on the underlying solution is one of the most intuitive ways to solve the steering problem in (2.1). Here, the entire complexity of the problem, such as the nonlinear system dynamics, is forwarded to the optimizer that either computes a locally optimal solution or fails due to the non-convex nature of the problem. Such an approach is, for instance, realized in [154] by combining the Hamilton-Jacobi framework with numerical optimization. In contrast to that, sequential quadratic programming is applied in [195] by iteratively converting (2.1) into a quadratic program (QP) (quadratic cost function with

linear constraints). A state-of-the art optimizer is then used to solve the QPs until convergence. A direct solution to the nonlinear program is computed in [156] using the Levenberg-Marquard algorithm. This approach, however, requires a relaxation of the hard constraints and thus might result in solutions that violate the physical limits of the robot. All of these approaches have in common that they typically require a high computational cost as well as a good initial guess in order to quickly converge to the optimal solution. A possible way to address these drawbacks is to simplify the optimization problem by explicitly constraining the solution to a given set of parametric curves.

Parametric optimization Such an approach is presented in [96], where the parameters of a polynomial spiral [39] are numerically optimized using a Lagrange method. It is shown that efficient solutions can be computed that satisfy the boundary conditions in (2.1b)–(2.1c). However, the presented method does not explicitly constrain the states and inputs, as required in (2.1e)–(2.1f), and therefore might compute paths that violate the actuator limits. Also, the solutions only allow the vehicle to move either forwards or backwards and do not consider any switching points, which are often necessary for various maneuvering tasks. Therefore, one of the applications of this method is on-street driving as shown in [125]. The problem of a potential constraint violation, as mentioned before, is tackled in [46]. Here, a post-processing step is conducted that enforces the vehicle constraints at the potential risk of violating the terminal boundary condition in (2.1c). Another approach in the domain of on-street driving is presented in [193], where fifth-order polynomials are used to steer the vehicle to terminal manifolds. Similar to [96], the vehicle constraints are not explicitly considered in the generation of the trajectories, but must be verified in a subsequent step. Satisfying theses constraints along the entire path, as shown in the context of B-spline fitting in [90], might significantly increase the complexity of the optimization problem leading to a potential loss in efficiency.

Geometric primitives The insight that path planning is mostly a geometric problem has raised the question whether (2.1) can be solved by concatenating a set of geometric primitives. Indeed, Dubins proves in [43] that the shortest path for a car-like robot with bounded curvature and constrained driving direction (either forwards or backwards) consists of straight lines and circular arcs of maximum curvature. Moreover, this path can be computed very efficiently on the basis of the analytic expressions presented in [43].

Building on these results, Reeds and Shepp achieve another breakthrough in [152] by deriving the shortest path for a vehicle that is allowed to move both forwards and backwards and not just in one direction as in [43]. The corresponding solution can still be computed analytically and thus results in an extremely efficient steering function. The only remaining disadvantage is that the computed paths are discontinuous in curvature. In order to improve the continuity of these paths, the set of geometric primitives can be extended with curves that allow a smooth transition of the vehicle from straight lines to circular arcs. Within this context, possible candidates are splines, such as B-splines [102], Bézier curves [126], or the various η-splines [60, 141, 142], as well as polynomial spirals [39] like clothoids [172] or cubic spirals [89]. Compared to splines, polynomial spirals are often preferred as they describe the curvature and its derivatives by polynomials that can be easily constrained to the steering limits of the vehicle. For example, CC steer [53] extends the geometric set of primitives with clothoids and by doing so, extends the RS paths to curvature continuity. The insight that CC steer has certain drawbacks in tight environments as well as its limitation to zero curvature at both ends of the path are the two main motivations behind the contributions in Sec. 2.3.2 and Sec. 2.3.3.

Raising the continuity to even higher dimensions is addressed in [57, 132, 136]. For instance, [57] enforces G^3 continuity using a nonlinear parametric curve obtained from feedback control. The presented approach, however, does not allow to constrain the maximum curvature rate nor the maximum curvature acceleration to arbitrary values. Hence, a careful design of the velocity profile is required in the execution phase such that the dynamic actuator limits of the robot are not violated. Furthermore, it remains unclear how elementary paths [159, 160] can be built into this method. In contrast to that, the idea in [132] is to compute G^3 continuous paths that enforce hard constraints on the maximum steering angle as well as on its first and second derivative. While this idea is similar to the one in Sec. 2.4 ([132] and the underlying paper of Sec. 2.4 [214] appeared at about the same time), both approaches differ in the realization of the G^3 continuous transitions as well as of the elementary paths [131, 132]. Unlike the previous methods, which integrate the enhanced smoothness directly into the generation of the path, [136] proposes to only locally smooth a Dubins path with an infinitely differentiable curve. However, it remains unanswered whether such a smoothing procedure can always be conducted and what the computational burden of the numerical operations is.

Sinusoidal inputs Especially in the 1980s and 1990s, it was discovered that sinusoidal inputs have the capability to (sub)optimally steer a nonholonomic system to its goal. Much of the work in this field goes back to [21]. Here, it is shown that for a certain class of dynamic systems, the optimal solution of the steering problem with respect to the minimization of squared effort consists of sinusoidal inputs. Building on these results, a suboptimal stepwise approach is presented in [129] that steers a car-like robot to a desired goal. The sequential nature of this algorithm is then removed in [187] by applying a linear combination of sinusoidal inputs. A similar idea is realized in [49] that constructs a Fourier series whose coefficients have to be determined numerically. The two major drawbacks of these approaches are that the computed paths might be highly suboptimal [129, 170] and that the constraints in (2.1e)–(2.1f) are not explicitly taken into account. As previously mentioned, this might lead to a violation of the actuator limits leading to a potential deviation from the planned path when executed in closed loop.

Out of the four different approaches presented above, the following sections focus on the method using geometric primitives. The reasons for this are threefold: (1) the concatenation of geometric primitives can be implemented very efficiently while at the same time guaranteeing robustness (solutions can always be found), (2) this method allows to compute high quality paths by explicitly minimizing the path length of the steering procedure, and (3) geometric primitives make it possible to enforce the vehicle constraints along the entire path by constraining the curvature profile accordingly. These aspects along with the foundations in this field of research are briefly highlighted in the following. Furthermore, the corresponding drawbacks of the existing methods are outlined along the text and afterwards addressed by the contributions in Sections 2.3–2.5.

2.2 G^1 Continuous Steering Functions

This section reviews the pioneering works of Dubins in Sec. 2.2.1 and Reeds and Shepp in Sec. 2.2.2. Most of the publications in this field are based on those two results as they came up with novel ways to solve (2.1) optimally for a car-like robot.

The derived solution in both cases describes a G^1 continuous path with a discrete curvature profile. Such a path is represented by a sequence of vehicle states $\check{\mathbf{x}}_k = \begin{pmatrix} \check{x}_k & \check{y}_k & \check{\theta}_k \end{pmatrix}^\mathsf{T}$, where the position of the rear axle's center is given

by $\begin{pmatrix}\check{x}_k & \check{y}_k\end{pmatrix}$, and the heading angle of the vehicle by $\check{\theta}_k$. The corresponding inputs are defined as $\check{\mathbf{u}}_k = \begin{pmatrix}\Delta\check{s}_k & \check{\kappa}_k\end{pmatrix}^\mathsf{T}$, where $\Delta\check{s}_k$ denotes the signed arc length and $\check{\kappa}_k$ the curvature of the path at $\begin{pmatrix}\check{x}_k & \check{y}_k\end{pmatrix}$.

2.2.1 Dubins Steer

Back in 1957, Dubins discovered a novel way to compute the shortest path for a particle moving in the plane with a bounded turning radius [43]. It turns out that the underlying system, which is nowadays referred to as Dubins car, corresponds to the simplest version of a car-like robot [115]. This section briefly highlights the ideas of the Dubins steering function including the motion model in Sec. 2.2.1.1, the so-called Dubins turn in Sec. 2.2.1.2, and the construction of a Dubins path in Sec. 2.2.1.3. Notice that the term Dubins steer is used in the following to refer to the steering function itself whose output is a Dubins path between two given states.

2.2.1.1 Dubins Car

A vehicle that only moves forwards or backwards on a path with bounded curvature is called a Dubins car. Its equation of motion can be formulated similar to (2.1d) as

$$\check{\mathbf{x}}_{k+1} = \begin{pmatrix}\check{x}_{k+1}\\ \check{y}_{k+1}\\ \check{\theta}_{k+1}\end{pmatrix} = \begin{pmatrix}\check{x}_k\\ \check{y}_k\\ \check{\theta}_k\end{pmatrix} + \int_0^{\Delta\check{s}_k} \begin{pmatrix}\cos(\check{\theta}_{k+1}(s))\\ \sin(\check{\theta}_{k+1}(s))\\ \check{\kappa}_k\end{pmatrix} \mathrm{d}s. \tag{2.2}$$

Note that throughout this thesis, the equations of motion are given in discrete form, which facilitates the implementation on a computing hardware.

The solution of (2.2) either describes a motion on a straight line ($\check{\kappa}_k = 0$) or on a circular arc ($\check{\kappa}_k \neq 0$) with radius $1/\check{\kappa}_k$. Both cases can be described analytically as shown in Sec. A.1.1 and Sec. A.1.2. The corresponding motion constraints, which take into account the selected driving direction and the maximum steering angle, are given as

$$\check{\mathbf{u}}_{\min} = \begin{pmatrix}0 & -\kappa_{\max}\end{pmatrix}^\mathsf{T}, \tag{2.3a}$$

$$\check{\mathbf{u}}_{\max} = \begin{pmatrix}\infty & \kappa_{\max}\end{pmatrix}^\mathsf{T}, \tag{2.3b}$$

where $\kappa_{\max}$ denotes the maximum curvature of the vehicle. Notice that (2.3) is given for forwards driving and has to be adjusted accordingly for going in reverse.

2.2.1.2 Dubins Turn

The insight from [43] that the shortest path of the Dubins car only consists of straight lines and circular arcs of maximum curvature motivates the introduction of the so-called Dubins turn. Such a turn is visualized in Fig. 2.3 and describes a robot motion on a circle that starts in $\check{\mathbf{x}}_s$ and terminates in $\check{\mathbf{x}}_g$. The angular change between both states is called deflection and is denoted by $\delta \in [0, 2\pi)$.

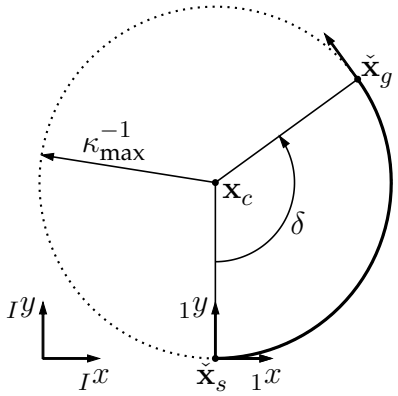


Figure 2.3 Visualization of a Dubins turn for a forward motion to the left. The start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ are connected with a circular arc of maximum curvature.

For the fast computation of the Dubins path in the next section, it is required to correlate all variables shown in Fig. 2.3. To do this, two additional parameters have to be introduced, namely the gear of the vehicle $g \in \{-1, 1\}$ (backwards, forwards) at the start state $\check{\mathbf{x}}_s$ and the desired turning direction $t \in \{-1, 1\}$ (right, left). The center of a Dubins turn $\mathbf{x}_c$ can now be computed as

$$ {}_1\mathbf{x}_c = \begin{pmatrix} {}_1x_c \\ {}_1y_c \end{pmatrix} = \begin{pmatrix} 0 \\ t\kappa_{\max}^{-1} \end{pmatrix}, \tag{2.4} $$

where ${}_1(\bullet)$ denotes a variable given in the local frame 1 (see Fig. 2.3). The output of this equation can then be used along with the deflection δ to compute the goal state $\check{\mathbf{x}}_g$ as

$$ {}_1\check{\mathbf{x}}_g = \begin{pmatrix} {}_1\check{x}_g \\ {}_1\check{y}_g \\ {}_1\check{\theta}_g \end{pmatrix} = \begin{pmatrix} {}_1x_c + t\kappa_{\max}^{-1}\sin(gt\delta) \\ {}_1y_c - t\kappa_{\max}^{-1}\cos(gt\delta) \\ gt\delta \end{pmatrix}. \tag{2.5} $$

While (2.4)–(2.5) are given in the local frame 1, a transformation to the inertial frame I (see Fig. 2.3) can be conducted by

$$ {}_I\check{\mathbf{x}}_{(\bullet)} = \begin{pmatrix} {}_I\check{x}_s \\ {}_I\check{y}_s \\ {}_I\check{\theta}_s \end{pmatrix} + \begin{pmatrix} \cos({}_I\check{\theta}_s) & -\sin({}_I\check{\theta}_s) & 0 \\ \sin({}_I\check{\theta}_s) & \cos({}_I\check{\theta}_s) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}_1\check{x}_{(\bullet)} \\ {}_1\check{y}_{(\bullet)} \\ {}_1\check{\theta}_{(\bullet)} \end{pmatrix}, \tag{2.6} $$

where ${}_I(\bullet)$ denotes a variable given in the inertial frame I. Notice that in Fig. 2.3, both the inertial frame I as well as the local frame 1 are aligned, which might not necessarily be the case in practice.

Lastly, the arc length $l(\delta)$ of a Dubins turn can be computed by

$$ l(\delta) = \kappa_{\text{max}}^{-1}\delta. \tag{2.7} $$

2.2.1.3 Dubins Path

The aim in this section is to outline the computation of a Dubins path and to briefly discuss its strengths and weaknesses. As mentioned before, such a path minimizes the length between two arbitrary states while satisfying the motion constraints of the Dubins car. The underlying objective function is given in the form of (2.1a) as

$$ J(\check{\mathcal{U}}) = \sum_{k=0}^{\bar{N}-1} \left|\Delta\check{s}_k\right|, \tag{2.8} $$

where $\Delta\check{s}_k$ describes the signed arc length of the various segments required to steer the vehicle to the goal. Without loss of generality, it is assumed in the following that the Dubins car only moves forwards.

In the first step of the computations, two circles of maximum curvature, but different turning directions, have to be fitted to the given start state $\check{\mathbf{x}}_s$ and goal state $\check{\mathbf{x}}_g$. This can be seen in Fig. 2.4, where $t_{i=1:4}$ either describes a turn to the right (-1) or to the left $(+1)$. Note that the center of the visualized circles $\mathbf{x}_{c,i=1:4}$ can be obtained using (2.4).

Next, the obtained circles have to be connected with up to six candidate paths out of which one describes the shortest connection [43]. These candidates are constructed using tangent conditions and the Dubins families listed in Tab. 2.1. It can be seen that only two families are required for the description of the shortest path. Note, however, that both turning directions at the start and goal state have to be considered, which yields six candidate

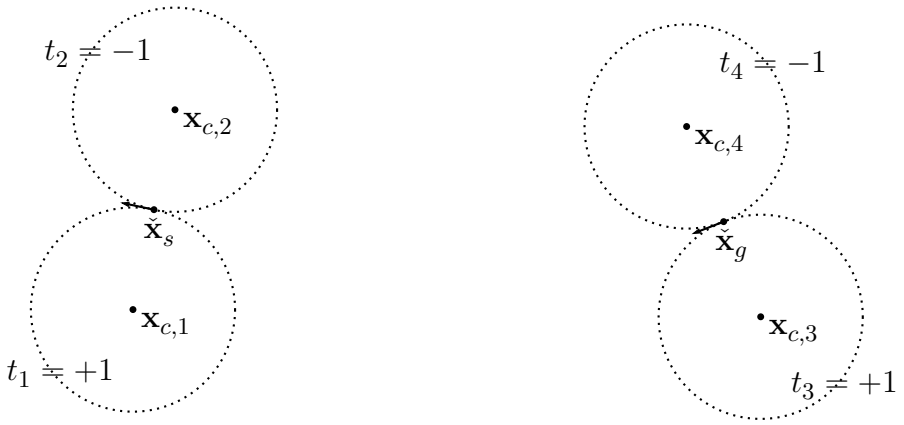


Figure 2.4 Two circles of maximum curvature, but different turning directions, fitted to the start and goal state.

Table 2.1 Dubins families, where C describes a circle of maximum curvature and S a straight line.

Dubins families
CSC
CCC

paths as mentioned above. For instance, the family CSC denotes a path that starts on a circle C, moves on to a straight line S, and ends on a circle C again as illustrated in Fig. 2.5. Furthermore, it has to be noted that the families CS, SC, C, and S are special cases of CSC and therefore not listed separately in Tab. 2.1.

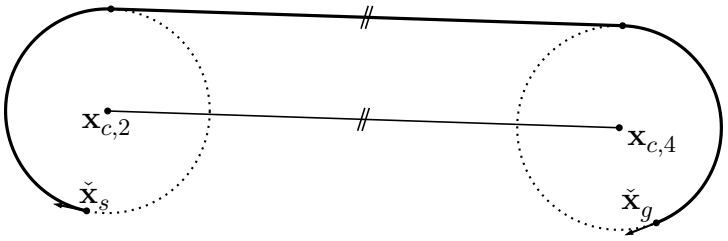


Figure 2.5 Extension of Fig. 2.4 with a Dubins candidate path based on the family CSC.

Every Dubins family comes with a set of existence conditions that specify under which circumstances a solution can be computed with the respective primitives. For instance, the corresponding conditions for the visualized candidate path in Fig. 2.5, where the tangent runs parallel to the line connecting

both circle centers, are given as

$$t_i \cdot t_j \overset{!}{=} +1, \tag{2.9a}$$

$$\left\| \mathbf{x}_{c,j} - \mathbf{x}_{c,i} \right\|_2 \overset{!}{\geq} 0, \tag{2.9b}$$

where $i \in \{1, 2\}$, $j \in \{3, 4\}$, and $\|\bullet\|_2$ measures the Euclidean distance between both circle centers. These two equations imply that the turning direction of the two Dubins turns must be the same while there is no constraint on the distance between them. Note that the corner case $\left\| \mathbf{x}_{c,j} - \mathbf{x}_{c,i} \right\|_2 = 0$ describes a motion on a single turn. The fact that (2.9) can always be fulfilled given arbitrary start and goal states already shows that Dubins steer is complete in the sense that a solution can always be found.

Having computed all candidate paths according to Tab. 2.1, the shortest one including the inputs for the Dubins car can finally be obtained by selecting the path with the minimal length [43]. Based on the implementation from [179], a visualization of such a shortest path connection is displayed in Fig. 2.6.

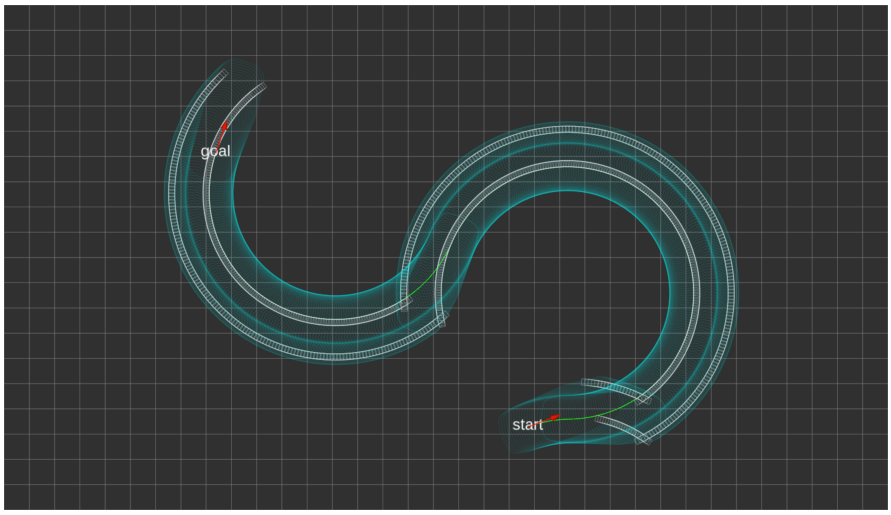


Figure 2.6 Visualization of the shortest path for a vehicle that only moves forwards. The front wheels of the vehicle are displayed in white, its contour in cyan, and the path is colored green.

It can be seen that the illustrated path is G^1 continuous as the vehicle's orientation changes smoothly between start and goal. However, the track of the front wheels depicts the curvature discontinuities at the transitions between circles. This has to be taken into account when computing the velocity profile along the path because no steering actuator can immediately switch between $+\kappa_{\max}$ and $-\kappa_{\max}$. A possible solution is to stop the car at the

curvature discontinuities in order to adjust the steering angle. However, such a trajectory not only appears unnatural, but also imposes a high mechanical stress on the steering actuator. As a result, paths with a higher geometric continuity are required that smoothly steer the vehicle from start to goal.

Apart from optimality and completeness, it is worth mentioning that Dubins steer is not symmetric and does not fulfill the topological property [111, 170]. Symmetry within this context describes the property of a steering function to output the same path when both start and goal state are swapped. This is not the case here as the Dubins car is not allowed to switch directions. The concept behind the topological property is more complex and further discussed in Sec. 2.3.2.4. Here, it is sufficient to know that a motion planner, which relies on a steering function, can only be complete if that steering function fulfills the topological property. This is, however, not the case for Dubins steer [111].

2.2.2 Reeds-Shepp Steer

In 1990, 33 years after Dubins' publication, Reeds and Shepp found a way to compute the shortest path for a car-like robot with bounded curvature and no constraint on the driving direction [152]. This breakthrough was so fundamental that it still can be found in various state-of-the-art motion planning algorithms [40, 100]. The underlying concepts are briefly reviewed in this section including the so-called RS car in Sec. 2.2.2.1 and the computation of a RS path in Sec. 2.2.2.2. Note that the RS turn is identical to the Dubins turn from Sec. 2.2.1.2 and therefore not further detailed here.

2.2.2.1 Reeds-Shepp Car

Similar to the Dubins car, the RS car moves on a path with bounded curvature, but without a constraint on the driving direction. While the corresponding motion model remains the same as in (2.2), the input is now constrained by

$$\check{\mathbf{u}}_{\min} = \begin{pmatrix} -\infty & -\kappa_{\max} \end{pmatrix}^\mathsf{T}, \tag{2.10a}$$

$$\check{\mathbf{u}}_{\max} = \begin{pmatrix} +\infty & +\kappa_{\max} \end{pmatrix}^\mathsf{T}, \tag{2.10b}$$

and allows the vehicle to move both forwards and backwards. This yields a symmetric system [113] that can now move from a start state to some destination and back again to the initial state on exactly the same path.

2.2.2.2 Reeds-Shepp Path

Given two arbitrary vehicle states, it is proven in [152] that the shortest connection can be computed analytically using straight lines and circular arcs of maximum curvature. The resulting solution is a generalization of Dubins steer as it guarantees to find the same or a shorter path by allowing direction switches (cusps) between start and goal. The computation of a RS path is very similar to the one of a Dubins path and briefly outlined in the following.

The first computation step requires the same circle fitting as in the Dubins case. This can be seen in Fig. 2.7, where the two circles at the given states $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_g$ allow the vehicle to turn in both directions. The only difference to the

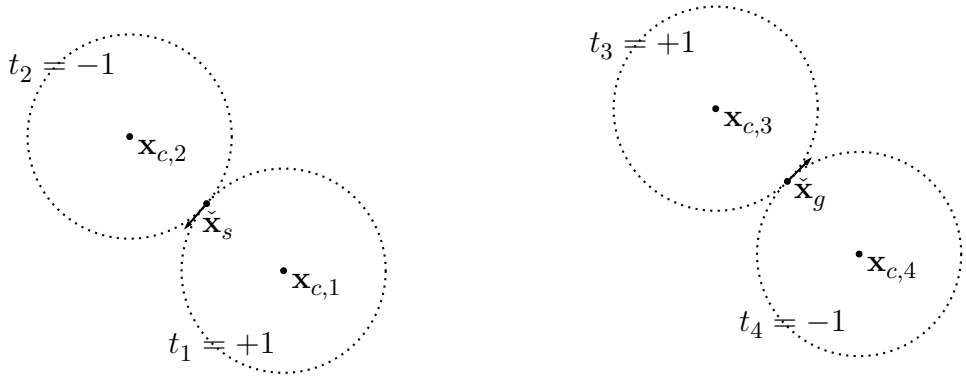


Figure 2.7 Two circles of maximum curvature, but different turning directions, fitted to the start and goal state.

computation of the Dubins path now lies in the fact that the RS car is allowed to move both forwards and backwards on all four circles.

Obviously, this increases the computational complexity of the optimal solution, which now requires to evaluate 46 possible candidate paths [182]. These candidates have to be constructed in the second step of the algorithm on the basis of the RS families given in Tab. 2.2.

Table 2.2 RS families, where C describes a circle of maximum curvature, S a straight line, and $|$ a direction switch.

RS families		
CSC	$CSC\|C$	$C\|CSC\|C$
$CC\|C$	$C\|CSC$	
$C\|CC$	$CC\|CC$	
$C\|C\|C$	$C\|CC\|C$	

In comparison to the Dubins families, it can be observed that direction switches are now allowed raising the number of possible families from two to nine. For instance, the family $CSC|C$ describes a path that starts on a circle C, moves on to a straight line S, transitions to another circle C, and switches the driving direction before reaching the terminal circle C. An example of such a path can be found in Fig. 2.8.

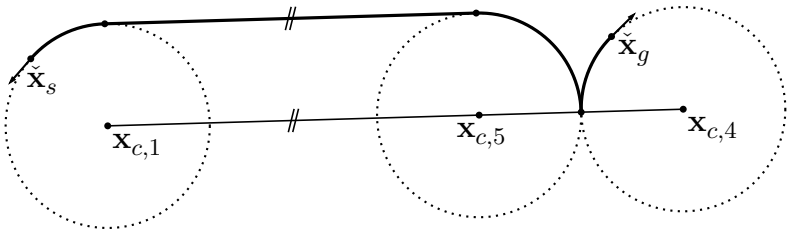


Figure 2.8 Extension of Fig. 2.7 with a RS candidate path based on the family $CSC|C$.

Similar to the Dubins case, not every RS family allows to connect the $i \in \{1, 2\}$ start circles with the $j \in \{3, 4\}$ goal circles (see Fig. 2.7). For instance, the existence conditions for the visualized candidate path in Fig. 2.8, where the tangent runs parallel to the line connecting the center of the start and goal circle, are given as

$$t_i \cdot t_j \overset{!}{=} -1, \tag{2.11a}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 2\kappa_{\max}^{-1}. \tag{2.11b}$$

Here, the turning direction at the start must be different to the one at the goal, and the distance between the corresponding circle centers must be larger or equal to the given value in (2.11b). Notice that $\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 = 2\kappa_{\max}^{-1}$ describes the corner case, where both the RS turn at the start and the one in the middle become a single turn.

Having constructed all candidate paths as described above, the optimal solution to the shortest path problem can finally be obtained by selecting the one with the minimal length [152]. An implementation of this procedure can be found in [179], and a visualization of the shortest path for the same start and goal state as in Fig. 2.6 is given in Fig. 2.9.

It can be seen in Fig. 2.9 that in contrast to Dubins steer (see Fig. 2.6), RS steer requires the vehicle to first move backwards and to switch direction before arriving at the goal. The resulting path is G^1 continuous as the orientation of the vehicle changes smoothly along the path. Just as with Dubins

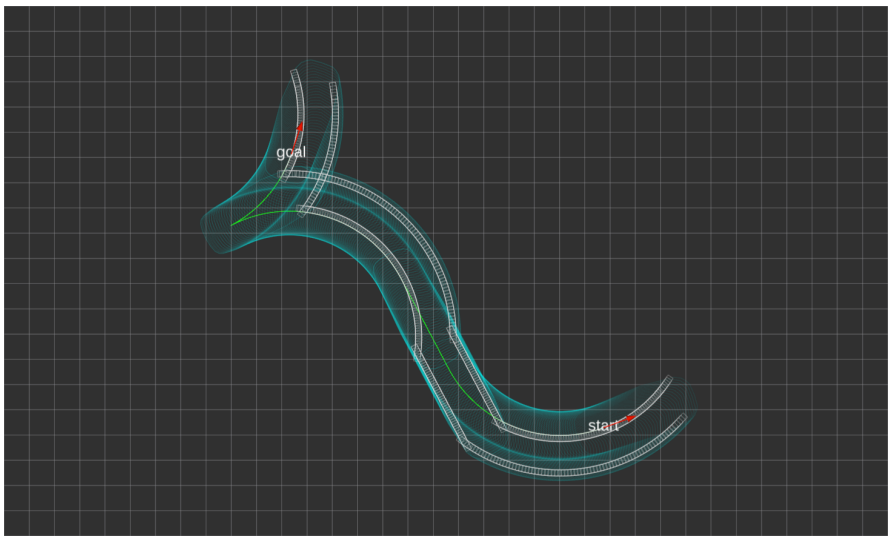


Figure 2.9 Visualization of the shortest path for a vehicle that moves both forwards and backwards. The front wheels of the vehicle are displayed in white, its contour in cyan, and the path is colored green.

steer, a precise closed-loop execution of the RS path forces the vehicle to stop at the curvature discontinuities in order to adjust the steering angle. This restriction is tackled in the next section, where the computed path is required to be continuous in curvature.

Additionally, it has to be mentioned that RS steer is complete, symmetric, and fulfills the topological property [111]. Completeness can be easily verified on the basis of the RS family CSC and the argumentation provided in Sec. 2.2.1.3. In contrast to Dubins steer, symmetry can be guaranteed due to three reasons: (1) the RS car is allowed to move both forwards and backwards, which yields a symmetric system [113], (2) the RS families are symmetrical by definition like $C|C|C$, or (3) a symmetrical counterpart exists such as in the case of $CC|C$ and $C|CC$. The proof with respect to the topological property of RS steer can be found in [111]. Here, it is sufficient to note that RS steer fulfills that property and thus guarantees completeness when integrated into a complete motion planner.

2.3 G^2 Continuous Steering Functions

The curvature discontinuities of Dubins and RS steer motivate the development of steering functions with higher continuity. Within this context, two G^2 continuous steering functions are highlighted in this section. Both

approaches require to increase the state space by one dimension to $\mathbb{R}^4$. The vehicle's state $\check{\mathbf{x}}_k = \begin{pmatrix} \check{x}_k & \check{y}_k & \check{\theta}_k & \check{\kappa}_k \end{pmatrix}^\mathsf{T}$ now also includes the curvature $\check{\kappa}_k$ of the path at the position $\begin{pmatrix} \check{x}_k & \check{y}_k \end{pmatrix}$. The input is adjusted accordingly such that the vehicle is now actuated by $\check{\mathbf{u}}_k = \begin{pmatrix} \Delta\check{s}_k & \check{\sigma}_k \end{pmatrix}^\mathsf{T}$, where $\check{\sigma}_k$ denotes the change in curvature (curvature rate) at the current arc length $\check{s}_k$.

This section is organized as follows: Sec. 2.3.1 reviews the concepts behind CC steer [53], and Sec. 2.3.2 introduces a novel steering function called HC steer. The latter is motivated by the fact that CC steer has certain drawbacks in tight environments as further detailed below. Sec. 2.3.3 then extends both CC as well as HC steer to start and goal states with arbitrary curvatures, and Sec. 2.3.4 finally evaluates the described approaches experimentally.

2.3.1 Continuous Curvature Steer

An extension of Dubins and RS steer to G^2 continuity was first presented in [53]. The derived steering function, called CC steer, enforces curvature continuity along the path and satisfies the vehicle's maximum curvature and maximum curvature rate constraint. The underlying ideas are reviewed in the following sections including the motion model in Sec. 2.3.1.1, the so-called CC turns in Sec. 2.3.1.2, and the computation of a CC path in Sec. 2.3.1.3.

2.3.1.1 Continuous Curvature Car

In analogy to the RS car, a CC car [53] is defined as a vehicle that moves forwards and backwards on a path with bounded curvature and bounded curvature rate. Its motion model can be formulated as

$$\check{\mathbf{x}}_{k+1} = \begin{pmatrix} \check{x}_{k+1} \\ \check{y}_{k+1} \\ \check{\theta}_{k+1} \\ \check{\kappa}_{k+1} \end{pmatrix} = \begin{pmatrix} \check{x}_k \\ \check{y}_k \\ \check{\theta}_k \\ \check{\kappa}_k \end{pmatrix} + \int_0^{\Delta\check{s}_k} \begin{pmatrix} \cos(\check{\theta}_{k+1}(s)) \\ \sin(\check{\theta}_{k+1}(s)) \\ \check{\kappa}_{k+1}(s) \\ \operatorname{sgn}(s)\check{\sigma}_k \end{pmatrix} \mathrm{d}s, \tag{2.12}$$

where $\operatorname{sgn}(\bullet)$ denotes the sign of a variable. The solution of (2.12) can be split into three cases. If the input $\check{\sigma}_k \neq 0$, the vehicle moves on a clothoid, also known as Euler or Cornu spiral, whose curvature changes linearly with the traveled distance. If $\check{\sigma}_k = 0$, the vehicle either moves on a circular arc ($\check{\kappa}_k \neq 0$) or on a straight line ($\check{\kappa}_k = 0$). All three robot motions can be evaluated explicitly as outlined in Sec. A.1.1, Sec. A.1.2, and Sec. A.1.3. Note, however, that the computation of a clothoid requires a numerical evaluation

of the Fresnel integrals [1], which can be efficiently approximated using e.g. Chebyshev polynomials [122].

As the CC car enforces curvature continuity without a constraint on the driving direction, its input is bounded by

$$\check{\mathbf{u}}_{\min} = \begin{pmatrix} -\infty & -\sigma_{\max} \end{pmatrix}^{\mathsf{T}}, \tag{2.13a}$$

$$\check{\mathbf{u}}_{\max} = \begin{pmatrix} +\infty & +\sigma_{\max} \end{pmatrix}^{\mathsf{T}}, \tag{2.13b}$$

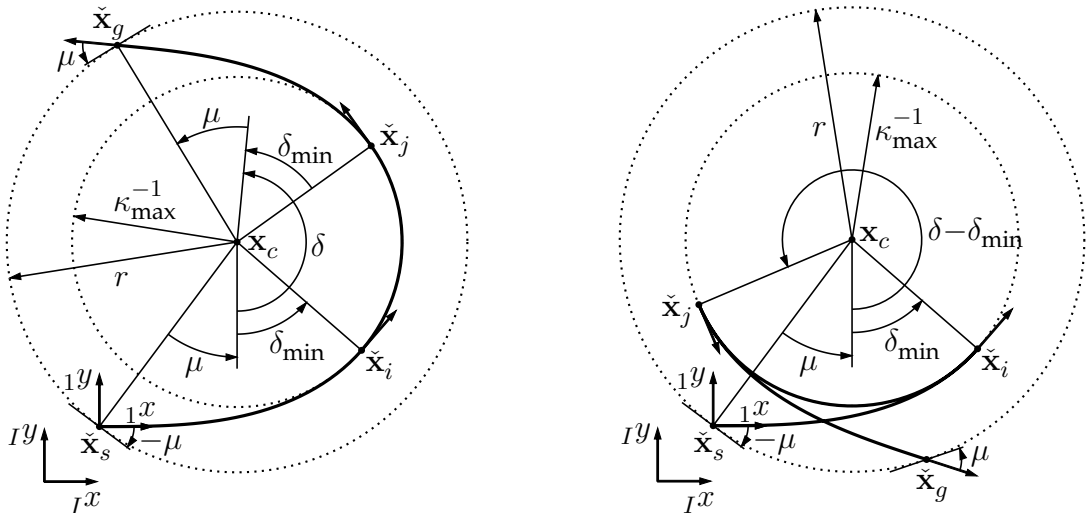
where $\sigma_{\max}$ denotes the maximum curvature rate. This parameter can be derived according to [53] and ensures that the vehicle's maximum steering rate is satisfied. In addition to that, the maximum steering angle of the CC car constrains the vehicle's curvature to

$$-\kappa_{\max} \leq \check{\kappa}_{k+1} \leq \kappa_{\max}. \tag{2.14}$$

The properties of the CC car without the constraint in (2.14) have been extensively studied in previous publications [14, 38, 181], especially with respect to the shortest path problem. An extension of the analysis to the fully constrained system is conducted in [158, 162]. Briefly summarized, it is known that an optimal path with minimal length exists for arbitrary start and goal states. Such a path consists of straight lines, circular arcs of maximum curvature, and clothoids of maximum curvature rate. However, a closed-form solution, as in the case of the Dubins and RS car, does not exist anymore because the shortest path may consist of an infinite number of clothoids. Therefore, a hierarchical approach is presented in [53, 161] that computes a suboptimal solution to the shortest path problem for the CC car. An integral part of this approach are the so-called CC turns, which are described in the next section.

2.3.1.2 Continuous Curvature Turns

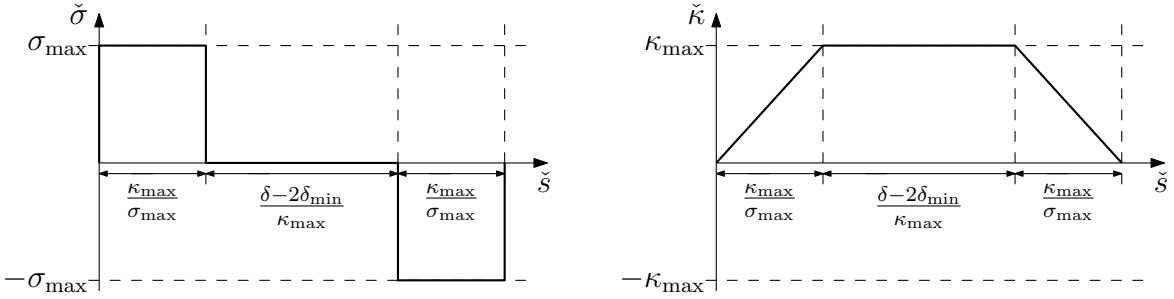
The main idea of CC turns [53, 161] is to design a novel circle that enforces curvature continuity when being concatenated with other CC turns or straight lines. This is achieved by enforcing zero curvature at the two states $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_g$ that enter and exit such a turn. Inspired by the fact that an optimal path for the CC car consists of circular arcs of maximum curvature and clothoids of maximum curvature rate, a CC turn connects both states in three steps (see Fig. 2.10): (1) the vehicle moves on a clothoid of maximum curvature rate $\sigma_{\max}$ from $\check{\mathbf{x}}_s$ to the first intermediate state $\check{\mathbf{x}}_i$, (2) it continues on a circular arc



(a) Regular CC turn. **(b)** Irregular CC turn.

Figure 2.10 Visualization of a regular and irregular CC turn for a forward motion to the left. Both turns connect the start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ with two clothoids and a circular arc. The irregular turn yields a shorter connection for $\delta > 2\delta_{\min} + \pi$.

with radius $\kappa_{\max}^{-1}$ until it reaches the second intermediate state $\check{\mathbf{x}}_j$, and (3) the vehicle arrives at the goal state $\check{\mathbf{x}}_g$ on a clothoid of curvature rate $-\sigma_{\max}$. The corresponding profiles of the curvature rate $\check{\sigma}(\check{s})$ and the curvature $\check{\kappa}(\check{s})$ with respect to the traveled arc length $\check{s}$ are displayed in Fig. 2.11.



(a) Curvature rate profile. **(b)** Curvature profile.

Figure 2.11 Curvature rate and curvature profile of a CC turn to the left. The curvature profile is obtained by integrating the curvature rate along the arc length $\check{s}$.

Depending on the deflection $\delta \in [0, 2\pi)$, which measures the angular change in orientation between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_g$, it is distinguished between a regular CC turn and an irregular one as illustrated in Fig. 2.10. In contrast to the regular CC turn, the irregular one allows a direction switch at the intermediate states $\check{\mathbf{x}}_i$ and $\check{\mathbf{x}}_j$. This results in a shorter connection for $\delta > 2\delta_{\min} + \pi$,

where the angle $\delta_{\min}$[4] describes the deflection between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_i$ as

$$\delta_{\min} = \kappa_{\max}^2/(2\sigma_{\max}). \tag{2.15}$$

A major advantage of the CC turn is that all its variables (see Fig. 2.10) can be described explicitly given the vehicle's initial gear $g \in \{-1, 1\}$ (backwards, forwards), the desired turning direction $t \in \{-1, 1\}$ (right, left), the start state $\check{\mathbf{x}}_s$, and the deflection δ. For instance, the first intermediate state $\check{\mathbf{x}}_i$ is computed as

$${}_1\check{\mathbf{x}}_i = \begin{pmatrix} {}_1\check{x}_i \\ {}_1\check{y}_i \\ {}_1\check{\theta}_i \\ {}_1\check{\kappa}_i \end{pmatrix} = \begin{pmatrix} g\sqrt{\pi/\sigma_{\max}} C_f(\sqrt{2\delta_{\min}/\pi}) \\ t\sqrt{\pi/\sigma_{\max}} S_f(\sqrt{2\delta_{\min}/\pi}) \\ gt\delta_{\min} \\ t\kappa_{\max} \end{pmatrix}, \tag{2.16}$$

where ${}_1(\bullet)$ describes a variable in the local frame 1 (see Fig. 2.10) and $C_f(\bullet)$, $S_f(\bullet)$ the Fresnel integrals given in (A.14). The center of the CC circle $\mathbf{x}_c$ can be obtained by

$${}_1\mathbf{x}_c = \begin{pmatrix} {}_1x_c \\ {}_1y_c \end{pmatrix} = \begin{pmatrix} {}_1\check{x}_i - {}_1\check{\kappa}_i^{-1} \sin({}_1\check{\theta}_i) \\ {}_1\check{y}_i + {}_1\check{\kappa}_i^{-1} \cos({}_1\theta_i) \end{pmatrix}, \tag{2.17}$$

and its outer radius r by

$$r = \|{}_1\mathbf{x}_c\|_2 \,. \tag{2.18}$$

Rotating $\check{\mathbf{x}}_i$ by $\delta - 2\delta_{\min}$ on a circle with radius $\kappa_{\max}^{-1}$ and center $\mathbf{x}_c$ yields the second intermediate state $\check{\mathbf{x}}_j$ as

$${}_1\check{\mathbf{x}}_j = \begin{pmatrix} {}_1\check{x}_j \\ {}_1\check{y}_j \\ {}_1\check{\theta}_j \\ {}_1\check{\kappa}_j \end{pmatrix} = \begin{pmatrix} {}_1x_c + {}_1\check{\kappa}_i^{-1} \sin({}_1\check{\theta}_i + gt(\delta - 2\delta_{\min})) \\ {}_1y_c - {}_1\check{\kappa}_i^{-1} \cos({}_1\check{\theta}_i + gt(\delta - 2\delta_{\min})) \\ {}_1\check{\theta}_i + gt(\delta - 2\delta_{\min}) \\ {}_1\check{\kappa}_i \end{pmatrix}. \tag{2.19}$$

The goal state $\check{\mathbf{x}}_g$ can finally be calculated according to

$${}_1\check{\mathbf{x}}_g = \begin{pmatrix} {}_1\check{x}_g \\ {}_1\check{y}_g \\ {}_1\check{\theta}_g \\ {}_1\check{\kappa}_g \end{pmatrix} = \begin{pmatrix} {}_1x_c + tr\sin(gt(\delta + \mu)) \\ {}_1y_c - tr\cos(gt(\delta + \mu)) \\ gt\delta \\ 0 \end{pmatrix}, \tag{2.20}$$

[4]Throughout this thesis, the minimal deflection $\delta_{\min}$ is assumed to be less than $\pi/2$, which is typically fulfilled at low speeds. An extension to arbitrary minimal deflections can be found in [208].

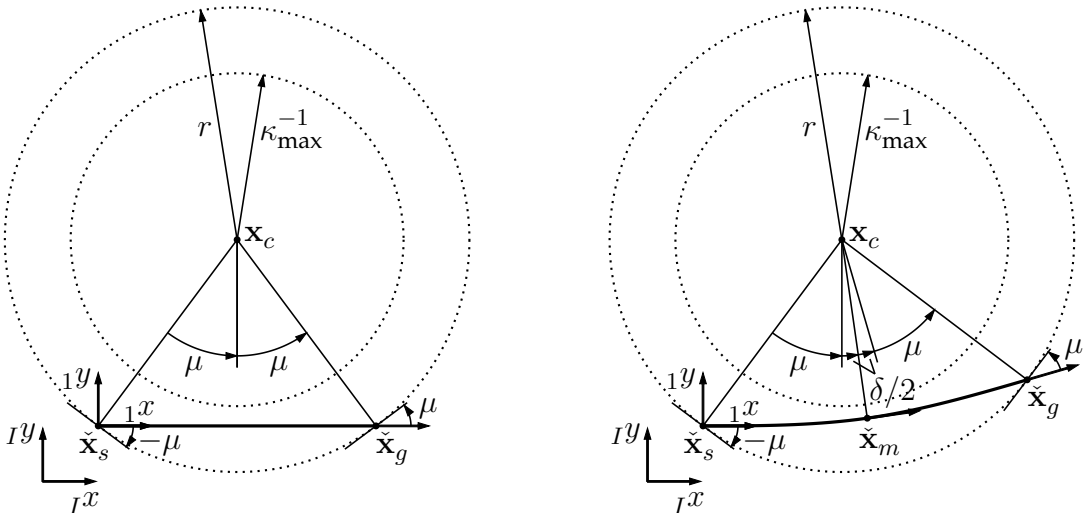
where the angle μ is given by

$$\mu = gt \arctan({}_1x_c / {}_1y_c). \tag{2.21}$$

Note that all states $\check{\mathbf{x}}_{(\bullet)}$ are given in the local frame 1 and can be transformed to the inertial frame I (see Fig. 2.10) with

$${}_I\check{\mathbf{x}}_{(\bullet)} = \begin{pmatrix} {}_I\check{x}_s \\ {}_I\check{y}_s \\ {}_I\check{\theta}_s \\ 0 \end{pmatrix} + \begin{pmatrix} \cos({}_I\check{\theta}_s) & -\sin({}_I\check{\theta}_s) & 0 & 0 \\ \sin({}_I\check{\theta}_s) & \cos({}_I\check{\theta}_s) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}_1\check{x}_{(\bullet)} \\ {}_1\check{y}_{(\bullet)} \\ {}_1\check{\theta}_{(\bullet)} \\ {}_1\check{\kappa}_{(\bullet)} \end{pmatrix}. \tag{2.22}$$

As the regular and irregular CC turns always steer the vehicle to maximum curvature, two special cases have to be considered (see Fig. 2.12) that result in a shorter arc length for small deflections. In case there is no change in orientation between the start and goal state, the CC turn reduces to a straight line as visualized in Fig. 2.12(a). For $0 < \delta < 2\delta_{\min}$, it is proposed in [53, 161] to connect both states with an elementary path [159, 160] as illustrated in Fig. 2.12(b).



(a) Straight line. **(b)** Elementary path.

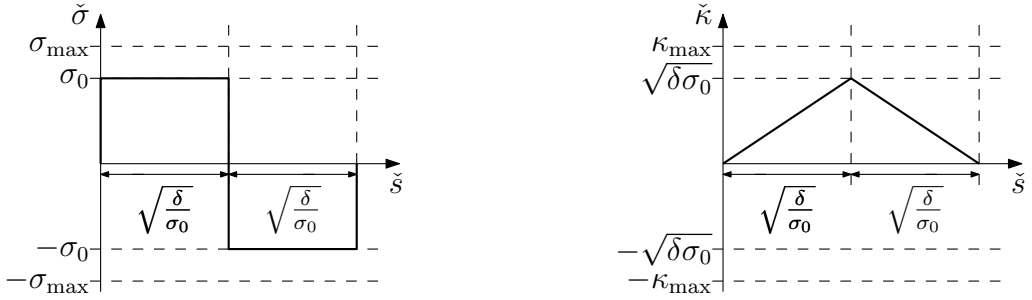
Figure 2.12 Two special cases of the CC turn for small deflections: a straight line for $\delta = 0$ and an elementary path for $0 < \delta < 2\delta_{\min}$.

The main idea of such an elementary path is to connect both vehicle states with two symmetric clothoids that satisfy the constraints of the CC car. It is shown in [160, 161] that a unique solution to this problem exists in the

interval $0 < \delta < 2\delta_{\min}$ if $\delta_{\min}$ is smaller than $\approx 0.7313\pi$. The curvature rate σ_0 of the two clothoids can then be explicitly computed according to

$$\sigma_0 = \frac{\pi(\cos(\delta/2)C_f(\sqrt{\delta/\pi}) + \sin(\delta/2)S_f(\sqrt{\delta/\pi}))^2}{(r\sin(\delta/2+\mu))^2}. \tag{2.23}$$

Fig. 2.13 visualizes both the curvature rate and the curvature profile of an elementary path.



(a) Curvature rate profile. **(b)** Curvature profile.

Figure 2.13 Curvature rate and curvature profile of an elementary path to the left.

It can be seen that such a path not only satisfies the vehicle's constraints given by $\sigma_{\max}$ and $\kappa_{\max}$, but also yields a symmetric connection between the start and goal state. This can also be observed in Fig. 2.12(b), where the symmetry axis is marked by the line joining the circle center $\mathbf{x}_c$ and the intermediate state $\check{\mathbf{x}}_m$. The latter can be explicitly expressed in the local frame 1 as

$${}_1\check{\mathbf{x}}_m = \begin{pmatrix} {}_1\check{x}_m \\ {}_1\check{y}_m \\ {}_1\check{\theta}_m \\ {}_1\check{\kappa}_m \end{pmatrix} = \begin{pmatrix} g\sqrt{\pi/\sigma_0}C_f(\sqrt{\delta/\pi}) \\ t\sqrt{\pi/\sigma_0}S_f(\sqrt{\delta/\pi}) \\ gt\delta/2 \\ t\sqrt{\delta\sigma_0} \end{pmatrix}. \tag{2.24}$$

On the basis of the details given above, the arc length $l(\delta)$ of a CC turn can now be derived. In the regular case, it is given by

$$l(\delta) = \begin{cases} 2r\sin(\mu), & \text{if } \delta = 0, \\ 2\sqrt{\delta/\sigma_0}, & \text{if } 0 < \delta < 2\delta_{\min}, \\ 2l_{\min} + \kappa_{\max}^{-1}(\delta - 2\delta_{\min}), & \text{if } \delta \geq 2\delta_{\min}, \end{cases} \tag{2.25}$$

where the length $l_{\min}$ of the clothoids are calculated as

$$l_{\min} = \kappa_{\max}/\sigma_{\max}. \tag{2.26}$$

The first two cases in (2.25) describe the length of a straight line and of an elementary path while the third case gives the length of two clothoids concatenated with a circular arc. Similarly, the arc length of an irregular CC turn is defined by

$$l(\delta) = \begin{cases} 2r\sin(\mu), & \text{if } \delta = 0, \\ 2\sqrt{\delta/\sigma_0}, & \text{if } 0 < \delta < 2\delta_{\min}, \\ 2l_{\min} + \kappa_{\max}^{-1}(\delta - 2\delta_{\min}), & \text{if } 2\delta_{\min} \leq \delta \leq 2\delta_{\min} + \pi, \\ 2l_{\min} + \kappa_{\max}^{-1}(2\pi - \delta + 2\delta_{\min}), & \text{if } \delta > 2\delta_{\min} + \pi, \end{cases} \tag{2.27}$$

where the fourth case accounts for the direction switches at the intermediate states $\check{\mathbf{x}}_i$ and $\check{\mathbf{x}}_j$. Note that using irregular rather than regular CC turns in the computation of continuous curvature paths remains a design choice to the user, which trades off path length against the number of direction switches.

2.3.1.3 Continuous Curvature Path

A CC path can now be computed such that it connects a given start and goal state while satisfying the constraints of the CC car. This steering function is called CC steer [53] and comes in two variants: CC-Dubins, which only allows driving forwards or backwards, and CC-RS, which poses no constraint on the driving direction. As indicated by their names, they extend both Dubins as well as RS steer to G^2 continuity. In the original publication [53], CC steer only connects start and goal states with zero curvature, in the following referred to as CC^{00}-Dubins and CC^{00}-RS. As this is a rather strong limitation, the extension in Sec. 2.3.3 describes a novel way that also takes into account the actual curvature at both ends of the path. Before that, a better understanding of the original computation steps is required, which are briefly outlined in the following on the basis of CC^{00}-RS steer.

Given a start state $\check{\mathbf{x}}_s$ and a goal state $\check{\mathbf{x}}_g$, both with zero curvature, four CC circles are fitted to each state such that all possible driving and turning directions of the CC car are covered. This is visualized in Fig. 2.14, where each circle center is obtained according to (2.17). As it can be seen in this figure, the initial gear of the vehicle at the start and goal state is abbreviated by $g_{i=1:4}$ and $g_{j=5:8}$ and its initial turning direction by $t_{i=1:4}$ and $t_{j=5:8}$. As both states are treated equally, it is required to invert g_j when deriving the actual driving direction on the goal circle.

In the next computation step, candidate paths have to be constructed using CC turns, straight lines, and tangency conditions to concatenate them. Op-

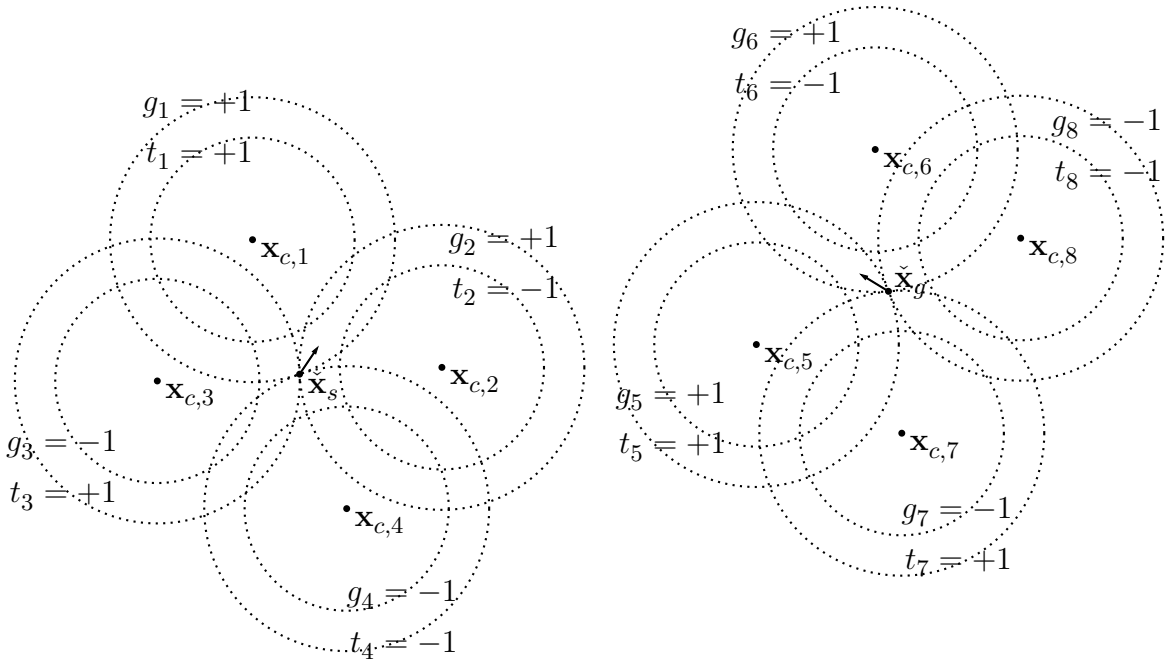


Figure 2.14 Four CC circles fitted to the start and goal state. Each circle accounts for a different driving and turning direction of the vehicle at the respective state.

posed to Dubins and RS steer, the shortest connection cannot be expressed by a closed set of candidate paths anymore as the solution might require an infinite number of clothoids. Thus, it is proposed in [53] to construct the candidate paths based on the Dubins and RS families given in Tab. 2.1 and Tab. 2.2. Depending on the available computation time, additional families [52] can be added, which might further reduce the length of the computed paths. Therefore, CC-RS steer is evaluated throughout this thesis with the families listed in Tab. 2.3, and CC-Dubins steer with the Dubins families given in the first column of that table.

Each family in Tab. 2.3 comes with a set of existence conditions that allow to quickly assess whether a start and a goal circle can be connected by the respective family. For instance, the existence conditions for $CC|C$ can be derived as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{2.28a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{2.28b}$$

$$2r(1-\cos(\mu)) \overset{!}{\leq} \left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\leq} 2r(1+\cos(\mu)), \tag{2.28c}$$

where the first and second equation condition the driving and turning direction on the start and goal circle, and the third equation defines a valid

Table 2.3 CC-RS families, where C describes a CC turn, S a straight line, and $|$ a direction switch.

Dubins families	Additonal RS families	Additional families
CSC	$CC\|C$	$CS\|C$
CCC	$C\|CC$	$C\|SC$
	$C\|C\|C$	$C\|S\|C$
	$CSC\|C$	
	$C\|CSC$	
	$CC\|CC$	
	$C\|CC\|C$	
	$C\|CSC\|C$	

interval between both circle centers. If all three conditions are fulfilled, a candidate path can be constructed. This is, for instance, visualized in Fig. 2.15 for a candidate of the family $CC|C$.

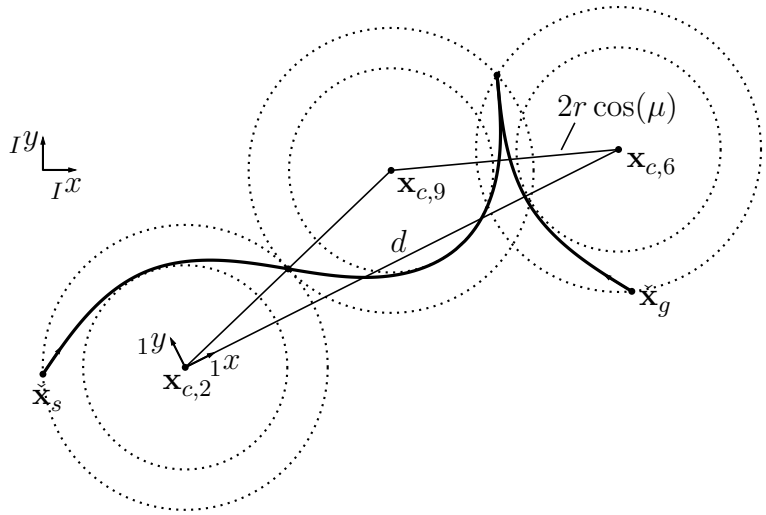


Figure 2.15 Extension of Fig. 2.14 with a CC^{00}-RS candidate path based on the family $CC|C$. The existence condition in (2.28c) can be verified by unrolling the start CC circle on the circle in the middle such that the distance d is mini-/maximized.

It can be seen that three CC turns are concatenated to a curvature continuous path that includes one direction switch. The geometric relations between the CC circles allow to compute the unknown circle center $\mathbf{x}_{c,9} = \begin{pmatrix} x_{c,9} & y_{c,9} \end{pmatrix}^\mathsf{T}$

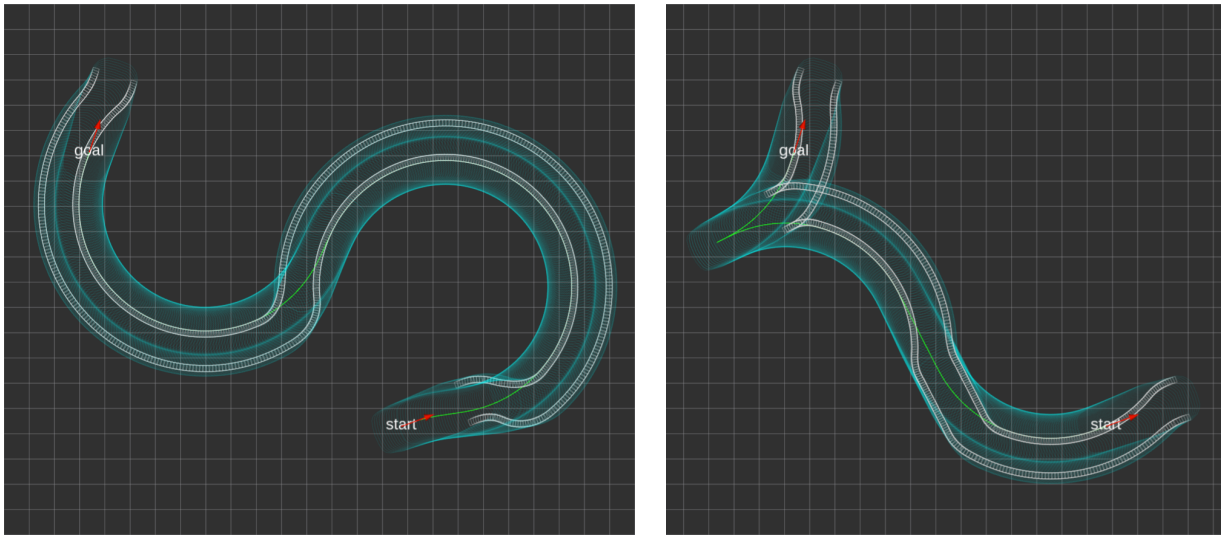
in the local frame 1 (see Fig. 2.15) according to

$$ {}_1x_{c,9} = \frac{4r^2(1-\cos^2(\mu)) + d^2}{2d}, \tag{2.29a} $$

$$ {}_1y_{c,9} = \sqrt{4r^2 - {}_1x_{c,9}^2}, \tag{2.29b} $$

where the law of cosines and the Pythagorean theorem are used. The overall length of a candidate path can now be computed by summing up the arc length of the CC turns and the length of the straight line, if one is contained within the respective family. This procedure has to be repeated for all possible candidate paths of all families. Additional information about the existence conditions of the other families including illustrations of possible candidate paths can be found in [68].

Having computed all candidates, CC steer selects the one with the minimal length as an approximation of the shortest connection. Additionally, it derives the inputs $\check{\mathbf{u}}_k$ that steer the CC car to the goal. This is implemented in [208] for the steering procedures CC^{00}-Dubins and CC^{00}-RS. A visualization of two example paths with the same start and goal pose (position and orientation) as in Fig. 2.6 and Fig. 2.9 is displayed in Fig. 2.16.



(a) CC^{00}-Dubins path. **(b)** CC^{00}-RS path.

Figure 2.16 Two illustrations of the shortest path approximation for a CC car that only moves forwards (a) and both forwards and backwards (b). The colors have the same meaning as in Fig. 2.6 and Fig. 2.9.

It can be observed that CC^{00}-Dubins steer chooses the family CCC to connect both states while CC^{00}-RS steer further optimizes the path length

by selecting the family $CSC|C$. The selected families are similar to the ones chosen by Dubins and RS steer in Fig. 2.6 and Fig. 2.9. This is, however, not always the case and depends on the two given states and the parameters of the vehicle. In addition to that, the track of the front wheels in Fig. 2.16 reveals the G^2 continuity of the computed CC paths. As a consequence, the CC car can smoothly follow the computed path without having to deal with curvature discontinuities as in the G^1 continuous case.

Just as with Dubins and RS steer, it is left to discuss the completeness, symmetry, and topological property [170] of CC steer. With respect to completeness, CC steer guarantees that a path exists for arbitrary start and goal states with zero curvature. This can be verified by comparing the existence conditions of all CC^{00}-Dubins and CC^{00}-RS families, which reveal that at least one family can always be constructed. In contrast to that, symmetry and the topological property can only be ensured by CC^{00}-RS and not by CC^{00}-Dubins steer. This is not surprising as Dubins and RS steer show the same characteristics. However, it has to be noted that in the case of CC^{00}-RS steer, satisfying the topological property requires the introduction of additional so-called topological paths [53]. Further details can be found in Sec. 2.3.2.4.

Albeit the advantages of CC steer compared to its G^1 continuous counterparts, three potential drawbacks remain:

(1) The computed bang-bang steering inputs (see Fig. 2.11(a) and Fig. 2.13(a)) require an infinite steering acceleration, which might violate the physical limits of the actuator. Possible outcomes include a high mechanical stress on the steering system as well as a potential deviation from the calculated path when it is executed by the vehicle in closed loop.

(2) CC^{00}-Dubins and CC^{00}-RS steer are restricted to zero curvature at the start and goal. This can be a significant limitation in the replanning phase of a motion planner when the vehicle is in a state with non-zero velocity and curvature.

(3) The CC car never allows to turn its wheels on the spot, which might cause unnatural solutions in tight environments such as encountered in parallel parking.

All three limitations are tackled in the following sections. A solution to (1) can be found in Sec. 2.4, a solution to (2) in Sec. 2.3.3, and the next section addresses the drawback stated in (3).

2.3.2 Hybrid Curvature Steer

As previously mentioned, maneuvering automated vehicles in tight environments without steering the wheels in place might result in unnatural paths with many cusps. Observations have shown, however, that humans often overcome this problem by allowing curvature discontinuities when switching the driving direction. This is one of the major motivations behind the introduction of HC steer, a novel steering function for automated vehicles in dense scenarios. HC steer enforces G^2 continuity between direction switches, but allows curvature discontinuity at cusps. By doing so, it computes directly executable paths with less curvature discontinuities than RS steer and shorter path length than CC steer.

The required computation steps of HC steer including its properties are outlined in the following sections: Sec. 2.3.2.1 introduces the HC car, Sec. 2.3.2.2 describes the so-called HC turns, and Sec. 2.3.2.3 outlines the computation of a HC path. In addition to that, the topological property of HC steer is discussed in Sec. 2.3.2.4.

2.3.2.1 Hybrid Curvature Car

Similar to the CC car, the so-called HC car moves on straight lines, clothoids, and circular arcs. Its motion model is identical to the one of the CC car, which constrains the maximum curvature as well as the maximum curvature rate (see Sec. 2.3.1.1). At direction switches, however, the HC car allows to change the curvature while the vehicle is not moving. In order to account for this in the arc length dependent motion model (2.12), it is required to directly modify the vehicle's current curvature to the desired value. As a result, the HC car behaves the same as the CC car between direction switches and acts similar to the RS car at cusps.

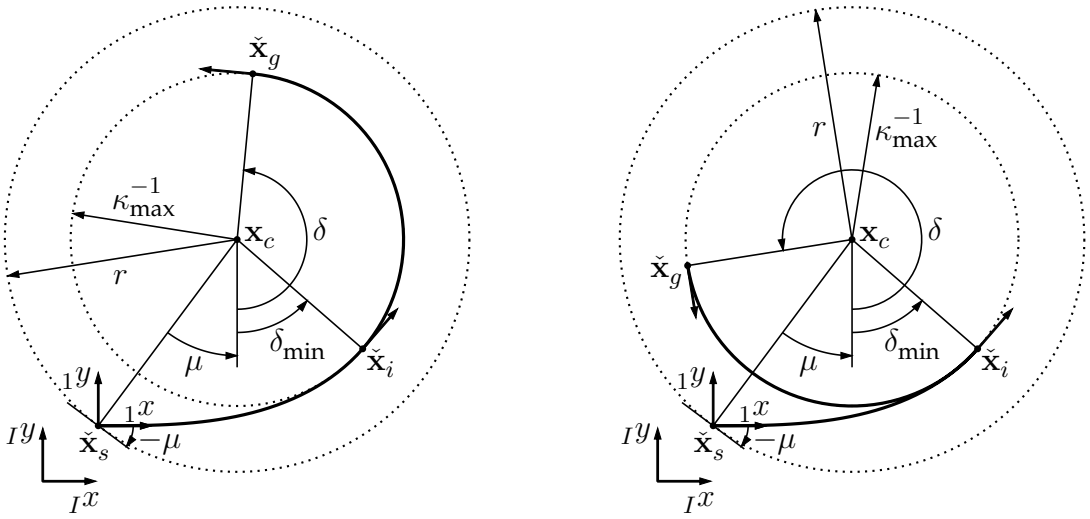
An interesting question is whether the shortest path of the HC car can be expressed similar to Dubins and RS steer with a closed set of families. Unfortunately, the insights from the analysis of the CC car allow to conclude that this is in general not the case. It can be verified by considering an optimal path that contains no direction switch. In this case, it is known for the CC car that: (1) if the start and goal state are far enough away from each other, the shortest path contains a straight line [38, 158], and (2) if a straight line is contained among other segments, an infinite number of clothoids are required to describe the optimal solution [14, 158]. As the HC car behaves the same as the CC car in the absence of cusps, the result that no closed-form

solution exists for this problem can be directly transferred to the HC car. In order to still derive a computationally tractable solution, a shortest path approximation is computed similarly to CC steer. This requires to introduce HC turns next as one of HC steer's basic components.

2.3.2.2 Hybrid Curvature Turns

The aim of this section is to introduce a novel curvature continuous turn called HC turn that transitions the vehicle from a start state $\check{\mathbf{x}}_s$ with zero curvature to a goal state $\check{\mathbf{x}}_g$ with maximum curvature. Such a turn can be seen as a combination of a CC turn, which also enforces zero curvature at the start state, and a RS turn, which terminates in a goal state with maximum curvature, too. This hybrid approach allows to concatenate HC turns with straight lines or other RS, CC, and HC turns such that curvature continuity is enforced at the start state and curvature discontinuity at the goal state. Thus, HC turns are an integral part of HC steer, which enforces curvature continuity except at cusps.

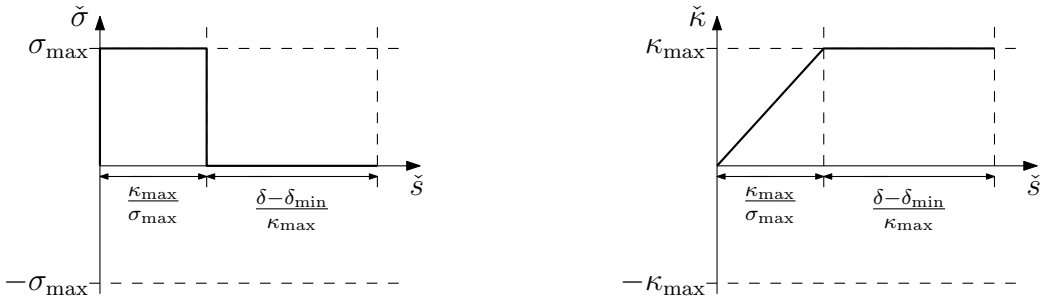
Similar to RS and CC turns, HC turns use clothoids of maximum curvature rate $\sigma_{\max}$ and circular arcs of maximum curvature $\kappa_{\max}$ to connect start and goal. This can be seen in Fig. 2.17, where a clothoid steers the vehicle from zero curvature at the start state $\check{\mathbf{x}}_s$ to maximum curvature at the intermediate



(a) Regular HC turn. **(b)** Irregular HC turn.

Figure 2.17 Visualization of a regular and irregular HC turn for a forward motion to the left. Both turns connect the start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ with a clothoid and a circular arc. The irregular turn results in a shorter connection for $\delta < \delta_{\min}$ and $\delta > \delta_{\min} + \pi$.

state $\check{\mathbf{x}}_i$. The goal state $\check{\mathbf{x}}_g$ is then reached on a circular arc with radius $\kappa_{\max}^{-1}$. The corresponding profiles of the curvature rate $\check{\sigma}(\check{s})$ and the curvature $\check{\kappa}(\check{s})$ with respect to the arc length $\check{s}$ are given in Fig. 2.18.



(a) Curvature rate profile. **(b)** Curvature profile.

Figure 2.18 Curvature rate and curvature profile of a HC turn to the left. The curvature profile is obtained by integrating the curvature rate along the arc length $\check{s}$.

Just as with CC turns, it is distinguished here between a regular and irregular HC turn (see Fig. 2.17). As opposed to the regular turn, the irregular one allows a direction switch at the intermediate state $\check{\mathbf{x}}_i$. This results in a shorter arc length for $\delta < \delta_{\min}$ and $\delta > \delta_{\min} + \pi$, where $\delta_{\min}$ is computed according to (2.15). Note, however, that a single direction switch at $\check{\mathbf{x}}_i$ inverts the initial driving direction, which has to be taken care of when computing HC paths.

Both HC turns in Fig. 2.17 can be completely described given the vehicle's initial gear $g \in \{-1, 1\}$, the desired turning direction $t \in \{-1, 1\}$, the start state $\check{\mathbf{x}}_s$, and the deflection $\delta \in [0, 2\pi)$. For example, the circle center $\mathbf{x}_c$ and the outer radius r can be evaluated according to (2.17)–(2.18). The angle μ is given by (2.21) and the intermediate state $\check{\mathbf{x}}_i$ by (2.16). A rotation by $\delta - \delta_{\min}$ of $\check{\mathbf{x}}_i$ on a circle with radius $\kappa_{\max}^{-1}$ and center $\mathbf{x}_c$ yields the goal state $\check{\mathbf{x}}_g$ in the local frame 1 as

$$
{}_1\check{\mathbf{x}}_g = \begin{pmatrix} {}_1\check{x}_g \\ {}_1\check{y}_g \\ {}_1\check{\theta}_g \\ {}_1\check{\kappa}_g \end{pmatrix} = \begin{pmatrix} {}_1x_c + {}_1\check{\kappa}_i^{-1} \sin({}_1\check{\theta}_i + gt(\delta - \delta_{\min})) \\ {}_1y_c - {}_1\check{\kappa}_i^{-1} \cos({}_1\check{\theta}_i + gt(\delta - \delta_{\min})) \\ {}_1\check{\theta}_i + gt(\delta - \delta_{\min}) \\ {}_1\check{\kappa}_i \end{pmatrix}, \tag{2.30}
$$

which can be transformed to the inertial frame I using (2.22).

The arc length $l(\delta)$ of a HC turn can now be derived. For a regular turn, it

is defined by

$$l(\delta) = \begin{cases} l_{\min} + \kappa_{\max}^{-1}(2\pi + \delta - \delta_{\min}), & \text{if } \delta < \delta_{\min}, \\ l_{\min} + \kappa_{\max}^{-1}(\delta - \delta_{\min}), & \text{if } \delta \geq \delta_{\min}, \end{cases} \tag{2.31}$$

where $l_{\min}$ is calculated according to (2.26). Note that small deflections ($\delta < \delta_{\min}$) result in longer connections than large deflections ($\delta \geq \delta_{\min}$). This is due to the fact that the regular HC turn does not allow a direction switch at $\check{\mathbf{x}}_i$. Thus, the vehicle is forced to go once all around the circle in order to reach a goal state with $\delta < \delta_{\min}$. As opposed to that, the irregular HC turn inverts the driving direction at $\check{\mathbf{x}}_i$ for $\delta < \delta_{\min}$ and $\delta > \delta_{\min} + \pi$ resulting in

$$l(\delta) = \begin{cases} l_{\min} + \kappa_{\max}^{-1}(-\delta + \delta_{\min}), & \text{if } \delta < \delta_{\min}, \\ l_{\min} + \kappa_{\max}^{-1}(\delta - \delta_{\min}), & \text{if } \delta_{\min} \leq \delta \leq \delta_{\min} + \pi, \\ l_{\min} + \kappa_{\max}^{-1}(2\pi - \delta + \delta_{\min}), & \text{if } \delta > \delta_{\min} + \pi. \end{cases} \tag{2.32}$$

Notice, however, that a single direction switch on a HC turn has to be kept in mind when constructing HC paths, which are supposed to only allow curvature discontinuities at cusps. This requires a case-by-case analysis to determine whether a regular HC turn can be replaced by an irregular one without violating the definition of HC steer.

2.3.2.3 Hybrid Curvature Path

This section details the computation steps of HC steer, a novel steering function that enforces curvature continuity except at cusps. The computed HC path minimizes path length and satisfies the constraints of the HC car. As this steering function can only be applied to vehicles that move both forwards and backwards, it is also denoted as HC-RS steer. Its computation is explicitly detailed in the following paragraphs for HC^{00}-RS, which enforces zero curvature at both the start and the goal state. This limitation is then removed in Sec. 2.3.3.

Given a start state $\check{\mathbf{x}}_s$ and a goal state $\check{\mathbf{x}}_g$, both with zero curvature, four HC circles are fitted to each state by iterating over all possible driving and turning directions. This can be seen in Fig. 2.19, where the center of each HC circle is computed according to (2.17). The variables $g_{i=1:4}$ and $g_{j=5:8}$ denote the vehicle's initial gear at the start and goal state while $t_{i=1:4}$ and $t_{j=5:8}$ describe its turning direction at the respective state. Similar to CC steer,

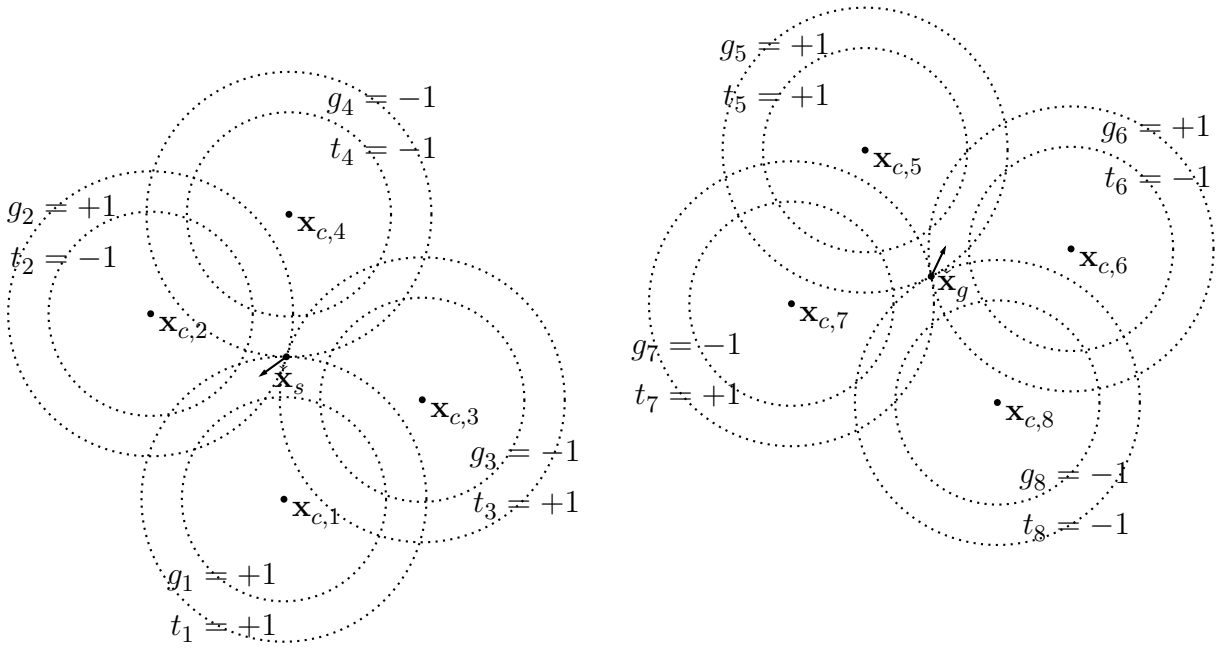


Figure 2.19 Four HC circles fitted to the start and goal state. Each circle describes a different driving and turning direction of the vehicle at the respective state.

it has to be noted that the vehicle's actual driving direction on the goal circle is inverse to g_j.

Next, candidate paths are computed that minimize path length while fulfilling the desired properties of HC steer. In contrast to Dubins and RS steer, the shortest path for the HC car can generally not be described by a finite set of candidate paths (see Sec. 2.3.2.1). To still obtain a computationally tractable solution, it is therefore proposed to construct candidate paths on the basis of the same heuristically selected families as CC-RS steer (see Tab. 2.3).

It is known that each family in Tab. 2.3 comes with a set of existence conditions. They allow to quickly determine which of the four start and goal HC circles (see Fig. 2.19) can be connected by the respective family. For instance, the existence conditions of $C|CC$ are defined as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{2.33a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{2.33b}$$

$$2(r - \kappa_{\max}^{-1}) \overset{!}{\leq} \left\| \mathbf{x}_{c,j} - \mathbf{x}_{c,i} \right\|_2 \overset{!}{\leq} 2(r + \kappa_{\max}^{-1}). \tag{2.33c}$$

The first two conditions in (2.33) formulate a constraint on the initial and final driving and turning direction. Opposed to that, the third condition gives a valid interval between the center of the circles $\mathbf{x}_{c,i=1:4}$ and $\mathbf{x}_{c,j=5:8}$. Only if all

three conditions are fulfilled, a candidate path of the respective family can be constructed between two HC circles. As these existence conditions are essential for a fast implementation of HC steer, they are listed for all families in Sec. A.2.

Now, straight lines as well as RS, CC, and HC turns are used to construct candidate paths on the basis of the given families and the definition of HC steer. For instance, Fig. 2.20 illustrates such a candidate path of the family $C|CC$.

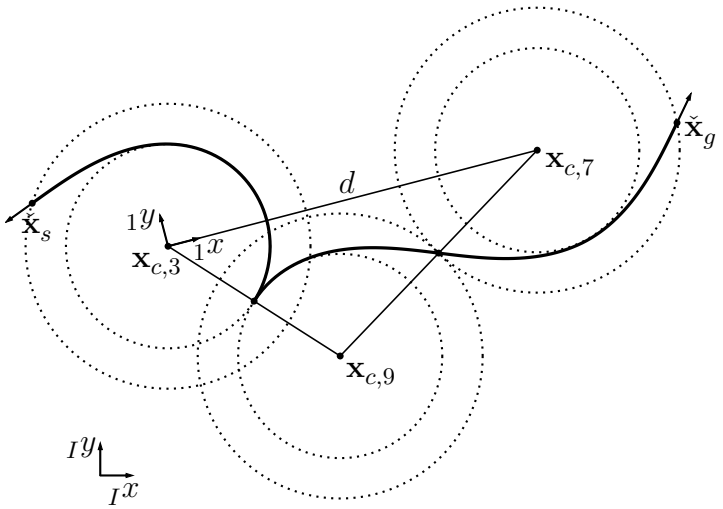


Figure 2.20 Extension of Fig. 2.19 with a HC^{00}-RS candidate path based on the family $C|CC$. The existence condition in (2.33c) can be verified by unrolling the start HC circle on the circle in the middle such that the distance d is mini-/maximized.

It can be seen that the computed path consists of two HC turns and one CC turn at the end. Thus, the steering procedure enforces curvature continuity everywhere except at the cusp as required by the definition of HC steer. In this example, the unknown circle center $\mathbf{x}_{c,9} = \begin{pmatrix} x_{c,9} & y_{c,9} \end{pmatrix}^\mathsf{T}$ can be computed in the local frame 1 (see Fig. 2.20) as

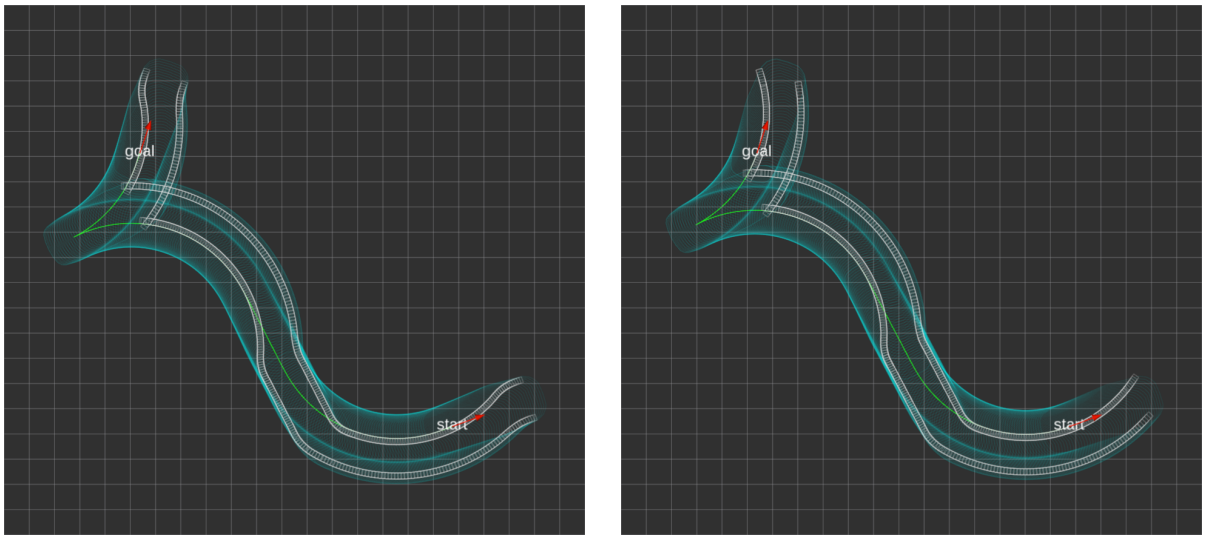
$${}_1x_{c,9} = \frac{4(\kappa_{\max}^{-2} - r^2) + d^2}{2d}, \tag{2.34a}$$

$${}_1y_{c,9} = \sqrt{4\kappa_{\max}^{-2} - {}_1x_{c,9}^2}, \tag{2.34b}$$

where the law of cosines and the Pythagorean theorem are used. The overall path length can now be derived by adding the arc length of the turns to the length of the straight line, if one is contained within the respective family. These steps have to be repeated for all possible candidate paths of all families,

which are further detailed in Sec. A.2. HC steer finally selects the candidate path with the minimal length and returns the corresponding inputs that move the vehicle to the goal. The resulting path is an approximation of the shortest path problem for the HC car with zero curvature at the start and goal.

The previously described steering function HC^{00}-RS enforces zero curvature at both ends of the path. Modifications to the start and/or the goal circle allow to also realize the steering functions $\mathrm{HC}^{0\pm}$-, $\mathrm{HC}^{\pm 0}$-, and $\mathrm{HC}^{\pm\pm}$-RS, where the superscript denotes the curvature at the start and goal (zero or $\pm\kappa_{\max}$). For instance, $\mathrm{HC}^{0\pm}$-RS steer denotes a steering procedure that starts with zero curvature and terminates with either $\pm\kappa_{\max}$ depending on the shorter path length. An implementation of all four steering functions can be found in [208] and an exemplary visualization of HC^{00}- and $\mathrm{HC}^{\pm\pm}$-RS steer in Fig. 2.21.



(a) HC^{00}-RS path. **(b)** $\mathrm{HC}^{\pm\pm}$-RS path.

Figure 2.21 Visualization of two shortest path approximations for a HC car. The illustrated paths are once computed for zero curvature at the start and goal state (a) and once for either $\pm\kappa_{\max}$ at both states (b). The colors are adopted from Fig. 2.9.

Both images highlight the nature of HC steer: computation of paths that enforce G^2 continuity everywhere except at cusps. This can be seen by looking at the track of the front wheels in Fig. 2.21. As a result, HC steer is closely related to RS steer at cusps (see Fig. 2.9) and to CC-RS steer everywhere else (see Fig. 2.16(b)).

Further important questions are whether HC steer is complete and symmetric, and whether it fulfills the topological property [170]. With respect to completeness, all four derivatives of HC steer, namely HC^{00}-, $\mathrm{HC}^{0\pm}$-, $\mathrm{HC}^{\pm 0}$,

and HC$^{\pm\pm}$-RS are complete. This can be verified by comparing the existence conditions of the families used for the computation of the candidate paths [208]. The symmetry analysis of RS steer in Sec. 2.2.2.2 can be directly transferred to HC^{00}- and HC$^{\pm\pm}$-RS steer allowing to conclude that both steering functions are symmetric. In contrast to that, HC$^{0\pm}$- and HC$^{\pm 0}$-RS steer cannot fulfill this property due to the different curvatures at both ends of the path. Regarding the topological property of HC steer, a discussion can be found in the next section.

2.3.2.4 Topological Property

Integrating a steering function into a (probabilistically) complete motion planner, such as RRT*, raises the question whether completeness can be preserved for the resulting combination. On the basis of [170], it is known that such a combination remains complete if the steering function respects the topological property. In order to proof that this is the case, it has to be shown that the steering function fulfills the underlying definition given in [170] as

$$\begin{aligned} &\forall \varepsilon > 0, \exists \eta > 0 \text{ such that } \forall \check{\mathbf{x}}_s \in \mathbb{R}^n, \\ &\forall \check{\mathbf{x}}_g \in B(\check{\mathbf{x}}_s, \eta) \Rightarrow \text{steer}(\check{\mathbf{x}}_s, \check{\mathbf{x}}_g) \in B(\check{\mathbf{x}}_s, \varepsilon), \end{aligned} \tag{2.35}$$

where ε and η are scalar values that describe the radius of a ball $B(\bullet)$ centered at the given start configuration $\check{\mathbf{x}}_s$, and steer$(\bullet)$ denotes the steering function that outputs a feasible connection from the start to the goal state. Within this context, it has to be noted that steer$(\bullet)$ is not restricted to a single steering function call, but rather allowed to concatenate multiple steering procedures in order to fulfill the given condition.

In other words, (2.35) says that a steering function fulfills the topological property if all states in an η-neighborhood can be reached by a path that completely remains within an ε-region. This allows to immediately conclude that all steering functions whose path length is lower bounded can not fulfill the topological property. Combining such steering functions with a motion planner yields an incomplete algorithm that might not be able to solve a feasible planning problem.

Out of the previously discussed G^2 continuous steering functions, HC$^{\pm\pm}$-RS steer directly fulfills the topological property without any further modification. The same can be guaranteed for the other derivatives of HC steer and for CC^{00}-RS steer if additional so-called topological paths [53] are introduced. Otherwise, the path length of these steering functions would be lower

bounded and thus violate the topological property as discussed above. While more information on this and the topological paths can be found in [53], the following paragraphs focus on HC$^{\pm\pm}$-RS steer and its ability to satisfy the topological property without such an extension.

Remember that HC$^{\pm\pm}$-RS steer computes hybrid curvature paths by evaluating the families in Tab. 2.3 and enforcing $\pm\kappa_{\max}$ at the start and goal. An in-depth analysis of the families shows that in this setup, both $C|C|C$ and $C|S|C$ only contain RS turns and no CC or HC turns. The advantage of RS turns is that they do not possess a lower bound on the arc length (compare (2.7), (2.25), and (2.31)) making them suitable candidates for paths that respect (2.35). Indeed, it can now be shown that the families $C|C|C$ and $C|S|C$ are sufficient in the HC$^{\pm\pm}$-RS case to satisfy the topological property.

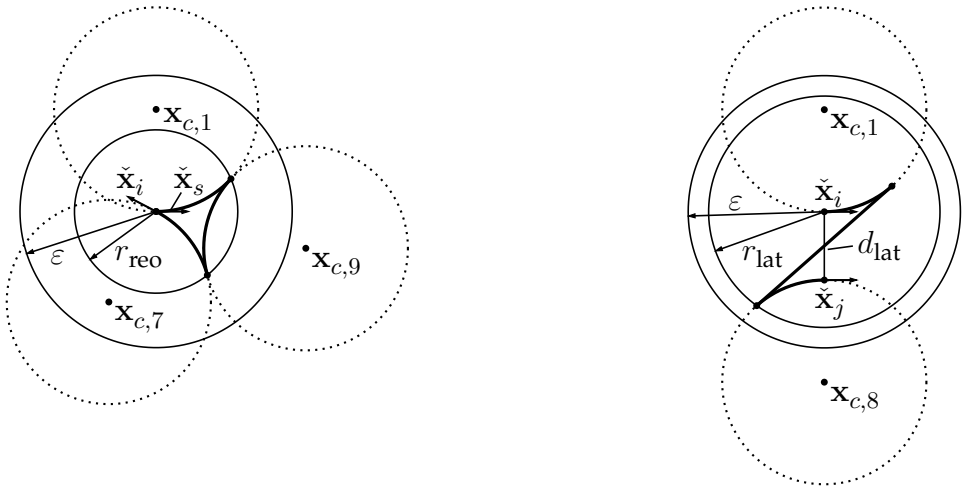
To this end, an η-neighborhood must be derived such that (2.35) can be fulfilled. Without loss of generality, the following two assumptions are made. First, the dimension n in (2.35) is set to three and the curvature at the start and goal state is neglected. This simplification is allowed in path planning as the vehicle can always be brought to a stop in order to adjust the wheels to the desired value [53]. Second, it is assumed similar to [53] that the goal state $\check{\mathbf{x}}_g = \begin{pmatrix}\check{x}_g & \check{y}_g & \check{\theta}_g\end{pmatrix}^\mathsf{T}$, where $\check{\mathbf{x}}_g \in B(\check{\mathbf{x}}_s, \eta)$, is reached from the start state $\check{\mathbf{x}}_s = \begin{pmatrix}0 & 0 & 0\end{pmatrix}^\mathsf{T}$ in three steps: (1) reorient the vehicle to the first intermediate state $\check{\mathbf{x}}_i = \begin{pmatrix}0 & 0 & \check{\theta}_i\end{pmatrix}^\mathsf{T}$ (see Fig. 2.22(a)), (2) move it laterally to the second intermediate state $\check{\mathbf{x}}_j = \begin{pmatrix}\check{x}_g & \check{y}_g & \check{\theta}_i\end{pmatrix}^\mathsf{T}$ (see Fig. 2.22(b)), and (3) apply another reorientation similar to (1) in order to reach the terminal state $\check{\mathbf{x}}_g$.

The reorientation path in Fig. 2.22(a) is given by the family $C|C|C$ of HC$^{\pm\pm}$-RS steer. It moves the vehicle on three consecutive RS turns from $\check{\mathbf{x}}_s$ to $\check{\mathbf{x}}_i$ without leaving the circle with radius $0 < r_{\text{reo}} \leq \min(\varepsilon, \kappa_{\max}^{-1})$. The deflections δ_1, δ_7, and δ_9 on those three circles can be computed analytically as

$$\delta_1(r_{\text{reo}}) = \delta_7(r_{\text{reo}}) = 2\arcsin\left(\frac{r_{\text{reo}}\kappa_{\max}}{2}\right), \tag{2.36a}$$

$$\delta_9(r_{\text{reo}}) = 2\arccos\left(\frac{\kappa_{\max}^{-1} + r_{\text{reo}}\sin(\delta_1/2)}{\sqrt{r_{\text{reo}}^2 + \kappa_{\max}^{-2} + 2r_{\text{reo}}\kappa_{\max}^{-1}\sin(\delta_1/2)}}\right). \tag{2.36b}$$

In other words, given an arbitrary intermediate orientation $\check{\theta}_i \in [-\pi, \pi)$, one can always find a radius $r_{\text{reo}} > 0$ that allows to reach $\check{\theta}_i$ without leaving the ε-bound. Note that if $\varepsilon < \kappa_{\max}^{-1}$, it might be required to concatenate multiple reorientation paths to end up at $\check{\theta}_i$. Additionally, it has to be mentioned that



(a) Reorientation path. **(b)** Lateral path.

Figure 2.22 Illustration of a reorientation and a lateral path. The reorientation path moves the robot on three RS turns from $\check{\mathbf{x}}_s$ to $\check{\mathbf{x}}_i$ without leaving the circle with radius r_{reo}. In contrast to that, the lateral path uses two RS turns and one straight line to shift the vehicle sideways to $\check{\mathbf{x}}_j$ while remaining within the circle of radius r_{lat}. Notice that both paths are derivatives of HC$^{\pm\pm}$-RS steer's families $C|C|C$ and $C|S|C$. Furthermore, it has to be noted that $\check{\mathbf{x}}_i$ describes the same vehicle pose in both images (only aligned horizontally in (b) for better readability).

due to the symmetry of the family $C|C|C$, negative orientations can also be realized by starting with a right turn instead of a left turn as in Fig. 2.22(a).

Now that it is shown that there always exists an η-neighborhood for the reorientation of the car, the same must be true for its lateral displacement. This requires an in-depth analysis of the lateral path visualized in Fig. 2.22(b). The general goal of such a path, which belongs to HC$^{\pm\pm}$-RS steer's family $C|S|C$, is to shift the vehicle sideways from $\check{\mathbf{x}}_i$ to the parallel vehicle pose $\check{\mathbf{x}}_j$ without leaving the circle with radius $0 < r_{\text{lat}} \leq \varepsilon$. To achieve this, the deflections $\delta_1 = \delta_8$ on the two RS turns in Fig. 2.22(b) can be computed by solving

$$\frac{r_{\text{lat}}^2 \kappa_{\max}^2}{4} = \frac{4\sin^4(\delta_1/2)}{\cos^2(\delta_1)} + \frac{4\sin^4(\delta_1/2)}{\cos(\delta_1)} + \sin^2(\delta_1/2). \tag{2.37}$$

The right hand side of (2.37) describes a monotonously increasing curve for $\delta_1 \in [0, \pi/2)$ that takes a value of zero for $\delta_1 = 0$ and goes to $+\infty$ as $\delta_1 \to \pi/2$. Therefore, it can be concluded that a numerical root finding algorithm can always solve (2.37) for arbitrary values of r_{lat}. The displacement d_{lat} of such a

lateral path (see Fig. 2.22(b)) can now be computed as

$$d_{\text{lat}}(r_{\text{lat}}) = \frac{4\sin^2(\delta_1/2)}{\kappa_{\max}\cos(\delta_1)}. \tag{2.38}$$

As a result, the lateral path allows to shift the vehicle sideways to arbitrary vehicle states $\check{\mathbf{x}}_j$, which are no further than $d_{\text{lat}}(r_{\text{lat}} = \varepsilon)$ away from $\check{\mathbf{x}}_i$. The resulting vehicle motion is guaranteed to lie within the ε-bound due to the definition of the lateral path.

Now that the vehicle has reached the second intermediate state $\check{\mathbf{x}}_j = (\check{x}_g \quad \check{y}_g \quad \check{\theta}_i)^{\mathsf{T}}$, another reorientation similar to the one in Fig. 2.22(a) can be applied to finally reach the goal $\check{\mathbf{x}}_g$. Here, it only has to be considered that the vehicle is not at the center of the ε-ball anymore, which requires to constrain r_{reo} in this step to $0 < r_{\text{reo}} \leq \min(\varepsilon - d_{\text{lat}}, \kappa_{\max}^{-1})$.

In summary, the previous analysis shows that the concatenation of multiple reorientation paths with a single lateral path allows to reach all states $\check{\mathbf{x}}_g \in B(\check{\mathbf{x}}_s, \eta)$, where $\eta = d_{\text{lat}}(r_{\text{lat}} = \varepsilon)$. Furthermore, it is guaranteed by the definition of both the reorientation as well as the lateral path that the resulting vehicle motion remains within the given ε-region. As these two paths are natively integrated into the computations of HC$^{\pm\pm}$-RS steer, it can be concluded that this steering function fulfills the topological property without any further modification.

2.3.3 Arbitrary Start and Goal Curvatures

While the previously described G^2 continuous steering functions restrict the curvature at both ends of the path to zero or $\pm\kappa_{\max}$, this section presents a novel approach that removes this limitation. Thus, the general goal is to compute a shortest path approximation for the CC or HC car with arbitrary start and goal states. This capability is especially required in the replanning phase of a motion planner when the vehicle is in a state with non-zero velocity and curvature. Another application is the goal extension in search-based planners (see Fig. 2.2(c)) that solve the motion planning problem in the G^2 continuous state-action space.

The required computation steps that extend both CC and HC steer to arbitrary start and goal curvatures are detailed in this section on the basis of CC-Dubins steer (forward motion only). A transfer to CC-RS and HC-RS steer is directly possible. Note that the naming of the steering functions without superscript denotes the case with arbitrary start and goal curvatures.

The general idea in this section consists of three steps: (1) propagate the start state $\check{\mathbf{x}}_s = \begin{pmatrix} \check{x}_s & \check{y}_s & \check{\theta}_s & \check{\kappa}_s \end{pmatrix}^\mathsf{T}$ and the goal state $\check{\mathbf{x}}_g = \begin{pmatrix} \check{x}_g & \check{y}_g & \check{\theta}_g & \check{\kappa}_g \end{pmatrix}^\mathsf{T}$ to zero and maximum curvature using (2.12), (2) connect all combinations of these new states with a shortest path approximation, and (3) select the path with the overall minimum length. As a result, the transformation in (1) allows to apply the efficient G^2 continuous solutions from the previous sections in (2). A visualization of the first step can be found in Fig. 2.23.

Figure 2.23 Propagation of the start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ to the intermediate states $\check{\mathbf{x}}_{s,i=1:2}$ and $\check{\mathbf{x}}_{g,j=1:2}$ with zero and maximum curvature.

Here, the start state $\check{\mathbf{x}}_s$ is propagated forwards on two clothoids of maximum curvature rate $\pm\sigma_{\max}$ until they reach the two novel states

$$\check{\mathbf{x}}_{s,1} = \begin{pmatrix} \check{x}_{s,1} & \check{y}_{s,1} & \check{\theta}_{s,1} & \operatorname{sgn}(\kappa_s)\kappa_{\max} \end{pmatrix}^\mathsf{T}, \tag{2.39a}$$

$$\check{\mathbf{x}}_{s,2} = \begin{pmatrix} \check{x}_{s,2} & \check{y}_{s,2} & \check{\theta}_{s,2} & 0 \end{pmatrix}^\mathsf{T}, \tag{2.39b}$$

The usage of clothoids of maximum curvature rate is motivated by the fact that they are along with circles and straight lines the components of the shortest-path solution (see Sec. 2.3.1.1). The same approach is used to propagate the goal state $\check{\mathbf{x}}_g$ backwards resulting in

$$\check{\mathbf{x}}_{g,1} = \begin{pmatrix} \check{x}_{g,1} & \check{y}_{g,1} & \check{\theta}_{g,1} & \operatorname{sgn}(\kappa_g)\kappa_{\max} \end{pmatrix}^\mathsf{T}, \tag{2.40a}$$

$$\check{\mathbf{x}}_{g,2} = \begin{pmatrix} \check{x}_{g,2} & \check{y}_{g,2} & \check{\theta}_{g,2} & 0 \end{pmatrix}^\mathsf{T}. \tag{2.40b}$$

Note that if the vehicle is allowed to move both forwards and backwards, the start and the goal state also have to be propagated forwards and backwards resulting in a total of eight new states.

In order to connect the derived states, the steering functions CC^{00}-, $\mathrm{CC}^{0\pm}$-, $\mathrm{CC}^{\pm 0}$, and $\mathrm{CC}^{\pm\pm}$-Dubins are required. While CC^{00}-Dubins is already known from Sec. 2.3.1.3, the other three steering functions with maximum curvature at the start and/or the goal can be derived in a similar way using HC turns. The reader is referred to [208] for an implementation.

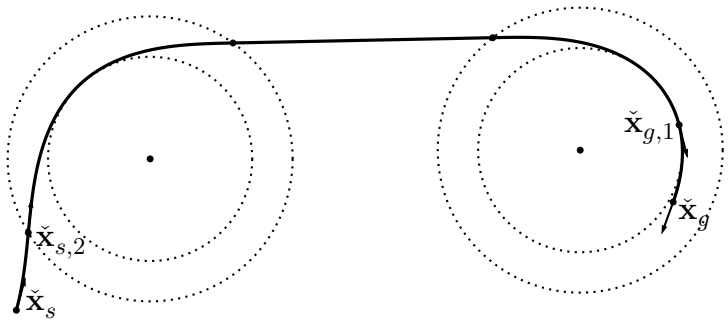
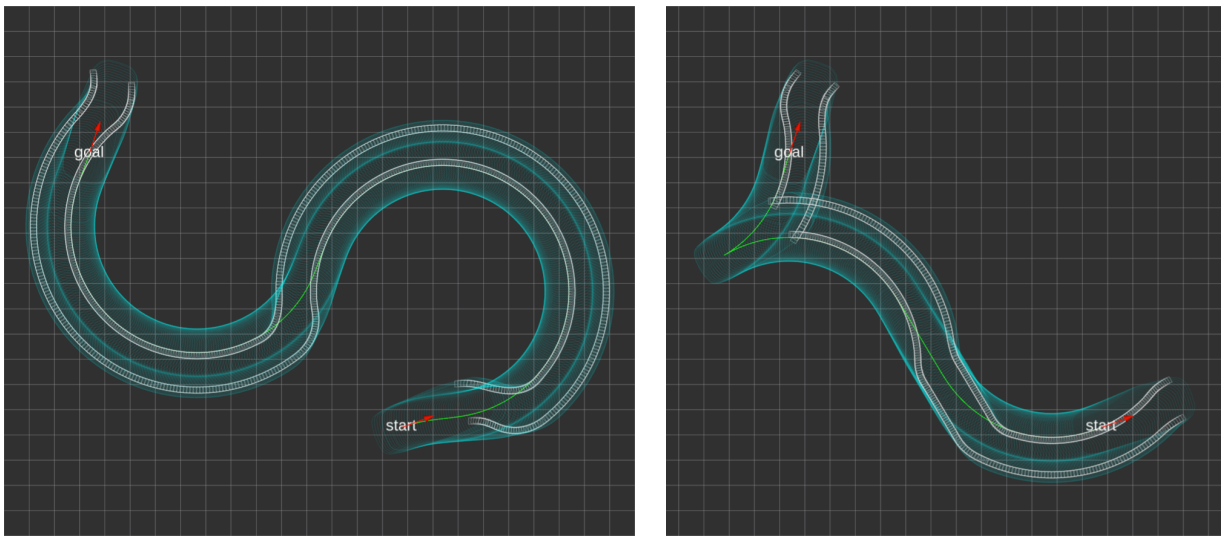


Figure 2.24 Connection of $\check{\mathbf{x}}_{s,2}$ and $\check{\mathbf{x}}_{g,1}$ from Fig. 2.23 with CC^{0-}-Dubins steer.

An extension of Fig. 2.23 with a shortest path approximation between $\check{\mathbf{x}}_{s,2}$ and $\check{\mathbf{x}}_{g,1}$ is shown in Fig. 2.24. It can be seen that the computed path of CC^{0-}-Dubins steer connects both states with a CC turn, a straight line, and a HC turn at the end. Similar connections can be computed for the other intermediate start and goal states using one of the steering functions listed above.

At the end, the path with the overall minimal length is selected as an approximation of the shortest path for the CC-Dubins car. This procedure is provided as open source for CC-Dubins and HC-RS steer in [208], and two example paths are visualized in Fig. 2.25.



(a) CC-Dubins path. **(b)** HC-RS path.

Figure 2.25 Visualization of a shortest path approximation for a CC car that only moves forwards (a) and a HC car that moves both forwards and backwards (b). The curvature at the start and goal is randomly chosen, and the colors are adopted from Fig. 2.6 and Fig. 2.9.

Analyzing the track of the front wheels in both images shows that the computed paths now take into account the randomly selected curvature at the start and goal. This can also be seen by comparing the two results in Fig. 2.25 with the corresponding counterparts in Fig. 2.16 and Fig. 2.21, which either enforce zero or maximum curvature at both ends of the path. It can also be observed that apart from the extension to arbitrary curvatures, the steering functions keep their properties, namely the computation of directly executable G^2 continuous paths. The computational effort of the presented concepts along with a benchmark against the G^1 continuous counterparts is highlighted in the next section.

2.3.4 Experimental Evaluation

The previously described G^1 and G^2 continuous steering functions are benchmarked in this section with regard to computation time, path length, and curvature discontinuities. Every steering function is evaluated on 10^5 random steering procedures with $\kappa_{\max} = 1\,\mathrm{m}^{-1}$ and $\sigma_{\max} = 1\,\mathrm{m}^{-2}$. The vehicle's start and goal position is uniformly sampled within a $20\,\mathrm{m} \times 20\,\mathrm{m}$ area, and the corresponding heading angle is randomly chosen from the interval $[0, 2\pi)$. If the steering function also takes into account the start and the goal curvature, a random value between $[-\kappa_{\max}, +\kappa_{\max}]$ is selected as the initial and final curvature. Additionally, irregular CC and HC turns are allowed to be deployed in the computation of the G^2 continuous paths. The presented results are based on the open-source implementation in [208], which is executed on a single core of an Intel Xeon E5@3.5 GHz. Note, however, that the underlying computations are well suited for parallel computing because the families of each steering function can be evaluated independently.

Tab. 2.4 shows the average computation time of the G^1 and G^2 continuous Dubins steering functions. Remember that in the curvature continuous case, a superscript denotes a restriction of the initial and final curvature to the indicated values.

It can be seen in Tab. 2.4 that all steering functions can be evaluated on average below 14.69 µs with a maximum standard deviation of 3.88 µs. Dubins steer performs on average about four times faster than CC^{00}-Dubins and about one order of magnitude faster than CC-Dubins. This is because the CC approach comes with an increase in complexity, such as the evaluation of CC turns including the numerical approximation of the Fresnel integrals. The difference in computation time between CC^{00}-Dubins and CC-Dubins is due

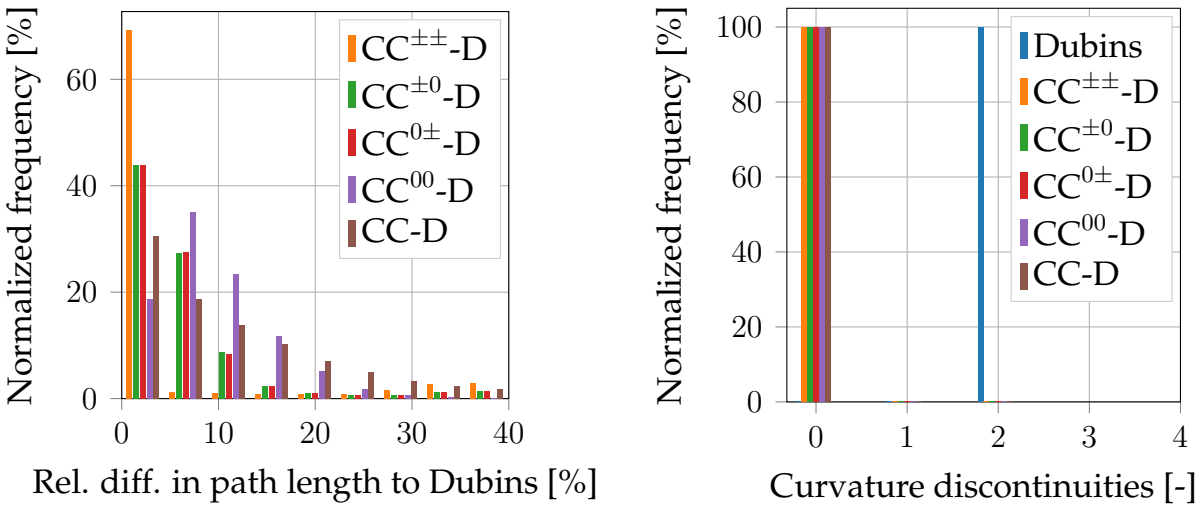
Table 2.4 Average computation time of the G^1 and G^2 continuous Dubins steering functions.

	computation time	
	mean [µs]	std [µs]
Dubins	1.29	±0.65
$CC^{\pm\pm}$-Dubins	7.16	±2.98
$CC^{\pm 0}$-Dubins	5.70	±1.85
$CC^{0\pm}$-Dubins	5.51	±1.72
CC^{00}-Dubins	4.90	±1.67
CC-Dubins	14.69	±3.88

to the fact that CC-Dubins requires four independent steering procedures to connect the intermediate start and goal states as described in the previous section. The other G^2 continuous Dubins steering functions with maximum curvature at the start and/or the goal perform about as fast as CC^{00}-Dubins. The reason for this is that they only require to replace the initial and/or the final CC circle with a HC circle to achieve the desired behavior.

The path length comparison in Fig. 2.26(a) visualizes how well the CC-Dubins steering functions approximate the provably shortest connections of Dubins steer. It can be observed that start and goal states with maximum curvature better approximate the shortest path length than states with either zero or arbitrary curvature. For instance, the length of the $CC^{\pm\pm}$-Dubins paths deviate in approximately 70 % of the 10^5 steering procedures less than 5 % from the length of the optimal Dubins paths. In contrast to that, only about 20 % of the CC^{00}-Dubins paths achieve this 5 % path length deviation. The reason for this is that the Dubins paths typically start and end with $\pm\kappa_{\max}$ similar to the paths computed by $CC^{\pm\pm}$-Dubins steer. As opposed to this, the CC^{00}-Dubins paths generally steer the vehicle from zero to maximum curvature first and back to zero curvature at the destination (see Fig. 2.16(a)). These additional transitions at both ends of the path finally result in an overall higher path length compared to the steering functions $CC^{\pm\pm}$-, $CC^{\pm 0}$-, and $CC^{0\pm}$-Dubins.

As previously mentioned, the major disadvantage of Dubins steer is that its paths are discrete in curvature. This may result in up to two curvature discontinuities at the concatenation of the geometric primitives as it can be seen in Fig. 2.26(b). In contrast to that, the greatest strength of CC-Dubins



(a) Path length comparison. **(b)** Curvature discont. comparison.

Figure 2.26 Benchmark of the G^2 continuous Dubins (D) steering functions against their G^1 continuous counterpart with respect to path length (a) and curvature discontinuities (b).

steer is that it enforces curvature continuity along the entire path. As a result, the computed G^2 continuous paths can be directly executed by a vehicle without having to deal with curvature discontinuities.

A similar evaluation as for the Dubins steering functions is conducted for the G^1 and G^2 continuous RS steering functions below. The current implementation in [208] includes RS steer, all derivatives of HC steer, and CC^{00}-RS steer as initially introduced in [53] without topological paths. Tab. 2.5 lists the corresponding average computation times.

In contrast to the Dubins steering functions, the vehicle is now allowed to move both forwards and backwards. It can be seen in Tab. 2.5 that this results in higher computation times as more families with a higher complexity have to be evaluated. For instance, RS steer is about seven times and CC^{00}-RS steer about eleven times slower than their corresponding counterpart in Tab. 2.4. While the evaluation of RS steer only takes on average 7.34 µs, the mean computation time of HC-RS steer with arbitrary start and goal curvatures raises to 449.89 µs. The reason for this is that the procedure described in Sec. 2.3.3 requires to connect four intermediate start states with four intermediate goal states. This results in 16 steering procedures, which are currently evaluated independently. However, some of these steering procedures compute the same candidate paths leaving room for further optimization in the future.

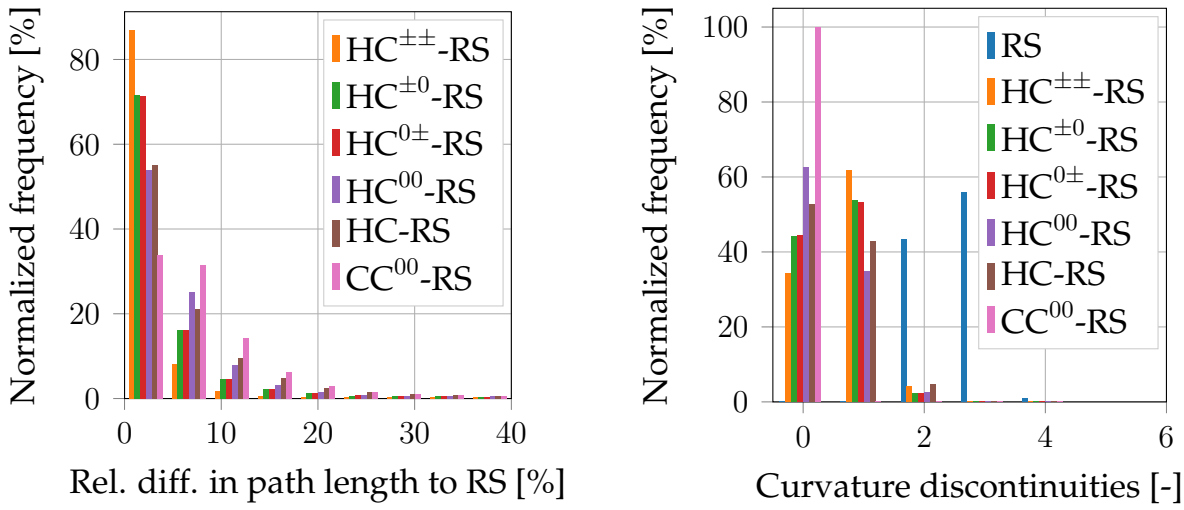
Table 2.5 Average computation time of the G^1 and G^2 continuous RS steering functions.

	computation time	
	mean [µs]	std [µs]
RS	7.34	±1.69
$HC^{\pm\pm}$-RS	55.71	±9.49
$HC^{\pm 0}$-RS	53.82	±8.25
$HC^{0\pm}$-RS	47.59	±7.52
HC^{00}-RS	53.74	±7.87
HC-RS	449.89	±67.15
CC^{00}-RS	53.19	±7.93

Tab. 2.5 also illustrates the variations in computation time measured by the standard deviation. The reason for these variations is that each of the underlying families (see Tab. 2.2 and Tab. 2.3) comes with a set of existence conditions that determine whether it can be evaluated for a given start and goal state. In general, less valid families lead to the evaluation of less candidate paths and thus to faster computations. Conversely, more valid families increase the computation time. Tab. 2.5 shows that the standard deviation is generally higher for the G^2 continuous steering functions as the computation of a candidate path is more complex than in the G^1 continuous case.

A comparison of the path length computed by the G^1 and G^2 continuous RS steering procedures is visualized in Fig. 2.27(a). It is shown that all derivatives of HC steer approximate the length of the provably shortest RS path better than CC^{00}-RS steer. This is due to the fact that similar to RS steer, HC steer allows curvature discontinuities at direction switches, which is not permitted by CC^{00}-RS steer. The latter requires transitions to zero curvature at cusps resulting in longer paths. As previously discussed, the G^2 continuous steering functions with maximum curvature at the start and/or the goal state yield a shorter path length compared to their counterpart with zero curvature at both ends of the path. For instance, more than 80 % of the 10^5 computed $HC^{\pm\pm}$-RS paths are less than 5 % longer than the respective RS path. In contrast to that, only about 50 % of the HC^{00}-RS paths achieve such a 5 % deviation.

While RS steer outperforms the G^2 continuous steering functions with respect to computation time and path length, the G^1 continuous paths suffer



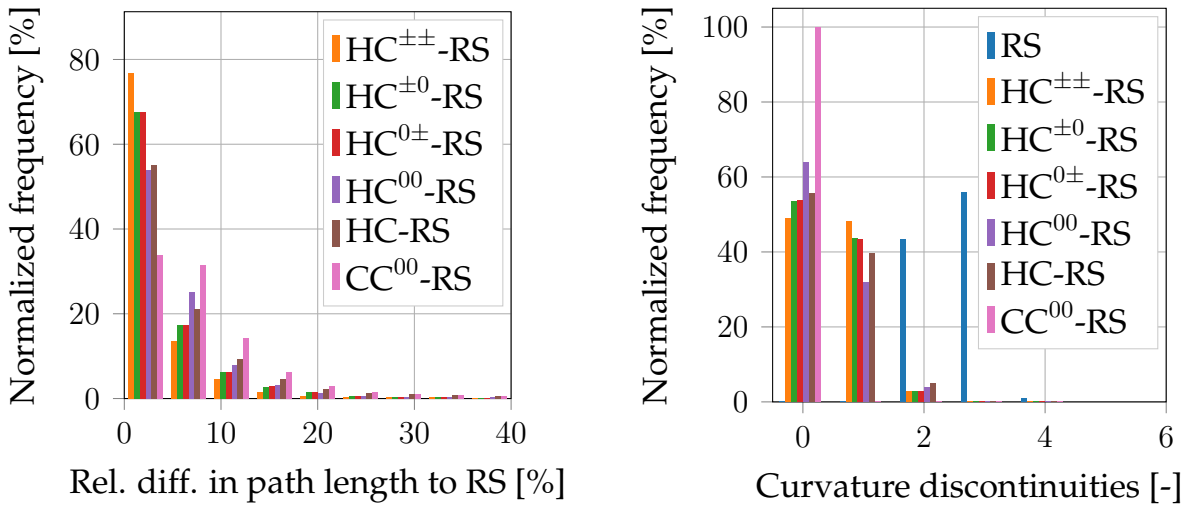
(a) Path length comparison. **(b)** Curvature discont. comparison.

Figure 2.27 Benchmark of the G^2 continuous RS steering functions against their G^1 continuous counterpart with respect to path length (a) and curvature discontinuities (b).

from up to four curvature discontinuities as shown in Fig. 2.27(b). In contrast to that, the major advantage of CC^{00}-RS steer is that it enforces curvature continuity along the entire path. A trade-off between path length and the number of curvature discontinuities is made by the derivatives of HC steer, whose paths mostly contain none or just one curvature discontinuity. As a result, the G^2 continuous steering functions compute smoother paths than RS steer. Furthermore, these paths can be directly executed without having to cope with curvature discontinuities between direction switches as in the G^1 continuous case.

The last benchmark in this section evaluates the influence of the additional families $CS|C$, $C|SC$, and $C|S|C$ (see Tab. 2.3) on the performance of the G^2 continuous RS steering functions. Excluding these families from the evaluations reduces the computation time of HC-RS steer by about 80 µs compared to Tab. 2.5. All other G^2 continuous RS steering functions only see a 7.7 µs improvement compared to the previous benchmark.

Regarding the path length and the curvature discontinuities, the impact of excluding the additional families from the computations is shown in Fig. 2.28. A comparison of Fig. 2.27 and Fig. 2.28 shows that the additional families mainly influence the results of HC$^{\pm\pm}$-, HC$^{\pm0}$-, and HC$^{0\pm}$-RS steer. For instance, the number of HC$^{\pm\pm}$-RS paths that deviate less than 5 % in path length from the RS solution drops by about 10 %. At the same time, the



(a) Path length comparison. **(b)** Curvature discont. comparison.

Figure 2.28 Benchmark of the G^2 continuous RS steering functions against their G^1 continuous counterpart, where CC and HC steer are evaluated without the additional families $CS|C$, $C|SC$, and $C|S|C$ (see Tab. 2.3).

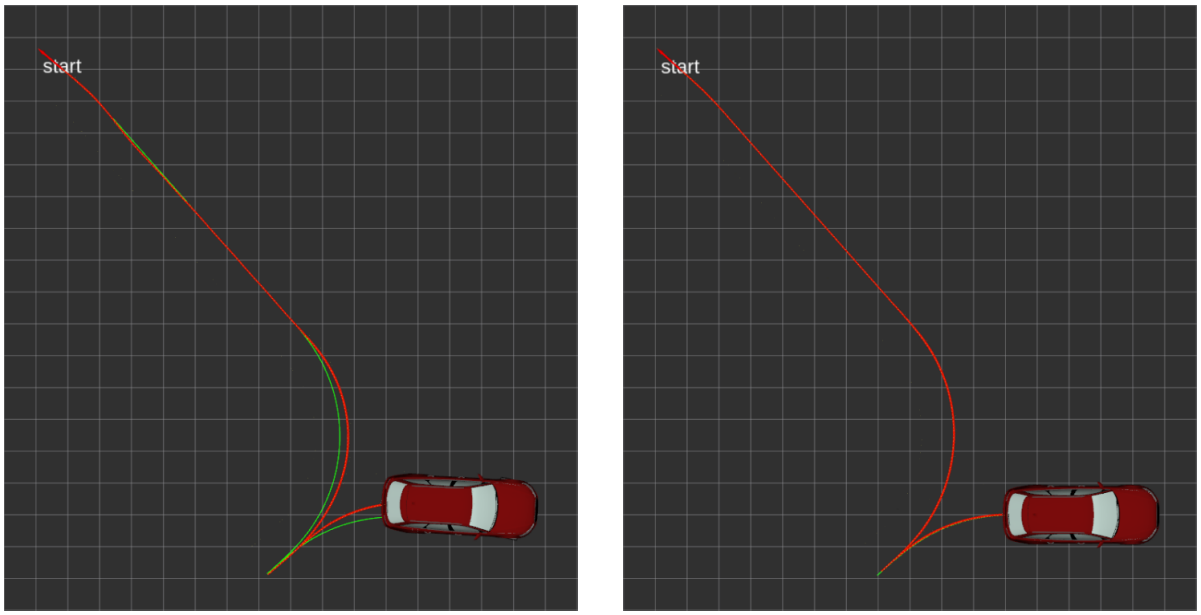
number of HC$^{\pm\pm}$-RS steering procedures without curvature discontinuities increases by 10 %. As a consequence, excluding the additional families from the computations might require the steering function to select an alternative candidate path with possibly higher path length, but potentially also fewer direction switches. Note that these results may vary for vehicles with different parameters and must therefore be evaluated on a case-by-case basis.

Briefly summarized, the results in this section show that HC steer computes smoother paths than RS steer and outperforms CC steer with respect to path length. While the computation times of both HC and CC steer are similar, enforcing G^2 continuity increases the computing effort compared to the simpler G^1 continuous case. Future implementations could, however, leverage the parallelizability of the underlying problem and by doing so, decrease the computation times of both the G^1 and the G^2 continuous steering functions.

2.4 G^3 Continuous Steering Functions

One of the drawbacks of the previously computed G^2 continuous paths are the bang-bang steering inputs, which require an infinite steering acceleration. This might not only result in a high mechanical stress on the steering system,

but also in a violation of the actuator's physical limits. A potential outcome might be a deviation from the calculated path when it is executed by a motion controller in closed loop as shown in Fig. 2.29(a). In order to overcome this



(a) G^2 continuous path. **(b)** G^3 continuous path.

Figure 2.29 Closed-loop tracking performance of a G^2 and a G^3 continuous path. The green line depicts the planned path while the red line shows the executed track.

problem, a possible approach is to increase the smoothness of the computed paths to G^3 continuity. This allows to not only enforce hard constraints on the maximum curvature and maximum curvature rate, but also on the maximum curvature acceleration. Such a path can improve the closed-loop tracking performance of the motion controller as illustrated in Fig. 2.29(b) and highlighted from a control perspective in [132].

Therefore, the goal in this section is to extend the previously described G^2 continuous steering functions to G^3 continuity, which requires to increase the vehicle's state space by one dimension to $\mathbb{R}^5$. The state of the vehicle is now described by $\check{\mathbf{x}}_k = \begin{pmatrix} \check{x}_k & \check{y}_k & \check{\theta}_k & \check{\kappa}_k & \check{\sigma}_k \end{pmatrix}^\mathsf{T}$ and the input by $\check{\mathbf{u}}_k = \begin{pmatrix} \Delta\check{s}_k & \check{\rho}_k \end{pmatrix}^\mathsf{T}$, where $\check{\rho}_k$ denotes the second derivative of curvature with respect to arc length.

The following sections are organized as follows: Sec. 2.4.1 and Sec. 2.4.2 introduce two novel G^3 continuous steering functions called continuous curvature rate (CCR) and hybrid curvature rate (HCR) steer, and Sec. 2.4.3 compares them with the G^1 and G^2 continuous approaches.

2.4.1 Continuous Curvature Rate Steer

The modular approach of RS and CC steer is leveraged in this section to derive a novel steering function called CCR steer that enforces curvature rate continuity along the entire path. In addition to that, hard constraints on the maximum curvature, maximum curvature rate, and maximum curvature acceleration are satisfied in order to take into account the physical constraints of the vehicle. The details of CCR steer are outlined in the following: Sec. 2.4.1.1 presents the underlying motion model, Sec. 2.4.1.2 introduces CCR turns, and Sec. 2.4.1.3 finally describes the computation of a CCR path.

2.4.1.1 Continuous Curvature Rate Car

Extending the CC car [53] to curvature rate continuity leads to the CCR car whose motion model is given by

$$\check{\mathbf{x}}_{k+1} = \begin{pmatrix} \check{x}_{k+1} \\ \check{y}_{k+1} \\ \check{\theta}_{k+1} \\ \check{\kappa}_{k+1} \\ \check{\sigma}_{k+1} \end{pmatrix} = \begin{pmatrix} \check{x}_{k} \\ \check{y}_{k} \\ \check{\theta}_{k} \\ \check{\kappa}_{k} \\ \check{\sigma}_{k} \end{pmatrix} + \int_0^{\Delta \check{s}_k} \begin{pmatrix} \cos(\check{\theta}_{k+1}(s)) \\ \sin(\check{\theta}_{k+1}(s)) \\ \check{\kappa}_{k+1}(s) \\ \operatorname{sgn}(s)\check{\sigma}_k + \check{\rho}_k s \\ \operatorname{sgn}(s)\check{\rho}_k \end{pmatrix} \mathrm{d}s. \tag{2.41}$$

For $\check{\rho}_k = 0$, the given equation of motion reduces to the one of the CC car whose solution either describes a straight line, a circular arc, or a clothoid (see Sec. 2.3.1.1). For $\check{\rho}_k \neq 0$, the solution of (2.41) is given in Sec. A.1.4 and describes a third-order polynomial spiral [39] also known as cubic spiral [89]. Similar to clothoids, the position update on a cubic spiral has no closed-form solution. However, efficient numerical techniques, such as Gauss-Legendre quadrature [1], exist to evaluate the corresponding integrals.

In order to bound the maximum steering acceleration of the CCR car, its input is limited by

$$\check{\mathbf{u}}_{\min} = \begin{pmatrix} -\infty & -\rho_{\max} \end{pmatrix}^{\mathsf{T}}, \tag{2.42a}$$

$$\check{\mathbf{u}}_{\max} = \begin{pmatrix} +\infty & +\rho_{\max} \end{pmatrix}^{\mathsf{T}}, \tag{2.42b}$$

where $\rho_{\max}$ denotes the maximum curvature acceleration. It can either be set to the physical limit of the steering actuator or alternatively be used as a tuning parameter, which determines the comfort of the resulting motion. Just as with the CC car, the maximum curvature and the maximum curvature rate

of the CCR car are bounded in order not to violate the physical constraints of the vehicle. The resulting state constraints are given as

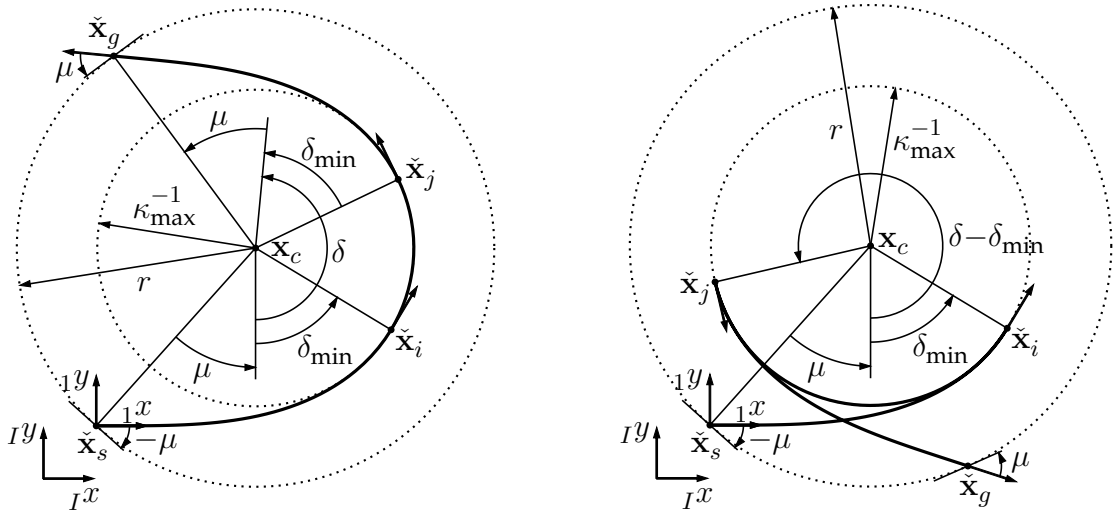
$$-\kappa_{\max} \leq \check{\kappa}_{k+1} \leq \kappa_{\max}, \tag{2.43a}$$

$$-\sigma_{\max} \leq \check{\sigma}_{k+1} \leq \sigma_{\max}. \tag{2.43b}$$

The goal now is to derive a steering function that takes into account the motion model of the CCR car while minimizing the length of the computed path. To do so, a similar approach as for RS and CC steer is pursued. This requires to encapsulate the system's nonlinear turning behavior into the so-called CCR turns, which are presented in the next section.

2.4.1.2 Continuous Curvature Rate Turns

This section introduces CCR turns as the basic component of CCR steer. The general objective is to come up with a modular description of the CCR car's turning behavior, which allows to abstract away the nonlinearity of the system. The resulting CCR turns are closely related to the previously described CC turns, however, differ in complexity due to the required increase in smoothness.

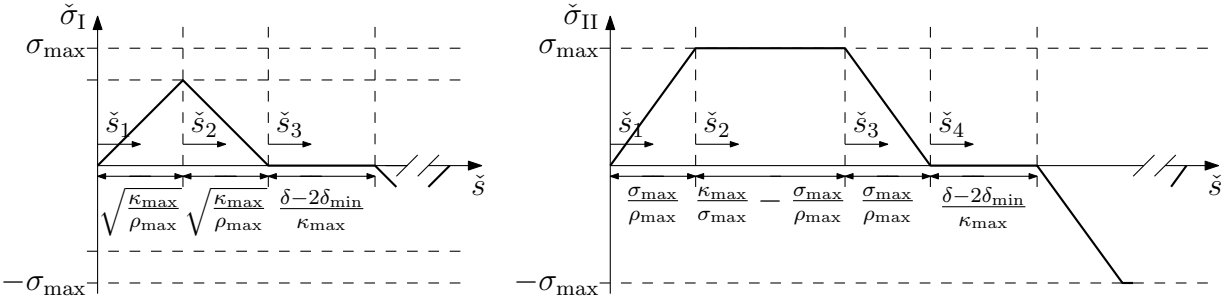


(a) Regular CCR turn. **(b)** Irregular CCR turn.

Figure 2.30 Visualization of a regular and irregular CCR turn for a forward motion to the left. The turns connect the start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ with a set of cubic spirals and a circular arc. The irregular turn results in a shorter connection for $\delta > 2\delta_{\min} + \pi$.

As visualized in Fig. 2.30, a CCR turn starts in $\check{\mathbf{x}}_s$ with zero curvature and zero curvature rate. Next, it transitions the vehicle on a set of cubic spirals to

the intermediate state $\check{\mathbf{x}}_i$, where it reaches the maximum curvature $\kappa_{\max}$. The CCR turn then moves the vehicle on a circular arc to the second intermediate state $\check{\mathbf{x}}_j$. Finally, the goal state $\check{\mathbf{x}}_g$ with zero curvature and zero curvature rate is reached on another set of cubic spirals. The corresponding curvature rate profile of the described motion is given in Fig. 2.31. Note that due to the symmetry of the CCR turns, only the transition between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_i$ is further detailed in the following.



(a) Case I. **(b)** Case II.

Figure 2.31 Curvature rate profile of a CCR turn to the left. The first case (a) has to be applied if $\kappa_{\max} \leq \sigma_{\max}^2/\rho_{\max}$ and the second case (b) for all other parameterizations.

First of all, it can be seen in Fig. 2.31 that two cases have to be distinguished for the transition between the start and intermediate state. Inspired by CC turns, both cases aim at transitioning the vehicle as fast as possible to the maximum curvature while enforcing hard constraints on the maximum curvature rate and the maximum curvature acceleration. If $\kappa_{\max} \leq \sigma_{\max}^2/\rho_{\max}$, the maximum curvature can be reached with no saturation of the curvature rate as illustrated in Fig. 2.31(a). In this case, two cubic spirals of maximum curvature acceleration $\pm\rho_{\max}$ are used to connect $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_i$. In contrast to that, $\kappa_{\max} > \sigma_{\max}^2/\rho_{\max}$ leads to the second case visualized in Fig. 2.31(b). Here, three cubic spirals are concatenated such that the maximum curvature rate $\sigma_{\max}$ is taken into account.

The piecewise affine functions that describe the continuous curvature rate profiles in Fig. 2.31 can now be derived. In the first case, they are given as

$$\check{\sigma}_{\mathrm{I}}(\check{s}_1) = \rho_{\max}\check{s}_1, \qquad \check{s}_1 \in [0, \sqrt{\tfrac{\kappa_{\max}}{\rho_{\max}}}], \tag{2.44a}$$

$$\check{\sigma}_{\mathrm{I}}(\check{s}_2) = -\rho_{\max}\check{s}_2 + \sqrt{\kappa_{\max}\rho_{\max}}, \qquad \check{s}_2 \in [0, \sqrt{\tfrac{\kappa_{\max}}{\rho_{\max}}}], \tag{2.44b}$$

and in the second case as

$$\check{\sigma}_{\mathrm{II}}(\check{s}_1) = \rho_{\max}\check{s}_1, \qquad \check{s}_1 \in [0, \tfrac{\sigma_{\max}}{\rho_{\max}}], \tag{2.45a}$$

$$\check{\sigma}_{\mathrm{II}}(\check{s}_2) = \sigma_{\max}, \qquad \check{s}_2 \in [0, \tfrac{\kappa_{\max}}{\sigma_{\max}} - \tfrac{\sigma_{\max}}{\rho_{\max}}], \tag{2.45b}$$

$$\check{\sigma}_{\mathrm{II}}(\check{s}_3) = -\rho_{\max}\check{s}_3 + \sigma_{\max}, \qquad \check{s}_3 \in [0, \tfrac{\sigma_{\max}}{\rho_{\max}}], \tag{2.45c}$$

where $\check{s}_1$, $\check{s}_2$, and $\check{s}_3$ denote the respective arc length as illustrated in Fig. 2.31. Integrating (2.44)–(2.45) leads to a quadratic curvature profile, which is defined as

$$\check{\kappa}_{\mathrm{I}}(\check{s}_1) = \frac{\rho_{\max}}{2}\check{s}_1^2, \tag{2.46a}$$

$$\check{\kappa}_{\mathrm{I}}(\check{s}_2) = -\frac{\rho_{\max}}{2}\check{s}_2^2 + \sqrt{\kappa_{\max}\rho_{\max}}\check{s}_2 + \frac{\kappa_{\max}}{2}, \tag{2.46b}$$

and

$$\check{\kappa}_{\mathrm{II}}(\check{s}_1) = \frac{\rho_{\max}}{2}\check{s}_1^2, \tag{2.47a}$$

$$\check{\kappa}_{\mathrm{II}}(\check{s}_2) = \sigma_{\max}\check{s}_2 + \frac{\sigma_{\max}^2}{2\rho_{\max}}, \tag{2.47b}$$

$$\check{\kappa}_{\mathrm{II}}(\check{s}_3) = -\frac{\rho_{\max}}{2}\check{s}_3^2 + \sigma_{\max}\check{s}_3 + \kappa_{\max} - \frac{\sigma_{\max}^2}{2\rho_{\max}}, \tag{2.47c}$$

where the properties $\kappa_s = 0$ and $\kappa_i = \kappa_{\max}$ are used. Another integration of (2.46)–(2.47) results in the heading angle $\check{\theta}(\check{s})$, which is, however, omitted here for brevity. Note that, hereinafter, the indices I and II of the different cases above are dropped for better readability.

Similar to the CC turn, it is distinguished here between a regular and irregular CCR turn as shown in Fig. 2.30. In contrast to the regular CCR turn, the irregular one allows a direction switch at the intermediate states $\check{\mathbf{x}}_i$ and $\check{\mathbf{x}}_j$. By doing so, a shorter connection between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_g$ can be generated for $\delta > 2\delta_{\min} + \pi$, where the minimal deflection $\delta_{\min}$ is defined as

$$\delta_{\min} = \begin{cases} \sqrt{\frac{\kappa_{\max}^3}{\rho_{\max}}}, & \text{if } \kappa_{\max} \leq \sigma_{\max}^2/\rho_{\max}, \\ \frac{\kappa_{\max}^2}{2\sigma_{\max}} + \frac{\kappa_{\max}\sigma_{\max}}{2\rho_{\max}}, & \text{else.} \end{cases} \tag{2.48}$$

The remaining variables of both turns (see Fig. 2.30) can now be derived given the vehicle's initial gear $g \in \{-1, 1\}$ (backwards, forwards), the desired turning direction $t \in \{-1, 1\}$ (right, left), the start state $\check{\mathbf{x}}_s$, and the deflection δ. At this point, the focus lies on those variables that are calculated

differently than the ones in Sec. 2.3.1.2. The first intermediate state $\check{\mathbf{x}}_i$, for instance, can be obtained in the local frame 1 (see Fig. 2.30) by

$$ {}_1\check{\mathbf{x}}_i = \begin{pmatrix} {}_1\check{x}_i \\ {}_1\check{y}_i \\ {}_1\check{\theta}_i \\ {}_1\check{\kappa}_i \\ {}_1\check{\sigma}_i \end{pmatrix} = \begin{pmatrix} g \int_0^{l_{\min}} \cos({}_1\check{\theta}(s)) \ \mathrm{d}s \\ t \int_0^{l_{\min}} \sin({}_1\check{\theta}(s)) \ \mathrm{d}s \\ g t \delta_{\min} \\ t \kappa_{\max} \\ 0 \end{pmatrix}, \tag{2.49} $$

where the length $l_{\min}$ of the transition between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_i$ is given as

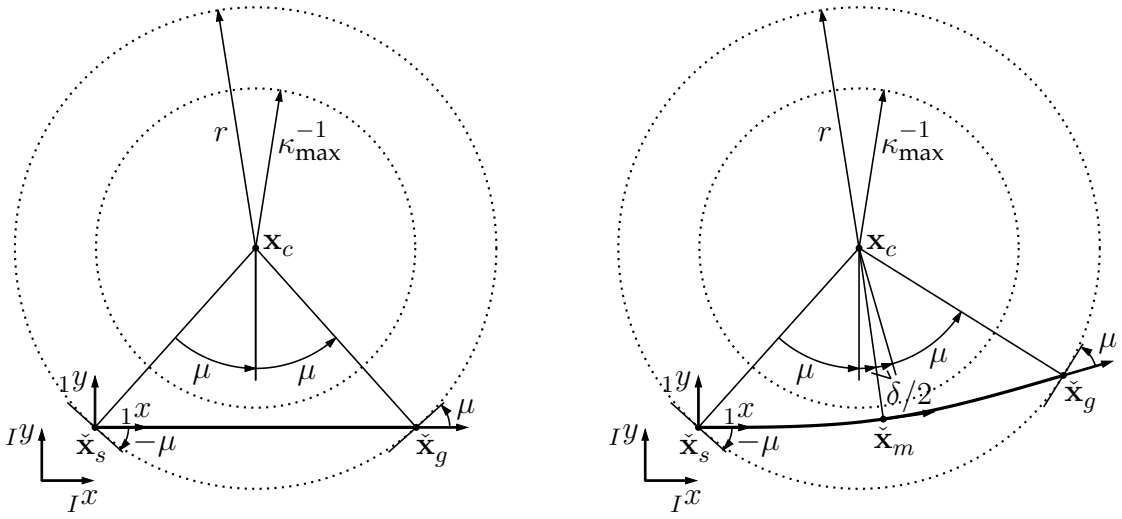
$$ l_{\min} = \begin{cases} \sqrt{\frac{4\kappa_{\max}}{\rho_{\max}}}, & \text{if } \kappa_{\max} \leq \sigma_{\max}^2 / \rho_{\max}, \\ \frac{\kappa_{\max}}{\sigma_{\max}} + \frac{\sigma_{\max}}{\rho_{\max}}, & \text{else.} \end{cases} \tag{2.50} $$

The second intermediate state $\check{\mathbf{x}}_j$ and the goal state $\check{\mathbf{x}}_g$, both with zero curvature rate, are computed according to (2.19)–(2.20). All states $\check{\mathbf{x}}_{(\bullet)}$ can be transformed from the local frame 1 to the inertial frame I (see Fig. 2.30) with

$$ {}_I\check{\mathbf{x}}_{(\bullet)} = \begin{pmatrix} {}_I\check{x}_s \\ {}_I\check{y}_s \\ {}_I\check{\theta}_s \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \cos({}_I\check{\theta}_s) & -\sin({}_I\check{\theta}_s) & 0 & 0 & 0 \\ \sin({}_I\check{\theta}_s) & \cos({}_I\check{\theta}_s) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}_1\check{x}_{(\bullet)} \\ {}_1\check{y}_{(\bullet)} \\ {}_1\check{\theta}_{(\bullet)} \\ {}_1\check{\kappa}_{(\bullet)} \\ {}_1\check{\sigma}_{(\bullet)} \end{pmatrix}. \tag{2.51} $$

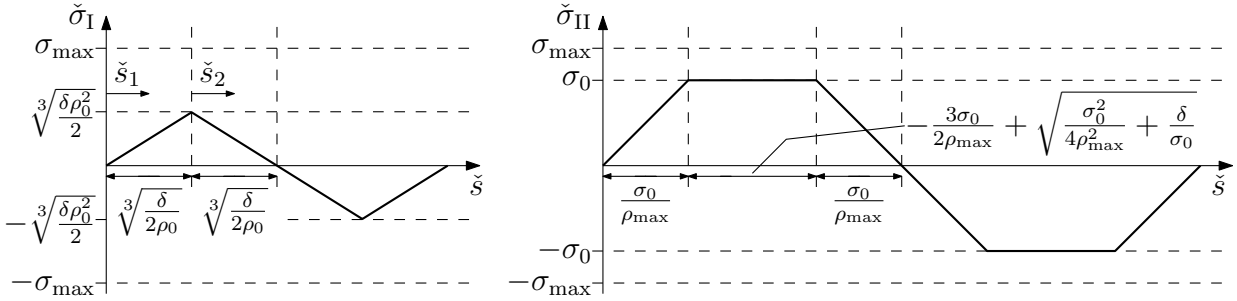
As already known from the CC turn, two special cases exist for $\delta < 2\delta_{\min}$ (see Fig. 2.32) that result in a shorter overall arc length than the regular and irregular case described above. For example, if the deflection between $\check{\mathbf{x}}_s$ and $\check{\mathbf{x}}_g$ is zero, the CCR turn reduces to a straight line as illustrated in Fig. 2.32(a). For $0 < \delta < 2\delta_{\min}$, a G^3 continuous elementary path can be constructed that connects the start and goal state without steering the CCR car to maximum curvature. This is illustrated in Fig. 2.32(b). In contrast to the G^2 continuous elementary path from Sec. 2.3.1.2, two cases have to be considered here as visualized in Fig. 2.33. Note that the symmetry of the elementary path allows to only detail the left part of the illustrated continuous curvature rate profiles in the following.

As it can be seen in Fig. 2.33(a), the first elementary path concatenates two cubic spirals in order to connect the start state $\check{\mathbf{x}}_s$ with the intermediate state $\check{\mathbf{x}}_m$ (see Fig. 2.32(b)). For a turn to the left, the corresponding curvature rate profile initially accelerates with $\rho_0 \leq \rho_{\max}$ and then decelerates back to



(a) Straight line. **(b)** Elementary path.

Figure 2.32 Two special cases of the CCR turn for small deflections: a straight line for $\delta = 0$ and an elementary path for $0 < \delta < 2\delta_{\min}$.



(a) Elementary path I. **(b)** Elementary path II.

Figure 2.33 Curvature rate profiles of a G^3 continuous elementary path to the left.

zero curvature rate with $-\rho_0$. The arc length of both cubic spirals is given in Fig. 2.33(a) using the properties $\check{\kappa}_s = 0$ and $\check{\theta}_m = \delta/2$. Contrary to this, the second elementary path in Fig. 2.33(b) uses three cubic spirals to reach $\check{\mathbf{x}}_m$. In this case, the first and the third cubic spiral accelerate with $\pm\rho_{\max}$ while the second one saturates the curvature rate at $\sigma_0 \leq \sigma_{\max}$. The same properties as before can be applied to determine the arc length of the cubic spirals given in Fig. 2.33(b). Both elementary paths have exactly one unknown parameter, namely ρ_0 and σ_0. They can be computed by projecting $\check{\mathbf{x}}_m$ (see Fig. 2.32(b)) onto the line joining the start state $\check{\mathbf{x}}_s$ and the goal state $\check{\mathbf{x}}_g$ according to

$$\begin{pmatrix}\cos(\delta/2)\\ \sin(\delta/2)\end{pmatrix}^{\mathsf{T}} \cdot \begin{pmatrix}\check{x}_m\\ \check{y}_m\end{pmatrix} = \frac{1}{2}\left\| \begin{pmatrix}\check{x}_g\\ \check{y}_g\end{pmatrix} - \begin{pmatrix}\check{x}_s\\ \check{y}_s\end{pmatrix} \right\|_2, \tag{2.52}$$

where $\begin{pmatrix}\check{x}_m & \check{y}_m\end{pmatrix}$, $\begin{pmatrix}\check{x}_g & \check{y}_g\end{pmatrix}$, and $\begin{pmatrix}\check{x}_s & \check{y}_s\end{pmatrix}$ denote the positions of the states $\check{\mathbf{x}}_m$, $\check{\mathbf{x}}_g$, and $\check{\mathbf{x}}_s$. The right side in (2.52) can be replaced with the term $r\sin(\delta/2+\mu)$ (see Fig. 2.32(b)) resulting in

$$\check{x}_m\cos(\delta/2)+\check{y}_m\sin(\delta/2)=r\sin(\delta/2+\mu). \tag{2.53}$$

In order to solve (2.53) for the unknown parameter ρ_0 and σ_0, respectively, $\check{x}_m$ and $\check{y}_m$ have to be substituted by their corresponding integral. In case of the first elementary path (see Fig. 2.33(a)), these integrals are given as

$$\begin{aligned}\check{x}_{m,\mathrm{I}} = \sqrt[3]{\tfrac{6}{\rho_0}}\int_0^{\sqrt[3]{\frac{\delta}{12}}} &\cos\left(\check{t}^3\right)+\ldots \\ &+\cos\left(-\check{t}^3+\sqrt[3]{\tfrac{9\delta}{4}}\check{t}^2+\sqrt[3]{\tfrac{3\delta^2}{16}}\check{t}+\tfrac{\delta}{12}\right)\,\mathrm{d}\check{t},\end{aligned} \tag{2.54a}$$

$$\begin{aligned}\check{y}_{m,\mathrm{I}} = \sqrt[3]{\tfrac{6}{\rho_0}}\int_0^{\sqrt[3]{\frac{\delta}{12}}} &\sin\left(\check{t}^3\right)+\ldots \\ &+\sin\left(-\check{t}^3+\sqrt[3]{\tfrac{9\delta}{4}}\check{t}^2+\sqrt[3]{\tfrac{3\delta^2}{16}}\check{t}+\tfrac{\delta}{12}\right)\,\mathrm{d}\check{t},\end{aligned} \tag{2.54b}$$

where the initial integration variables $\check{s}_1$ and $\check{s}_2$ (see Fig. 2.33(a)) are substituted by $\check{s}_1=\check{s}_2=\sqrt[3]{6/\rho_0}\check{t}$. Combining (2.53) and (2.54) allows to compute the parameter ρ_0 according to

$$\rho_0=\frac{6D_{\mathrm{I}}^3(\delta)}{\left(r\sin(\delta/2+\mu)\right)^3}, \tag{2.55}$$

where the function $D_{\mathrm{I}}(\delta)$ is given as

$$\begin{aligned}D_{\mathrm{I}}(\delta) = \cos\left(\tfrac{\delta}{2}\right)\int_0^{\sqrt[3]{\frac{\delta}{12}}} &\cos\left(\check{t}^3\right)+\ldots \\ &+\cos\left(-\check{t}^3+\sqrt[3]{\tfrac{9\delta}{4}}\check{t}^2+\sqrt[3]{\tfrac{3\delta^2}{16}}\check{t}+\tfrac{\delta}{12}\right)\,\mathrm{d}\check{t}+\ldots \\ +\sin\left(\tfrac{\delta}{2}\right)\int_0^{\sqrt[3]{\frac{\delta}{12}}} &\sin\left(\check{t}^3\right)+\ldots \\ &+\sin\left(-\check{t}^3+\sqrt[3]{\tfrac{9\delta}{4}}\check{t}^2+\sqrt[3]{\tfrac{3\delta^2}{16}}\check{t}+\tfrac{\delta}{12}\right)\,\mathrm{d}\check{t}.\end{aligned} \tag{2.56}$$

A visualization of D_{I} in the interval $\delta\in[0,2\pi)$ can be found in Fig. 2.34.

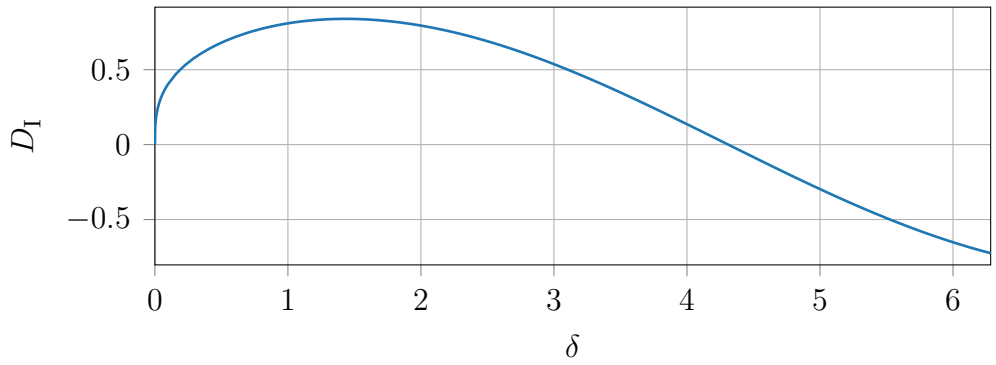


Figure 2.34 Visualization of D_I in the interval $\delta \in [0, 2\pi)$. The root of this function lies at $\approx 1.3721\pi$.

Once ρ_0 is computed using (2.55)–(2.56), it must be verified that the obtained solution does not violate the constraints of the CCR car. Unfortunately, it cannot be generally guaranteed that this is the case for arbitrary deflections $0 < \delta < 2\delta_\mathrm{min}$.

With respect to the second elementary path (see Fig. 2.33(b)), the unknown parameter σ_0 must also be determined on the basis of (2.53). In contrast to the previous case, the resulting equation does not possess an explicit solution and thus requires a numerical root-finding algorithm to obtain σ_0. Similar to the first elementary path, it has to be verified afterwards that the resulting solution is compliant with the constraints of the CCR car.

As neither the first nor the second elementary path can guarantee a feasible solution for $0 < \delta < 2\delta_\mathrm{min}$, the regular or irregular CCR turn have to be used as backup to avoid situations without a connection. The resulting arc length $l(\delta)$ for $0 < \delta < 2\delta_\mathrm{min}$ can therefore be derived as

$$l(\delta) = \begin{cases} \sqrt[3]{\frac{32\delta}{\rho_0}}, & \text{if } \rho_0 \neq \emptyset, \\ \frac{\sigma_0}{\rho_\mathrm{max}} + \sqrt{\frac{\sigma_0^2}{\rho_\mathrm{max}^2} + \frac{4\delta}{\sigma_0}}, & \text{elif } \sigma_0 \neq \emptyset, \\ 2l_\mathrm{min} + \kappa_\mathrm{max}^{-1}(-\delta + 2\delta_\mathrm{min}), & \text{elif irregular CCR turn}, \\ 2l_\mathrm{min} + \kappa_\mathrm{max}^{-1}(2\pi + \delta - 2\delta_\mathrm{min}), & \text{else (regular CCR turn)}, \end{cases} \tag{2.57}$$

where the first two cases describe the arc length of the two elementary paths and the last two cases the one of a regular or irregular CCR turn.

Based on the previous derivations, the arc length $l(\delta)$ of the CCR turn can now be determined in the interval $\delta \in [0, 2\pi)$. In case of a regular CCR turn,

it is given by

$$l(\delta) = \begin{cases} 2r\sin(\mu), & \text{if } \delta = 0, \\ \text{see (2.57)}, & \text{if } 0 < \delta < 2\delta_{\min}, \\ 2l_{\min} + \kappa_{\max}^{-1}(\delta - 2\delta_{\min}), & \text{if } \delta \geq 2\delta_{\min}. \end{cases} \tag{2.58}$$

Similarly, the arc length of an irregular CCR turn is defined as

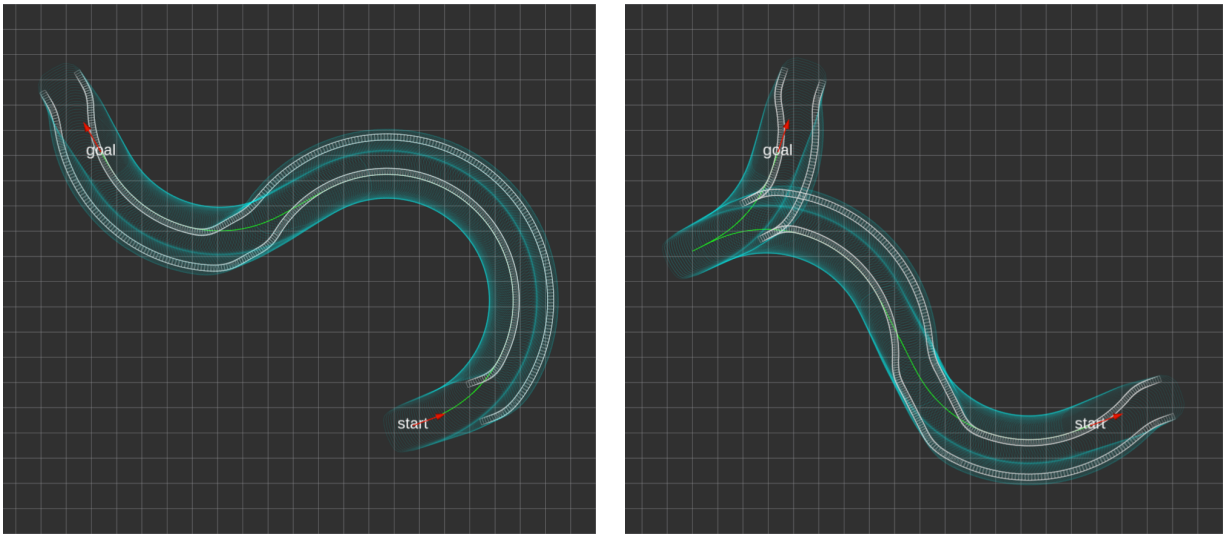
$$l(\delta) = \begin{cases} 2r\sin(\mu), & \text{if } \delta = 0, \\ \text{see (2.57)}, & \text{if } 0 < \delta < 2\delta_{\min}, \\ 2l_{\min} + \kappa_{\max}^{-1}(\delta - 2\delta_{\min}), & \text{if } 2\delta_{\min} \leq \delta \leq 2\delta_{\min} + \pi, \\ 2l_{\min} + \kappa_{\max}^{-1}(2\pi - \delta + 2\delta_{\min}), & \text{if } \delta > 2\delta_{\min} + \pi, \end{cases} \tag{2.59}$$

where the last case considers the two direction switches at the intermediate states $\check{\mathbf{x}}_i$ and $\check{\mathbf{x}}_j$. The derived CCR turns can now be used to compute G^3 continuous paths as further outlined in the following section.

2.4.1.3 Continuous Curvature Rate Path

The modularity of CCR turns allows to concatenate them along with straight lines to a CCR path. The underlying procedure can be directly adopted from CC steer (see Sec. 2.3.1.3) with the only difference that CC turns must be replaced with CCR turns. The resulting steering function is called CCR steer and minimizes path length while satisfying hard constraints on the maximum curvature, maximum curvature rate, and maximum curvature acceleration. Two variants of CCR steer can be distinguished: CCR-Dubins, which only allows the vehicle to move forwards or backwards, and CCR-RS, which does not restrict the driving direction. Similar to their G^2 continuous counterpart, both steering functions can be evaluated very efficiently if the start and goal state are given with either zero or maximum curvature and without curvature rate. For instance, CCR^{00}-Dubins and CCR^{00}-RS steer enforce zero curvature at the start and goal state as indicated by the corresponding superscript. In this case, zero curvature rate is implicitly assumed at both ends of the path and not further annotated in the description of the steering function. An illustration of a CCR^{00}-Dubins and CCR^{00}-RS steering procedure is shown in Fig. 2.35.

It can be observed that CCR^{00}-Dubins steer selects a CSC candidate path as an approximation of the shortest connection between start and goal while



(a) CCR^{00}-Dubins path. **(b)** CCR^{00}-RS path.

Figure 2.35 Visualization of two G^3 continuous shortest path approximations for a CCR car that only moves forwards (a) and both forwards and backwards (b). The front wheels of the vehicle are displayed in white, its contour in cyan, and the computed path in green.

CCR^{00}-RS steer outputs a path of the family $CSC|C$. The G^3 continuity of both paths is displayed in Fig. 2.36, where the corresponding curvature profile is plotted with respect to arc length $\check{s}$.

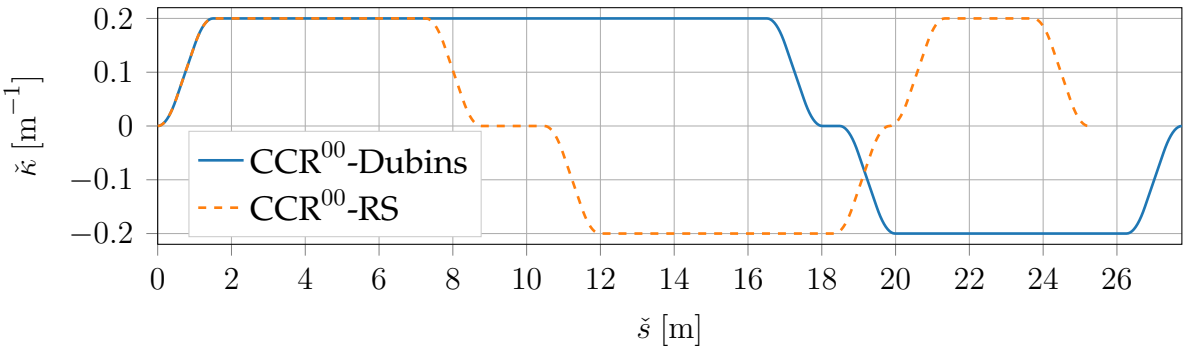


Figure 2.36 Curvature profile of the CCR^{00}-Dubins and CCR^{00}-RS path from Fig. 2.35.

It can be seen that both profiles are given by smooth quadratic functions that are bounded by the maximum curvature of the vehicle. Additionally, the first and second derivative of these functions are limited by the maximum curvature rate and the maximum curvature acceleration. Hence, the computed G^3 continuous paths can be directly executed by the CCR car.

Finally, it has to be mentioned that CCR steer's characteristic with respect to completeness, symmetry, and the topological property are identical to CC steer (see Sec. 2.3.1.3) and therefore not further discussed here.

2.4.2 Hybrid Curvature Rate Steer

Following the same motivation as in Sec. 2.3.2, HCR steer enforces G^3 continuity while the vehicle is moving either forwards or backwards, but allows curvature discontinuity at cusps. This hybrid approach combines the properties of RS steer at direction switches and the ones of CCR steer everywhere else. As a result, the computed paths are not only smoother than RS paths, but also shorter than the ones of CCR steer. Further details with respect to HCR steer are outlined in the following: Sec. 2.4.2.1 describes the underlying motion model, Sec. 2.4.2.2 introduces HCR turns as one of HCR steer's fundamental components, and Sec. 2.4.2.3 finally discusses the computation of a HCR path.

2.4.2.1 Hybrid Curvature Rate Car

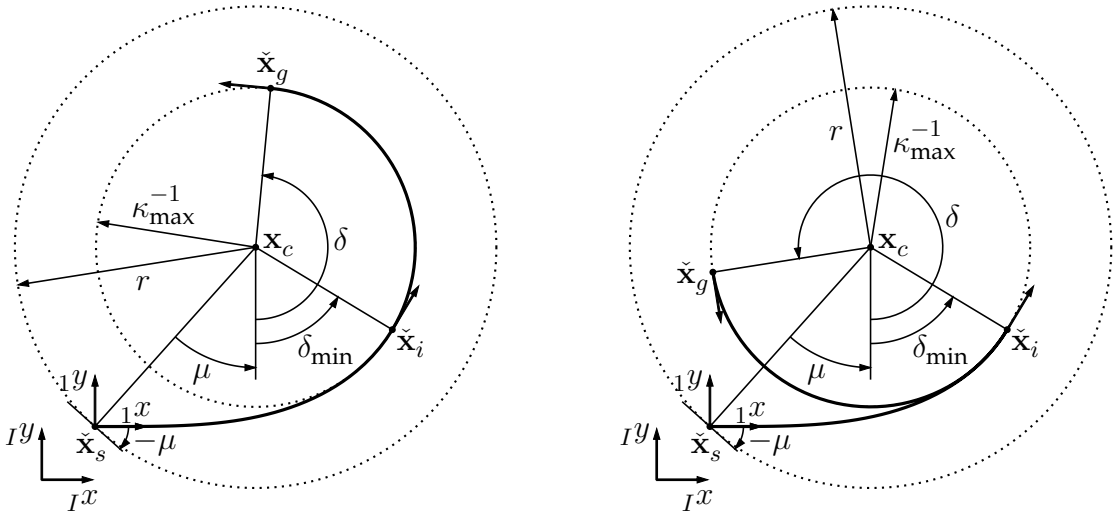
The so-called HCR car extends the CCR car from Sec. 2.4.1.1 with the capability to also steer the wheels in place while the vehicle is changing its driving direction. The evolution of motion between cusps is identical to the one of the CCR car, which bounds the maximum curvature, the maximum curvature rate, and the maximum curvature acceleration. In order to also take into account the described behavior at direction switches, the arc length dependent motion model (2.41) requires to directly modify the vehicle's curvature and possibly its curvature rate to the desired values. Hence, the HCR car behaves similar to the RS car at cusps and identical to the CCR car everywhere else.

The aim of the following sections is to develop a steering function that connects two vehicle states with a shortest path approximation and leverages the characteristics of the HCR car. To accomplish this in a similar way as in the previously described steering functions, it is required to introduce HCR turns next as one of HCR steer's fundamental components.

2.4.2.2 Hybrid Curvature Rate Turns

The motivation behind the so-called HCR turns in this section is to smoothly transition the HCR car from zero curvature to maximum curvature while taking into account its constraints. Terminating in a state with maximum

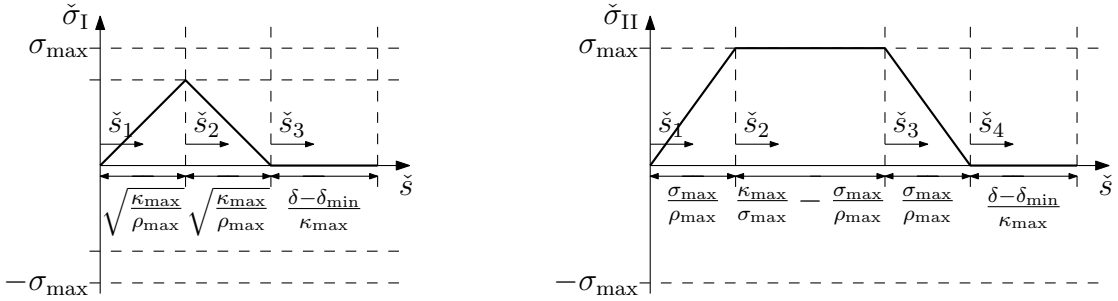
curvature allows to generate paths that permit steering in place at cusps. An exemplary visualization of two HCR turns can be found in Fig. 2.37.



(a) Regular HCR turn. **(b)** Irregular HCR turn.

Figure 2.37 Illustration of a regular and irregular HCR turn for a forward motion to the left. Cubic spirals and a circular arc are used to connect the start state $\check{\mathbf{x}}_s$ with the goal state $\check{\mathbf{x}}_g$. The irregular turn yields a shorter arc length for $\delta < \delta_{\min}$ and $\delta > \delta_{\min} + \pi$.

Similar to the CCR turn in Sec. 2.4.1.2, the HCR turn starts in $\check{\mathbf{x}}_s$ with zero curvature and zero curvature rate. It then transitions to the intermediate state $\check{\mathbf{x}}_i$, where it reaches the maximum curvature $\kappa_{\max}$, using a set of up to three cubic spirals. The goal state $\check{\mathbf{x}}_g$ with maximum curvature and zero curvature rate is finally reached on a circular arc. Fig. 2.38 illustrates the corresponding curvature rate profile including the two cases that are already known from the CCR turn.



(a) Case I. **(b)** Case II.

Figure 2.38 Curvature rate profile of a HCR turn to the left. The first case (a) has to be applied if $\kappa_{\max} \leq \sigma_{\max}^2/\rho_{\max}$ and the second case (b) for all other parameterizations.

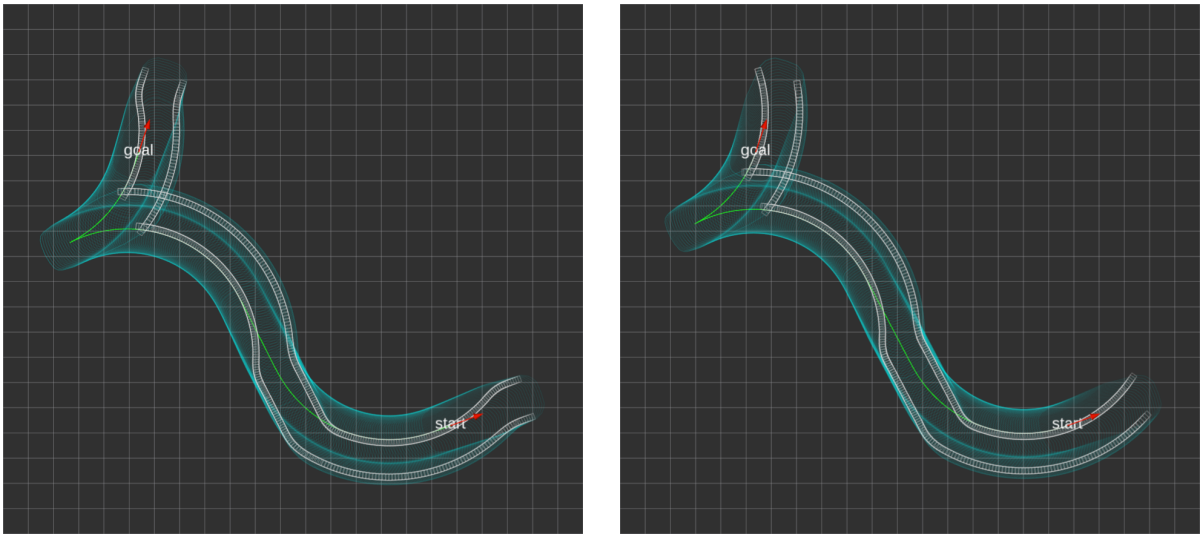
Apart from the length of the circular arc, both profiles are identical to the first part of the CCR turn's curvature rate profiles given in Fig. 2.31 and therefore not further analyzed here. Also, the distinction in Fig. 2.37 between a regular and an irregular HCR turn is not further elaborated in this section as it is identical to the already known one from HC turns (see Sec. 2.3.2.2).

The variables in Fig. 2.37, which completely define a HCR turn, can all be computed using the already derived equations from the previous sections. For instance, the computation of the intermediate state $\check{\mathbf{x}}_i$ is given in (2.49), and the evaluation of the goal state $\check{\mathbf{x}}_g$ with zero curvature rate can be conducted according to (2.30). The arc length of a regular and irregular HCR turn is calculated in the same way as in (2.31)–(2.32), where the variables $\delta_{\min}$ and $l_{\min}$ have to be replaced by the respective correlation given in (2.48) and (2.50). On this basis, G^3 continuous HCR paths can now be computed as further detailed in the next section.

2.4.2.3 Hybrid Curvature Rate Path

The computation of a HCR path that enforces G^3 continuity everywhere except at cusps follows the same procedure as the one described for HC steer in Sec. 2.3.2.3. The only difference is that the G^2 continuous CC and HC turns are replaced by their G^3 continuous counterparts, namely CCR and HCR turns. Similar to the G^2 continuous case, the resulting steering function can be evaluated in a particularly efficient way if (1) the curvature rate vanishes at the start and goal state, and (2) the curvature at both ends of the path takes either the value zero or $\pm\kappa_{\max}$. These cases are highlighted in the following with the already known superscript that denotes the respective curvature at the initial and terminal state. For example, HCR^{00}-RS steer computes a path with zero curvature at the start and goal, and HCR$^{\pm\pm}$-RS steer either chooses $\pm\kappa_{\max}$ at both ends of the path. An exemplary visualization of two shortest path approximations computed by these steering functions can be found in Fig. 2.39.

It can be seen that both steering procedures select a $CSC|C$ candidate path to steer the vehicle to the goal. The track of the front wheels reveals that a curvature discontinuity occurs at the direction switch while G^3 continuity is enforced everywhere else. This can also be seen in Fig. 2.40, which illustrates the curvature profile of the two paths with respect to the traveled distance $\check{s}$. Both profiles are described by a smooth quadratic function that takes into account the maximum curvature, the maximum curvature rate,



(a) HCR^{00}-RS path. **(b)** HCR$^{\pm\pm}$-RS path.

Figure 2.39 Illustration of two G^3 continuous shortest path approximations for a HCR car. Zero curvature at both ends of the path is enforced in (a) while the path in (b) chooses either $\pm\kappa_{\max}$ as the initial and terminal curvature. The colors are adopted from Fig. 2.35.

and the maximum curvature acceleration of the vehicle. In addition to that, a curvature discontinuity can be observed at about 19 m, which corresponds to the direction switch shown in Fig. 2.39. As a result, the computed paths satisfy the constraints of the HCR car and can therefore be directly executed by such a system.

Furthermore, it is worth mentioning that with respect to completeness, symmetry, and the topological property, HCR steer behaves identical to HC steer. Therefore, the reader is referred to Sec. 2.3.2.3 and Sec. 2.3.2.4 for further details.

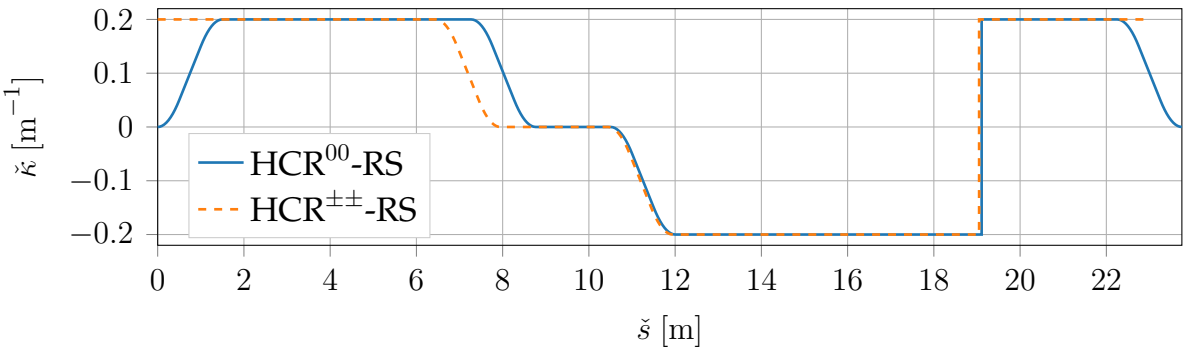
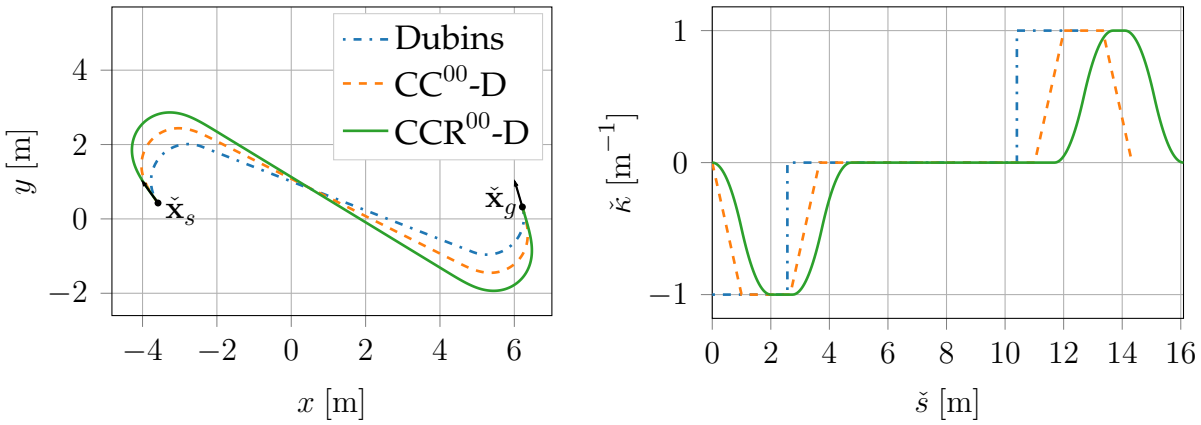


Figure 2.40 Curvature profile of the HCR^{00}-RS and HCR$^{\pm\pm}$-RS path from Fig. 2.39.

2.4.3 Experimental Evaluation

The experiments in this section evaluate and compare the performance of the G^3 continuous steering functions with their G^1 and G^2 continuous counterparts. The setup for the evaluation is the same as in Sec. 2.3.4, and the additional parameter $\rho_{\max}$ is set to $1\,\mathrm{m}^{-3}$. Furthermore, all G^3 continuous steering functions are allowed to use the irregular variant of the previously introduced CCR and HCR turns in order to further optimize the length of the computed paths. The implementation of CCR and HCR steer is based on the open-source code from [208] that has been extended to also compute G^3 continuous paths.

The qualitative comparison in Fig. 2.41 highlights how an increase in geometric continuity influences the computed path and the corresponding curvature profile.



(a) Path comparison. **(b)** Curvature profiles.

Figure 2.41 Qualitative comparison of a G^1, G^2, and G^3 continuous Dubins (D) steering procedure. The computed paths are shown in (a) and the corresponding curvature profiles in (b). Note that the legend in (a) also applies to (b).

It can be seen in Fig. 2.41(a) that compared to the provably shortest path of the G^1 continuous approach, increasing the smoothness also increases the length of the path. This effect depends strongly on the vehicle's maximum curvature rate $\sigma_{\max}$ and its maximum curvature acceleration $\rho_{\max}$ and becomes larger if one of these parameters decreases. With respect to curvature continuity, Fig. 2.41(b) shows that the G^1 continuous solution is discrete in curvature while the G^2 and G^3 continuous solutions are described by a linear and a quadratic function, respectively. Especially the G^3 continuous steering function with its bounded curvature acceleration represents a promising ap-

proach whenever a high comfort or a high tracking performance is required.

However, it also has to be considered that a higher degree of smoothness increases the average computation time. This is shown in Tab. 2.6, which compares the computation times of the G^1, G^2, and G^3 continuous Dubins steering functions.

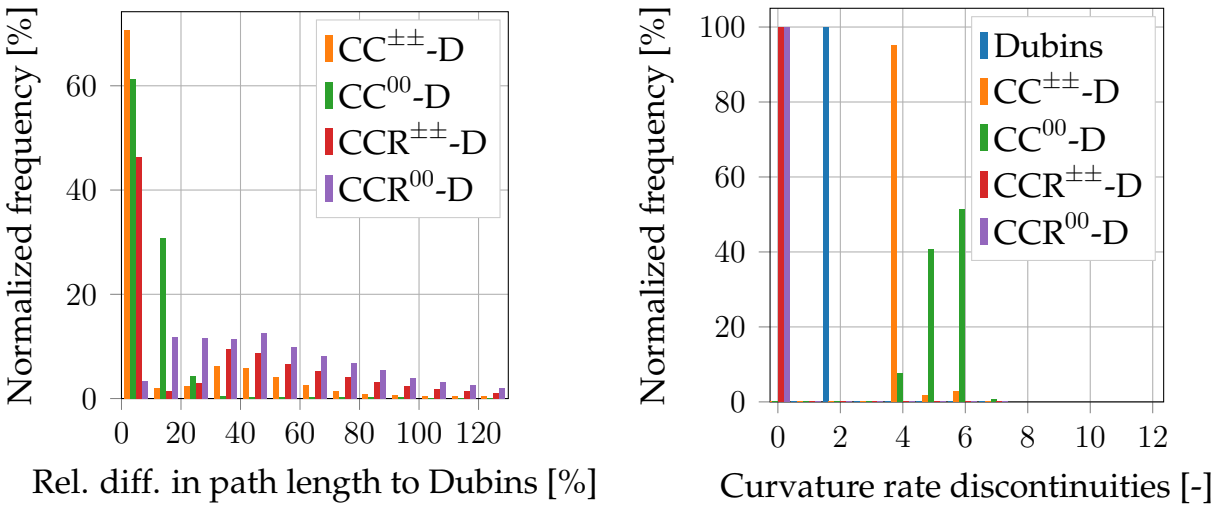
Table 2.6 Average computation time of the G^1, G^2, and G^3 continuous Dubins steering functions.

	computation time	
	mean [µs]	std [µs]
Dubins	1.29	±0.65
$CC^{\pm\pm}$-Dubins	7.16	±2.98
CC^{00}-Dubins	4.90	±1.67
$CCR^{\pm\pm}$-Dubins	10.85	±10.67
CCR^{00}-Dubins	13.12	±13.56

While the average computation time of all listed steering functions is below 13.12 µs, the G^3 continuous versions perform up to a factor of 2.7 slower than their respective G^2 continuous counterpart. In addition to that, the standard deviation raises to ±13.56 µs mostly due to the complexity of the G^3 continuous elementary paths.

As already discussed in the qualitative comparison above, increasing the smoothness of the paths also raises the path length. This effect is displayed for 10^5 random steering procedures in Fig. 2.42(a). It can be observed that the G^3 continuous steering function with maximum curvature at both ends of the path approximates the length of the Dubins paths more closely than the version with zero curvature at both states. This effect and the reasons behind it are identical to the G^2 continuous case that was analyzed in Sec. 2.3.4. Furthermore, it has to be noted that the width of the illustrated distribution in Fig. 2.42(a) significantly depends on the maximum curvature acceleration $\rho_{\max}$. In general, increasing $\rho_{\max}$ shifts the distribution of the G^3 continuous steering functions closer towards the one of the G^2 continuous steering functions that implicitly assume $\rho_{\max} = \infty$. The opposite is true for lower values of $\rho_{\max}$.

The major advantage of the G^3 continuous steering functions is that they enforce both curvature as well as curvature rate continuity. In contrast to that, the results in Fig. 2.42(b) show that the G^1 and G^2 continuous paths



(a) Path length comparison. **(b)** Curvature rate discont. comparison.

Figure 2.42 Comparison of the G^1, G^2, and G^3 continuous Dubins (D) steering functions with respect to path length (a) and curvature rate discontinuities (b).

contain between two and seven curvature rate discontinuities[5]. Remember that each discontinuity requires an infinite steering acceleration, which might cause a high lateral jerk or a potential deviation from the planned path when being executed in closed loop. Only in the G^1 continuous case, this can be overcome by bringing the vehicle to a stop as both the curvature and the curvature rate discontinuities occur together.

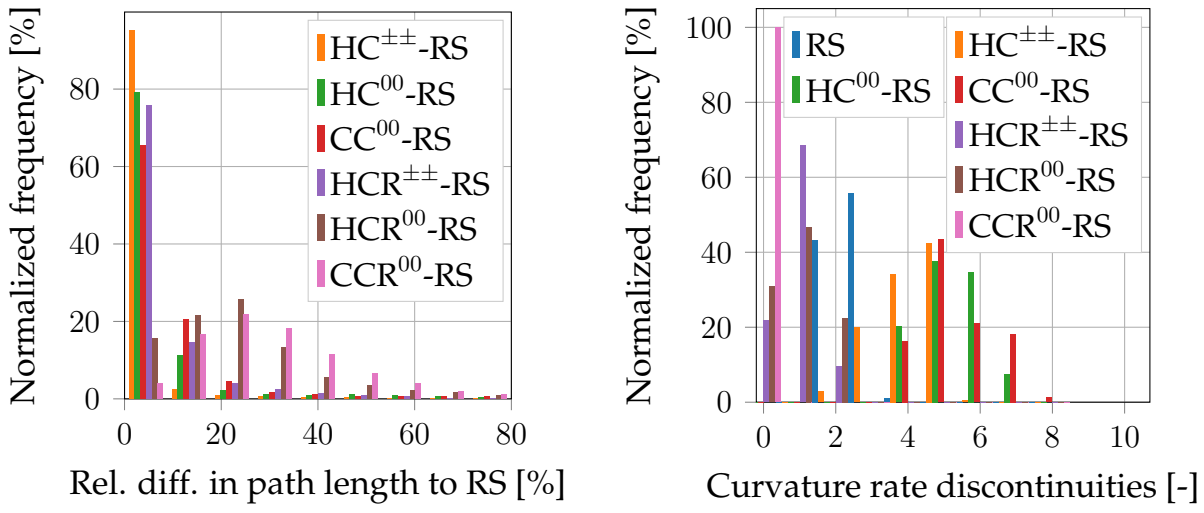
The same benchmark as above is also carried out for the G^1, G^2, and G^3 continuous RS steering functions. An overview of the average computation times is given in Tab. 2.7. It is interesting to see that HCR$^{\pm\pm}$-RS steer performs almost as fast as its G^2 continuous counterpart. As opposed to this, the mean computation time of CCR^{00}-RS steer is about 2.6 times higher than the one of CC^{00}-RS steer with a standard deviation that differs by a factor of 7.3. As the construction of HCR$^{\pm\pm}$-RS and CCR^{00}-RS paths only differ in the number of CCR turns used, it can be concluded that the additional complexity of the CCR turns causes the observed difference in performance. Especially the numerical computation of the G^3 continuous elementary paths currently slows down the evaluation of the CCR turns.

[5] As described at the beginning of this chapter, the term curvature rate denotes the first derivative of curvature with respect to arc length. This derivative is mathematically not defined at transitions between discrete curvatures. In order to still account for the infinitely high curvature rate impulses required to realize these transitions, every curvature discontinuity is also counted as a curvature rate discontinuity throughout this thesis.

Table 2.7 Average computation time of the G^1, G^2, and G^3 continuous RS steering functions.

	computation time	
	mean [µs]	std [µs]
RS	7.34	±1.69
$HC^{\pm\pm}$-RS	55.71	±9.49
HC^{00}-RS	53.74	±7.87
CC^{00}-RS	53.19	±7.93
$HCR^{\pm\pm}$-RS	59.28	±12.76
HCR^{00}-RS	88.90	±33.94
CCR^{00}-RS	139.83	±57.69

The path length comparison in Fig. 2.43(a) confirms the insight from above that an increase in geometric continuity also raises the length of the computed connections.



(a) Path length comparison. **(b)** Curvature rate discont. comparison.

Figure 2.43 Comparison of the G^1, G^2, and G^3 continuous RS steering functions with respect to path length (a) and curvature rate discontinuities (b).

However, it can also be observed that the length of the G^3 continuous RS paths deviates less from the shortest connection given by RS steer than in the Dubins benchmark before. This is due to the fact that the RS steering functions do not restrict the driving direction of the vehicle and also evaluate

more families than the respective Dubins counterparts.

The number of curvature rate discontinuities within the 10^5 evaluated paths is compared in Fig. 2.43(b). Here, it can be seen that CCR^{00}-RS steer enforces curvature rate continuity along the entire path. In contrast to that, HCR$^{\pm\pm}$-RS and HCR^{00}-RS steer allow to directly adjust the curvature at direction switches leading to the visualized number of curvature rate discontinuities. Note, however, that the latter is not a problem in practice as the vehicle has to stop at direction switches anyway allowing to smoothly adjust the steering angle to the desired value. In comparison to that, the G^2 continuous steering functions stand out with up to eight curvature rate discontinuities. Also in the G^1 continuous case, the number of curvature rate discontinuities ranges from two to four.

In conclusion, the G^3 continuous RS paths outperform the other state-of-the-art approaches with respect to smoothness, however, require more computation time and a higher path length in order to connect two given vehicle states.

2.5 Steering Functions in Belief Space

The steering functions in Sections 2.2–2.4 compute a sequence of $\bar{N}$ nominal inputs $\check{\mathcal{U}} = \langle \check{\mathbf{u}}_0, \dots, \check{\mathbf{u}}_{\bar{N}-1} \rangle$ that steer the vehicle to a desired goal. Given these inputs and the vehicle's initial state, the deterministic motion models of the form (2.1d) can then be used to obtain the corresponding states $\check{\mathcal{X}} = \langle \check{\mathbf{x}}_0, \dots, \check{\mathbf{x}}_{\bar{N}} \rangle$. In reality, however, various errors lead to a deviation from the planned path when it is executed by a motion controller in closed loop. To still guarantee a bounded collision probability, the goal in this section is to compute a belief distribution over vehicle states along the nominal path of the steering functions.

In this regard, two uncertainties are distinguished: (1) control/execution uncertainty resulting from model errors and external disturbances, such as wind forces or uneven road surfaces, and (2) localization uncertainty due to measurement errors. In the presence of both uncertainties, the deterministic motion model in (2.1d) changes to

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \tag{2.60a}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k), \tag{2.60b}$$

where $\mathbf{f}(\bullet)$ describes the evolution of the disturbed vehicle state $\mathbf{x}_k \in \mathbb{R}^n$ given the input $\mathbf{u}_k \in \mathbb{R}^m$ and the motion noise $\mathbf{w}_k \in \mathbb{R}^p$. The actual mea-

surement is denoted by $\mathbf{z}_k \in \mathbb{R}^q$, the measurement model by $\mathbf{h}(\bullet)$, and the measurement noise by $\mathbf{v}_k \in \mathbb{R}^r$. The input $\mathbf{u}_k$, which tries to keep the vehicle close to the nominal path, is computed by

$$\mathbf{u}_k = \check{\mathbf{u}}_k - \mathbf{k}(\boldsymbol{\mu}_k, \check{\mathbf{x}}_k), \tag{2.61}$$

where $-\mathbf{k}(\bullet)$ describes the feedback term that tries to minimize the error between the mean $\boldsymbol{\mu}_k \in \mathbb{R}^n$ of the state estimate and the desired nominal state $\check{\mathbf{x}}_k \in \mathbb{R}^n$. Note that in the presence of localization uncertainty, only a state estimate instead of the true vehicle state $\mathbf{x}_k$ is available to the system. On the basis of (2.60), such an estimate can be computed using e.g. the extended Kalman filter (EKF), the unscented Kalman filter (UKF), or the particle filter [186].

In order to capture the uncertainty of the vehicle motion, the goal now is to compute the belief distribution along the nominal path of a steering procedure (see Fig. 2.44). This information can then be used, for example, in

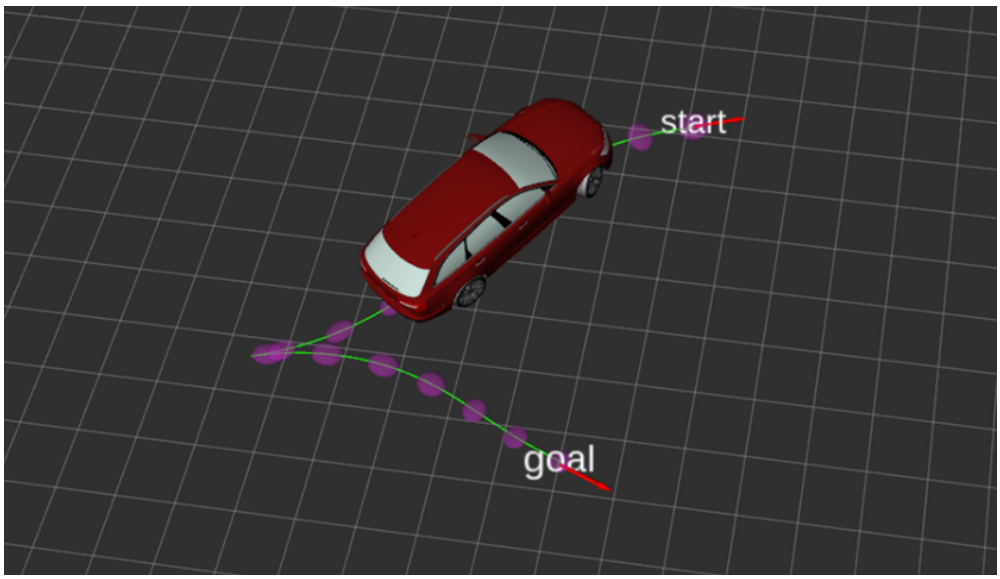


Figure 2.44 Visualization of a sequence of beliefs (purple ellipsoids) propagated along the nominal path (green line) of a steering procedure. In this image, the beliefs are represented by multivariate normal distributions that describe the uncertainty arising from control and localization errors. Reprinted from [213], © 2018 IEEE.

a probabilistic collision checker to evaluate the collision probability of the computed path. Following the definition in [186], a belief is defined here as

$$bel(\mathbf{x}_k) = p(\mathbf{x}_k | \mathbf{z}_{0:k}, \mathbf{u}_{0:k}), \tag{2.62}$$

where $p(\bullet)$ denotes the probability density function (PDF) of the true vehicle state $\mathbf{x}_k$ given the past measurements $\mathbf{z}_{0:k}$ and the inputs $\mathbf{u}_{0:k}$.

In the following sections, two approaches are highlighted for the computation of the beliefs. Sec. 2.5.1 presents a MC simulation that computes a discrete approximation of the true belief distribution. Despite the flexibility of this method, it is well known that MC approaches typically come with a high computational cost [84, 165]. As a result, they are either used as a reference method for comparison or require advanced techniques that exploit the parallelizability of the underlying problem [76]. More efficient, but also more restrictive methods exist in Gaussian belief space. Here, the belief is parametrized by a multivariate normal distribution. A closed-form solution for the belief update can then be derived as in [8, 23, 117, 145], where [117, 145] assume a maximum likelihood observation and [8, 23] consider all possible measurements. On this basis, the EKF-MP [23] is used in Sec. 2.5.2 to propagate a Gaussian belief along the nominal path of the steering functions.

2.5.1 Monte Carlo Simulation

As previously mentioned, the MC simulation computes a discrete approximation of the true belief distribution by propagating samples of the noise models through the system. This is shown in Alg. 1, where M closed-loop executions $\mathcal{X}^{[i=1:M]}$ are computed given an initial belief $bel(\mathbf{x}_0)$ and the nominal path $\check{\mathcal{X}}$ as well as the nominal inputs $\check{\mathcal{U}}$ from a steering procedure.

At the end of the simulation, each belief $bel(\mathbf{x}_k)$ is approximated by M simulated vehicle states $\mathbf{x}_k^{[i=1:M]}$ that can be retrieved from the computed closed-loop executions. The underlying calculations make no assumption on the system given in (2.60)–(2.61) and only require that sampling from the initial belief as well as from the motion and measurement noise is possible. In addition to that, a state estimator is required to evaluate the feedback term in (2.61). Without loss of generality, it is assumed in Alg. 1 that a Gaussian filter [186] is used to compute that estimate, which is in this case represented by the mean $\boldsymbol{\mu}_k \in \mathbb{R}^n$ and the covariance $\boldsymbol{\Sigma}_k \in \mathbb{R}^{n \times n}$. Note that other estimators, such as a particle filter, can also be integrated into Alg. 1 by adjusting Line 10 accordingly.

To start the MC simulation in Alg. 1, the expected value $E[\bullet]$ and the covariance $\mathrm{Cov}[\bullet]$ of the initial belief distribution are computed in Line 2 for future use. The remainder of the algorithm is then composed of two nested for loops: the outer loop iterates over the M MC runs while the

Algorithm 1 Monte Carlo Simulation (adapted from [213], © 2018 IEEE)

mc_sim$(bel(\mathbf{x}_0), \check{\mathcal{X}}, \check{\mathcal{U}})$**:**

1: $\mathcal{X}^{[i=1:M]} = \emptyset$
2: $\boldsymbol{\mu}_0^{[i=1:M]} = E_{\mathbf{x}\sim bel(\mathbf{x}_0)}[\mathbf{x}];\ \boldsymbol{\Sigma}_0^{[i=1:M]} = \mathrm{Cov}_{\mathbf{x}\sim bel(\mathbf{x}_0)}[\mathbf{x}]$
3: **for** $i = 1 : M$ **do**
4: sample $\mathbf{x}_0^{[i]} \sim bel(\mathbf{x}_0)$
5: $\mathcal{X}^{[i]} \mathrel{+}= \langle \mathbf{x}_0^{[i]} \rangle$
6: **for** $k = 0 : \bar{N} - 1$ **do**
7: $\mathbf{u}_k^{[i]} = \check{\mathbf{u}}_k - \mathbf{k}(\boldsymbol{\mu}_k^{[i]}, \check{\mathbf{x}}_k)$
8: $\mathbf{x}_{k+1}^{[i]} = \mathbf{f}(\mathbf{x}_k^{[i]}, \mathbf{u}_k^{[i]}, \mathbf{w}_k^{[i]})$
9: $\mathbf{z}_{k+1}^{[i]} = \mathbf{h}(\mathbf{x}_{k+1}^{[i]}, \mathbf{v}_{k+1}^{[i]})$
10: $\boldsymbol{\mu}_{k+1}^{[i]}, \boldsymbol{\Sigma}_{k+1}^{[i]} = \text{estimator}(\boldsymbol{\mu}_k^{[i]}, \boldsymbol{\Sigma}_k^{[i]}, \mathbf{u}_k^{[i]}, \mathbf{z}_{k+1}^{[i]})$
11: $\mathcal{X}^{[i]} \mathrel{+}= \langle \mathbf{x}_{k+1}^{[i]} \rangle$
12: **end for**
13: **end for**
14: **return** $\mathcal{X}^{[i=1:M]}$

inner loop propagates the state of the vehicle $\bar{N}$ steps along the nominal path. This procedure starts with the sampling of an initial vehicle state $\mathbf{x}_0^{[i]}$ from the belief $bel(\mathbf{x}_0)$ (see Line 4). The obtained state can then be updated using the input $\mathbf{u}_k^{[i]}$ in Line 7, a sample $\mathbf{w}_k^{[i]}$ from the motion noise, and the motion model in Line 8. Next, an estimate $\boldsymbol{\mu}_{k+1}^{[i]}$ of the vehicle state $\mathbf{x}_{k+1}^{[i]}$ must be computed such that the control law $-\mathbf{k}(\bullet)$ can be evaluated in the next iteration. To do so, a state estimator is used in Line 10 that not only requires the previous state estimate and the current input $\mathbf{u}_k^{[i]}$, but also a measurement $\mathbf{z}_{k+1}^{[i]}$ of the updated vehicle state. The latter is generated in Line 9 by sampling $\mathbf{v}_{k+1}^{[i]}$ from the measurement noise and by evaluating the measurement model accordingly. Repeating the previously described steps $\bar{N}$ times yields one closed-loop execution $\mathcal{X}^{[i]} = \langle \mathbf{x}_0^{[i]}, \ldots, \mathbf{x}_{\bar{N}}^{[i]} \rangle$ of the system. The algorithm finally terminates after having computed M runs $\mathcal{X}^{[i=1:M]}$ in an identical way.

A visualization of such an MC simulation along the path of a RS steering procedure can be found in Fig. 2.45. In this example, it can be seen that both the stabilizing feedback controller and the fact that measurements are taken into account keep the uncertainty bounded as the vehicle moves from the

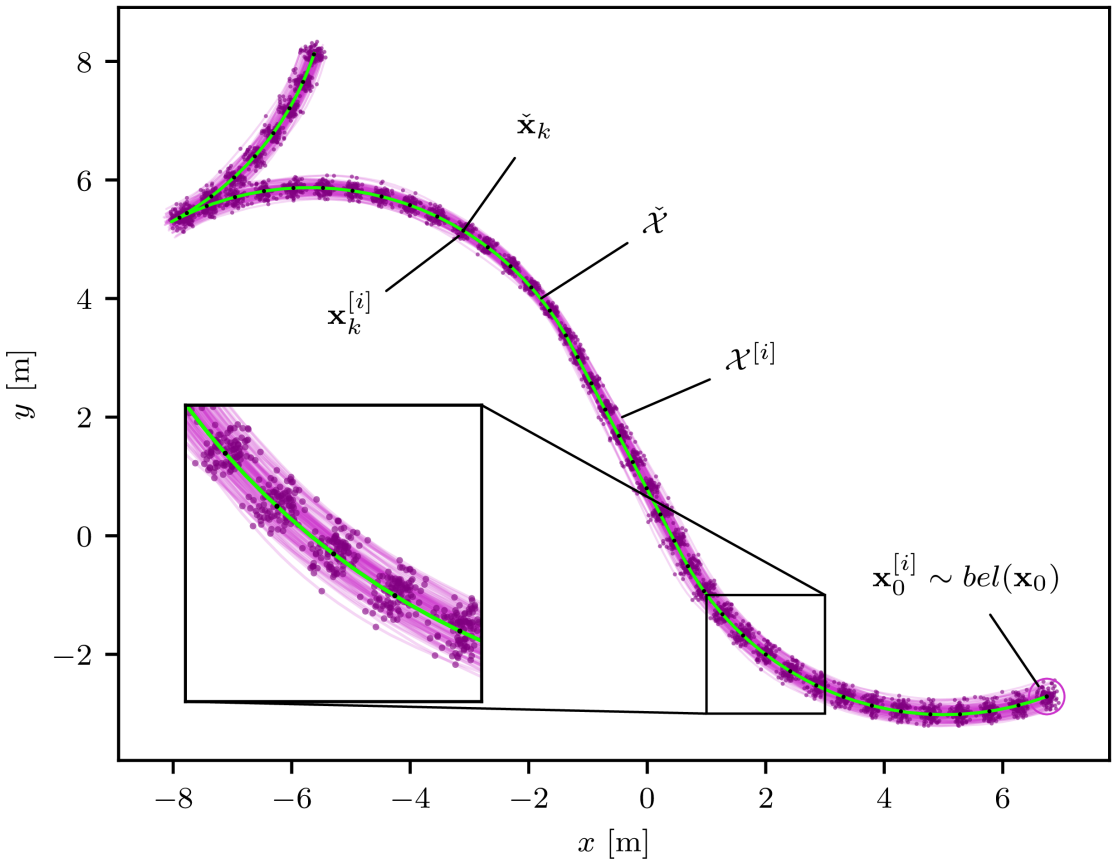


Figure 2.45 RS path from Fig. 2.9 augmented with 100 MC runs. The initial belief $bel(\mathbf{x}_0)$ visualized in the lower right corner (three-sigma ellipse) is given by a Gaussian distribution. A subset of the nominal states $\check{\mathbf{x}}_k$ is shown by the black dots on the green line that illustrates the nominal path $\check{\mathcal{X}}$, and the corresponding disturbed vehicle states are highlighted by the dark purple dots.

initial belief $bel(\mathbf{x}_0)$ to the goal. Details on the underlying implementation are given in the next section.

2.5.1.1 Implementation Details

The following paragraphs detail the design choices that have been made on the basis of [31, 186] for the implementation of Alg. 1 in this thesis. First of all, the state-action space for the MC simulation is chosen to be the one of the G^1 continuous steering functions that is commonly used in low-speed driving. As a result, the vehicle state is given by $\mathbf{x}_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^\mathsf{T}$ and the input by $\mathbf{u}_k = \begin{pmatrix} \Delta s_k & \kappa_k \end{pmatrix}^\mathsf{T}$. The motion model $\mathbf{f}(\bullet)$ is assumed to be given by the constant curvature model in (2.2) with the only difference that the nominal input $\check{\mathbf{u}}_k$ is replaced by $\mathbf{u}_k + \mathbf{w}_k$. This implies that the motion

noise $\mathbf{w}_k$, which is described by a Gaussian distribution with zero mean and covariance $\mathbf{Q}_k$, is added to the control input $\mathbf{u}_k$ and propagated through $\mathbf{f}(\bullet)$. Similar to [186], the covariance $\mathbf{Q}_k$ is defined as

$$\mathbf{Q}_k = \begin{pmatrix} \alpha_1 \Delta\check{s}_k^2 + \alpha_2 \check{\kappa}_k^2 & 0 \\ 0 & \alpha_3 \Delta\check{s}_k^2 + \alpha_4 \check{\kappa}_k^2 \end{pmatrix}, \tag{2.63}$$

where $\alpha_{1:4} \in \mathbb{R}_{\geq 0}$ are parameters of the model, and $\Delta\check{s}_k$ and $\check{\kappa}_k$ the nominal inputs of the steering procedure.

In order to keep the system close to the nominal path, a stabilizing feedback controller has to be implemented. Within this context, a linear controller is selected whose output is computed according to

$$\mathbf{k}(\boldsymbol{\mu}_k, \check{\mathbf{x}}_k) = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & \operatorname{sgn}(\Delta\check{s}_k) k_3 \end{pmatrix} \cdot \mathbf{T}_{kI}(\check{\theta}_k) \cdot ({}_I\boldsymbol{\mu}_k - {}_I\check{\mathbf{x}}_k), \tag{2.64}$$

where $k_{1:3} \in \mathbb{R}$ denote the parameters of the controller and $\mathbf{T}_{kI}(\check{\theta}_k)$ the transformation matrix that rotates a state from the inertial frame I to the local frame k defined by the nominal state $\check{\mathbf{x}}_k$.

The assumption that a localization system provides the measurements $\mathbf{z}_k$ allows to define the corresponding measurement model as

$$\mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) = \mathbf{x}_k + \mathbf{v}_k, \tag{2.65}$$

where the measurement noise $\mathbf{v}_k$ is drawn from a Gaussian distribution with zero mean and covariance $\mathbf{R}_k$. Currently, it is assumed that $\mathbf{R}_k$ is constant throughout the state space and given as

$$\mathbf{R}_k = \operatorname{diag}(\sigma_x^2,\, \sigma_y^2,\, \sigma_\theta^2), \tag{2.66}$$

where $\operatorname{diag}(\bullet)$ describes a diagonal matrix and σ_x^2, σ_y^2, and σ_θ^2 the variance of the localization module. Note, however, that the assumption of $\mathbf{R}_k$ being constant can be removed if precomputed localizability maps are available that allow to deduce the measurement covariance at a given vehicle pose.

Although the MC simulation provides the greatest flexibility with no assumptions on the underlying system, its computational complexity with the two nested for loops restricts its usage in real-time motion planning. Therefore, the next section details the EKF-MP [23] that allows to propagate a Gaussian belief along the nominal path of the steering functions.

2.5.2 Extended Kalman Filter for Motion Planning

The basic idea of the EKF-MP is to propagate a Gaussian distribution along the nominal path of the vehicle while considering the localization and control uncertainty. This is, for example, visualized in Fig. 2.44, which highlights a subset of the Gaussian beliefs $\mathcal{B} = \langle bel(\mathbf{x}_0), \ldots, bel(\mathbf{x}_{\bar{N}}) \rangle$ computed along the path of a G^2 continuous steering procedure. Based on the derivation in [23], the beliefs are parametrized by

$$bel(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mu_k, \mathbf{\Sigma}_k + \mathbf{\Lambda}_k), \tag{2.67}$$

where $\mathcal{N}(\bullet)$ denotes a multivariate normal distribution with mean $\mu_k \in \mathbb{R}^n$ and covariance $(\mathbf{\Sigma}_k + \mathbf{\Lambda}_k) \in \mathbb{R}^{n \times n}$. While the meaning of the covariance $\mathbf{\Sigma}_k$ is generally known from the standard EKF [186], the interesting part in (2.67) is the additive term $\mathbf{\Lambda}_k$. It accounts for the fact that in motion planning, the future measurements are not known at planning time. Therefore, the EKF-MP takes into account all possible future measurements, which are assumed to be given by a Gaussian distribution. As stated in [23], this leads to the fact that the mean $\boldsymbol{\mu}_k$ of the state estimate is not a deterministic variable anymore, as in the case of the standard EKF [186], but instead distributed as

$$\boldsymbol{\mu}_k = \mathcal{N}(\boldsymbol{\mu}_k; \mu_k, \mathbf{\Lambda}_k). \tag{2.68}$$

Further details on the derivation of (2.67) using (2.68) can be found in [23].

The EKF-MP requires to linearize both the motion and the measurement model in (2.60) as well as the feedback controller in (2.61) along the nominal path of the steering function. A linearization of (2.60) yields

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{F}_{\mathbf{x},k}\tilde{\mathbf{x}}_k + \mathbf{F}_{\mathbf{u},k}\tilde{\mathbf{u}}_k + \mathbf{F}_{\mathbf{w},k}\mathbf{w}_k, \tag{2.69a}$$

$$\tilde{\mathbf{z}}_k = \mathbf{H}_{\mathbf{x},k}\tilde{\mathbf{x}}_k + \mathbf{H}_{\mathbf{v},k}\mathbf{v}_k, \tag{2.69b}$$

where the deviations from the nominal path are given as $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \check{\mathbf{x}}_k$, $\tilde{\mathbf{u}}_k = \mathbf{u}_k - \check{\mathbf{u}}_k$, and $\tilde{\mathbf{z}}_k = \mathbf{z}_k - \check{\mathbf{z}}_k$. The matrices $\mathbf{F}_{\mathbf{x},k} = \nabla_{\mathbf{x}_k}\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$, $\mathbf{F}_{\mathbf{u},k} = \nabla_{\mathbf{u}_k}\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$, $\mathbf{F}_{\mathbf{w},k} = \nabla_{\mathbf{w}_k}\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$ describe the partial derivatives of (2.60a) with respect to the indexed variable. Similarly, the partial derivatives of (2.60b) are given as $\mathbf{H}_{\mathbf{x},k} = \nabla_{\mathbf{x}_k}\mathbf{h}(\mathbf{x}_k, \mathbf{v}_k)$ and $\mathbf{H}_{\mathbf{v},k} = \nabla_{\mathbf{v}_k}\mathbf{h}(\mathbf{x}_k, \mathbf{v}_k)$. All matrices have to be recomputed at every step k at the respective nominal state $\check{\mathbf{x}}_k$ and input $\check{\mathbf{u}}_k$. Furthermore, it is assumed that both the motion noise $\mathbf{w}_k$ as well as the measurement noise $\mathbf{v}_k$ are drawn from two independent zero-mean Gaussians with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively.

With respect to the feedback controller in (2.61), a linearization yields

$$\mathbf{u}_k = \check{\mathbf{u}}_k - \mathbf{K}_{\mathbf{x},k}(\boldsymbol{\mu}_k - \check{\mathbf{x}}_k), \tag{2.70}$$

where $\mathbf{K}_{\mathbf{x},k} = \nabla_{\mathbf{x}_k}\mathbf{k}(\boldsymbol{\mu}_k, \check{\mathbf{x}}_k)$ denotes the linearized controller gain that also has to be evaluated at every step k at the respective nominal state $\check{\mathbf{x}}_k$ and input $\check{\mathbf{u}}_k$.

Based on (2.69)–(2.70), the EKF-MP can now be used to update the current belief $bel(\mathbf{x}_k)$ given a nominal input $\check{\mathbf{u}}_k$. The required computation steps, which can be divided similar to the standard EKF into a prediction and update step, are extracted from [23] and listed in Alg. 2.

Algorithm 2 EKF for Motion Planning (reprinted from [213], © 2018 IEEE)

ekf_mp$(\mu_k, \boldsymbol{\Sigma}_k, \boldsymbol{\Lambda}_k, \check{\mathbf{u}}_k)$**:**

Prediction

1: $\mathbf{F}_{\mathbf{K},k} = \mathbf{F}_{\mathbf{x},k} - \mathbf{F}_{\mathbf{u},k}\mathbf{K}_{\mathbf{x},k}$
2: $\bar{\mu}_{k+1} = \mathbf{f}(\mu_k, \check{\mathbf{u}}_k, \mathbf{0})$
3: $\bar{\boldsymbol{\Sigma}}_{k+1} = \mathbf{F}_{\mathbf{x},k}\boldsymbol{\Sigma}_k\mathbf{F}_{\mathbf{x},k}^\intercal + \mathbf{F}_{\mathbf{w},k}\mathbf{Q}_k\mathbf{F}_{\mathbf{w},k}^\intercal$
4: $\bar{\boldsymbol{\Lambda}}_{k+1} = \mathbf{F}_{\mathbf{K},k}\boldsymbol{\Lambda}_k\mathbf{F}_{\mathbf{K},k}^\intercal$

Update

5: $\boldsymbol{\Sigma}_{\mathbf{xz},k+1} = \bar{\boldsymbol{\Sigma}}_{k+1}\mathbf{H}_{\mathbf{x},k+1}^\intercal$
6: $\boldsymbol{\Sigma}_{\mathbf{z},k+1} = \mathbf{H}_{\mathbf{x},k+1}\boldsymbol{\Sigma}_{\mathbf{xz},k+1} + \mathbf{H}_{\mathbf{v},k+1}\mathbf{R}_{k+1}\mathbf{H}_{\mathbf{v},k+1}^\intercal$
7: $\mathbf{L}_{k+1} = \boldsymbol{\Sigma}_{\mathbf{xz},k+1}\boldsymbol{\Sigma}_{\mathbf{z},k+1}^{-1}$
8: $\mu_{k+1} = \bar{\mu}_{k+1}$
9: $\boldsymbol{\Sigma}_{k+1} = \bar{\boldsymbol{\Sigma}}_{k+1} - \mathbf{L}_{k+1}\boldsymbol{\Sigma}_{\mathbf{xz},k+1}^\intercal$
10: $\boldsymbol{\Lambda}_{k+1} = \bar{\boldsymbol{\Lambda}}_{k+1} + \mathbf{L}_{k+1}\boldsymbol{\Sigma}_{\mathbf{xz},k+1}^\intercal$
11: **return** $\mu_{k+1}, \boldsymbol{\Sigma}_{k+1}, \boldsymbol{\Lambda}_{k+1}$

In the prediction step, it can be observed that the stabilizing controller only acts on $\boldsymbol{\Lambda}_k$ and not on $\boldsymbol{\Sigma}_k$ (see Lines 3–4). This is because the EKF-MP uses the linear(ized) system from (2.69) to which the separation principle [108] applies, i.e. the possibility to treat both control and state estimation independently. The latter is also known from linear quadratic Gaussian (LQG) control [177], where the optimal controller can be designed separately from the optimal filter.

Another interesting observation can be made in the update step of Alg. 2. Here, it can be seen that the updated covariance $\boldsymbol{\Sigma}_{k+1}$ is smaller than its predicted counterpart $\bar{\boldsymbol{\Sigma}}_{k+1}$ (see Line 9). This behavior is already known

from the standard EKF, where the availability of measurements leads to a reduction in uncertainty. As opposed to this, it can be noticed in Line 10 that the updated covariance $\mathbf{\Lambda}_{k+1}$ is larger than its predicted counterpart $\bar{\mathbf{\Lambda}}_{k+1}$. This is due to the fact that the actual measurements are not available during planning and therefore the entire distribution of possible measurements has to be considered. Although this leads to an increase in uncertainty (see Line 10), the stabilizing controller has the capability to keep the corresponding covariance bounded (see Line 4, where $\mathbf{F}_{\mathbf{K},k}\mathbf{\Lambda}_k\mathbf{F}_{\mathbf{K},k}^{\mathsf{T}}$ is contractive for a stabilized system [23]).

The last insight from Alg. 2 is that the sum of the predicted covariances $\bar{\mathbf{\Sigma}}_{k+1} + \bar{\mathbf{\Lambda}}_{k+1}$ is equal to the covariance of the updated belief $\mathbf{\Sigma}_{k+1} + \mathbf{\Lambda}_{k+1}$. This highlights the fact that during planning, the overall uncertainty can only be reduced by the controller in Line 4 and not in the update step of Alg. 2 due to the absence of actual measurements.

A visualization of the Gaussian beliefs from the EKF-MP along with the output from the MC simulation is shown in Fig. 2.46. Here, it can be seen that the Gaussian distributions result in a good approximation of the true belief distribution. However, it has to be kept in mind that this depends strongly on the nonlinearity of the system as well as on the intensity of both the motion and the measurement noise. Therefore, it is highly recommended to always use the MC simulation in order to verify if the EKF-MP can be used for a given problem.

2.5.2.1 Implementation Details

This section details the implementation of the EKF-MP used in this thesis to propagate Gaussian beliefs along the nominal path of the steering functions. Similar to Sec. 2.5.1.1 and [31], the vehicle state is represented here by $\mathbf{x}_k = \begin{pmatrix} x_k & y_k & \theta_k \end{pmatrix}^{\mathsf{T}}$ and the input by $\mathbf{u}_k = \begin{pmatrix} \Delta s_k & \kappa_k \end{pmatrix}^{\mathsf{T}}$. This results in the computation of three-dimensional Gaussian distributions that describe the uncertainty in position and orientation as required for probabilistic collision checking (see Ch. 3).

One of the prerequisites of Alg. 2 is a linearization of the vehicle's motion along the nominal path according to (2.69a). For the previously introduced steering functions, the corresponding partial derivatives $\mathbf{F}_{\mathbf{x},k}$ and $\mathbf{F}_{\mathbf{u},k}$ are given in Sec. A.1. The assumption that the motion noise acts directly on the input of the system, as in Sec. 2.5.1.1, leads to $\mathbf{F}_{\mathbf{w},k} = \mathbf{F}_{\mathbf{u},k}$. Furthermore, it is assumed here that the motion noise $\mathbf{w}_k$ is given by a zero-mean Gaussian

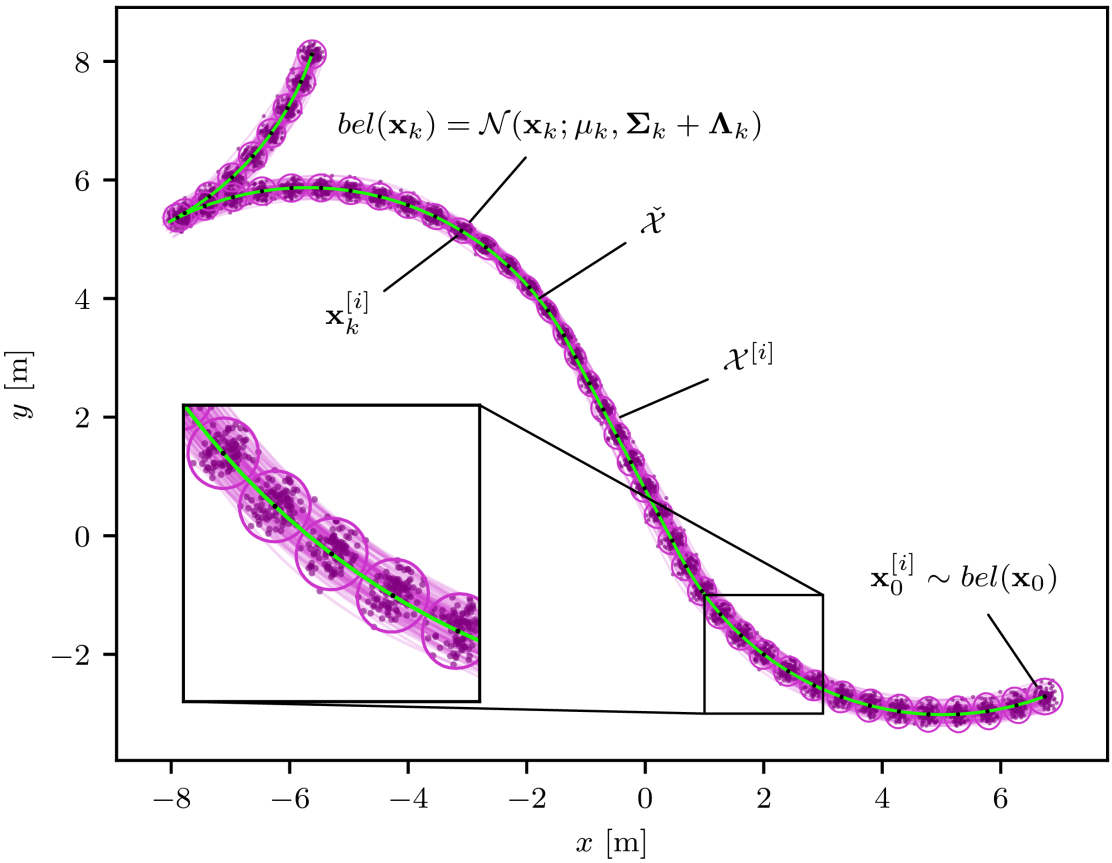


Figure 2.46 Extension of Fig. 2.45 with the Gaussian beliefs from the EKF-MP. The ellipses illustrate the three-sigma uncertainty of the beliefs $bel(\mathbf{x}_k)$ while the meaning of the other markers is identical to Fig. 2.45.

whose covariance $\mathbf{Q}_k$ is computed analogously to (2.63).

The stabilizing feedback controller from (2.64) can be used here directly as it is already given in a linear form. The corresponding controller gain $\mathbf{K}_{\mathbf{x},k}$ is defined as

$$\mathbf{K}_{\mathbf{x},k} = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & \operatorname{sgn}(\Delta \check{s}_k) k_3 \end{pmatrix} \cdot \mathbf{T}_{kI}(\check{\theta}_k), \tag{2.71}$$

where $k_{1:3}$ and $\mathbf{T}_{kI}(\check{\theta}_k)$ have the same meaning as in (2.64).

The assumptions in Sec. 2.5.1.1 with respect to the localization system are adopted here. The matrices $\mathbf{H}_{\mathbf{x},k}$ and $\mathbf{H}_{\mathbf{v},k}$ in Alg. 2 can therefore be directly derived from (2.65) as $\mathbf{H}_{\mathbf{x},k} = \mathbf{H}_{\mathbf{v},k} = \mathbf{I}_3$, where $\mathbf{I}_3$ denotes the three-dimensional identity matrix. On the basis of these equations, the EKF-MP can now be used to propagate the beliefs along the nominal path. A benchmark of the computational complexity is highlighted in the next section.

2.5.2.2 Experimental Evaluation

This section compares the computation times of the nominal steering functions with the ones in Gaussian belief space, where the belief distribution is computed along the nominal path using the EKF-MP. For this purpose, the implementation of the G^1 and G^2 continuous steering functions has been extended with Alg. 2. The corresponding source code is publicly available in [208]. Note that on the basis of the provided details, the G^3 continuous steering functions can also be extended to Gaussian belief space, which is, however, beyond the scope of this section.

The setup for the conducted experiments is similar to the one in Sec. 2.3.4. Briefly summarized, 10^5 random steering procedures are evaluated on a single core of an Intel Xeon E5@3.5 GHz. The vehicle's start and goal position is uniformly sampled within a 20 m $\times$ 20 m area and its orientation is randomly chosen from the interval $[0, 2\pi)$. The parameters of the vehicle are set to $\kappa_{\max} = 1\,\mathrm{m}^{-1}$ and $\sigma_{\max} = 1\,\mathrm{m}^{-2}$, and the states/beliefs are computed at equidistant arc lengths with a discretization of 10 cm. In contrast to Tab. 2.4 and Tab. 2.5, which only list the time for the computation of the actions $\check{\mathbf{u}}_k \in \check{\mathcal{U}}$, the evaluation here also includes the derivation of the states/beliefs using the corresponding motion model. The source code is implemented in C++ and uses the Eigen library [69] for the linear algebra in Alg. 2.

The results of the conducted benchmark are listed in Tab. 2.8.

Table 2.8 Average computation time of the G^1 and G^2 continuous steering functions with/without the usage of the EKF-MP for the computation of the belief distribution.

	without EKF-MP		with EKF-MP	
	mean [µs]	std [µs]	mean [µs]	std [µs]
Dubins	8.4	±2.9	65.0	±23.5
CC^{00}-Dubins	16.1	±3.2	85.6	±23.5
RS	13.4	±2.9	63.3	±23.6
$HC^{\pm\pm}$-RS	56.8	±7.0	112.6	±23.0
HC^{00}-RS	57.5	±6.1	117.2	±20.3
CC^{00}-RS	64.6	±7.6	126.3	±19.7

It can be seen that the computation of the beliefs increases the average computation times between 49.6 µs and 69.5 µs. This is due to the additional complexity of Alg. 2 that is executed in the given setup every 10 cm. A

relative comparison shows that especially in the G^1 continuous case, the EKF-MP increases the average computation times by up to a factor of 7.7. In contrast to that, this factor is only 2.0 in case of the G^2 continuous RS steering functions. The underlying reason is that the computation time for the belief update is fairly decoupled from the smoothness of the path. This makes steering functions with an originally low computation time relatively more expensive. Additionally, it has to be considered that the length of the path determines the number of beliefs and thus how often Alg. 2 needs to be executed. The belief propagation is therefore more expensive for longer paths as it can be seen by comparing the mean computation time with/without EKF-MP for CC^{00}-Dubins (+69.5 µs) and RS steer (+49.9 µs). Nevertheless, Alg. 2 is currently one of the most efficient approaches to also consider the localization and control uncertainty in motion planning.

2.6 Summary

Solving the steering problem in (2.1) is one of the major problems in motion planning for car-like robots. It requires to find a preferably optimal solution that connects a start and goal state while satisfying the constraints of the system. Obstacle avoidance can then be enforced in a subsequent step by iteratively probing the computed states for collision with the environment.

On the basis of the prominent steering functions Dubins [43], RS [152], and CC steer [53], several novel methods are presented and benchmarked in this chapter to tackle the drawbacks of those three approaches. For instance, the novel steering function HC steer in Sec. 2.3.2 enforces curvature continuity between direction switches, but allows to turn the wheels at cusps. As shown in Sec. 2.3.4, the resulting paths are not only smoother than the curvature-discrete ones of RS steer, but also shorter than the paths of CC steer. Especially in tight environments, as it will be shown in Sec. 5.3.1, these properties allow to compute motion plans with less direction switches than CC steer while providing a higher degree of comfort for passengers than RS steer.

An extension of HC and CC steer to arbitrary start and goal curvatures is then introduced in Sec. 2.3.3. While the original publication of CC steer [53] requires zero curvature at both ends of the path, the novel approach allows to connect two arbitrary four-dimensional states (position, orientation, curvature) with either a HC or a CC path. This feature is especially useful when the vehicle is in a state with non-zero velocity and curvature.

One of the remaining problems that both HC and CC steer impose infinite steering acceleration on the vehicle is tackled in Sec. 2.4. Here, the smoothness of the paths is raised to curvature rate continuity while at the same time satisfying the vehicle's constraints on the maximum curvature, maximum curvature rate, and maximum curvature acceleration. Among the advantages of the so-called HCR and CCR steering functions are a better closed-loop tracking performance, less mechanical stress on the steering system, and more comfort due to a reduction in lateral jerk. However, it also has to be considered that the corresponding paths are typically longer than their G^1 and G^2 continuous counterparts and also more expensive to compute. Both aspects were elaborated in detail in Sec. 2.4.3.

While the previous approaches assumed perfect state estimation and control, an extension of these nominal steering functions to belief space is finally presented in Sec. 2.5. Here, two methods are described that allow to compute the belief distribution along the nominal path by explicitly taking into account the localization and control uncertainty. This distribution can then be used in the motion planner to compute more robust solutions with bounded collision probability as further detailed in Ch. 3 and Ch. 5.

In the future, researchers are encouraged to build upon the presented results by using the steering function package in [208] that has been developed and provided as open source as part of this thesis. Two robots that currently leverage the capabilities of this implementation are the Bosch Campus Shuttle and the automated forklifts of the ILIAD project [79] both visualized in Fig. 2.47. Further details on how the provided steering functions can be used in a sampling-based motion planner are given in Ch. 5.

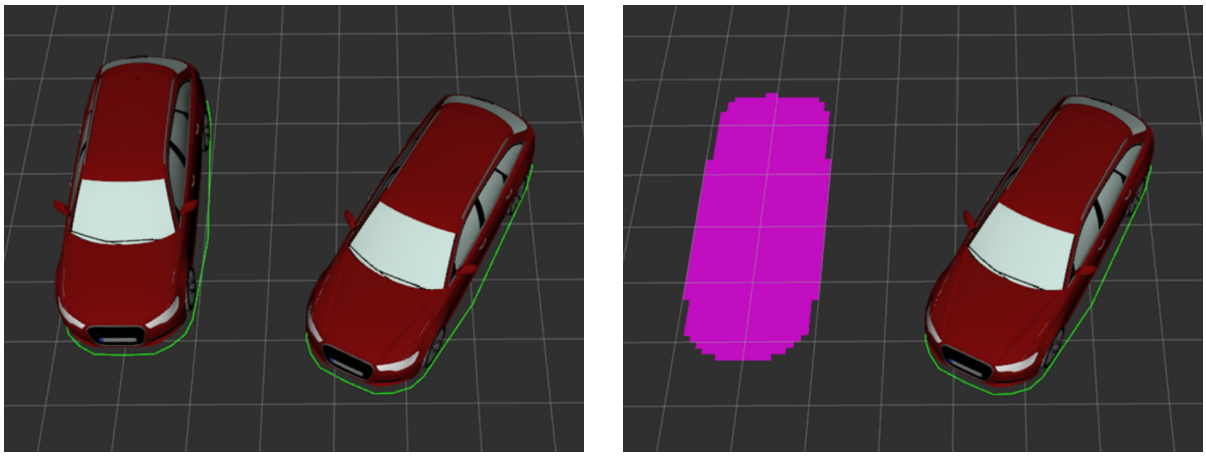
(a) Bosch Campus Shuttle. **(b)** Automated forklifts [79].

Figure 2.47 Two robots that currently use the open-source implementation [208] of the steering functions. (b) Reprinted with permission from [79], © 2018 Timm Linder.

3 From Footprints to Beliefprints: Probabilistic Collision Checking

Collision checking is one of the important aspects in motion planning. It requires to determine whether a sequence of states can be executed safely by probing the footprint of the vehicle for collision with the environment. Within this context, the footprint is defined as the robot's contour projected onto the ground. For efficient collision checking, the exact footprint of the vehicle is typically approximated by several disks [205] or as in this chapter by a convex polygon (see Fig. 3.1).

Depending on the representation of the surrounding obstacles, different algorithms must be applied for collision checking. For instance, Fig. 3.1(a) illustrates the case where both the ego-vehicle as well as the obstacle are represented by a polygon. In contrast to that, an occupancy grid [124] is used in Fig. 3.1(b) to describe the area covered by the obstacle. In both cases, efficient algorithms exist in the literature for collision checking.

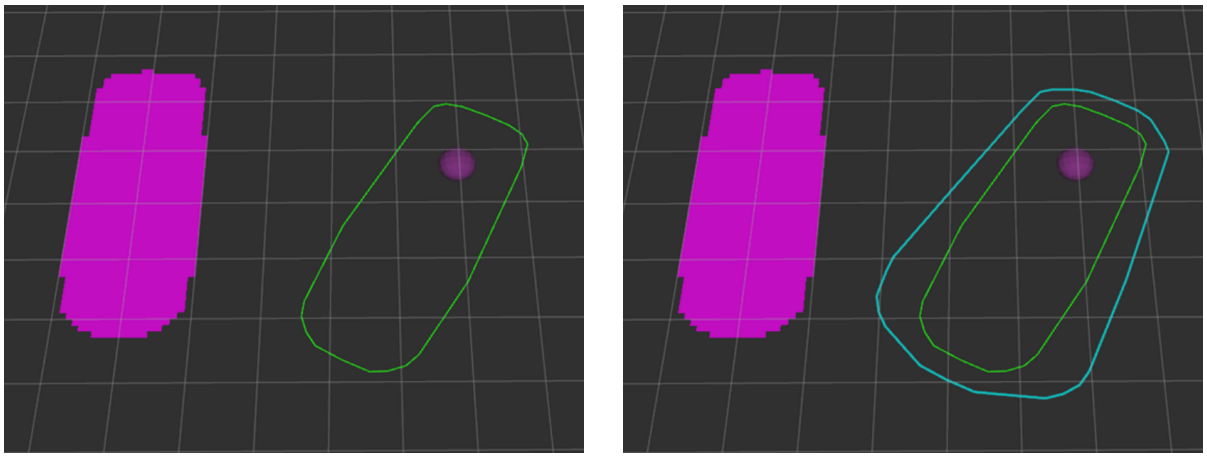


(a) Polygon–polygon collision check. **(b)** Polygon–occ.-grid collision check.

Figure 3.1 Different representations of the environment require different approaches for collision checking. Two polygons (green contours) must be examined for collision in (a) while (b) requires to check a polygon (green contour) against an occupancy grid (purple area). Reprinted from [209].

For example, a well-known approach for the polygon–polygon collision check in Fig. 3.1(a) is the Gilbert-Johnson-Keerthi algorithm [62]. It is based on the fact that two convex polygons are collision-free if the corresponding Minkowski difference [45] does not contain the origin. If, however, the environment is represented by an occupancy grid as in Fig. 3.1(b), the Bresenham algorithm [20] allows to efficiently evaluate whether the edges of a polygon intersect with the occupied area. In order to not only check the edges, but also the interior of that polygon, filling algorithms from computer graphics [51] or a convolution-based approach [184] can be applied. In summary, efficient approaches exist for collision checking when both the state of the robot as well as the shape of the obstacles are given without uncertainty.

In reality, however, this is typically not the case due to e.g. noisy measurements or imperfect execution of the nominal motion plan. As outlined in Sec. 2.5, the latter leads to the fact that the state of the vehicle is no longer given by a deterministic value, but rather described by a belief distribution. This can be seen in Fig. 3.2(a), where the purple ellipsoid at the center of the rear axle illustrates the uncertainty of the ego-vehicle's state.



(a) Vehicle state with uncertainty. **(b)** Beliefprint for collision checking.

Figure 3.2 Transformation of the footprint (green polygon) into the beliefprint (cyan polygon) for collision checking under uncertainty. The environment is assumed to be given by the same occupancy grid as in Fig. 3.1(b) while the uncertain vehicle pose is described by a three-dimensional Gaussian distribution (purple ellipsoid). Reprinted from [209].

A natural question in such a situation is how to assess whether the collision probability (CP) with the environment remains below a user-defined threshold. This motivates the introduction of the so-called beliefprint $\mathcal{B}_k \subset \mathbb{R}^2$, which describes the area occupied by the robot given its footprint $\mathcal{F} \subset \mathbb{R}^2$, a

belief $bel(\mathbf{x}_k)$ with $\mathbf{x}_k \in \mathbb{R}^{n\leq 3}$, and a confidence $P \in (0.5, 1]$. A visualization of such a beliefprint can be found in Fig. 3.2(b). The general idea behind this concept is to derive a new polygon that guarantees a CP below $1 - P$ if it does not collide with the surrounding obstacles. Therefore, beliefprints are especially interesting from a planning perspective as they allow to enforce the frequently arising chance constraint [12, 23, 121, 191]

$$\Pr(\mathbf{x}_k \in \mathcal{X}_{\text{obs}}) < 1 - P, \tag{3.1}$$

where $\Pr(\bullet)$ measures probability and $\mathcal{X}_{\text{obs}} \subset \mathbb{R}^n$ denotes the states that lead to a collision with the surrounding obstacles. Note that these obstacles are assumed to be given without uncertainty, however, an extension to a probabilistic environment representation is also possible. Furthermore, the ideas presented in this chapter are not only restricted to a single two-dimensional rigid body, but rather transferable to a three-dimensional multi-body system. Both aspects are, however, beyond the scope of this thesis and subject of future work.

The remainder of this chapter, which is based on [213, 218], is organized as follows: Sec. 3.1 reviews the state of the art in probabilistic collision checking, and Sec. 3.2 introduces two novel algorithms for the computation of the beliefprint in Gaussian belief space. A summary is finally given in Sec. 3.3.

3.1 State of the Art

Over the years, various approaches have emerged in the literature to evaluate collisions from a probabilistic point of view. Typically, these methods differ in the assumptions they make about the shape of the robot and the representation of both the belief distribution as well as of the environment. Within this context, the following paragraphs first highlight the state-of-the-art Monte Carlo (MC) approaches for probabilistic collision checking followed by the currently available methods in Gaussian belief space.

Monte Carlo sampling Computing CPs using MC sampling is one of the most flexible tools that requires the fewest assumptions about the problem. For instance, [109] presents an MC algorithm that approximates the CP of two vehicles whose states are described by two independent PDFs. An extension of this algorithm to the more general case of a joint PDF describing the underlying distribution is given in [42]. In order to not only evaluate

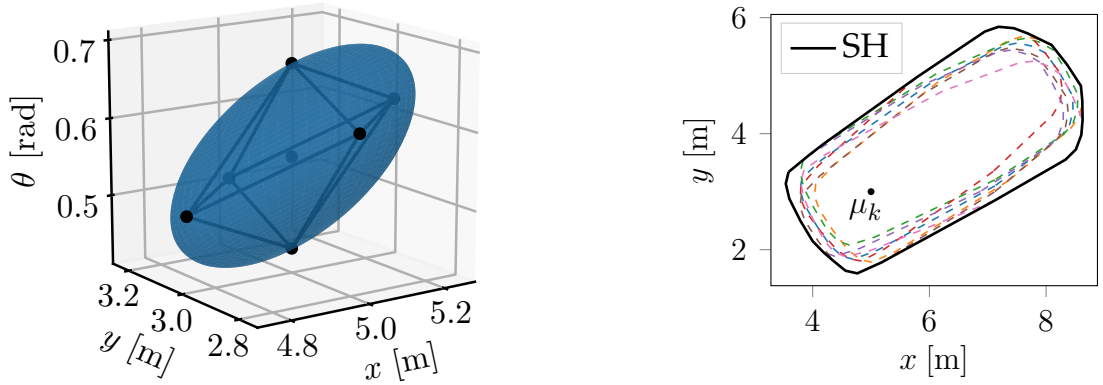
the CP of a single state but rather the one of an entire motion plan, the approaches in [35, 84, 165] perform trajectory rollouts using MC sampling. For instance, [35] considers the case where the future motion of the target vehicles is described by Gaussian processes. In contrast to that, the focus in [84, 165] lies on the ego-motion being subject to uncertainty similar to Sec. 2.5.1. In general, all of the previously mentioned MC approaches come with a high computational cost especially for accurate approximations of the true CP. Although advanced techniques, such as importance sampling [84, 165], can reduce this cost, only highly parallelized implementations allow an application in real-time motion planning [76]. More efficient methods for probabilistic collision checking can be derived in Gaussian belief space as highlighted in the next paragraphs.

Gaussian belief space Under the assumption of two point robots with a joint Gaussian distribution, it is known from e.g. [42] that the exact CP can be computed analytically. However, if both robot shapes are modified to a disk, no closed-form solution exists anymore. In this case, one must either rely on an approximation or on an upper bound on the true CP [42, 134]. Even in the case where only one of the two circular robots is subject to uncertainty, no analytic solution can be found. Instead, it is proposed in [106, 140] to either store the underlying integral in a look-up table [106] or to evaluate it numerically on the fly [140]. A different approach for a circular robot in a known polygonal environment is presented in [8]. Here, the authors compute the minimum probability that no collision occurs with the obstacles at a given time step. Unfortunately, the efficiency of this approach can not be maintained in the case of a non-circular robot, where an optimization problem must be solved for every Gaussian belief [180]. Note that both approaches [8, 180] can also be used in e.g. [137] to approximate the CP of an entire trajectory.

In automated driving, most of the obstacles as well as the ego-vehicle itself have a non-circular shape. This is for example considered in [196], where a conservative hull is constructed such that the translational uncertainty of the ego-vehicle is taken into account. While this might be sufficient for probabilistic collision checking, the grid-based approach in [139] allows to actually compute the CP (up to a discretization error) in case that only one of the two (colliding) objects is subject to translational uncertainty. If, however, the position of both objects is uncertain, the algorithm in [135] can be used to compute an upper bound on the true CP. In order to also consider the

uncertainty of the rotational degree of freedom (DOF), a two-step approach is introduced in [71]. Here, both the translational as well as the rotational DOF are treated separately as the corresponding uncertainties are assumed to be independent.

In contrast to that, the idea of sigma hulls (SHs) [117] is to derive a convex hull that captures the uncertainty of a Gaussian belief with respect to a user-defined confidence P. To do so, it is proposed to first transform the robot to the sigma points known from the UKF [87] and then to compute a convex hull that encloses the robot's shape at all of these configurations. An exemplary visualization of this procedure for a vehicle with one rotational and two translational DOFs can be found in Fig. 3.3.



(a) Sigma points. **(b)** Sigma hull (SH).

Figure 3.3 Visualization of the sigma hull given the Gaussian belief from (a) with mean μ_k. The underlying procedure computes the convex hull of the footprints (dashed lines in (b)) that are transformed to the sigma points of the Gaussian.

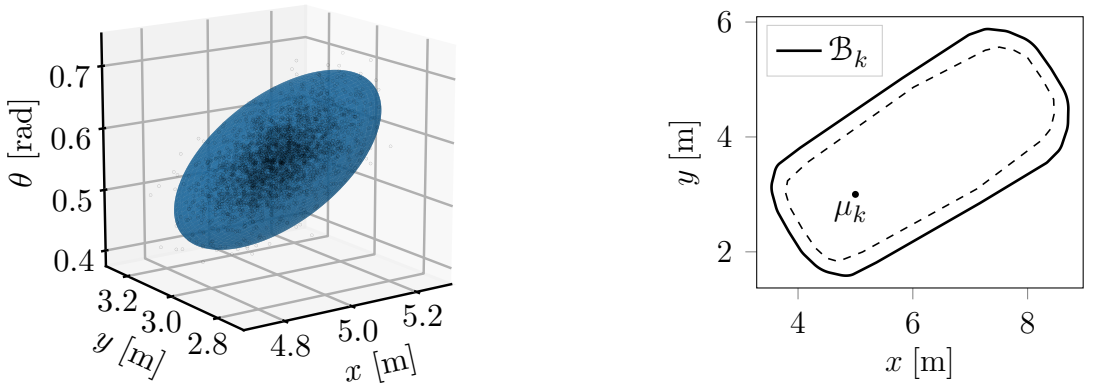
Unfortunately, the sigma points only yield an optimistic approximation of the Gaussian belief as indicated by the polyhedron in Fig. 3.3(a). Therefore, checking a SH for collision with the environment provides no guarantee that the corresponding CP is below the given threshold $1 - P$. This insight is one of the main motivations behind the two novel approaches presented in the following section.

Finally, it has to be mentioned that there are also methods that take into account the perception uncertainty in the evaluation of the CP [3, 13, 54, 139]. Especially [54] has to be highlighted here as it outlines the computation of the CP in a probabilistic occupancy grid [44] that is frequently used in automated driving [63, 189, 194]. Note that this procedure can also be combined with

the previously introduced concept of beliefprints in order to compute CPs that account for both perception and ego-motion uncertainty.

3.2 Beliefprint Computation

This section introduces two novel algorithms for the computation of the beliefprint given a Gaussian belief, the vehicle's footprint, and a confidence P. The development of both algorithms is motivated by the fact that state-of-the-art approaches [117, 196] fail to calculate a conservative approximation of the actual beliefprint $\mathcal{B}_k$. To better understand this, Fig. 3.4 illustrates how $\mathcal{B}_k$ can be computed for a three-dimensional vehicle state with uncertain position and orientation.



(a) Gaussian belief $bel(\mathbf{x}_k)$. **(b)** Beliefprint $\mathcal{B}_k$.

Figure 3.4 Illustration of a three-dimensional Gaussian belief $bel(\mathbf{x}_k)$ along with the corresponding beliefprint $\mathcal{B}_k$. The latter can be obtained by transforming the footprint $\mathcal{F}$ (dashed line in (b)) to all states on the surface of the blue ellipsoid shown in (a).

Briefly summarized, one must first compute the ellipsoid visualized in Fig. 3.4(a) such that it encloses the probability mass P of the Gaussian belief $bel(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mu_k, \mathbf{\Gamma}_k)$[6] with mean $\mu_k \in \mathbb{R}^n$ and covariance $\mathbf{\Gamma}_k \in \mathbb{R}^{n \times n}$. The beliefprint $\mathcal{B}_k$ can then be obtained by taking the union of all footprints transformed to the (infinitely many) states on the surface of the previously

[6]Throughout this section, the belief is represented by a multivariate normal distribution as computed, for example, by the EKF-MP in Alg. 2. Note, however, that the underlying PDF neglects that angles are typically wrapped to a 2π interval (see e.g. wrapped normal distribution [81]) and thus only yields a good approximation for small angular uncertainties.

computed ellipsoid. Comparing this procedure with the state of the art, it can be seen that [196] completely neglects the rotational uncertainty while [117] approximates the belief optimistically with the $2n + 1$ sigma points as shown in Fig. 3.3(a). Furthermore, the procedure described above also reveals that $\mathcal{B}_k$ itself cannot be calculated efficiently due to the infinitely many transformations involved in the underlying computations.

Therefore, the goal in this section is to derive an efficient and systematic approach that allows to approximate the beliefprint $\mathcal{B}_k$ conservatively. To do so, two novel algorithms are proposed on the basis of [213] in Sec. 3.2.1 and [218] in Sec. 3.2.2. While the first algorithm can only guarantee conservatism in certain cases, it serves as an integral part of the second algorithm that provides this guarantee for arbitrary three-dimensional Gaussians (uncertain position and orientation).

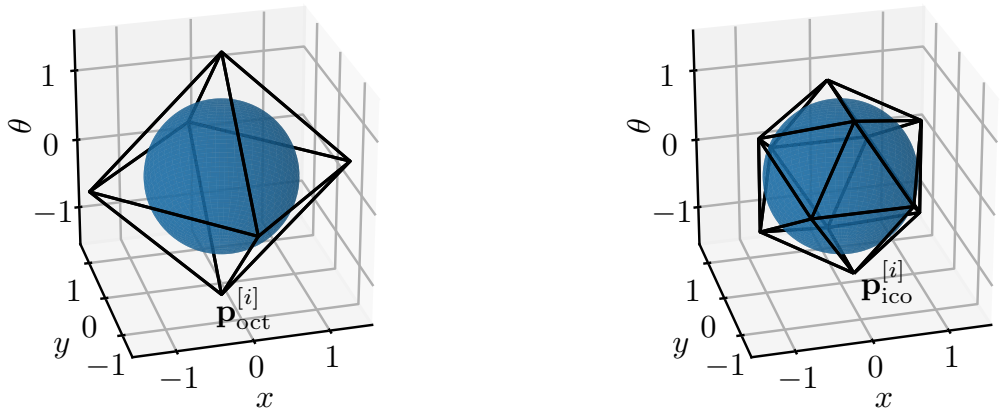
3.2.1 Approximate Beliefprint

The basic idea in this section lies in the approximation of the beliefprint by a convex hull that encloses the vehicle's footprint $\mathcal{F}$ transformed to a finite set of vehicle states. The key aspect of such an approach is the derivation of these states such that the resulting beliefprint $\hat{\mathcal{B}}_k$ yields a conservative approximation of $\mathcal{B}_k$. In this section, it is proposed to use the vertices of a polyhedron $\mathcal{P}_k$, that encloses the probability mass P of the Gaussian belief $bel(\mathbf{x}_k)$, for the required transformations. In certain cases, such a procedure allows a conservative approximation of $\mathcal{B}_k$ as further detailed below. Note that the following derivations are given for the two-dimensional case ($\mathcal{F}, \mathcal{B}_k, \hat{\mathcal{B}}_k \subset \mathbb{R}^2$ and $\mathbf{x}_k \in \mathbb{R}^{n\leq 3}$), but can also be transfered to three dimensions ($\mathcal{F}, \mathcal{B}_k, \hat{\mathcal{B}}_k \subset \mathbb{R}^3$ and $\mathbf{x}_k \in \mathbb{R}^{n\leq 6}$).

The computation of the approximate beliefprint $\hat{\mathcal{B}}_k$ is divided into two stages. The first one requires the definition of a polyhedron $\mathcal{P} \subset \mathbb{R}^n$ with s vertices $\mathbf{p}^{[i=1:s]} \in \mathbb{R}^n$ such that it encloses the probability mass P of an n-dimensional standard normal distribution as follows

$$P = \frac{1}{\sqrt{(2\pi)^n}} \int_{\mathcal{P}} \exp\left(-\frac{1}{2}\mathbf{x}^\intercal\mathbf{x}\right) \, \mathrm{d}\mathbf{x}, \tag{3.2}$$

where the vector $\mathbf{x} \in \mathbb{R}^n$ contains the n states of the robot. For $n = 3$, interesting candidates for $\mathcal{P}$ are the Platonic solids [37] such as the octahedron or the icosahedron both shown in Fig. 3.5.



(a) Octahedron. **(b)** Icosahedron.

Figure 3.5 Visualization of an octahedron with six vertices and of an icosahedron with twelve vertices that both enclose a unit sphere. Adapted from [213], © 2018 IEEE.

The advantage of these polyhedrons is that their symmetry facilitates the integration in (3.2). In case of the octahedron, for example, (3.2) can be written as

$$P_{\text{oct}}(R) = \frac{8}{\sqrt{(2\pi)^3}} \int_0^R \int_0^{R-x} \int_0^{R-x-y} \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{x}\right) \, \mathrm{d}\theta \, \mathrm{d}y \, \mathrm{d}x, \tag{3.3}$$

where the state of the robot is given by $\mathbf{x} = \begin{pmatrix} x & y & \theta \end{pmatrix}^\mathsf{T}$, and the circumscribed radius by $R = \sqrt{3}r$ with r being the inscribed radius visualized by the blue sphere in Fig. 3.5(a). In a similar way, the probability mass can be derived that is contained within an icosahedron given, for example, its inscribed radius. A numerical evaluation of these integrals for different values of r can be found in Fig. 3.6, which also includes the corresponding correlation for a sphere. In this case, the integration in (3.2) can be conducted very efficiently using a spherical coordinate system, which yields

$$P_{\text{sph}}(r) = \operatorname{erf}\left(\frac{\sqrt{2}}{2}r\right) - \sqrt{\frac{2}{\pi}}r \exp\left(-\frac{1}{2}r^2\right), \tag{3.4}$$

where r denotes the radius of the sphere and $\operatorname{erf}(\bullet)$ the error function [1].

It can be seen in Fig. 3.6 that for a given radius r, both the octahedron as well as the icosahedron enclose more probability mass than the sphere. This is due to the fact that both polyhedrons enclose the sphere entirely and also

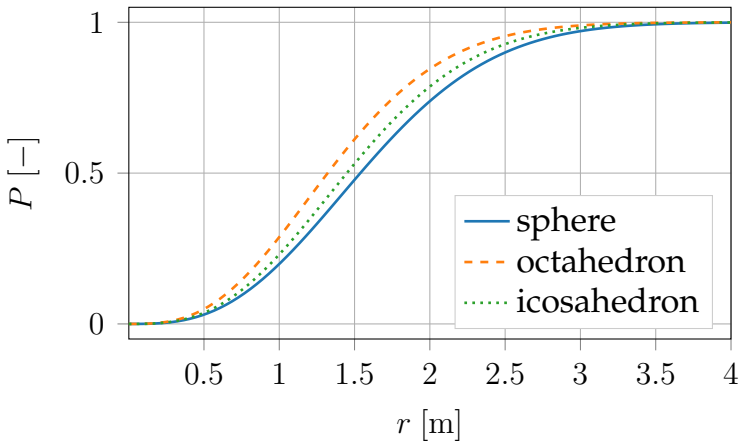


Figure 3.6 Comparison of the probability mass P contained within a sphere, an octahedron, and an icosahedron with regard to the inscribed radius r given a standard normal distribution. Adapted from [213], © 2018 IEEE.

capture additional probability mass at the corners (see Fig. 3.5). However, adding more vertices to the polyhedron in order to better approximate the sphere reduces this effect as it can be observed by comparing the curves of the octahedron and the icosahedron in Fig. 3.6.

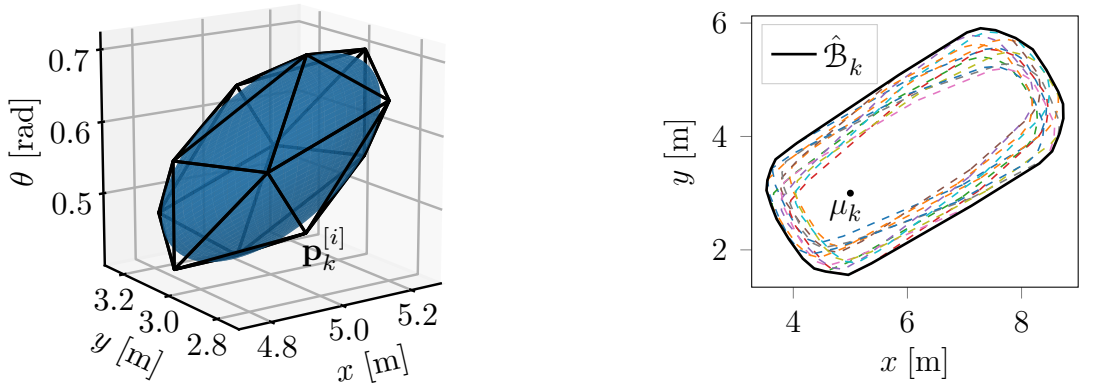
Apart from this analysis, Fig. 3.6 also allows to derive the inscribed radius of the two polyhedrons given a user-defined confidence P. This value can then be used to specify the vertices $\mathbf{p}^{[i]}$ (see Fig. 3.5) that are required for the remaining computations.

While all of the previous steps can be executed offline, the second phase of the proposed algorithm must be repeated online for every Gaussian belief $bel(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mu_k, \mathbf{\Gamma}_k)$. The goal now is to compute the s vertices $\mathbf{p}_k^{[i]}$ of the polyhedron $\mathcal{P}_k$, which encloses the probability mass P of the belief. In fact, it is possible to show that these vertices can be calculated efficiently by transforming the vertices $\mathbf{p}^{[i]}$ of the polyhedron $\mathcal{P}$ according to

$$\mathbf{p}_k^{[i]} = \mathbf{L}_k \mathbf{p}^{[i]} + \mu_k, \tag{3.5}$$

where $\mathbf{L}_k \in \mathbb{R}^{n \times n}$ denotes the lower triangular matrix obtained from the Cholesky factorization of $\mathbf{\Gamma}_k$ [65]. From a geometric point of view, (3.5) first deforms the polyhedron $\mathcal{P}$ linearly using $\mathbf{L}_k$ and then shifts it to the mean μ_k as shown in Fig. 3.7(a). Note that this affine transformation is also known from the generation of multivariate Gaussian samples when only sampling from a standard normal distribution is possible [59].

Now that the polyhedron $\mathcal{P}_k$ is known, the approximate beliefprint $\hat{\mathcal{B}}_k$



(a) Transformed polyhedron $\mathcal{P}_k$. **(b)** Approximate beliefprint $\hat{\mathcal{B}}_k$.

Figure 3.7 Illustration of the transformed polyhedron $\mathcal{P}_k$ and the approximate beliefprint $\hat{\mathcal{B}}_k$. The latter describes the convex hull of the footprints transformed to the s vertices $\mathbf{p}_k^{[i]}$ of $\mathcal{P}_k$. Note that the ellipsoid in (a) is already known from Fig. 3.4(a) and encloses the same probability mass as $\mathcal{P}_k$. Adapted from [213], © 2018 IEEE.

can be computed similar to [117] in two steps: (1) transform the vehicle's footprint $\mathcal{F}$ to the s vertices $\mathbf{p}_k^{[i]}$ of $\mathcal{P}_k$, and (2) compute the convex hull of these transformed footprints in order to end up with $\hat{\mathcal{B}}_k$. A visualization of both steps can be found in Fig. 3.7(b), and a summarizing description of the algorithm in Alg. 3.

Algorithm 3 Approximate Beliefprint (adapted from [213], © 2018 IEEE)

approx_beliefprint$(\mu_k, \mathbf{\Gamma}_k, \mathcal{P}, \mathcal{F})$**:**

1: $\mathbf{L}_k \mathbf{L}_k^\intercal = \text{cholesky}(\mathbf{\Gamma}_k)$
2: $\mathcal{F}_k = \emptyset$
3: **for** $i = 1 : s$ **do**
4: $\quad \mathbf{p}_k^{[i]} = \mathbf{L}_k \mathbf{p}^{[i]} + \mu_k$
5: $\quad \mathcal{F}_k^{[i]} = \text{transform}(\mathcal{F}, \mathbf{p}_k^{[i]})$
6: $\quad \mathcal{F}_k \mathrel{+}= \langle \mathcal{F}_k^{[i]} \rangle$
7: **end for**
8: $\hat{\mathcal{B}}_k = \text{convhull}(\mathcal{F}_k)$
9: **return** $\hat{\mathcal{B}}_k$

A step-wise analysis of the described computations allows to derive the time complexity of this algorithm. Remember that the dimension of the vehicle's state is denoted by n and the polyhedron's number of vertices by s.

Furthermore, it is assumed that the footprint is composed of p points. On this basis, the time complexity of Alg. 3 is given by $O(n^3 + sn^2 + sp\log(sp))$. More precisely, the Cholesky factorization in Line 1 has a complexity of $O(n^3)$ [65], the for loop in Lines 3–7 is $O(s(n^2 + p))$, and the complexity of the convex hull operation in Line 8 is given by $O(sp\log(sp))$ [148]. Based on these results, it is expected that the total number of generated points sp has a major influence on the overall runtime of this algorithm. Further details on the actual computation time for different values of s and p are highlighted in the next section.

Finally, it remains to be answered in which cases Alg. 3 yields a conservative approximation of the beliefprint $\mathcal{B}_k$. In general, conservatism can be guaranteed if the approximate beliefprint describes the union of all footprints transformed to the infinitely many points on the surface of $\mathcal{P}_k$. The reason for this is that the polyhedron $\mathcal{P}_k$ encloses the probability mass P exactly, however, its vertices lie further away from the mean than all points on the corresponding ellipsoid (see Fig. 3.7(a)) that is used to compute $\mathcal{B}_k$.

Under the assumption that the robot has only translational DOFs, the previously described problem of transforming the footprint $\mathcal{F}$ to all points on $\mathcal{P}_k$ corresponds to the computation of the Minkowski sum [45]. If both $\mathcal{P}_k$ and $\mathcal{F}$ are convex, it is known from e.g. [61] that Alg. 3 computes exactly this Minkowski sum and by doing so, derives a conservative approximation of $\mathcal{B}_k$. In case of a non-convex footprint, a convexification must first be conducted in order to approximate $\mathcal{B}_k$ conservatively [166].

In contrast to that, if the state of the robot also contains a rotational DOF, it can generally no longer be ensured that the output of Alg. 3 yields a conservative approximation of $\mathcal{B}_k$. This is due to the nonlinearity of the rotation, which not only requires the evaluation of $\mathcal{F}$ at the vertices of $\mathcal{P}_k$, but rather on its whole surface. However, one exemption can be identified: if the footprint $\mathcal{F}$ only consists of a single point (or a disk), the rotational DOF has no effect on the actual shape of $\mathcal{B}_k$. In this case, the output from Alg. 3 yields a conservative approximation of $\mathcal{B}_k$ as it can be seen in Fig. 3.8

3.2.1.1 Experimental Evaluation

The experimental results in this section compare the performance of the proposed method with sigma hulls (SHs) [117] and an MC approach. The latter draws a given number of samples from the Gaussian distribution such that the user-defined confidence P is satisfied with a certain probability [25].

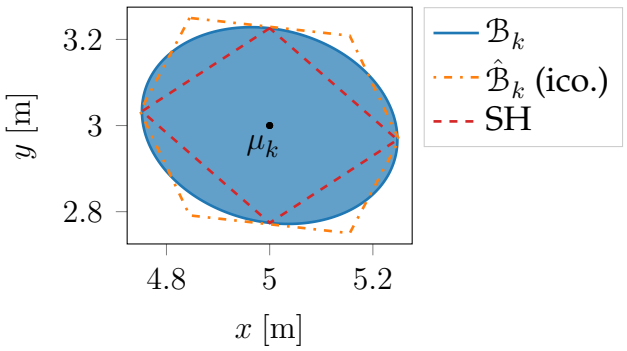


Figure 3.8 Beliefprint $\mathcal{B}_k$ of a point robot given the three-dimensional Gaussian distribution from Fig. 3.4(a). In contrast to SHs [117], the approximate beliefprint $\hat{\mathcal{B}}_k$ (computed here with an icosahedron) captures additional probability mass at the corners resulting in a more conservative/larger beliefprint than $\mathcal{B}_k$ itself.

In the experiments below, this probability is set to 99 %, and the three-sigma confidence in three dimensions (97.1 %) is chosen for P. On the basis of these parameters, Fig. 3.9 compares the shape of the beliefprint $\mathcal{B}_k$ with the approximations of the other three approaches given the Gaussian belief from Fig. 3.4(a).

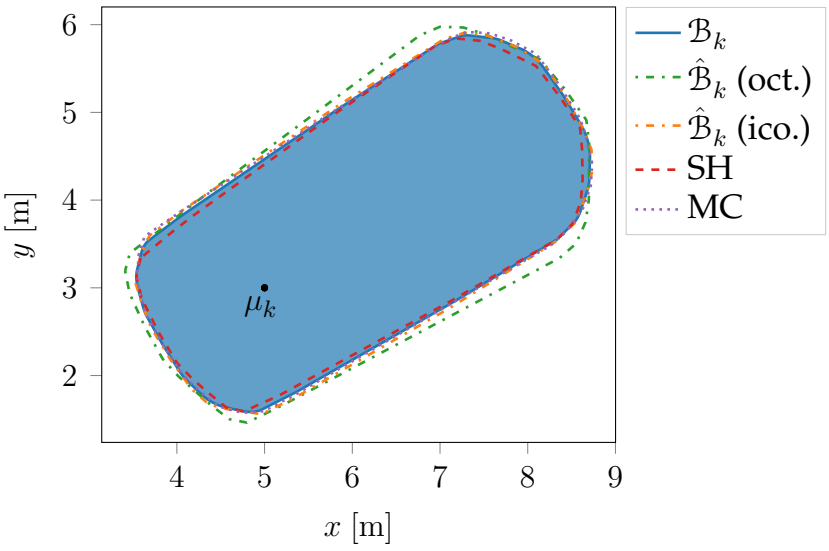


Figure 3.9 Comparison of the beliefprint $\mathcal{B}_k$ with the approximate beliefprint $\hat{\mathcal{B}}_k$ of an octahedron/icosahedron, a sigma hull (SH) [117], and the result of an MC approach given the Gaussian belief from Fig. 3.4(a).

It can be seen in Fig. 3.9 that the SH lies completely within $\mathcal{B}_k$ and thus might result in situations where the actual CP exceeds the user-defined threshold $1 - P$. As opposed to this, the approximate beliefprint $\hat{\mathcal{B}}_k$ of the

octahedron leads to a fairly conservative approximation of $\mathcal{B}_k$. The reason for this is that the octahedron consists of only six vertices, which must be placed further away from the mean to capture the same probability mass as e.g. an icosahedron with its twelve vertices (see Fig. 3.5). In fact, it can be observed that both the icosahedron as well as the MC approach yield the closest approximation of $\mathcal{B}_k$ in this example. However, both methods differ significantly with respect to their computational effort as it is shown in the following benchmark.

Especially in real-time applications, methods with a low computation time are required. Therefore, the different beliefprint approximations are now compared with respect to their average computation time given 10^4 random beliefs. All methods are implemented in C++ and benchmarked on a single core of an Intel Xeon E5@3.5 GHz. The corresponding values can be found in Tab. 3.1, where the shape of the footprint is varied between a rectangle with four points $\mathcal{F}_{\text{rectangle}}$ and a polygon with 20 points $\mathcal{F}_{\text{contour}}$.

Table 3.1 Benchmark of the average computation time [µs] for a single beliefprint approximation. Adapted from [213], © 2018 IEEE.

	MC	SH	oct.	ico.
$\mathcal{F}_{\text{rectangle}}$	750.6	3.5	3.6	5.2
$\mathcal{F}_{\text{contour}}$	1708.2	9.1	9.8	17.2

It can be seen that the computation of a sigma hull takes about as long as the execution of Alg. 3 in combination with an octahedron. In contrast to that, replacing the octahedron with an icosahedron raises the average computation time by up to 75.5 %. A similar trend can be observed when comparing the computation times for the different footprints $\mathcal{F}_{\text{rectangle}}$ and $\mathcal{F}_{\text{contour}}$. These results are aligned with the analysis of the time complexity above and therefore not further detailed here. In addition to that, it has to be mentioned that the MC approach is orders of magnitude slower than the three other approaches limiting its usability in real-time applications.

3.2.2 Robust Beliefprint

The insight that the nonlinearity of the rotation prevents the previous method from being conservative inspired the algorithm presented in this section. The general idea here is to first construct a hull that completely captures the

uncertainty of the rotational DOF and then use this result to also enclose the uncertainty of the vehicle's position. The output of this method is called robust beliefprint $\bar{\mathcal{B}}_k$ as it guarantees (under the premise of Footnote 6 (p. 102)) to approximate the beliefprint $\mathcal{B}_k$ conservatively.

The results in this section are currently limited to a three-dimensional Gaussian belief $bel(\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mu_k, \mathbf{\Gamma}_k)$ whose components are given as

$$bel(\mathbf{x}_k) = \mathcal{N}\left(\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}; \begin{pmatrix} \mu_{x,k} \\ \mu_{y,k} \\ \mu_{\theta,k} \end{pmatrix}, \begin{pmatrix} \Gamma_{xx,k} & \Gamma_{xy,k} & \Gamma_{x\theta,k} \\ \Gamma_{yx,k} & \Gamma_{yy,k} & \Gamma_{y\theta,k} \\ \Gamma_{\theta x,k} & \Gamma_{\theta y,k} & \Gamma_{\theta\theta,k} \end{pmatrix}\right). \tag{3.6}$$

Note that the covariance matrix $\mathbf{\Gamma}_k$ is symmetric and that x_k and y_k describe the vehicle's position and θ_k its heading angle.

Similar to Sec. 3.2.1, the overall goal in the following derivation is to derive a volume $\mathcal{V}_k \subset \mathbb{R}^3$ that encloses the probability mass P of the current belief $bel(\mathbf{x}_k) = p(x_k, y_k, \theta_k)$ according to

$$P = \int\int\int_{\mathcal{V}_k} p(x_k, y_k, \theta_k) \ \mathrm{d}x_k \ \mathrm{d}y_k \ \mathrm{d}\theta_k. \tag{3.7}$$

This equation can be modified by applying the definition of conditional probability [186] as

$$P = \int\int\int_{\mathcal{V}_k} p(\theta_k|x_k, y_k) p(x_k, y_k) \ \mathrm{d}x_k \ \mathrm{d}y_k \ \mathrm{d}\theta_k, \tag{3.8a}$$

$$P = \int\int_{\mathcal{A}_k} p(x_k, y_k) \int_{\mathcal{H}_k} p(\theta_k|x_k, y_k) \ \mathrm{d}\theta_k \ \mathrm{d}x_k \ \mathrm{d}y_k, \tag{3.8b}$$

where the volume $\mathcal{V}_k$ is decomposed into the two-dimensional area $\mathcal{A}_k \subset \mathbb{R}^2$ and into the one-dimensional bound $\mathcal{H}_k \subset \mathbb{R}$ that may depend on x_k and y_k. For a Gaussian belief, it is known that both the marginal distribution $p(x_k, y_k)$ as well as the conditional distribution $p(\theta_k|x_k, y_k)$ remain Gaussians. In particular, the latter is described by $\mathcal{N}(\theta|x, y; \mu_{\theta|x,y}, \Gamma_{\theta|x,y})$, where the index k is dropped in the following for better readability. The mean $\mu_{\theta|x,y}$ and the variance $\Gamma_{\theta|x,y}$ of this distribution are given according to [11] by

$$\mu_{\theta|x,y} = \mu_\theta + \begin{pmatrix} \Gamma_{\theta x} & \Gamma_{\theta y} \end{pmatrix} \begin{pmatrix} \Gamma_{xx} & \Gamma_{xy} \\ \Gamma_{yx} & \Gamma_{yy} \end{pmatrix}^{-1} \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} \right), \tag{3.9a}$$

$$\Gamma_{\theta|x,y} = \Gamma_{\theta\theta} - \begin{pmatrix} \Gamma_{\theta x} & \Gamma_{\theta y} \end{pmatrix} \begin{pmatrix} \Gamma_{xx} & \Gamma_{xy} \\ \Gamma_{yx} & \Gamma_{yy} \end{pmatrix}^{-1} \begin{pmatrix} \Gamma_{x\theta} \\ \Gamma_{y\theta} \end{pmatrix}. \tag{3.9b}$$

Two important facts can be observed in (3.9) that are used later on: (1) the mean $\mu_{\theta|x,y}$ is a linear function of the vehicle's position $\begin{pmatrix} x & y \end{pmatrix}$, and (2) the variance $\Gamma_{\theta|x,y}$ is constant and can be determined with the values from (3.6).

Using the previous results, (3.8b) can be reformulated as

$$P = \int \int_{\mathcal{A}} p(x,y) \int_{\mu_{\theta|x,y}-\Delta\theta}^{\mu_{\theta|x,y}+\Delta\theta} \mathcal{N}(\theta|x,y;\mu_{\theta|x,y},\Gamma_{\theta|x,y}) \ \mathrm{d}\theta \ \mathrm{d}x \ \mathrm{d}y, \tag{3.10}$$

where the integration bound $\mathcal{H}$ is replaced with $\mu_{\theta|x,y} \pm \Delta\theta$. The general idea behind this substitution is to capture the uncertainty of the rotational DOF in an interval $\pm\Delta\theta$ around the mean $\mu_{\theta|x,y}$ of the underlying distribution. This allows to further simplify (3.10) by shifting the integration bounds to

$$P = \int \int_{\mathcal{A}} p(x,y) \int_{-\Delta\theta}^{+\Delta\theta} \mathcal{N}(\theta|x,y;0,\Gamma_{\theta|x,y}) \ \mathrm{d}\theta \ \mathrm{d}x \ \mathrm{d}y. \tag{3.11}$$

The inner integral can now be solved leading to the final result

$$P = \int \int_{\mathcal{A}} \operatorname{erf}\left(\frac{\Delta\theta}{\sqrt{2\Gamma_{\theta|x,y}}}\right) p(x,y) \ \mathrm{d}x \ \mathrm{d}y, \tag{3.12a}$$

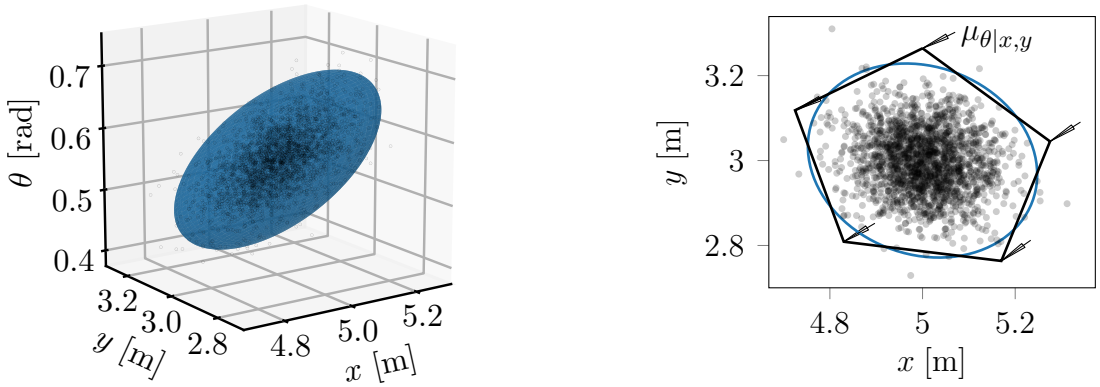
$$P = \underbrace{\operatorname{erf}\left(\frac{\Delta\theta}{\sqrt{2\Gamma_{\theta|x,y}}}\right)}_{P_{\text{rot}}} \underbrace{\int \int_{\mathcal{A}} p(x,y) \ \mathrm{d}x \ \mathrm{d}y}_{P_{\text{trans}}}, \tag{3.12b}$$

where the fact that the variance $\Gamma_{\theta|x,y}$ from (3.9b) does not depend on the variables x and y is leveraged. On this basis, it can be concluded that an alternative way to satisfy P is to compute an interval $\pm\Delta\theta$ and an area $\mathcal{A}$ such that they enclose the probability mass P_{rot} and P_{trans}, respectively. The corresponding values for P_{rot} and P_{trans} must be defined by the user such that the multiplication of both terms gives the confidence P (see (3.12b)). For instance, one can set P to the three-sigma value in three dimensions (97.1 %), and P_{trans} to the three-sigma value in two-dimensions (98.9 %), which yields $P_{\text{rot}} = P/P_{\text{trans}}$. These confidences can then be used to determine both $\Delta\theta$ and $\mathcal{A}$ analytically by applying the results from Sec. 3.2.1.

For instance, $\Delta\theta$ can be obtained by solving (3.2) offline ($n = 1$ and confidence P_{rot}) and by transforming the resulting two bounds similar to (3.5) with $\sqrt{\Gamma_{\theta|x,y}}$ [59]. This procedure has the advantage that it only requires a simple linear transformation in the online phase of the algorithm. The two-dimensional area $\mathcal{A}$ in (3.12b) can also be derived with the method from

Sec. 3.2.1 using $n = 2$ and the confidence P_{trans}. In this case, regular polygons are promising candidates for the integration in (3.2) due to their symmetry.

The results of the previous analysis are visualized in Fig. 3.10.



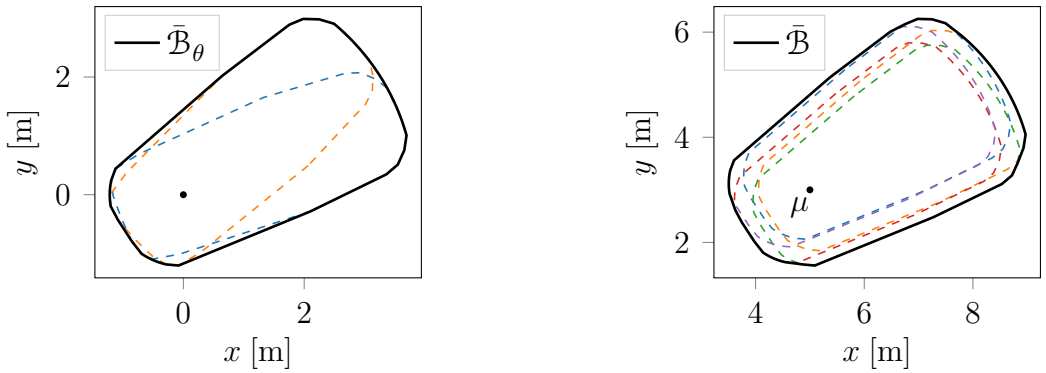
(a) Joint distribution $p(x, y, \theta)$. **(b)** Marginal distribution $p(x, y)$.

Figure 3.10 Visualization of the joint and the marginal distribution along with the ellipsoid (a) and the ellipse (b) that indicate the user-defined confidence P and P_{trans}. In addition to that, the polygon $\mathcal{A}$ and the mean $\mu_{\theta|x,y}$ at the corners of $\mathcal{A}$ are also shown in (b). Note that the illustrated wedges in (b) indicate the $\pm\Delta\theta$ bound derived from the confidence P_{rot}.

Here, a polygon with five vertices is chosen to enclose the probability mass P_{trans} of the marginal distribution $p(x, y)$. Furthermore, the wedges at the respective vertices display the range $\pm\Delta\theta$ that is required to also capture P_{rot}. From the previous analysis, it is known that $\Delta\theta$ is constant for all positions in $\mathcal{A}$, however, the mean heading angle $\mu_{\theta|x,y}$ changes linearly with respect to x and y (see (3.9a)). This effect can also be seen in Fig. 3.10(b) by comparing the corresponding values visualized at the corners of $\mathcal{A}$.

In order to ensure conservatism in the computation of the robust beliefprint, the dependence of $\mu_{\theta|x,y}$ on x and y must first be eliminated. Here, it is proposed to over-approximate this value with the maximum $\bar{\mu}_{\theta|x,y}$ and the minimum $\underline{\mu}_{\theta|x,y}$ that occur on $\mathcal{A}$. Under the assumption that a polygon is used to describe $\mathcal{A}$, finding these extrema is equal to solving a linear program, where (3.9a) defines the objective function and $\mathcal{A}$ the corresponding constraints. Fortunately, it is known from linear optimization that both extrema occur at the corners of the underlying polygon [9]. Therefore, it is sufficient to evaluate $\mu_{\theta|x,y}$ at these positions (see Fig. 3.10(b)) and then to take the maximum value as $\bar{\mu}_{\theta|x,y}$ and the minimum as $\underline{\mu}_{\theta|x,y}$.

On this basis, a conservative hull $\bar{\mathcal{B}}_\theta$ can be computed that fully encloses the uncertainty of the rotation. As shown in Fig. 3.11(a), the corresponding procedure requires to rotate the footprint $\mathcal{F}$ from $\underline{\mu}_{\theta|x,y} - \Delta\theta$ to $\bar{\mu}_{\theta|x,y} + \Delta\theta$ and to over-approximate the swept area with a polygon. Note that during



(a) Rotational hull $\bar{\mathcal{B}}_\theta$. **(b)** Robust beliefprint $\bar{\mathcal{B}}$.

Figure 3.11 Conservative approximation $\bar{\mathcal{B}}_\theta$ of the rotational uncertainty (a) and illustration of the resulting beliefprint $\bar{\mathcal{B}}$ (b) given the Gaussian belief from Fig. 3.10. The footprints (dashed lines in (a)) are obtained by rotating $\mathcal{F}$ from $\underline{\mu}_{\theta|x,y} - \Delta\theta$ to $\bar{\mu}_{\theta|x,y} + \Delta\theta$. In contrast to that, the dashed lines in (b) illustrate the rotational hull $\bar{\mathcal{B}}_\theta$ transformed to the five vertices of the polygon $\mathcal{A}$ visualized in Fig. 3.10(b).

this rotation, each point on the footprint moves on a circle centered at the origin, allowing to derive efficient methods for the computation of $\bar{\mathcal{B}}_\theta$.

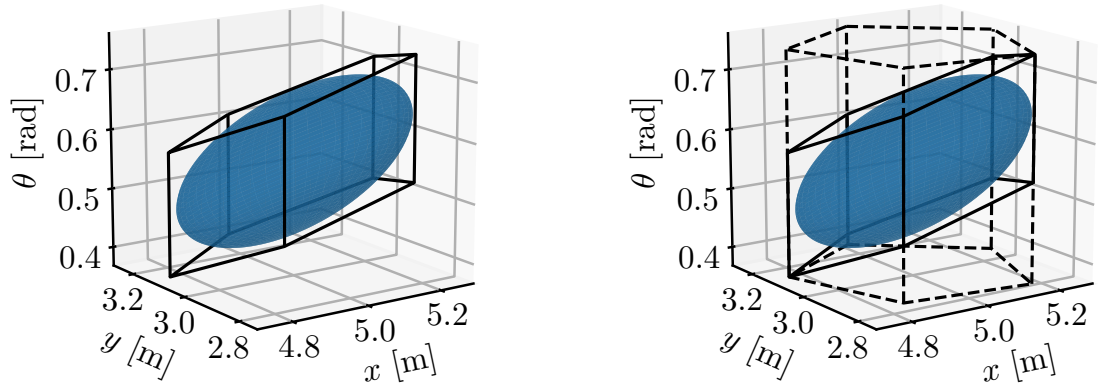
Building on the results of Sec. 3.2.1, the translational uncertainty can now be taken into account by translating $\bar{\mathcal{B}}_\theta$ to the corners of $\mathcal{A}$ and by taking the convex hull of the resulting polygons. A visualization of this procedure, which outputs the robust beliefprint $\bar{\mathcal{B}}$, is given in Fig. 3.11(b). It has to be noted that this approximation is guaranteed to be conservative as no assumptions are made in the integration process and only geometric over-approximations are used to derive $\bar{\mathcal{B}}$. Furthermore, the full correlation of the underlying belief is considered here, and no independence assumptions between the rotational and the translational uncertainty, as e.g. in [71], are made. To the author's best knowledge, this algorithm is the first approach that provides these guarantees while at the same time, only requires analytical evaluations (apart from the precomputations that can be executed offline).

3.2.2.1 Geometric Interpretation

The aim of this section is to briefly analyze the concept behind the previously introduced algorithm from a geometric point of view. This interpretation

is meant to provide a better understanding of the underlying computation steps that were only highlighted from a mathematical perspective above.

Starting with the insights from (3.9)–(3.12) that the rotational uncertainty can be captured in an interval $\mu_{\theta|x,y} \pm \Delta\theta$ and assuming that the area $\mathcal{A}$ is described by a polygon with five vertices leads to the visualization in Fig. 3.12(a).



(a) Pentagonal prism. **(b)** Over-approximated prism.

Figure 3.12 Geometric interpretation of (3.9)–(3.12) in (a) and visualization of the over-approximation (dashed lines) made in the robust beliefprint computation (b). Both images are based on the Gaussian belief given in Fig. 3.10(a).

Here, a pentagonal prism is constructed such that it encloses the probability mass P of the underlying belief while keeping its vertical edges parallel to the θ-axis. In fact, the height of this prism corresponds to the previously derived value $2\Delta\theta$, and the planes running through its upper and lower base are given by $\mu_{\theta|x,y} \pm \Delta\theta$. Furthermore, projecting all vertices of that prism onto the ground results in the area $\mathcal{A}$ known from (3.12b).

As the rotational DOF still depends on x and y in Fig. 3.12(a), the proposed algorithm conducts the over-approximation shown in Fig. 3.12(b). Here, a new prism (dashed lines) is constructed that fully encloses the previous one and whose upper and lower base run parallel to the xy-plane. This over-approximation corresponds to the computation of $\bar{\mu}_{\theta|x,y}$ and $\underline{\mu}_{\theta|x,y}$ that, augmented by $\pm\Delta\theta$, describe the two vertical limits shown in Fig. 3.12(a).

Now that the rotational and the translational DOFs are decoupled, one can proceed with the computation of $\bar{\mathcal{B}}_\theta$ and $\bar{\mathcal{B}}$ as described above and visualized in Fig. 3.11.

3.2.2.2 Experimental Evaluation

Finally, it remains to be answered how conservative the robust beliefprint $\bar{\mathcal{B}}_k$ is compared to $\mathcal{B}_k$ and the approximation $\hat{\mathcal{B}}_k$ from Sec. 3.2.1. For this purpose, Fig. 3.13 shows a comparison of the outputs obtained from the previously described methods given the Gaussian belief from Fig. 3.10(a).

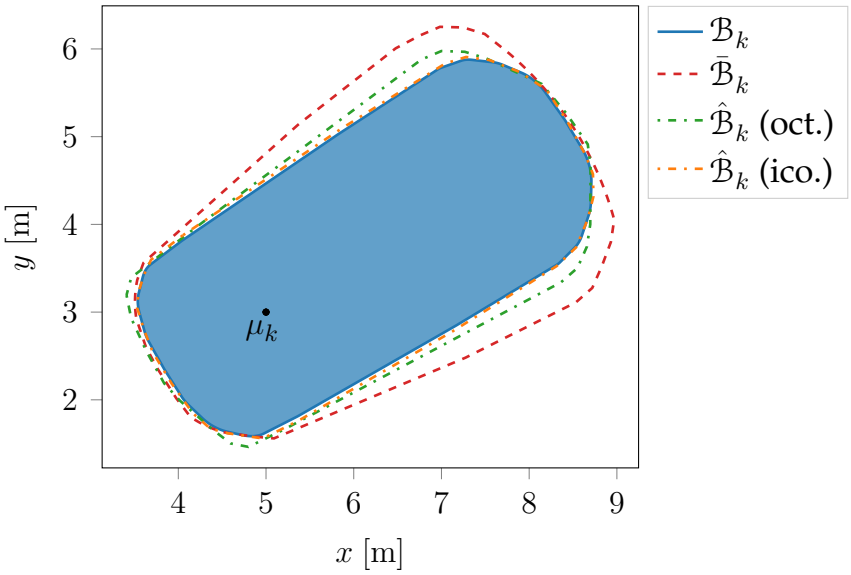


Figure 3.13 Comparison of the beliefprint $\mathcal{B}_k$ with the robust beliefprint $\bar{\mathcal{B}}_k$ and two variants of the approximate beliefprint $\hat{\mathcal{B}}_k$ computed with an octahedron/icosahedron given the Gaussian belief from Fig. 3.10(a).

Once again, it can be observed that the best approximation of $\mathcal{B}_k$ is generated by $\hat{\mathcal{B}}_k$ using an icosahedron. However, it cannot be generally guaranteed that the corresponding result approximates $\mathcal{B}_k$ conservatively. As opposed to that, the robust beliefprint $\bar{\mathcal{B}}_k$ enforces conservatism, which leads in the given example to a significant over-approximation of $\mathcal{B}_k$. This could be a problem in tight environments, where the robust beliefprint $\bar{\mathcal{B}}_k$ might indicate a collision although $\mathcal{B}_k$ is collision-free. From a practical point of view, it might therefore be required to rely on the approximate and not on the robust beliefprint in order to avoid a too conservative behavior of the robot.

3.3 Summary

Motion planning in belief space requires approaches that allow to evaluate whether a given belief exceeds a user-defined collision probability (CP). Under the assumption of Gaussian beliefs, two novel algorithms are introduced

in this chapter that calculate the so-called beliefprint: the area occupied by the robot given its footprint, a belief distribution, and the user-defined confidence P. These beliefprints can then be used for collision checking to ensure that the CP of a vehicle state does not exceed the given threshold $1 - P$.

Based on the insight that the actual shape of the beliefprint cannot be computed efficiently, the two algorithms in this chapter focus on conservative and efficient approximations thereof. While the approach in Sec. 3.2.1 can only ensure conservatism in certain cases, e.g. for a robot with only translational DOFs, the extension in Sec. 3.2.2 provides this guarantee for a robot with one rotational and two translational DOFs. The latter is typically the case in automated driving, where the vehicle's position and orientation are frequently used for two-dimensional collision checking. In this case, it is shown that the approximate beliefprint from Sec. 3.2.1 already results in a fairly precise approximation of the actual beliefprint. In contrast, the robust beliefprint in Sec. 3.2.2 outputs a more conservative area that might lead to a too defensive behavior when actually being deployed on a real robot.

With respect to efficiency, the presented approaches allow to perform all complex calculations offline such that only analytical evaluations are required in the online phase of the algorithms. This two-stage approach reduces the computation time to microseconds as it is shown in Sec. 3.2.1.1 for the calculation of the approximate beliefprint.

In the future, it would be interesting to extend the presented algorithms to belief representations other than the Gaussian distribution considered in this chapter. Moreover, it remains an interesting question how the provided concept can be used for the computation of an obstacle's beliefprint. In this case, the proposed algorithms must be extended such that the perception uncertainty can be taken into account as well.

4 Learning Pose Predictions for Guided Motion Planning

Motion planning in complex environments often requires problem-specific heuristics for the fast computation of a preferably optimal solution [28, 40, 77, 207]. In automated driving, however, the large variety of possible scenarios as well as the nonholonomic constraints of the vehicle make it difficult to design such heuristics manually. Therefore, a data-driven approach is proposed in this chapter that allows to guide a motion planner through complex scenarios. More specifically, a convolutional neural network (CNN) is presented that predicts future ego-vehicle poses from a start to a goal state given observations of the current environment. As shown in Fig. 4.1, these predicted poses can then be used by a sampling-based motion planner, such

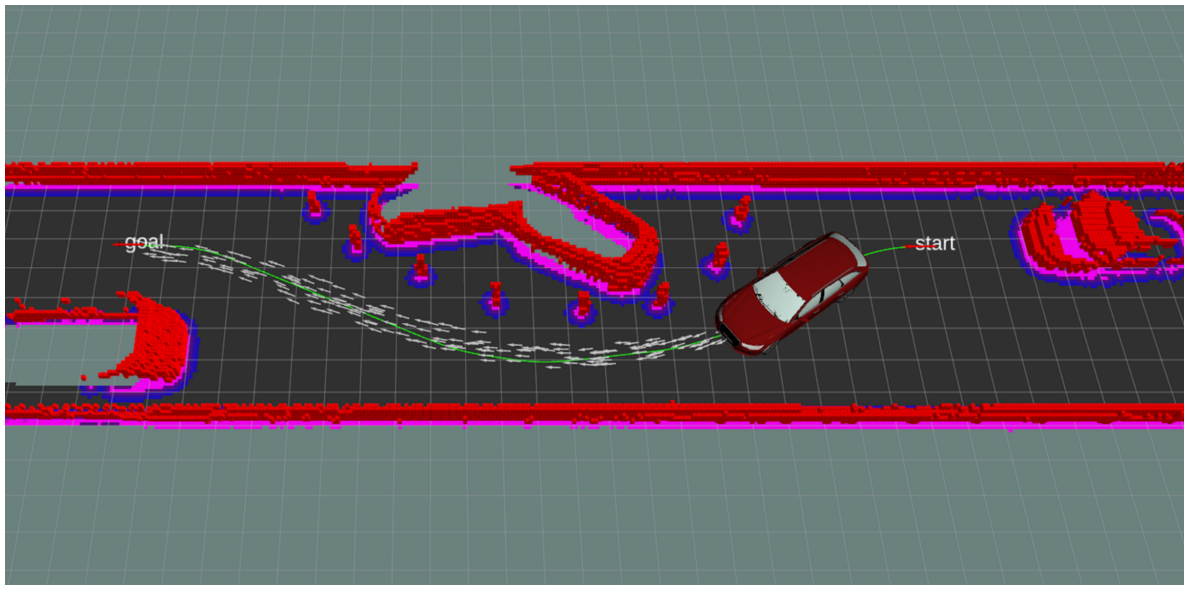


Figure 4.1 "Evasive maneuver due to an accident blocking the road. The path of the ego-vehicle (green line) is computed using the learned vehicle pose predictions (gray arrows). The obstacles in the environment are visualized by the red voxels, and the corresponding two-dimensional cost map for motion planning is depicted on the ground" [216]. Reprinted from [216], © 2019 IEEE.

as RRT* [93], along with one of the steering functions from Ch. 2 to compute a feasible path to the desired goal. Biasing the motion planner in this way has the potential to accelerate the convergence and improve the success rate in challenging environments. These two aspects are further detailed in the guided motion planning experiments in Sec. 5.3.5.

The remainder of this chapter is organized as follows: Sec. 4.1 reviews the state of the art, Sec. 4.2 highlights the generation process of the training data, and Sec. 4.3 describes the CNN used for the prediction of the future ego-vehicle poses. A benchmark of its performance against uniform sampling and an A*-based approach can be found in Sec. 4.4, and a concluding summary is finally given in Sec. 4.5. It has to be noted that parts of this chapter have previously been published in [216] and in the extended version [215].[7]

4.1 State of the Art

"Recent advances in deep learning have opened up new possibilities to improve or even replace existing motion planning algorithms for robot navigation. For instance, impressive results have been achieved in the field of imitation learning, where a robot is trained to act based on expert demonstrations, such as end-to-end learning for self-driving cars" [215].

End-to-end learning While early achievements in this field go back to 1989 [147], significant progress has been made with the possibility to train even more complex (deeper) models end-to-end. An example of such an approach for lane following is presented in [15], where a CNN learns to control the steering angle of a vehicle given raw camera inputs. Building on these results, the adapted models in [36, 74] also take into account a discrete set of navigation commands to initiate e.g. a turn at an intersection.

A known problem of the previously described approaches is that they only compute a reactive policy given the raw measurements of a front-facing sensor. This is especially an issue from a safety perspective as hard constraints, such as collision avoidance or actuator limits, cannot be explicitly enforced.

[7] Various passages in the text below are directly quoted (sometimes over multiple pages) from these two publications, where the following formatting is applied: (1) direct quotations are marked by "..." [•] as it can be seen in the caption of Fig. 4.1, (2) modified references (to figures, tables, equations, bibliography) for a consistent appearance in this thesis are not explicitly highlighted inside the quoted text, however, the quotation marks are changed to '...' [•], and (3) all other changes to the original content are indicated by square brackets.

In addition to that, a 360° surround view might be required to make safe decisions in challenging traffic situations. To overcome these problems, a recurrent neural network is trained in [5] that outputs an entire trajectory given a detailed top-down representation of the environment. One of the advantages of this approach is that it takes into account the entire surrounding while allowing for a further optimization of the computed trajectory with respect to comfort and safety. However, with its focus on corridor driving, this method might fail in situations that require a global solution to the motion planning problem.

Hybrid approaches The fact that learning-based approaches cannot (yet) enforce hard constraints motivates the introduction of so-called hybrid approaches that combine classic motion planning algorithms with machine learning techniques. By doing so, it is possible to maintain the theoretical guarantees of the motion planner (optimality and completeness) while improving its overall performance. 'Depending on the planning algorithm, different approaches exist in the literature. **Optimization-based planners** [emphasis added], for example, often suffer from short planning horizons. This issue can be resolved with a learned cost shaping term that aligns the short-term horizon with the long-term goal as proposed in [183].

In contrast to that, **search-based planners** [emphasis added] in continuous state-action space require an action sampling distribution for graph expansion and a cost-to-go heuristic for node selection. The former is addressed in [98], where a generative adversarial network [67] is trained to guide the search towards the goal. Minimizing search effort through learned cost-to-go heuristics is achieved in [34]. However, iteratively evaluating the heuristic during search limits the capacity of the neural network due to real-time constraints. Furthermore, optimality can only be guaranteed in a multi-heuristic setup' [215].

With respect to **sampling-based** motion **planning**, it is generally known that biasing the sampling distribution towards promising regions in the state space can significantly improve the performance of the planner [56, 197–199, 207]. Especially deep learning-based approaches allow to compute such non-uniform distributions conditioned on various features such as the environment of the robot or a desired goal state. For instance, both [75] and [107] train a neural network to guide a sampling-based local motion planner in an environmental aware fashion. In contrast to that, sampling distributions for global motion planning are learned in [77, 123] using a

conditional variational autoencoder (CVAE). Although these results appear promising, further studies are required to evaluate the CVAE's capability to avoid obstacles while generalizing to a broad set of scenarios. Another interesting approach in this field is presented in [150], where dropout instead of a generative model is used to compute samples for the motion planner. Future research must investigate how such a method performs in motion planning for automated vehicles, where the static environment is often represented by a high-dimensional occupancy grid.

Opposed to the previous approaches, [138] directly computes a sampling distribution over future robot positions using a fully convolutional neural network. Such a distribution can then be sampled to guide the planner towards a cost-minimizing solution. Note that this approach is similar to the one presented in this chapter, however, the method proposed below computes not only a distribution over future vehicle positions as in [138], but also provides the heading angle at a predicted position. Especially in nonholonomic motion planning, where the vehicle is not capable to make a turn on the spot, it is essential to provide the planner with this information to both accelerate the planning and improve the convergence rate.

Instead of directly computing a non-uniform sampling distribution, the approaches in [105, 200] still rely on uniform sampling, but train a neural network that either rejects [200] or modifies the generated samples locally [105]. In problems with narrow passages, such strategies may, however, still fail as uniform sampling remains at the core of both approaches.

Just recently, several novel planning algorithms [78, 149, 176, 178] are proposed "that conduct planning in a learned latent space rather than in the complex configuration space. The general idea is to simplify the planning problem by solving it in the lower-dimensional latent space. While still in an early development phase, the future of these approaches in safety-critical applications highly depends on the possibility to satisfy hard constraints like collision avoidance" [215].

4.2 Data Generation

This section is extracted from [215]: 'Learning ego-vehicle predictions in a supervised fashion requires a diverse dataset consisting of the vehicle's trajectory from start to goal and the corresponding observations of the environment. Such a dataset with a total number of 13 418 trajectories is recorded

in a [...] simulator using Gazebo and ROS. The implemented data generation process is fully automated and parallelized with multiple simulation instances, and requires no time-consuming manual labeling of the recorded training data.

Each recording is generated in one of the eleven scenarios visualized in Fig. 4.2. As this research is embedded in a project with a focus on low-speed maneuvering in tight environments, the designed scenarios focus on challenging setups that require the vehicle to act precisely in a highly non-convex workspace. Exemplary situations include blocked roads, dead ends, or different parking problems. Variation is introduced in each scenario by changing the start and goal pose of the ego-vehicle as well as the number and location of the static obstacles. For unstructured environments, this is implemented by randomly sampling the obstacles and the vehicle's start and goal pose. For semi-structured environments, obstacles are randomly assigned to predefined locations, such as parking spots. The ego-vehicle's start and goal pose are, in this case, randomly chosen within predefined locations corresponding to the entrance and exit of a scenario, the driveway, or a parking spot.

The following procedure is then applied to obtain a recording in the randomly configured instances of each scenario. First, the motion planner BiRRT* generates a curvature-continuous collision-free path from start to goal as detailed in [Ch. 5]. Here, BiRRT* is guided by the A*-based heuristic described in Sec. 4.4, and its initial motion plan is optimized for 5 s. Next, a motion controller executes the computed solution resulting in a trajectory with $t = 1, \dots, T$ vehicle states [$\mathcal{X} = \langle \mathbf{x}_1, \dots, \mathbf{x}_T \rangle$], where a state $\mathbf{x}_t = [\begin{pmatrix} x_t & y_t & \theta_t & \kappa_t & v_t \end{pmatrix}^\intercal]$ at time step t is defined by the vehicle's position [$\begin{pmatrix} x_t & y_t \end{pmatrix}$], orientation θ_t, curvature κ_t, and velocity v_t. Note that throughout this [thesis], the term vehicle pose is only used to describe a lower-dimensional subset of the state including the vehicle's position and orientation. Finally, the observations from a simulated LiDAR perception are fused in a two-dimensional 60 m $\times$ 60 m occupancy grid with a resolution of 10 cm and recorded along the trajectory. A visualization of such a recording can be found in Fig. 4.4 on the [top] left.

It has to be noted that, similar to [34], motion planning for data generation is conducted with full environmental information in order to guarantee fast convergence to a cost-minimizing solution. In contrast to that, the recorded occupancy grid only fuses the current history of sensor observations resulting in unobserved areas due to occlusions. This forces the model in the learning

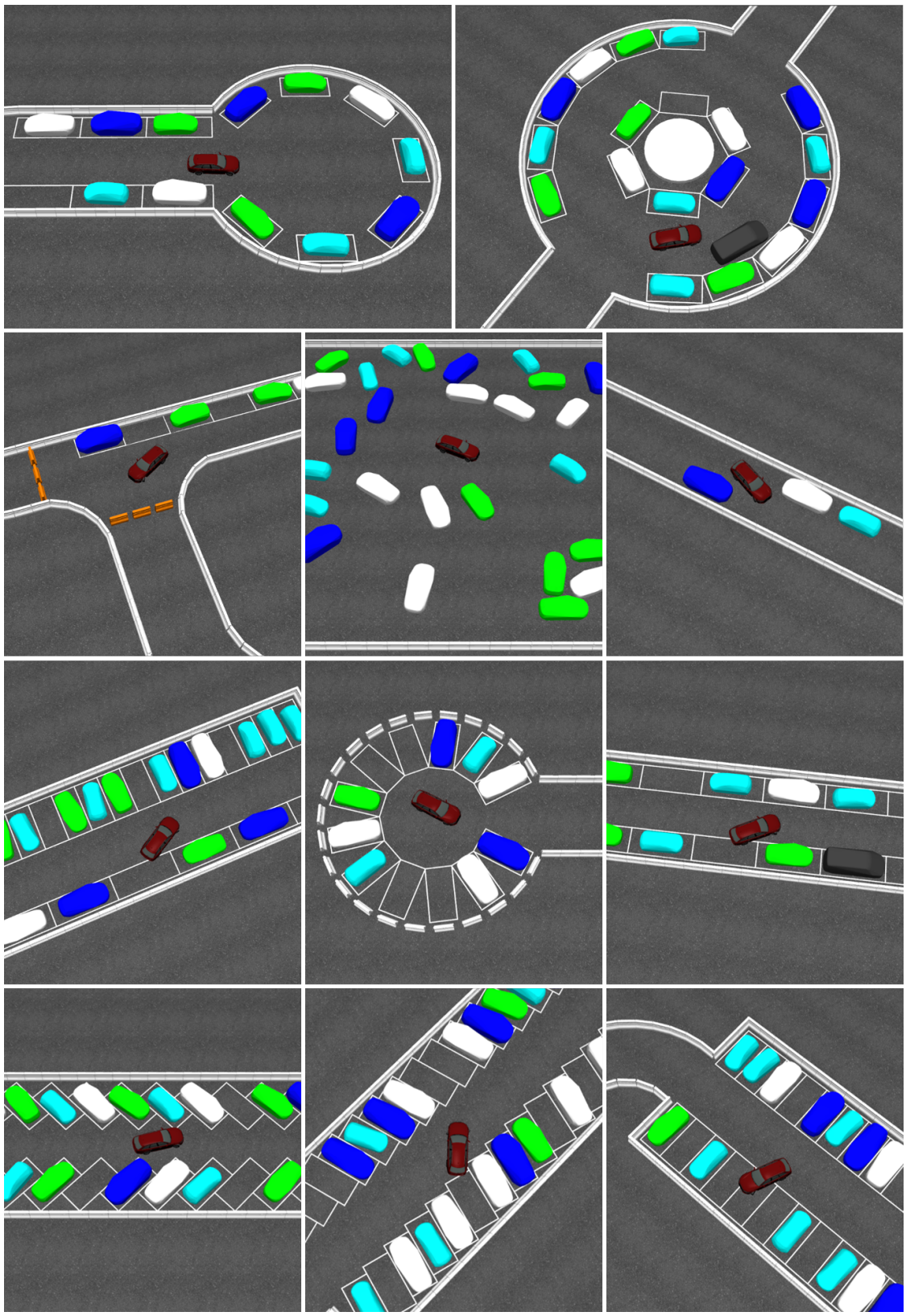


Figure 4.2 "Visualization of the eleven scenarios used in the generation of the dataset. Each scenario highlights a challenging driving task in the context of low-speed maneuvering in cluttered environments" [215]. First row: dead end and roundabout. Second row: blocked T-intersection, arena, and blocked road. Third and forth row: different parking scenarios. Adapted from [215].

phase to also resolve situations with partial observability that requires an intuition where possible solutions might lie' [215].

4.3 Learning Ego-Vehicle Pose Predictions

This section presents the model that predicts future ego-vehicle poses from a start to a goal state given observations of the current environment. Sec. 4.3.1 specifies the architecture of the model as well as the representation of its in- and output, and Sec. 4.3.2 highlights the generation of N discrete vehicle poses from the CNN's output. The training process including the metrics used for the evaluation are detailed in Sec. 4.3.3, and a hyperparameter optimization is finally conducted in Sec. 4.3.4. Note that all four sections are extracted from [215].

4.3.1 Model

'The model is designed to generate a sampling distribution over future vehicle poses connecting a start and goal state given an occupancy grid with environmental observations. Previous publications [7, 26, 138] have shown that CNNs are well suited for processing high dimensional inputs and predicting multi-modal distributions over future ego-vehicle locations. For nonholonomic motion planning, however, it is essential to enrich such a spatial distribution with the information about the robot's heading angle at all predicted positions. Therefore, it is proposed to jointly learn a sampling distribution over future vehicle positions as well as a mapping from position to heading angle for a complete representation of a vehicle pose.

This task is realized with the encoder-decoder architecture shown in Fig. 4.3. The illustrated CNN, which contains about 2.5 million parameters, is based on the well-known SegNet [4] from semantic segmentation. The main contribution is not the architecture of the model itself, as it can be easily exchanged with any other state-of-the-art architecture, but the representation of the input and output layers.

The proposed network takes five grids with a resolution of 256 px $\times$ 256 px as an input. These grids encode the static obstacles, the unknown environment, the past path, and the start and goal state of the vehicle (see Fig. 4.4). A simple and effective encoding of the vehicle state is achieved by a 7 px $\times$ 7 px square that describes three information: (1) its location in the grid marks the

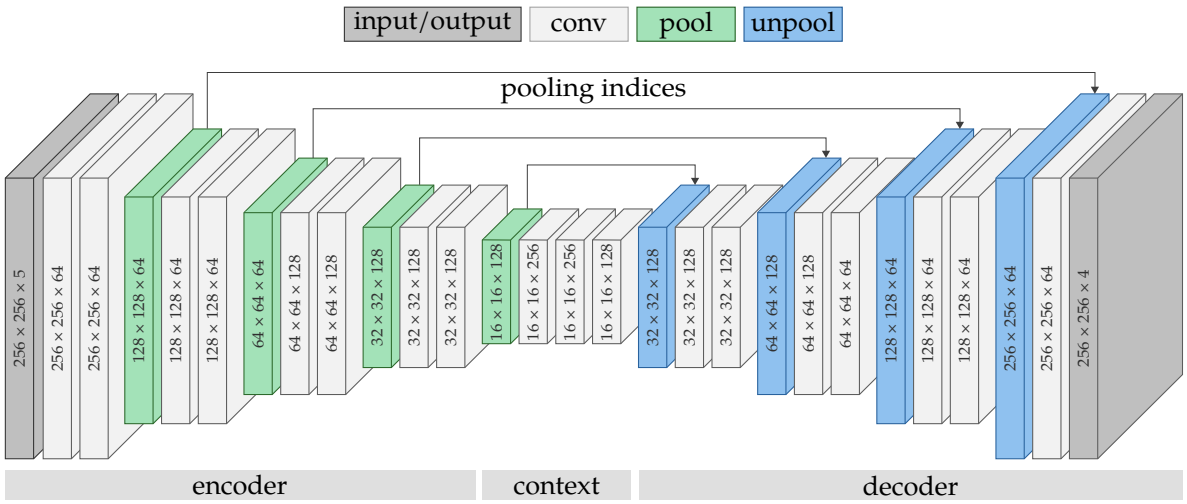


Figure 4.3 "Architecture of the CNN, which predicts a two-dimensional sampling distribution over future ego-vehicle positions [...] and the corresponding mapping from position to heading angle" [216]. Adapted from [216], © 2019 IEEE.

position of the vehicle, (2) its inner pixels encode the corresponding vehicle velocity, and (3) its outer pixels depict the respective heading angle. Note that the curvature of the start state is implicitly encoded in the past path as it can be seen in [Fig. 4.4(c)].

The input of the CNN is first encoded and then decoded back to its original size using alternating blocks of convolutional layers with 3×3 kernels and a stride of 1, 2×2 max pooling layers, and 2×2 unpooling layers. All convolutional layers except the last one are followed by a batch normalization [80] and a ReLU activation [130].

The CNN outputs four grids of the same resolution as the input grids. The first two grids give the results of a cell-wise classification that is trained to predict whether a given cell belongs to the future path or not. An example of such a prediction is shown in [Fig. 4.5(a)], where the intensity corresponds to the probability $p_{\text{path}}(c)$ of a cell c being part of the future path. In contrast to that, the last two output grids contain the results of a cell-wise regression for the sine and cosine components of the robot's heading angle. The decomposition into sine and cosine has three advantages: (1) a cell with zeros in both components represents an invalid orientation and can therefore be used to label cells without angle information (see Fig. 4.4), (2) computing the cell-wise norm of the predicted components can be interpreted as a confidence measure $p_\theta(c)$ indicating if an angle information is available at a respective cell c (see [Fig. 4.5(b)]), and (3) the prediction of the heading angle can be treated as a regression task rather than a classification task, which yields a

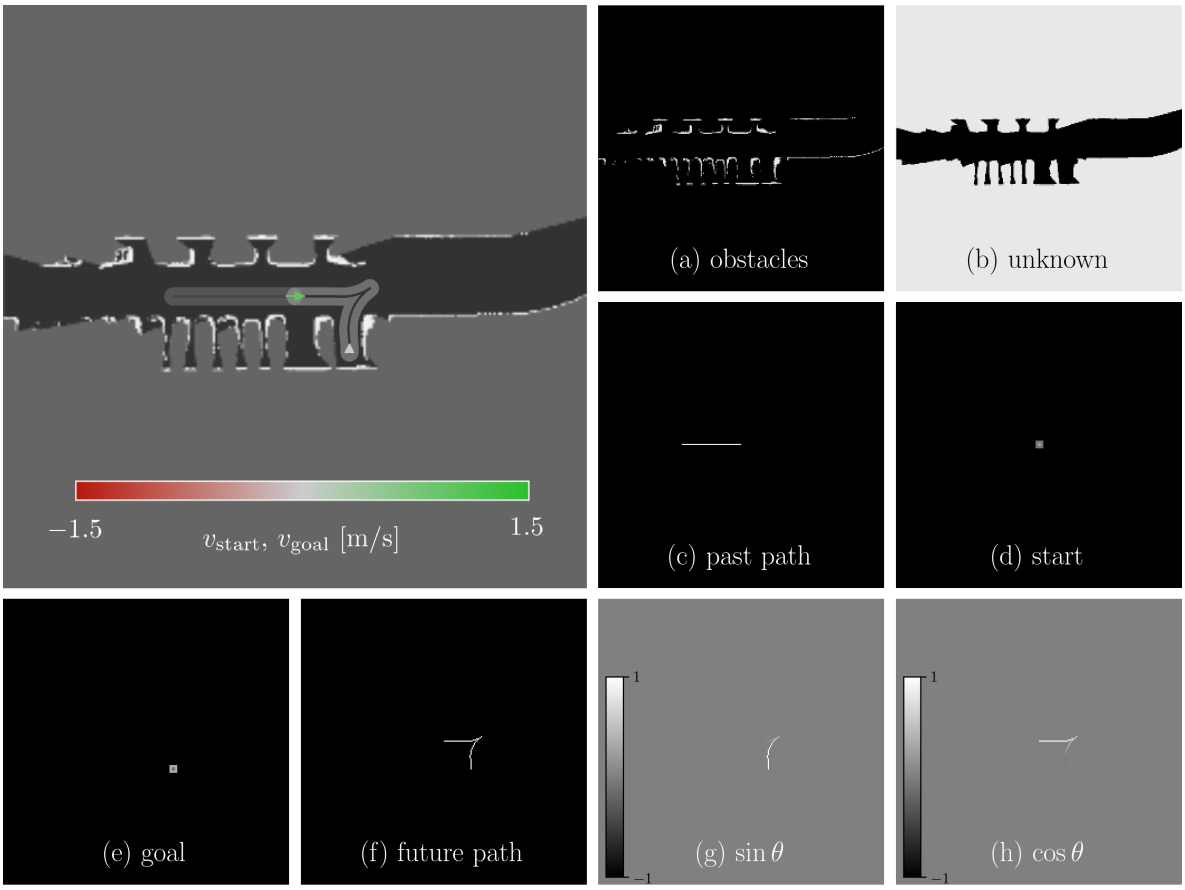
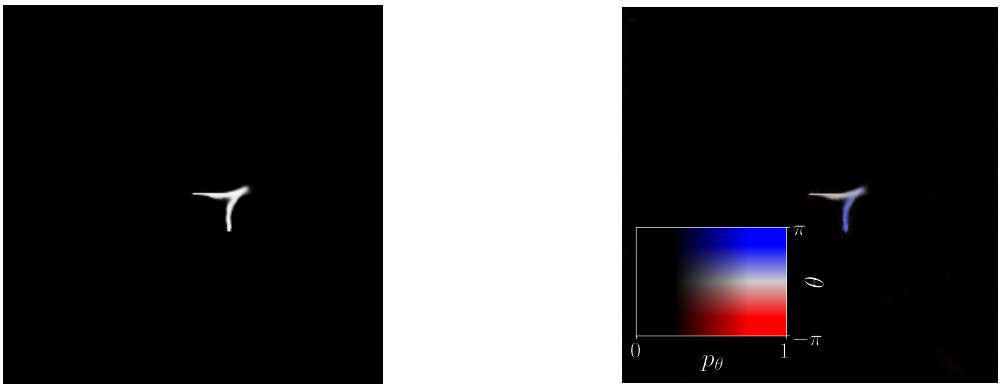


Figure 4.4 "Visualization of a perpendicular parking task on the [top] left, and the derived input grids (a)–(e) and label grids (f)–(h) for training the CNN. The ego-vehicle is represented by the green arrow in the center of the image on the [top] left and the target vehicle pose by the white arrow without a tail. The color of the arrows indicates the vehicle's velocity at the respective pose" [215].



(a) Path prediction. **(b)** Heading angle prediction.

Figure 4.5 Output of the CNN given the scenario from Fig. 4.4. While (a) gives a distribution over future vehicle positions, (b) provides a mapping from position to heading angle. Note that for a compact illustration, the four output grids (see Fig. 4.3) are blended into the two images shown above.

compact representation of the output and avoids an exponential growth of its dimension. However, the latter comes with a potential drawback of not being able to predict multi-modal, cell-wise heading angle predictions. The experiments have shown, however, that this is only an issue in rare corner cases as most of the scenarios do not require the vehicle to change its heading angle significantly while staying in the same region of the environment' [215].

4.3.2 Vehicle Pose Sampling

'Generating $i = 1, \ldots, N$ continuous vehicle poses $\mathbf{x}_{\text{pred}}^{[i]}$ from the output of the CNN is conducted in four steps. First, N random cells $c_{\text{pred}}^{[i]}$ are sampled from the CNN's path prediction $p_{\text{path}}(c)$, which can be seen as a probability mass function [(PMF)] that describes the relative probability of a cell c being part of the future path (see [Fig. 4.5(a)]). Sampling from the [PMF] can be realized efficiently using the low-variance sampling algorithm in [186]. Its linear time complexity yields a fast computation even for large values of N. Next, a continuous position prediction $[(x_{\text{pred}}^{[i]} \quad y_{\text{pred}}^{[i]})]$ is obtained by sampling uniformly within the previously computed discrete cell $c_{\text{pred}}^{[i]}$. The heading angle prediction $\theta_{\text{pred}}^{[i]}$ is now evaluated at the cell $c_{\text{pred}}^{[i]}$. If required, it can be interpolated with respect to the continuous position $[(x_{\text{pred}}^{[i]} \quad y_{\text{pred}}^{[i]})]$, which is omitted here for the sake of simplicity. In a final step, the sampled position and the corresponding heading angle are concatenated yielding a sampled vehicle pose $\mathbf{x}_{\text{pred}}^{[i]}$.

Note that throughout this [thesis], samples are exclusively drawn from cells with $p_{\text{path}}(c) > 0.5$ in order to only guide the motion planner towards regions with a high probability to be part of the future path. In contrast to that, visualizations of $p_{\text{path}}(c)$ are never thresholded and show the unmodified output of the CNN' [215].

4.3.3 Training and Metrics

'The proposed model is trained end-to-end using 64 % of the recorded trajectories. The training dataset is further augmented by randomly selecting up to 100 different start states on a recorded trajectory resulting in 807 273 (partly correlated) data points. This augmentation strategy does not only

upscale the dataset, but also forces the CNN to adjust its prediction as more information on the vehicle's environment becomes available.

The parameters of the network are optimized using Adam [99] and the loss function

$$L = \sum_{c \in \mathcal{C}} \left(f_{\text{CE}}(c) L_{\text{CE}}(c) + f_{\text{MSE}}(c) L_{\text{MSE}}(c) \right) + \sum_{w \in \mathcal{W}} \frac{\lambda w^2}{2}, \tag{4.1}$$

where the first term computes the cell-wise cross-entropy loss $L_{\text{CE}}(c)$ of the classification task, the second term the cell-wise mean squared error $L_{\text{MSE}}(c)$ of the regression task, and the last term the L2 regularization loss of the weights $w \in \mathcal{W}$ with a scaling factor λ. As the majority of cells in the label grids contain no path information (see Fig. 4.4), a weighting of the relevant cells is conducted with the functions $f_{\text{CE}}(c)$ and $f_{\text{MSE}}(c)$ given as

$$f_{\text{CE}}(c) = 1 + \mathbb{1}_c \cdot (\gamma_{\text{CE}} - 1), \tag{4.2}$$

$$f_{\text{MSE}}(c) = 1 + \mathbb{1}_c \cdot (\gamma_{\text{MSE}} - 1), \tag{4.3}$$

where γ_{CE} and γ_{MSE} are hyperparameters of the model, and $\mathbb{1}_c$ is the indicator function that is 1 if the future path crosses a cell c and 0 otherwise.

In order to evaluate the prediction capability of the model, two metrics are introduced below. The first metric measures the proximity of the N predicted samples [$\mathcal{X}_{\text{pred}} = \langle \mathbf{x}_{\text{pred}}^{[1]}, \ldots, \mathbf{x}_{\text{pred}}^{[N]} \rangle$] to the ground truth trajectory $\mathbf{x}_t \in \mathcal{X}$ by computing the average path deviation D according to

$$D(\mathcal{X}, \mathcal{X}_{\text{pred}}) = \frac{\sum_{i=1}^{N} \min_{\mathbf{x}_t \in \mathcal{X}} d(\mathbf{x}_t, \mathbf{x}_{\text{pred}}^{[i]})}{N}, \tag{4.4}$$

where $d(\bullet)$ describes the distance between a pose on the trajectory and a sample. It is computed by a weighted sum of the Euclidean distance and the angular deviation given as

$$d(\mathbf{x}_t, \mathbf{x}_{\text{pred}}^{[i]}) = w_{\text{pos}} \left\| \mathbf{x}_{\text{pred}}^{[i]} - \mathbf{x}_t \right\|_2 + w_\theta \left| \theta_{\text{pred}}^{[i]} - \theta_t \right|, \tag{4.5}$$

where the different units of both terms are taken into account by setting w_{pos} to 0.35 and w_θ to 0.65 throughout this [chapter].

As the predicted poses are supposed to guide the motion planner through complex scenarios, it is beneficial to have evenly distributed samples along the ground truth trajectory. Therefore, the second metric measures the maximum prediction gap G in the following two steps: (1) project every sample

onto the trajectory, and (2) evaluate the maximum arc length that is not covered by any sample. The projection in step 1 is defined as

$$\mathbf{x}_{\text{ref}}^{[i]} = \underset{\mathbf{x}_t \in \mathcal{X}}{\arg\min}\, d(\mathbf{x}_t, \mathbf{x}_{\text{pred}}^{[i]}), \tag{4.6}$$

where $\mathbf{x}_{\text{ref}}^{[i]}$ is the reference pose of the predicted sample $\mathbf{x}_{\text{pred}}^{[i]}$ on the trajectory. These reference poses are then added to the ordered list $\mathcal{X}_{\text{ref}}$ according to their arc length $s_{\text{ref}}^{[i]}$. The maximum prediction gap $G \in [0, 1]$ is finally obtained by

$$G(\mathcal{X}, \mathcal{X}_{\text{ref}}) = \frac{\max_{i=1:N-1} s_{\text{ref}}^{[i+1]} - s_{\text{ref}}^{[i]}}{s_T}, \tag{4.7}$$

where the length of the recorded trajectory is denoted by s_T' [215].

4.3.4 Hyperparameter Optimization

'This subsection evaluates the influence of the hyperparameters on the CNN's prediction performance based on the previously derived metrics. The analysis is conducted on a validation dataset containing 16 % of the recorded trajectories. Similar to the training dataset, up to five different start states are selected on each trajectory resulting in 10 730 (partly correlated) data points.

The different parametrizations of the model are trained on an Nvidia Titan X with a batch size of 20 using the Python interface of TensorFlow. Training a model to convergence takes about 30 h, which corresponds to 90 000 iterations. A visualization of the training statistics for one of the conducted optimizations can be found in Fig. 4.6.

In order to decrease the exponentially growing search effort, only the hyperparameters with the highest expected influence on the CNN's overall performance are optimized. Therefore, two out of the five hyperparameters, namely the exponential learning rate decay lrd and the L2 regularization scaling factor λ, are fixed to $lrd = 0.01$ after 10^6 iterations and $\lambda = 0.003$. The remaining hyperparameters are optimized in two steps. First, the learning rate lr is varied while keeping γ_{CE} and γ_{MSE} at 25, and second, lr is set to its best value while γ_{CE} and γ_{MSE} are optimized.

The quantitative results of the hyperparameter optimization are listed in Tab. 4.1. It can be seen that lower learning rates result in a better prediction performance as they decrease the maximum prediction gap metric G as well as the average path deviation metric D. The insights from [7] that higher

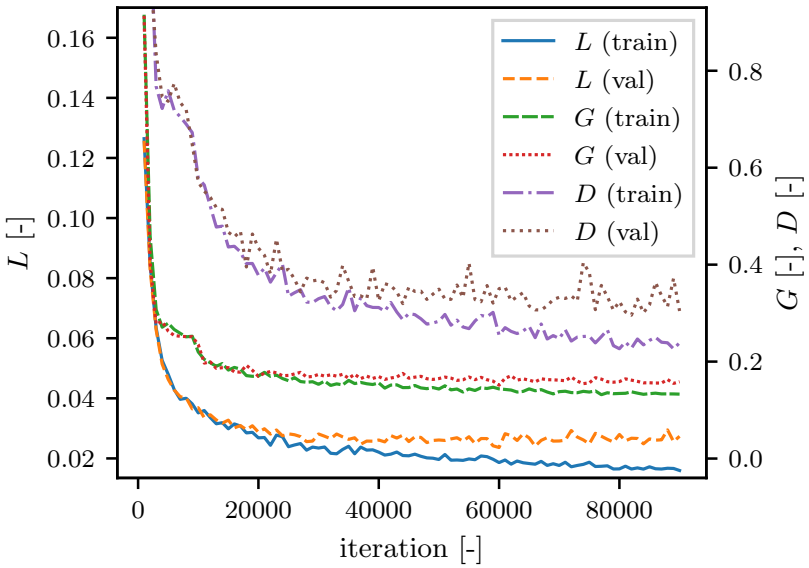
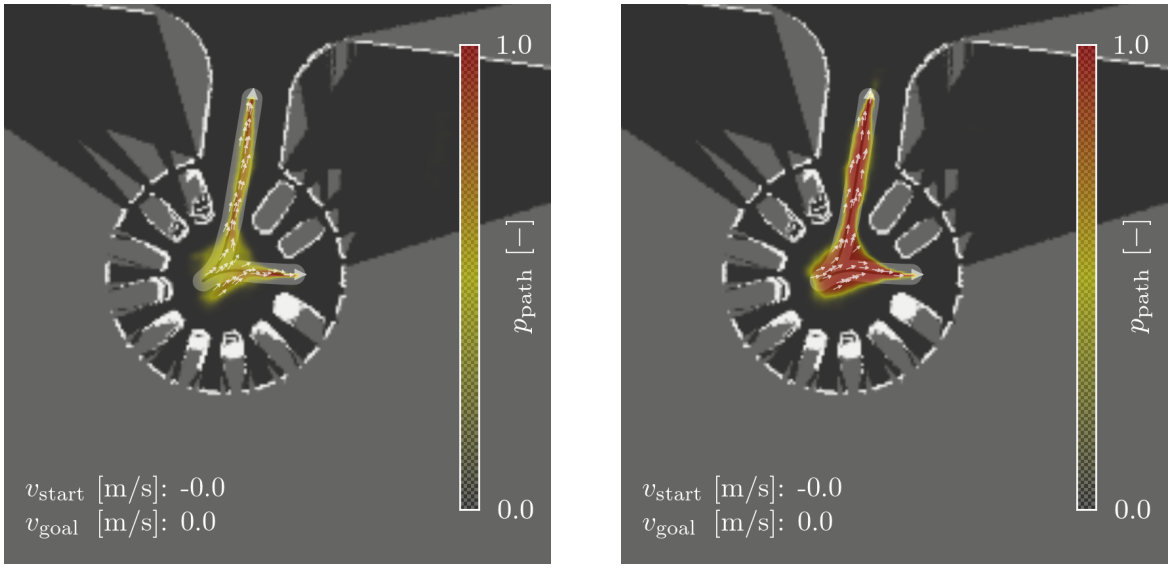


Figure 4.6 "Visualization of the training statistics with the hyperparameters set to $\gamma_{\text{CE}} = \gamma_{\text{MSE}} = 25$ and $lr = 10^{-5}$ (learning rate). Note that for better comparability, the loss L is visualized without the L2 regularization term as it is only computed at training and not at inference time. The metrics G and D are evaluated on 50 sampled vehicle poses" [215]. Reprinted from [215].

Table 4.1 "Quantitative comparison of the hyperparameter optimization on the basis of 200 sampled vehicle poses carried out on the validation dataset. The displayed values are given with mean and standard deviation" [215]. Reprinted from [215].

lr	γ_{CE}	γ_{MSE}	G [%]	D [−]
10^{-3}			20.7 ± 18.9	0.64 ± 0.26
10^{-4}	25	25	16.9 ± 18.6	0.35 ± 0.33
10^{-5}			10.1 ± 12.0	0.33 ± 0.35
10^{-5}	10	10	12.6 ± 14.9	0.26 ± 0.26
	10	25	12.7 ± 15.3	0.26 ± 0.28
	10	100	12.5 ± 14.9	0.25 ± 0.25
	25	10	11.2 ± 13.4	0.31 ± 0.35
	25	**25**	$\mathbf{10.1 \pm 12.0}$	$\mathbf{0.33 \pm 0.35}$
	25	100	10.8 ± 12.9	0.33 ± 0.33
	100	10	9.8 ± 10.6	0.45 ± 0.40
	100	25	10.3 ± 11.6	0.46 ± 0.45
	100	100	10.3 ± 11.5	0.42 ± 0.35



(a) $lr = 10^{-5}$, $\gamma_{\text{CE}} = 10$, $\gamma_{\text{MSE}} = 100$. **(b)** $lr = 10^{-5}$, $\gamma_{\text{CE}} = 100$, $\gamma_{\text{MSE}} = 10$.

Figure 4.7 "Qualitative comparison of the different hyperparameters on a test set trajectory with 50 sampled vehicle poses" [215]. The heatmap visualizes the relative probability of a cell belonging to the future path, and the gray arrows indicate the heading angle at the respective positions. A supplementary visualization for $lr = 10^{-5}$, $\gamma_{\text{CE}} = 25$, $\gamma_{\text{MSE}} = 25$ can be found in [215]. Adapted from [215].

values for γ_{CE} incentivize the CNN to classify more cells as part of the future path can also be observed in Tab. 4.1 and Fig. 4.7. Thus, increasing γ_{CE} reduces the maximum prediction gap G while raising the average path deviation D because more samples are placed further away from the ground truth. A potential risk of too large values for γ_{CE} is a deterioration of the heading angle prediction, which cannot be recovered by increasing γ_{MSE} (see Tab. 4.1). For the remaining evaluations, the model with $\gamma_{\text{CE}} = 25$, $\gamma_{\text{MSE}} = 25$, and $lr = 10^{-5}$ is selected as this combination yields a smooth distribution of the samples in close vicinity of the ground truth' [215].

4.4 Experimental Evaluation

This section is extracted from [215]: 'The following paragraphs analyze the prediction capability of the CNN and benchmark its performance against uniform sampling and the A*-based approach called [o]rientation-[a]ware [s]pace [e]xploration (OSE) [30]. To the best of the authors' knowledge, the latter is currently one of the most effective approaches for guiding a motion planner through complex environments. It is based on an A* graph search

and explores the workspace with oriented circles as illustrated in Fig. 4.8.

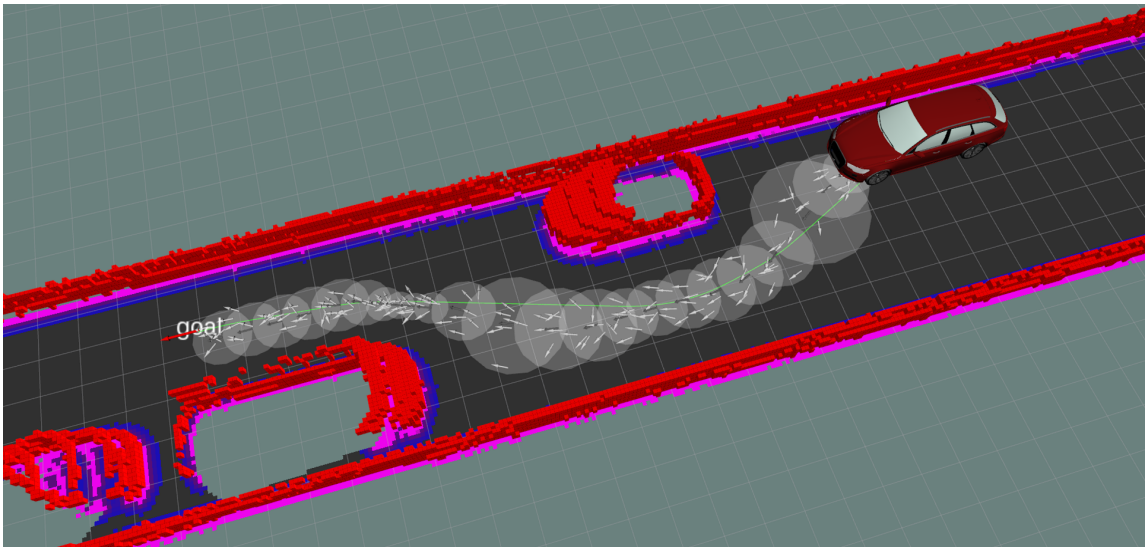


Figure 4.8 "Orientation-[a]ware [s]pace [e]xploration visualized by the white circles on the ground including 200 sampled vehicle poses (gray arrows)" [215]. Reprinted from [215].

Discrete vehicle poses can then be obtained by sampling from three-dimensional Gaussian distributions located at the center of the circles. The standard deviation of the Gaussians can be made dependent of the radius r and is set to $\sigma_x = \sigma_y = r/3$ for the translational and $\sigma_\theta = \pi/6$ for the rotational component, respectively. Additional parameters of the OSE heuristic used in this [thesis] are listed in Tab. 4.2.

Table 4.2 Parameters used for the OSE heuristic. Reprinted from [215].

parameter	value
minimum radius	0.2 m
maximum radius	5.0 m
minimum clearance	1.041 m
neighbors	32
maximum curvature	0.1982 m^{-1}
computation timeout	1.0 s

A benchmark of the three approaches on the test dataset can be found in Tab. 4.3. The 13 420 test cases with up to five different start states on each trajectory represent the remaining 20 % of the recordings. For a scenario-

Table 4.3 "Benchmark of uniform sampling, the OSE approach, and the CNN prediction on the test dataset. The metrics G and D as well as the computation time are evaluated on 200 sampled vehicle poses and are given with mean and standard deviation. Notice that the superscript in the notation of the CNN below indicates which input features have been used in the corresponding test" [216]. Adapted from [216], © 2019 IEEE.

	scenarios	G [%]	D [−]	time [ms]
uniform	all	13.5 ± 6.4	7.84 ± 0.44	0.1 ± 0.0
OSE	all	15.2 ± 17.4	0.57 ± 0.30	39.7 ± 43.1
CNN[1]	**all**	$\mathbf{10.4 \pm 12.5}$	$\mathbf{0.34 \pm 0.39}$	$\mathbf{41.7 \pm 14.0}$
CNN[2]	all	11.8 ± 14.1	0.35 ± 0.40	39.7 ± 14.5
CNN[3]	all	11.9 ± 14.3	0.34 ± 0.37	37.3 ± 13.7
CNN[4]	all	17.8 ± 19.6	0.44 ± 0.35	37.4 ± 14.0
CNN[5]	all	11.4 ± 14.0	0.32 ± 0.33	35.7 ± 14.5
CNN[6]	all	12.0 ± 14.8	0.32 ± 0.33	38.8 ± 14.1
CNN[7]	all	32.6 ± 22.4	0.82 ± 0.55	38.4 ± 13.4
CNN[1]	arena	14.4 ± 16.2	0.50 ± 0.47	42.5 ± 25.2
CNN[1]	parking	9.1 ± 9.6	0.23 ± 0.15	40.5 ± 25.1

[1] all input grids
[2] all but the obstacle grid
[3] all but the unknown grid
[4] all but the obstacle and unknown grid
[5] all but the past path grid
[6] all but the start grid
[7] all but the goal grid

specific evaluation, 5770 trajectories are extracted from this dataset, half of which come from the scenario arena and the other half from parallel and perpendicular parking. Both uniform sampling as well as the OSE procedure are evaluated on a single core of an Intel Xeon E5@3.5 GHz with the latter being implemented in C++. As opposed to that, the CNN is executed with its Python pipeline on an Nvidia Titan X and an Intel Xeon E5@3.1 GHz.

In comparison to uniform sampling and the OSE approach, Tab. 4.3 shows that the CNN[1] predicts the vehicle poses more evenly along the ground truth path and yields overall smaller deviation from it. The mean computation time of the CNN is comparable to the OSE heuristic, however, with a one-third smaller standard deviation. The OSE's high variance in computation time is due to the fact that the performance of graph search-based algorithms highly depends on the complexity of the environment. This also causes the problem

that a solution might not be found before the timeout is reached, which occurred here in 3.4 % of all test cases. In summary, the CNN, whose output is qualitatively visualized in Fig. 4.9, makes its predictions more reliably (no outages) with a lower latency. Both aspects are key features in safety-critical applications such as automated driving.

In order to better understand the effect of the different input grids on the performance of the CNN, an ablation study has been conducted. The results in Tab. 4.3 show that removing features from the CNN's input causes a deterioration of at least one of the analyzed metrics. Furthermore, it can be seen that excluding the goal or the observations causes the greatest decline in performance. In contrast to that, the metrics only change slightly if the CNN is trained and evaluated without the obstacle or the unknown grid (not both). One of the reasons for this is that in many cases, both grids are complementary in the sense that the unknown grid allows to infer the obstacles and vice versa (see Fig. 4.4).

The scenario-specific benchmark in Tab. 4.3 highlights that the CNN's performance in the parking scenario is almost a factor two better compared to the maze-like structure arena. The resulting insight is that learning stationary sampling distributions in completely unstructured environments is a much harder task for the network than in semi-structured environments. This can also be seen in Fig. 4.9 [in] the [middle] right, where the CNN prediction degenerates due to the complexity of the scenario. Possible reasons for this are longer prediction horizons, a larger variety of feasible maneuvers, and potentially heavier occlusions. It is left for future work to determine which network structure is the most suitable one for such challenging environments.

A visualization of the CNN prediction in two novel scenarios, namely a construction zone and and a cluttered 4-way intersection, can be found in the [two] lower [...] images of Fig. 4.9. Both scenarios were not seen during training and contain novel artifacts such as construction barrels and rectangular dumpsters. While the construction zone only requires the vehicle to follow the lane, the maneuver at the cluttered 4-way intersection consists of three steps: (1) exit the tight parking spot, (2) avoid the dumpster at the side of the road, and (3) make a turn at the intersection. The illustrated predictions showcase the models capability to deal with so far unseen artifacts as well as to generalize to novel scenarios. Only in the second case, the initial path prediction overlaps with the barriers in front of the vehicle. However, this is not a safety issue if the predictions are combined with a motion planner as highlighted in [Sec. 5.3.5]' [215].

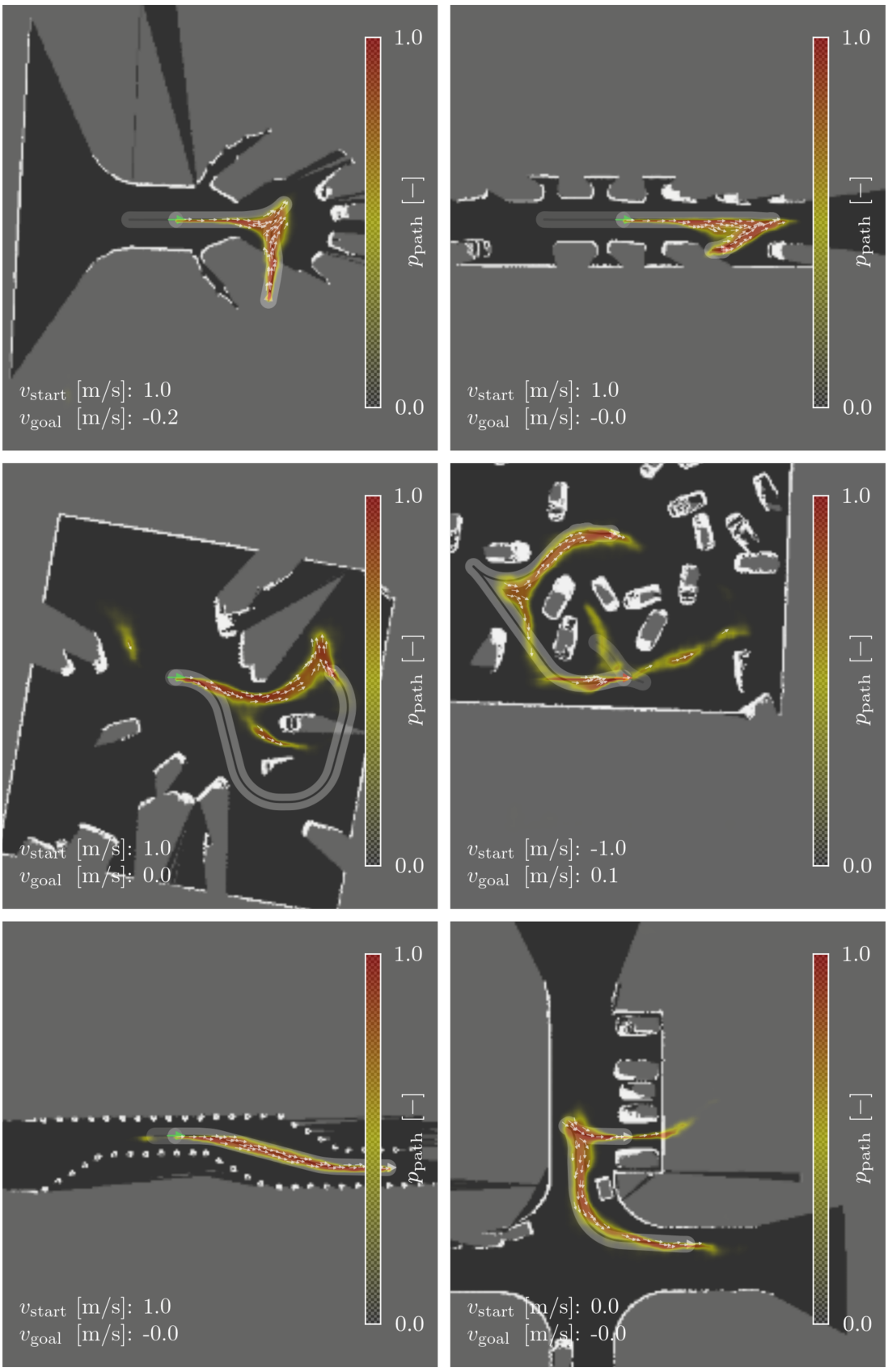


Figure 4.9 "Illustration of the CNN prediction including 50 sampled vehicle poses in different scenarios" [215]. The first two rows highlight the performance on four trajectories from the test dataset while the third row visualizes the prediction in two previously unseen scenarios called construction zone and 4-way intersection. A degeneration in performance can be seen in the middle right. Adapted from [215].

4.5 Summary

Solving the motion planning problem in real-time typically requires heuristics that accelerate the planning process and improve the convergence rate. As the large variety of scenarios in automated driving as well as the nonholonomic constraints of the vehicle make it challenging to design such heuristics manually, a data-driven approach is presented in this chapter. More precisely, a CNN is introduced in Sec. 4.3 that predicts a distribution over future vehicle poses given a start and goal state as well as observations of the environment. Based on the algorithm described in Sec. 4.3.2, discrete vehicle poses can then be computed from this output in order to guide, e.g., a sampling-based motion planner through complex environments. The latter is analyzed from a planning perspective in the next chapter.

Training the neural network requires a diverse dataset that was recorded in eleven distinct scenarios using an automated vehicle simulator. As detailed in Sec. 4.2, variation was introduced into the data generation process by constantly changing the ego-vehicle's start and goal pose as well as the number and the position of the obstacles in the environment. The resulting procedure was fully automated and required no time-consuming manual labeling to obtain the training data.

The benchmark in Sec. 4.4 finally shows that the CNN predictions are not only closer to the ground truth, but also more evenly distributed along it compared to the ones generated with uniform sampling and an A*-based approach. In addition to that, the CNN makes its predictions with significantly less variation in computation time than the A*-based approach making it particularly suitable for real-time applications such as automated driving. Furthermore, the ablation study shows that excluding single features from the CNN's input deteriorates its overall prediction performance and thus highlights the importance of the selected inputs. Lastly, it is demonstrated in two previously unseen scenarios that the CNN is able to adapt to completely new situations with novel artifacts.

Among the various future directions in this field, the following two research questions appear as a natural extension of the presented approach: (1) How can even higher dimensional distributions (e.g. position, orientation, and curvature) be predicted given the same input features as above? And (2) How can the temporal dimension as well as dynamic obstacles be considered in such an approach? Especially the second question needs to be addressed in order to overcome the current limitation to a static environment.

5 Sampling-Based Motion Planning in Dense Scenarios

This chapter combines the previously introduced methods to compute feasible collision-free motion plans in dense scenarios. To do so, the derived algorithms are integrated into the sampling-based motion planner RRT* [93] and its bidirectional extension BiRRT* [86]. While the presented steering functions and the concept of beliefprints could also be used in search-based algorithms, such as Hybrid A* [40], the focus here lies on sampling-based motion planning. This is mainly due to the fact that search-based approaches require expert knowledge to come up with a proper discretization of the state-action space. Otherwise, and especially in tight situations, the planner might fail although a feasible solution exists. In contrast to that, sampling-based approaches do not rely on such a discretization as they compute a solution by continuously adding samples from a given distribution to the problem.

The remainder of this chapter, which is based on [209, 211, 213, 214, 216], is organized as follows: Sec. 5.1 formulates the underlying motion planning problem, Sec. 5.2 describes the platforms and the setups of the experiments, and Sec. 5.3 highlights the experimental results. A summary is finally given in Sec. 5.4.

5.1 Problem Formulation

Solving a motion planning problem requires to connect a start and goal state while taking into account the surrounding obstacles, the vehicle's constraints, and the comfort requirements of the passengers. This can be formulated as a mathematical optimization problem, which is given in the absence of uncertainty as

$$\underset{\check{x},\check{u}}{\arg\min} \quad J(\check{\mathcal{X}}, \check{\mathcal{U}}) \tag{5.1a}$$

$$\text{s.t.} \quad \check{\mathbf{x}}_0 = \check{\mathbf{x}}_s, \tag{5.1b}$$

$$\check{\mathbf{x}}_N = \check{\mathbf{x}}_g, \tag{5.1c}$$
$$\check{\mathbf{x}}_{k+1} = \check{\mathbf{f}}(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k), \tag{5.1d}$$
$$\check{\mathbf{x}}_{\min} \leq \check{\mathbf{x}}_{k+1} \leq \check{\mathbf{x}}_{\max}, \tag{5.1e}$$
$$\check{\mathbf{u}}_{\min} \leq \check{\mathbf{u}}_k \leq \check{\mathbf{u}}_{\max}, \tag{5.1f}$$
$$\check{\mathbf{x}}_{k+1} \notin \mathcal{X}_{\text{obs}}, \quad k = 0 : N-1, \tag{5.1g}$$

where $\check{\mathbf{x}}_k \in \mathbb{R}^n$ denotes the nominal state of the system, $\check{\mathbf{u}}_k \in \mathbb{R}^m$ the nominal input, and $\check{\mathbf{f}}(\bullet)$ the respective motion model. The start state is described by $\check{\mathbf{x}}_s$, the goal state by $\check{\mathbf{x}}_g$, the state and input constraints of the vehicle by $\check{\mathbf{x}}_{\min}$, $\check{\mathbf{x}}_{\max}$, $\check{\mathbf{u}}_{\min}$, $\check{\mathbf{u}}_{\max}$, and the states in collision with the environment by $\mathcal{X}_{\text{obs}} \subset \mathbb{R}^n$. The user-defined objective function is represented by $J(\bullet)$ and transforms the $N+1$ nominal states $\check{\mathcal{X}} = \langle \check{\mathbf{x}}_0, \dots, \check{\mathbf{x}}_N \rangle$ and the corresponding inputs $\check{\mathcal{U}} = \langle \check{\mathbf{u}}_0, \dots, \check{\mathbf{u}}_{N-1} \rangle$ into a scalar cost.

Informally speaking, the general goal in (5.1) is to find a sequence of $N+1$ nominal vehicle states $\check{\mathcal{X}}$ and the respective inputs $\check{\mathcal{U}}$ such that they minimize the objective function in (5.1a) without violating the given constraints. Note that in comparison to the steering problem in (2.1), collision avoidance must now be enforced as indicated by the additional constraint in (5.1g).

Computing a (sub)optimal solution to the given motion planning problem is conducted here using the well-known path-velocity decomposition [91]. The underlying idea is to first compute a feasible path from start to goal by only taking into account the static environment and then to derive a velocity profile such that collisions with dynamic obstacles are avoided. This chapter focuses on solving the first part of this decomposition in a sampling-based fashion, which typically requires at least three steps: (1) iteratively sample a new state, (2) steer the system from an already explored state to that new sample, and (3) probe the computed connection against collision with the environment.

One of the most prominent motion planners that implements these steps by incrementally building a tree from start to goal is RRT*. It guarantees asymptotic convergence to the globally optimal solution by locally rewiring the resulting tree in an additional forth step. Another important feature of RRT* is its probabilistic completeness, which means that the probability of solving a feasible motion planning problem goes to one in the limit of infinite samples. Further details on the algorithm can be found in [93], and an open-source implementation is available in [179]. In deterministic path planning, RRT* can be replaced by its two-tree version called BiRRT* [86, 100]. The latter is known to outperform RRT* in challenging environments

as it solves the problem from start to goal and vice versa at the same time. Therefore, BiRRT* is used in the experiments below to solve (5.1).

As previously mentioned, the general goal in motion planning is to minimize the cost function in (5.1a) that allows to optimize multiple objectives. This function is required to return a positive scalar cost for non-trivial collision-free paths [93] and is given here as

$$J(\check{\mathcal{X}}, \check{\mathcal{U}}) = \mathbf{w}_J^\intercal \cdot \mathbf{J}(\check{\mathcal{X}}, \check{\mathcal{U}}), \tag{5.2}$$

where $\mathbf{w}_J$ weights the user-defined cost terms stored in $\mathbf{J}$. For nominal motion planning, $\mathbf{J}$ consists here of four scalar terms that are evaluated according to

$$J_{\text{length}} = \sum_{k=0}^{N-1} |\Delta \check{s}_k|, \tag{5.3a}$$

$$J_{\text{cusp}} = \sum_{k=0}^{N-2} \mathbb{1}_{\Delta \check{s}_{k+1} \cdot \Delta \check{s}_k < 0}, \tag{5.3b}$$

$$J_{\text{curv}} = \sum_{k=0}^{N-1} \int_0^{|\Delta \check{s}_k|} |\check{\kappa}_{k+1}(s)| \ \mathrm{d}s, \tag{5.3c}$$

$$J_{\text{print}} = \sum_{k=0}^{N} \mathbf{footprint\_cost}(\mathcal{F}, \check{\mathbf{x}}_k), \tag{5.3d}$$

where both the nominal arc length $\Delta \check{s}_k$ as well as the nominal curvature $\check{\kappa}_k$ are provided by the steering function (see Ch. 2). The first term J_{length} computes the total length of the path, and the second term J_{cusp} evaluates the number of direction switches, where $\mathbb{1}_{(\bullet)}$ denotes the indicator function. The third term J_{curv} integrates the curvature along the path, and the fourth term J_{print} sums up the cost of the footprint $\mathcal{F}$ at the nominal states $\check{\mathbf{x}}_{k=0:N}$. The latter allows to increase the safety of a motion plan by penalizing paths with only little clearance to surrounding obstacles.

While (5.1)–(5.3) assume perfect state observation and precise execution of the planned path, taking into account both the localization and control uncertainty requires to adapt the presented problem formulation. Under the assumption of a Gaussian belief space, the goal now is to compute a sequence of $N+1$ nominal states $\check{\mathcal{X}} = \langle \check{\mathbf{x}}_0, \ldots, \check{\mathbf{x}}_N \rangle$ and the corresponding inputs $\check{\mathcal{U}} = \langle \check{\mathbf{u}}_0, \ldots, \check{\mathbf{u}}_{N-1} \rangle$ such that the expected cost is minimized. Furthermore, a chance constraint is required to ensure that the collision probability (CP)

remains below a user-defined threshold $1 - P$, where $P \in (0.5, 1]$. The resulting optimization problem can be formulated as

$$\begin{aligned}
\underset{\check{\mathcal{X}},\check{\mathcal{U}}}{\arg\min} \quad & J(\mathcal{B}, \check{\mathcal{U}}) && \text{(5.4a)}\\
\text{s.t.} \quad & \check{\mathbf{x}}_0 = \mu_s, && \text{(5.4b)}\\
& \check{\mathbf{x}}_N = \mu_g, && \text{(5.4c)}\\
& \check{\mathbf{x}}_{k+1} = \check{\mathbf{f}}(\check{\mathbf{x}}_k, \check{\mathbf{u}}_k), && \text{(5.4d)}\\
& bel(\mathbf{x}_0) = bel(\mathbf{x}_s), && \text{(5.4e)}\\
& bel(\mathbf{x}_{k+1}) = \mathbf{ekf_mp}(bel(\mathbf{x}_k), \check{\mathbf{u}}_k), && \text{(5.4f)}\\
& \check{\mathbf{x}}_{\min} \leq \check{\mathbf{x}}_{k+1} \leq \check{\mathbf{x}}_{\max}, && \text{(5.4g)}\\
& \check{\mathbf{u}}_{\min} \leq \check{\mathbf{u}}_k \leq \check{\mathbf{u}}_{\max}, && \text{(5.4h)}\\
& \Pr(\mathbf{x}_{k+1} \in \mathcal{X}_{\text{obs}}) < 1 - P, \quad k = 0 : N-1, && \text{(5.4i)}
\end{aligned}$$

where the initial belief is described by $bel(\mathbf{x}_s) = \mathcal{N}(\mathbf{x}_s; \mu_s, \mathbf{\Sigma}_s)$ with mean $\mu_s \in \mathbb{R}^n$ and covariance $\mathbf{\Sigma}_s \in \mathbb{R}^{n \times n}$, the desired goal by $\mu_g \in \mathbb{R}^n$, and the $N+1$ Gaussian beliefs obtained from the EKF-MP by $\mathcal{B} = \langle bel(\mathbf{x}_0), \ldots, bel(\mathbf{x}_N) \rangle$. The other variables are identical to the nominal optimization problem in (5.1) and therefore not further specified here.

As the overall goal in (5.4) is to minimize the expected cost, the objective function $J(\bullet)$ now depends on the beliefs $\mathcal{B}$ and the nominal inputs $\check{\mathcal{U}}$. In the experiments below, this function evaluates similar to (5.2) a weighted sum of five cost terms. While the first two terms are identical to (5.3a)–(5.3b), the other three terms are given as

$$J_{\text{curv}} = \sum_{k=0}^{N-1} \int_0^{|\Delta \check{s}_k|} |E[\kappa_{k+1}(s)]| \, \mathrm{d}s, \tag{5.5a}$$

$$J_{\text{cov}} = \sum_{k=0}^{N} \frac{1}{n} \operatorname{tr}(\mathbf{\Sigma}_k + \mathbf{\Lambda}_k), \tag{5.5b}$$

$$J_{\text{print}} = \sum_{k=0}^{N} \mathbf{beliefprint\_cost}(\mathcal{B}_k). \tag{5.5c}$$

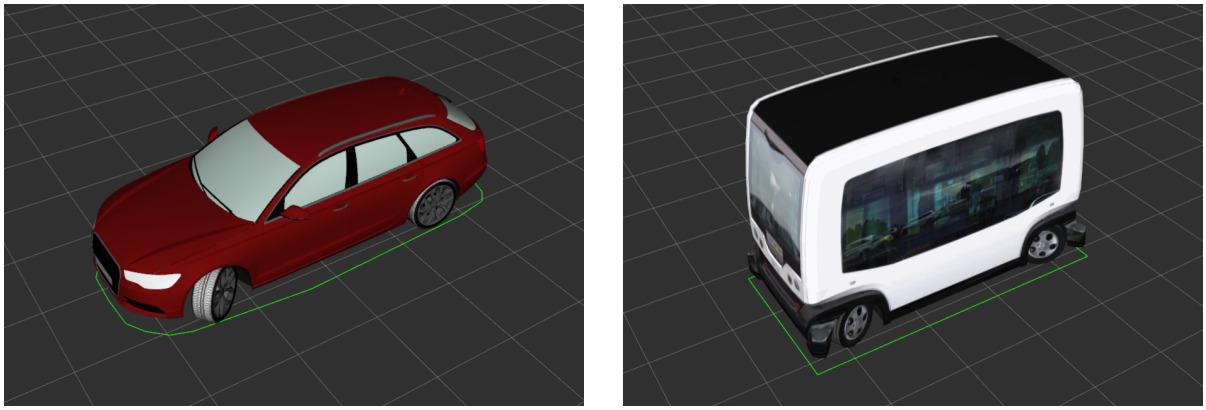
The third term J_{curv} integrates the expected curvature along the path, and the fourth term J_{cov} considers the uncertainty of the beliefs according to the A-optimality criterion [32]. Here, $\operatorname{tr}(\bullet)$ denotes the trace of the $n \times n$ covariance matrix $\mathbf{\Sigma}_k + \mathbf{\Lambda}_k$ that is obtained from the EKF-MP. The fifth term J_{print} sums

up the cost of the beliefprints $\mathcal{B}_k$ and allows to penalize paths with small distance to surrounding obstacles.

A solution to the belief space planning problem in (5.4)–(5.5) can still be computed with RRT*, however, asymptotic convergence to the optimal solution can no longer be guaranteed. This is because paths in belief space can, in general, only be partially ordered [23] while RRT* relies on a total ordering. To overcome this problem, two alternatives exist: one can switch to the rapidly-exploring random belief tree (RRBT) [23], which keeps multiple beliefs at the sampled nodes, or one can adjust the cost function such that it fulfills the optimal substructure property [17, 171]. Both approaches are beyond the scope of this chapter and therefore not further detailed here. Moreover, it has to be noted that planning in belief space cannot be tackled bidirectionally any longer. This is because the evolution of uncertainty depends on the previous belief that is initially only known at the start and not at the goal state.

5.2 Platforms and Setups

In order to show the flexibility of the developed approaches, the experiments are conducted on two different vehicle platforms that are shown in Fig. 5.1.



(a) Audi A6 Avant. **(b)** EasyMile EZ10.

Figure 5.1 Illustration of the platforms used in the experiments including the approximated footprint visualized by the green polygon on the ground.

The red Audi A6 Avant represents a conventional full-size car whose footprint is approximated by a convex polygon with 20 vertices. If not stated differently in the experiments below, the A6 is only allowed to turn its front wheels, which is denoted here as single Ackermann steering. In contrast to

that, the white EasyMile EZ10 represents a so-called people mover that is capable to transport up to 12 passengers to their destination. Its steering system allows to turn both the front and the rear wheels at the same time, which is in the following referred to as double Ackermann steering. Due to the boxy design of the EZ10, its footprint is approximated by a rectangular polygon with four vertices. Additional parameters of both platforms can be found in Tab. 5.1. Note that the given maximum curvature $\kappa_{\max}$ as well as the maximum curvature rate $\sigma_{\max}$ already include 10 % control reserve for closed-loop execution.

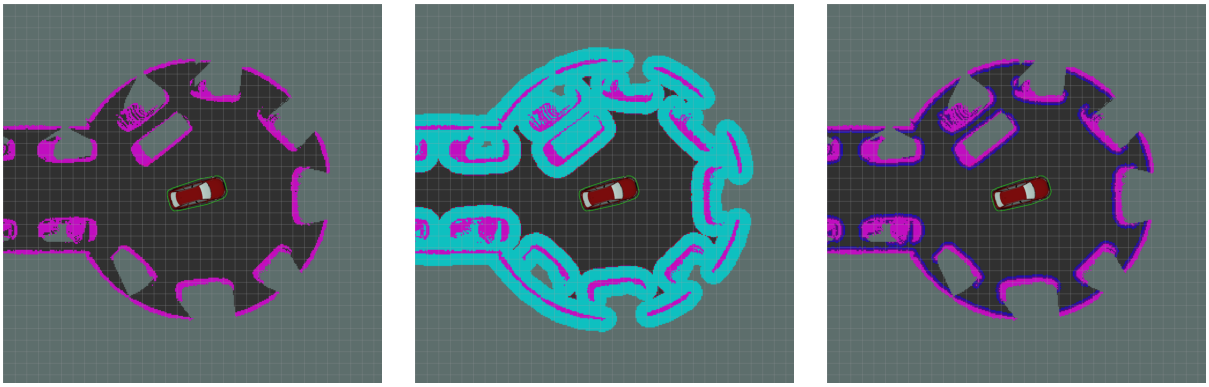
Table 5.1 Vehicle parameters of the two platforms.

parameter	symbol	A6	EZ10
length	-	4.926 m	3.946 m
width	-	2.086 m	1.983 m
wheel base	-	2.912 m	2.8 m
max. curvature	$\kappa_{\max}$	$0.1982\,\mathrm{m}^{-1}$	$0.2274\,\mathrm{m}^{-1}$
max. curvature rate	$\sigma_{\max}$	$0.1868\,\mathrm{m}^{-2}$	$0.1736\,\mathrm{m}^{-2}$
footprint	$\mathcal{F}$	20 vertices	4 vertices

Before executing the developed motion planner on a real-word vehicle, extensive testing has been conducted in simulation. For this purpose, a simulator was built up from scratch using Gazebo [58] and ROS [155]. It implements the different kinematics of both vehicles and allows to actuate them with a motion controller in closed loop. Furthermore, arbitrary three-dimensional environments can be created in simulation and then perceived by different sensors including, for example, a simulated LiDAR system. A visualization of the developed simulator, which runs in real-time on an Intel Xeon E5@3.5 GHz and an Nvidia Quadro K2200, can be found in Fig. 5.3.

The current implementation uses an occupancy grid to represent the static environment by fusing the measurements of the LiDAR perception. This can be seen in Fig. 5.2(a), which shows the perceived environment in a dead end scenario. The size and the resolution of the grid can be varied and are given in the respective experiments.

To speed-up the motion planning process, two cost maps are derived from the current occupancy grid. The first one is shown in Fig. 5.2(b) and inflates the obstacles by the inscribed radius of the footprint. The resulting cost map allows a simple collision check by looking up whether the planned



(a) Occupancy grid. **(b)** Cost map I. **(c)** Cost map II.

Figure 5.2 Visualization of the environment model used for motion planning. The occupancy grid (a) distinguishes between occupied (purple), unknown (greenish gray), and free space (transparent). The first cost map (b) inflates the obstacles by the inscribed radius of the footprint (cyan) and the second cost map (c) applies a user-defined inflation (here 25 cm) of the obstacles (bluish purple).

position of the robot is within the inflated area [47]. If this is not the case, the footprint/beliefprint is convolved with the second cost map shown in Fig. 5.2(c), which inflates the obstacles by a user-defined radius. In case no collision is detected, the integrated cost of that footprint/beliefprint is returned to evaluate (5.3d) and (5.5c), respectively. As a result, the second cost map implements a soft safety margin as the algorithm is incentivized to stay outside the inflated region. In addition to that, the footprint of the vehicle can be padded by a hard safety buffer in order to account for noisy measurements. This is realized in Fig. 5.2, where a footprint padding of 10 cm is applied. The entire collision checking process is repeated every 10 cm with the parameters listed in Tab. 5.2.

During the planning process, samples for (Bi)RRT* are uniformly generated in $SE(2)$ [115] if not stated differently in the experiments. The sampling region for possible vehicle positions is defined by the size of the occupancy grid, and vehicle orientations are drawn from the interval $[-\pi, \pi)$. As summarized in Tab. 5.2, a goal sampling frequency of 5 % is applied in order to slightly bias the planner towards the goal. The sampling duration for an initial solution is fixed to 10 s. If a solution can be found during that time, an additional 3 s are assigned for optimization of the initial plan. Otherwise, the run is classified as failed. The constant γ of (Bi)RRT* [93] is set to 6, and a total of 100 simulation runs are conducted per experiment to compensate for randomization effects.

Table 5.2 Auxiliary parameters used in the motion planning experiments, where the safety margins are adapted between simulation and real-world testing.

parameter	value
cost map II inflation	25 cm (sim.), 50 cm (real)
footprint padding	10 cm (sim.), 20 cm (real)
collision checking	every 10 cm
goal sampling frequency	5 %
sampling duration	max. 10 s for initial solution 3 s for optimization
(Bi)RRT* constant γ [93]	6
simulation runs	100

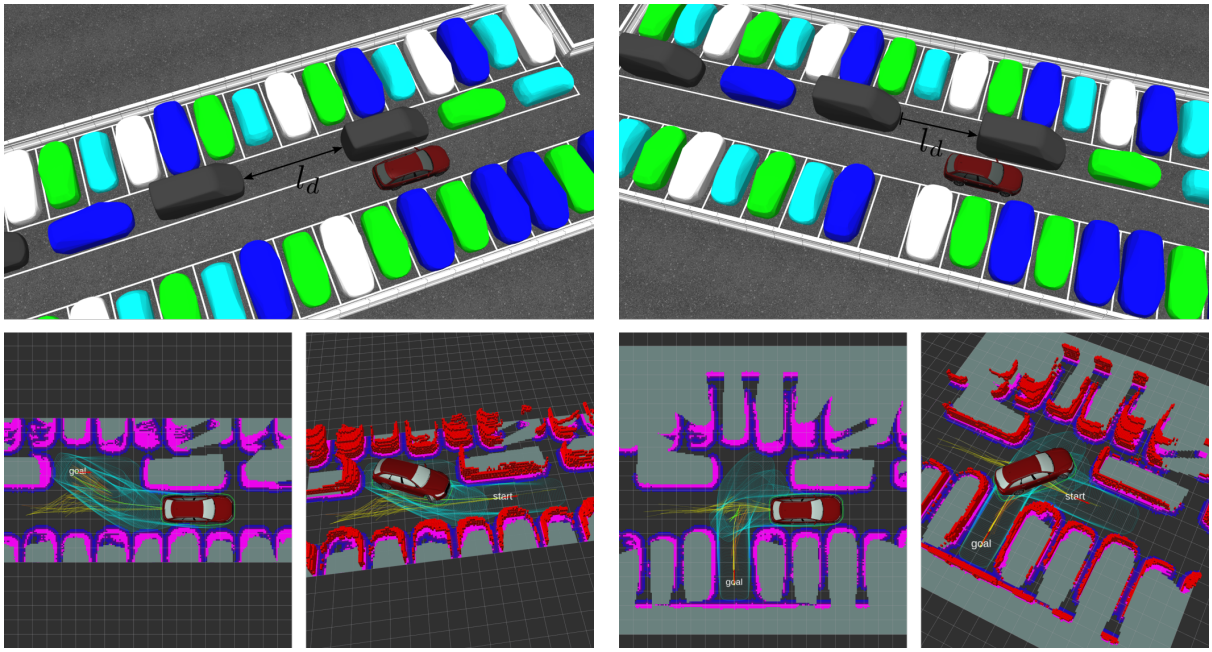
5.3 Experimental Results

A variety of challenging automated driving scenarios has been created to benchmark the algorithms presented in this thesis. In particular, Sections 5.3.1–5.3.3 analyze the performance of the different steering functions along with BiRRT* in various tight environments. The advantages and drawbacks of planning in Gaussian belief space are presented in Sec. 5.3.4, and the effects of guiding the motion planner with the previously described neural network are finally highlighted in Sec. 5.3.5.

5.3.1 Maneuvering in Tight Parking Space

The overall goal of the experiments in this section is to compare the performance of HC-RS steer against RS and CC-RS steer in two extremely tight environments. To do so, two high density parking scenarios are created that not only allow vehicles to park perpendicular to the driveway, but also on one side of it [210]. This can be seen in Fig. 5.3, where the red ego-vehicle has to execute a parallel parking maneuver in Scenario I and a perpendicular one in Scenario II.

The environment is represented by an occupancy grid with a resolution of 10 cm and a size of 20 m $\times$ 10 m in Scenario I and 20 m $\times$ 20 m in Scenario II. In both scenarios, the free space can be controlled by modifying the variable l_d, which describes the distance between the two black transporters as shown in the top row of Fig. 5.3. This parameter is incrementally decreased in the



(a) Scenario I. **(b)** Scenario II.

Figure 5.3 Two high density parking scenarios [210] used to benchmark the G^1 and G^2 continuous steering functions along with BiRRT*. The top row illustrates both scenarios, which can be modified by varying the distance l_d between the two black transporters. The bottom row shows two example paths computed with HC$^{\pm\pm}$-RS steer, the two trees generated by BiRRT* (yellow and orange lines on the ground), and the execution of both paths. Additionally, the perceived environment of the simulated LiDAR system is displayed by the red voxels in the two images on the bottom right with the corresponding cost maps drawn on the ground.

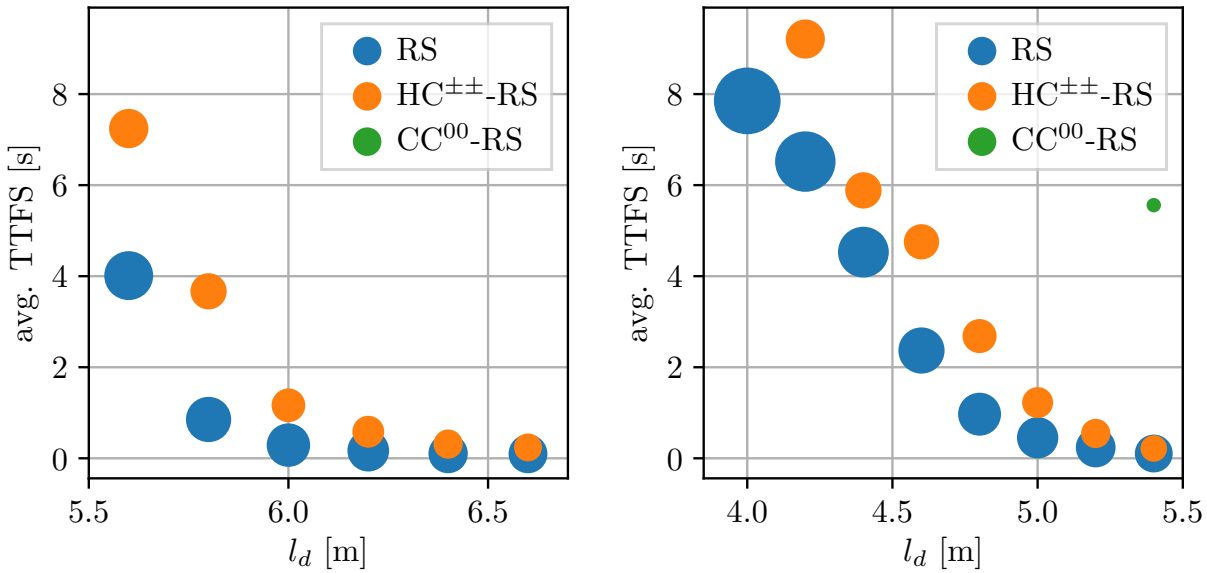
experiments below until no more solution can be computed within the given time limit. It has to be kept in mind that the length of the padded footprint is approximately 5.13 m (see Tab. 5.1 and Tab. 5.2), which makes it impossible to maneuver into parallel parking spots with l_d being smaller than this value.

A summary of the quantitative results in both scenarios can be found in Tab. 5.3. The performance of BiRRT* along with the different steering functions is evaluated based on the following five metrics: (1) the time-to-first-solution (TTFS), which measures the elapsed time until the first solution is found, (2) the number of curvature discontinuities, (3) the number of direction switches (cusps), (4) the length of the final path, and (5) the success rate, which indicates the relative number of simulation runs that succeeded with a solution. Note that the metrics (2)–(4) are given for the final solution after 3 s of optimization.

It can be seen both in Tab. 5.3 and Fig. 5.4 that using RS steer for motion

Table 5.3 Quantitative results of the experiments in the two high density parking scenarios shown in Fig. 5.3. The time-to-first-solution (TTFS), the number of curvature discontinuities, the number of cusps, and the length of the final path are given with mean and standard deviation, and the best value in each column is highlighted in bold.

		TTFS [s]			#curvature discont. [−]			#cusps [−]			length [m]			success rate [%]		
	l_d [m]	RS	HC$^{\pm\pm}$-RS	CC^{00}-RS	RS	HC$^{\pm\pm}$-RS	CC^{00}-RS	RS	HC$^{\pm\pm}$-RS	CC^{00}-RS	RS	HC$^{\pm\pm}$-RS	CC^{00}-RS	RS	HC$^{\pm\pm}$-RS	CC^{00}-RS
Scenario I	5.4	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
	5.6	4.0±2.4	7.2±2.7	-	15.1±3.3	9.1±1.3	-	8.5±1.4	10.5±1.3	-	17.1±4.0	19.6±7.1	-	79	12	0
	5.8	0.9±0.7	3.7±2.3	-	12.7±3.0	7.5±1.6	-	6.7±1.7	9.1±2.1	-	**16.6±2.9**	18.0±5.3	-	**100**	93	0
	6.0	0.3±0.2	1.2±0.9	-	11.5±2.6	6.2±1.4	-	5.6±1.2	7.4±1.7	-	16.9±4.3	**17.9±4.3**	-	100	**100**	0
	6.2	0.2±0.1	0.6±0.4	-	10.5±2.3	5.4±1.5	-	4.7±1.0	6.7±1.8	-	16.9±4.7	18.5±5.5	-	100	100	0
	6.4	0.1±0.1	0.3±0.3	-	9.1±2.2	4.3±1.3	-	3.8±1.2	5.6±1.7	-	16.7±4.7	18.2±5.7	-	100	100	0
	6.6	0.1±0.1	0.2±0.2	-	8.8±2.2	3.8±1.2	-	3.6±1.2	5.1±1.6	-	17.5±4.9	17.7±4.8	-	100	100	0
	⋮															
	7.2	**0.0±0.0**	**0.1±0.1**	-	8.6±2.2	**3.4±1.4**	-	3.4±1.2	4.4±1.6	-	18.7±4.5	18.7±5.0	-	100	100	0
	7.4	0.0±0.0	0.1±0.1	**0.4±0.2**	**8.1±1.8**	3.4±1.4	**0.0±0.0**	**3.3±0.9**	**4.3±1.6**	**6.3±1.6**	19.9±4.1	18.7±5.1	**29.7±6.0**	100	100	**100**
Scenario II	3.8	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
	4.0	7.9±1.8	-	-	30.0±3.0	-	-	14.0±0.0	-	-	28.1±0.8	-	-	2	0	0
	4.2	6.5±1.4	9.2±0.0	-	24.2±5.5	9.0±0.0	-	12.5±2.8	11.0±0.0	-	22.8±4.7	27.7±0.0	-	23	1	0
	4.4	4.5±2.4	5.9±2.7	-	16.5±5.0	7.5±1.8	-	7.9±2.8	9.5±2.6	-	17.8±4.3	17.5±4.4	-	76	22	0
	4.6	2.4±1.8	4.8±2.8	-	13.3±3.1	7.0±1.7	-	6.3±1.6	8.8±2.3	-	15.9±3.0	16.7±4.8	-	**100**	66	0
	4.8	1.0±0.7	2.7±2.2	-	11.2±2.7	6.4±1.9	-	5.2±1.4	7.8±2.4	-	14.6±2.0	16.1±3.7	-	100	97	0
	5.0	0.5±0.5	1.2±1.3	-	10.2±2.0	5.0±1.5	-	4.5±1.2	6.1±1.7	-	14.1±2.0	14.4±2.6	-	100	**100**	0
	5.2	0.2±0.2	0.5±0.6	-	9.3±1.5	4.3±1.4	-	3.9±1.1	5.4±1.7	-	14.0±1.9	14.5±2.6	-	100	100	0
	5.4	0.1±0.1	0.2±0.3	5.6±2.2	8.3±1.3	3.2±1.4	**0.0±0.0**	3.6±0.9	4.2±1.5	8.7±1.0	13.6±1.4	13.2±1.4	28.0±3.9	100	100	7
	5.6	0.1±0.1	0.1±0.1	5.3±3.0	7.5±1.6	2.5±1.0	0.0±0.0	3.4±1.0	3.4±0.9	7.7±1.2	12.9±1.0	12.6±0.8	23.2±4.0	100	100	44
	5.8	**0.0±0.0**	0.1±0.1	3.1±2.4	7.0±1.5	2.2±0.7	0.0±0.0	2.9±1.0	3.3±0.7	6.6±1.7	12.6±1.2	12.4±0.6	22.2±4.3	100	100	97
	6.0	0.0±0.0	**0.0±0.0**	1.2±1.2	6.5±1.3	2.1±0.4	0.0±0.0	2.4±0.8	3.2±0.6	5.6±1.6	12.3±1.3	**12.3±0.2**	20.0±3.8	100	100	**100**
	6.2	0.0±0.0	0.0±0.0	0.1±0.2	6.4±1.4	2.1±0.3	0.0±0.0	2.4±0.8	3.1±0.7	4.1±0.8	**12.1±1.2**	12.4±0.6	16.1±2.6	100	100	100
	6.4	0.0±0.0	0.0±0.0	**0.1±0.1**	**6.3±1.3**	**1.9±0.5**	0.0±0.0	**2.3±0.7**	**2.9±0.8**	**3.7±0.9**	12.1±1.6	12.9±1.5	**15.7±1.9**	100	100	100



(a) Scenario I. **(b)** Scenario II.

Figure 5.4 Illustration of the quantitative results from Tab. 5.3. The diagrams compare the average TTFS of BiRRT* in combination with the different G^1 and G^2 continuous steering functions when l_d is incrementally increased in both scenarios. The size of the circles qualitatively visualizes the number of curvature discontinuities, where CC^{00}-RS marks the baseline with no discontinuity along the entire path.

planning requires the least computation time, but results on average in 66 % to 232 % more curvature discontinuities than HC$^{\pm\pm}$-RS steer. Remember that every curvature discontinuity forces the vehicle to stop and adjust the steering angle at standstill. This is neither favorable from a passenger's point of view nor from the steering system (high mechanical stress).

Enforcing curvature continuity along the entire path can be realized with CC^{00}-RS steer, which, however, requires much more free space and on average 28 % to 126 % more cusps than HC$^{\pm\pm}$-RS steer. Especially in Scenario I, BiRRT* combined with CC^{00}-RS steer needs almost a 2 m longer parking spot to succeed compared to the other two steering functions. Within this context, it has to be noted that CC^{00}-RS steer is currently implemented without the topological paths [53]. Integrating them could possibly improve the performance in such tight environments while keeping the problem of (too) many direction switches. In contrast to that, a good trade-off between direction switches and curvature discontinuities can be realized with HC$^{\pm\pm}$-RS steer that balances the respective values between the ones of RS and CC^{00}-RS steer.

A comparison of the success rates in Tab. 5.3 reveals that the scenarios, which can be solved with RS steer, can generally also be tackled with HC$^{\pm\pm}$-

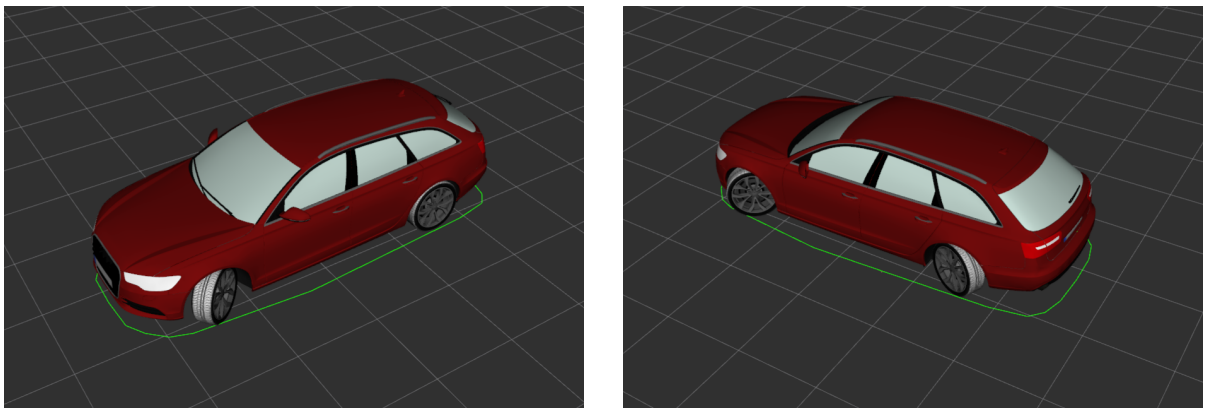
RS steer. This is due to the fact that both steering functions fulfill the topological property as described in Ch. 2 and therefore yield probabilistic completeness when integrated into BiRRT*. The reasons why RS steer still allows to realize higher success rates than $HC^{\pm\pm}$-RS steer are twofold: (1) the computation time of a single RS steering procedure is about a factor of 8 smaller (see Tab. 2.5) allowing to explore more states during planning, and (2) the computed RS paths are typically shorter (see Fig. 2.27(a)) and therefore less likely in collision with the environment.

Nevertheless, the combination of BiRRT* and $HC^{\pm\pm}$-RS steer yields a powerful approach for tight environments as it integrates smoothness into the motion plans while still preserving the probabilistic completeness of the planner.

5.3.2 From Single to Double Ackermann Steering

The definition of the steering functions with respect to curvature and its derivatives allows to directly apply them on vehicles with double Ackermann steering. This is shown in this section, where the first part analyzes the effects on motion planning while the second part highlights two real-world experiments.

As previously mentioned, double Ackermann steering denotes the capability to turn both the front and the rear wheels at the same time. Here, only the case is considered which increases the maximum curvature by turning the wheels on both axles in opposite directions (see Fig. 5.5). This increases

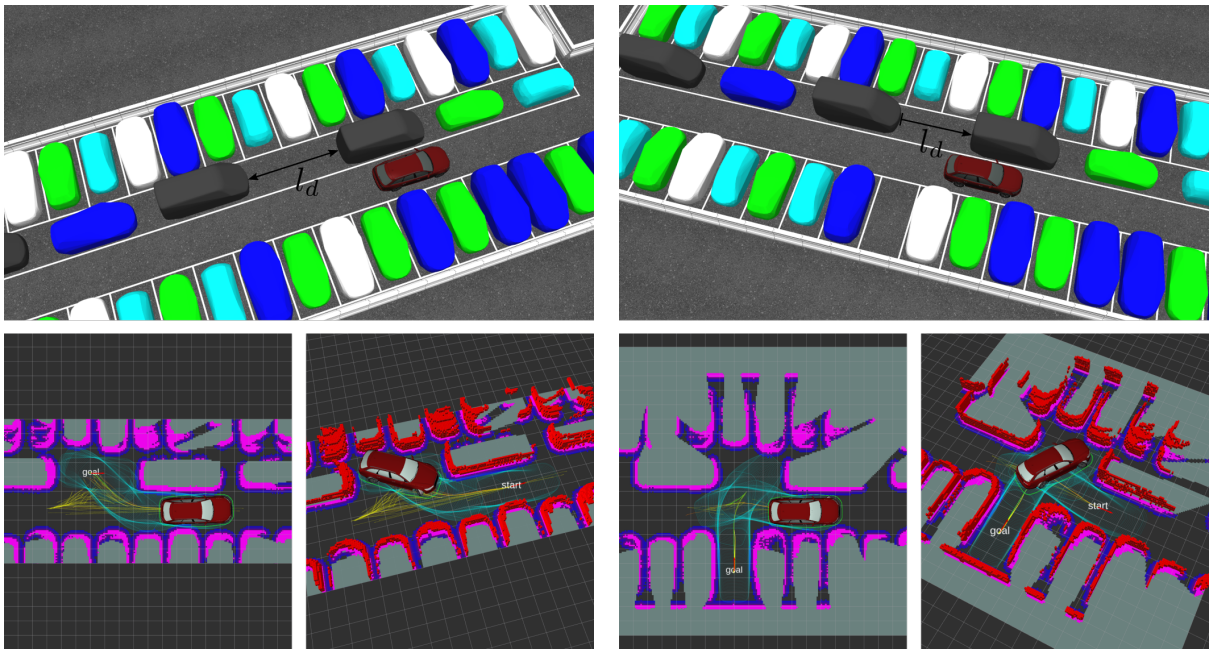


(a) Single Ackermann steering. **(b)** Double Ackermann steering.

Figure 5.5 Comparison of the Audi A6 with single and double Ackermann steering. In the image on the right, the maximum steering angle at the rear axle is set to be half as large as the one in the front.

the previously given parameters of the Audi A6 to $\kappa_{\max} = 0.2888\,\mathrm{m}^{-1}$ and $\sigma_{\max} = 0.2722\,\mathrm{m}^{-2}$, where the ratio of the maximum steering angle between the front and the rear axle is set to 1:0.5.

In order to analyze the effects of planning with the novel kinematic, the same experiments as in the previous section are repeated. The general setup is kept the same (see Fig. 5.6) with the only difference that the ego-vehicle's maximum curvature and maximum curvature rate are modified to the values above.



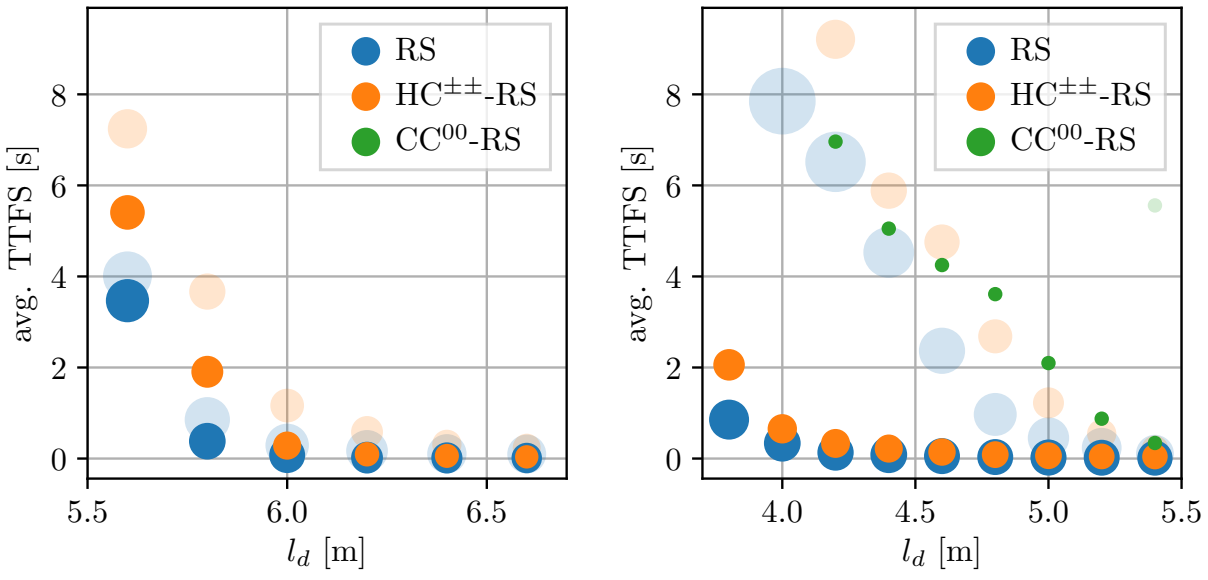
(a) Scenario I. **(b)** Scenario II.

Figure 5.6 High density parking scenarios used to analyze the effects of planning with double Ackermann steering. The motion plans visualized in the bottom row are calculated with BiRRT* and HC$^{\pm\pm}$-RS steer by taking into account the novel kinematic of the ego-vehicle.

A summary of the motion planning experiments including a comparison to single Ackermann steering is given in Tab. 5.4 and in Fig. 5.7. It can be seen that especially in Scenario II, double Ackermann steering has a significant effect on the computed motion plans. Solutions cannot only be found earlier, but also with less curvature discontinuities, less direction switches, and higher success rates. Tab. 5.4 shows that, for instance, the average number of curvature discontinuities can be reduced by at least 26 % and the average number of cusps by more than 27 % when double Ackermann steering is enabled in BiRRT* and HC$^{\pm\pm}$-RS steer. At the same time, the average TTFS

Table 5.4 Quantiative comparison of single and double Ackermann steering in the two high density parking scenarios shown in Fig. 5.3 and Fig. 5.6. The metrics TTFS, number of curvature discontinuities, and number of cusps are given with mean and standard deviation, and the best value in each column is highlighted in bold.

			TTFS [s]		#curvature discont. [–]		#cusps [–]		success rate [%]	
	l_d [m]	steering	RS	HC$^{\pm\pm}$-RS	RS	HC$^{\pm\pm}$-RS	RS	HC$^{\pm\pm}$-RS	RS	HC$^{\pm\pm}$-RS
Scenario I	5.4	single Ack.	-	-	-	-	-	-	0	0
		double Ack.	-	-	-	-	-	-	0	0
	5.6	single Ack.	4.0±2.4	7.2±2.7	15.1±3.3	9.1±1.3	8.5±1.4	10.5±1.3	79	12
		double Ack.	3.5±2.5	5.4±2.6	11.4±2.8	6.7±1.4	6.4±1.6	6.8±1.4	89	35
	5.8	single Ack.	0.9±0.7	3.7±2.3	12.7±3.0	7.5±1.6	6.7±1.7	9.1±2.1	**100**	93
		double Ack.	0.4±0.4	1.9±1.7	7.7±2.4	5.4±1.4	3.9±1.4	6.6±1.8	100	98
	6.0	single Ack.	0.3±0.2	1.2±0.9	11.5±2.6	6.2±1.4	5.6±1.2	7.4±1.7	100	**100**
		double Ack.	**0.1±0.1**	**0.3±0.2**	**7.3±1.7**	**3.9±1.3**	**3.3±1.0**	**4.8±1.6**	100	100
Scenario II	3.8	single Ack.	-	-	-	-	-	-	0	0
		double Ack.	0.9±1.1	2.1±2.0	9.5±1.7	5.2±1.9	3.8±0.8	7.2±2.2	**100**	99
	4.0	single Ack.	7.9±1.8	-	30.0±3.0	-	14.0±0.0	-	2	0
		double Ack.	0.3±0.4	0.7±0.7	7.7±1.5	4.3±1.6	2.7±1.0	5.9±1.6	100	**100**
	4.2	single Ack.	6.5±1.4	9.2±0.0	24.2±5.5	9.0±0.0	12.5±2.8	11.0±0.0	23	1
		double Ack.	0.1±0.2	0.3±0.3	**7.5±1.4**	4.2±1.4	**2.5±0.9**	5.6±1.6	100	100
	4.4	single Ack.	4.5±2.4	5.9±2.7	16.5±5.0	7.5±1.8	7.9±2.8	9.5±2.6	76	22
		double Ack.	**0.1±0.1**	**0.2±0.2**	7.7±1.6	**3.8±1.2**	2.5±0.9	**5.0±1.3**	100	100



(a) Scenario I. **(b)** Scenario II.

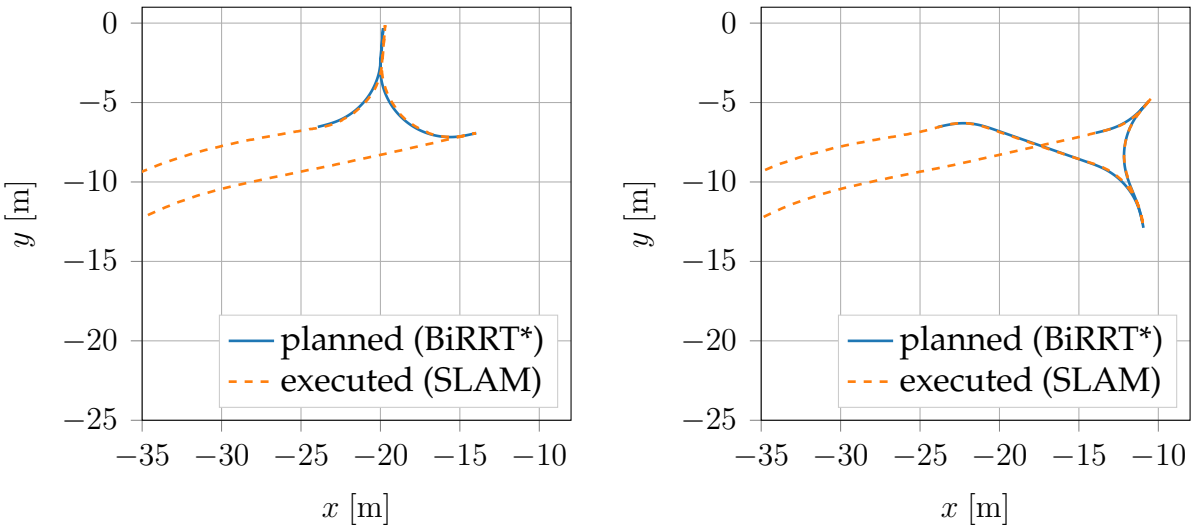
Figure 5.7 Visualization of the average TTFS and the number of curvature discontinuities (size of the circles) with respect to l_d when double Ackermann steering is enabled. For a better comparison, the transparent circles illustrate the results from Fig. 5.4, where the vehicle is only allowed to turn its front wheels.

can be decreased by at least 25 % and the success rates increased by up to 100 %. As double Ackermann steering improves all mean values in Tab. 5.4, it can be concluded that the ability to steer all four wheels in such tight environments is highly favorable from a motion planning perspective.

The sampling-based motion planner that has been so far benchmarked in simulation is also integrated into the motion planning framework of the Bosch Campus Shuttle. The underlying platform is an EasyMile EZ10 with double Ackermann steering (see Fig. 5.1(b)), whose vehicle parameters are listed in Tab. 5.1. Obstacles in the environment are perceived by a LiDAR system and fused in an occupancy grid with a resolution of 15 cm and a size of 50 m $\times$ 50 m. The reference pose of the vehicle is determined without GPS using a simultaneous localization and mapping (SLAM) algorithm [186].

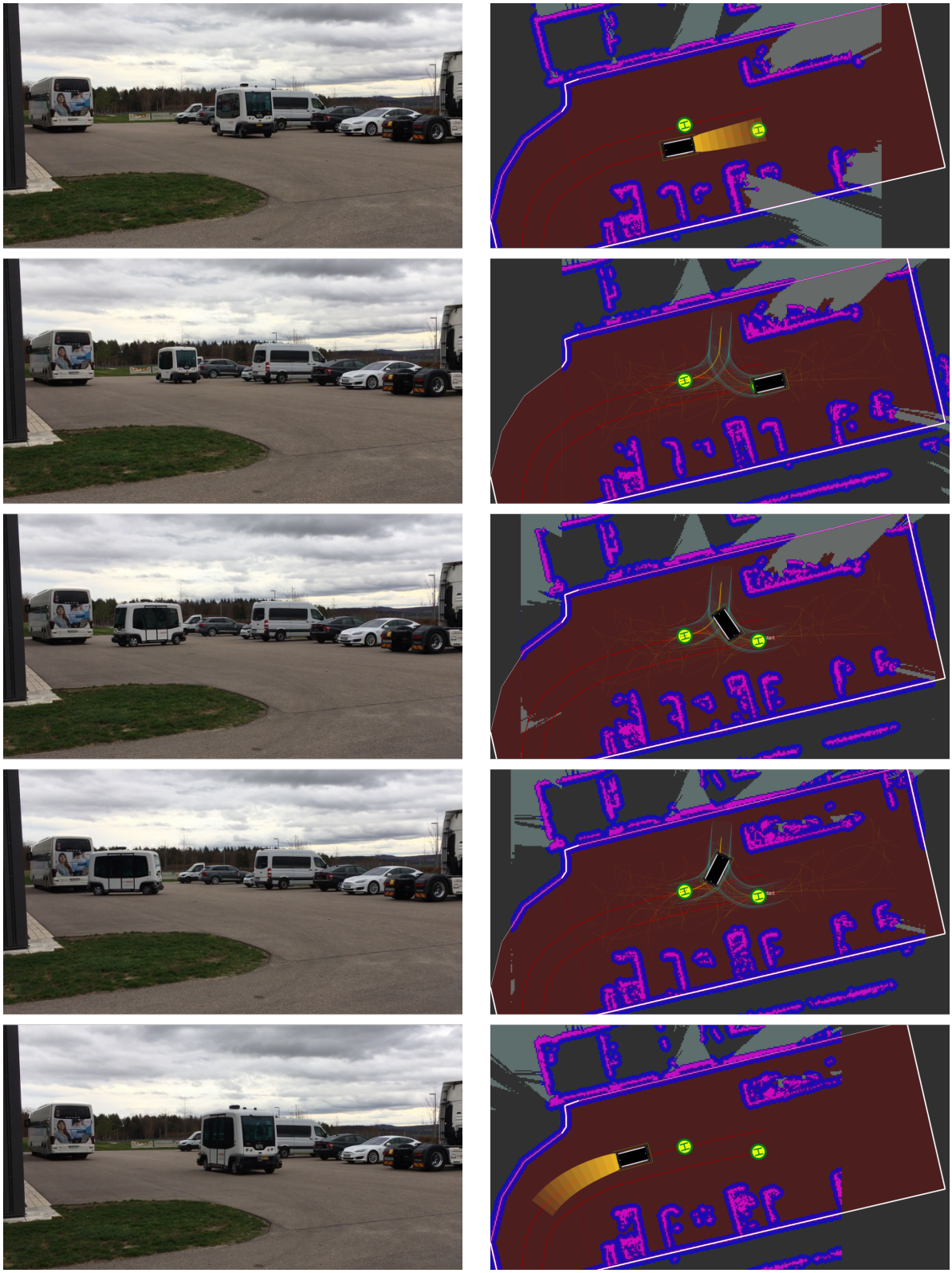
A demonstration of the planner's capability in two real-world scenarios can be found in Fig. 5.9 and Fig. 5.10. The first scenario requires the shuttle to execute a turn at the end of a line while the second scenario requires an autonomous parking maneuver at the shuttle's depot. In both scenarios, the maneuvers are computed online using BiRRT* and CC^{00}-RS steer, whose solution is then executed by a feedback linearizing motion controller [97].

It can be seen both in Fig. 5.9 and Fig. 5.10 that the computed motion plans smoothly transition the vehicle from start to goal while avoiding the surrounding obstacles. The visualization in Fig. 5.8 also shows that the tracking error is generally small along the entire path.



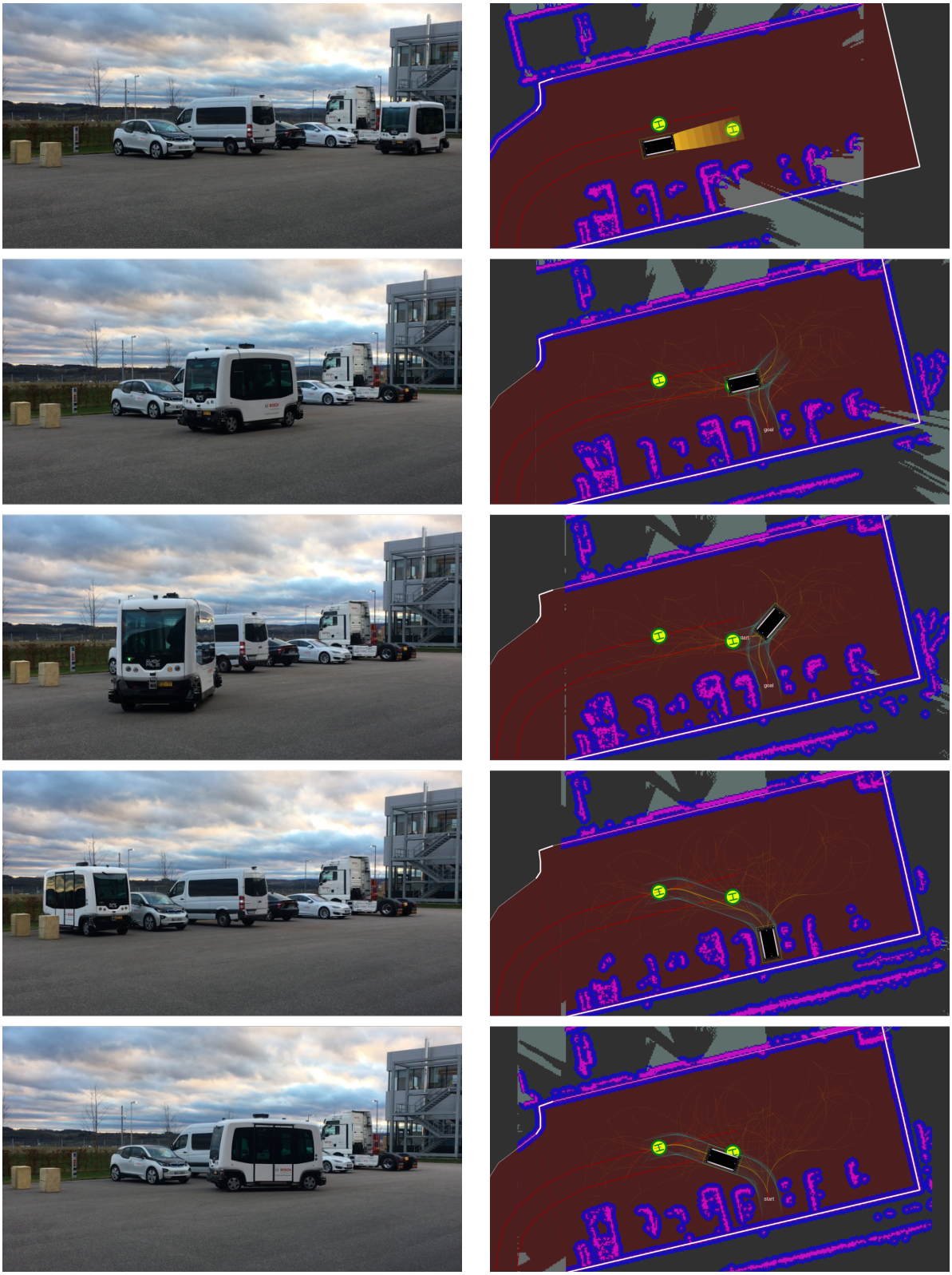
(a) Turn at the end of the line. **(b)** Parking at the depot.

Figure 5.8 Visualization of the planned and the executed path in the two shuttle experiments shown in Fig. 5.9 and Fig. 5.10.



(a) Real-world view. **(b)** Robot view.

Figure 5.9 Execution of a turn at the end of the line. The sequence on the left displays the executed maneuver while the one on the right illustrates the corresponding robot view. After having executed the turn, the robot switches back to corridor driving as shown in the image on the bottom right.



(a) Real-world view. **(b)** Robot view.

Figure 5.10 Perpendicular parking after arriving at the shuttle's depot followed by a maneuver that moves the shuttle back to the start of the line.

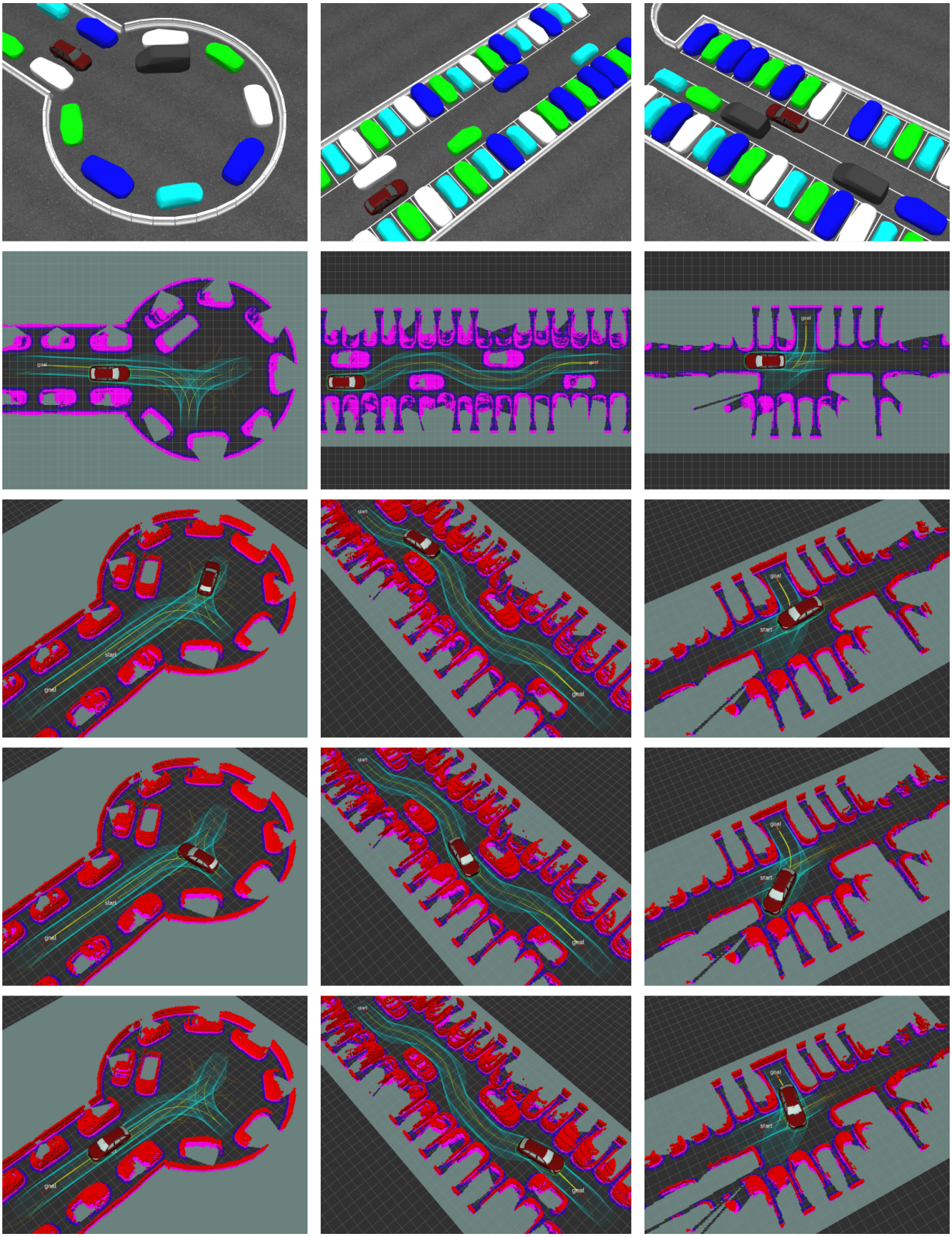
However, small deviations from the planned path can still be observed in the real-world experiments mostly due to localization and control uncertainty as well as due to the curvature rate discontinuities of the G^2 continuous paths. The latter is addressed in the next section by increasing the continuity of the motion plans to G^3.

5.3.3 G^3 Continuous Motion Planning

The previous two sections have shown the planner's capability to compute G^1 and G^2 continuous paths in dense scenarios. Certain situations might require, however, to plan even smoother motion plans that reduce lateral jerk and improve the tracking performance in closed loop. For this purpose, BiRRT* is combined with HCR and CCR steer to compute G^3 continuous paths in three challenging automated driving scenarios. A visualization of all three scenarios including example paths and a sequence of their execution is given in Fig. 5.11.

The first scenario illustrates a dead end, where the red ego-vehicle is required to execute a multi-point turn due to the parked black transporter. Multiple narrow passages have to be passed in the second scenario, which represents a blocked driveway, and parking on a high density parking lot has to be conducted in the third scenario. The three different environments are represented by an occupancy grid with a resolution of 10 cm and a size of $40\,\mathrm{m} \times 40\,\mathrm{m}$ in Scenario I, $50\,\mathrm{m} \times 20\,\mathrm{m}$ in Scenario II, and $40\,\mathrm{m} \times 20\,\mathrm{m}$ in Scenario III. The steering system is switched back to single Ackermann steering, and the maximum curvature acceleration $\rho_{\max}$ is set to $0.3905\,\mathrm{m}^{-3}$.

A summary of the motion planning experiments in the previously described scenarios can be found in Tab. 5.5. As expected, the G^1 continuous steering functions RS and Dubins require on average the least computation time to find an initial solution. Furthermore, the corresponding final paths have on average the shortest length and a low number of direction switches. However, an average of up to 14.2 curvature discontinuities makes it impractical to directly execute these paths. In contrast to that, curvature continuity can be enforced with the G^2 continuous steering functions, but the resulting paths suffer from on average up to 24.7 curvature rate discontinuities. Remember that every curvature rate discontinuity implies an infinite steering acceleration. In closed loop, this might lead to a jerky motion, a high mechanical load on the steering system, or a potential deviation from the planned path.



(a) Scenario I. **(b)** Scenario II. **(c)** Scenario III.

Figure 5.11 G^3 continuous motion planning in three distinct scenarios. The red ego-vehicle is required to execute a multi-point turn at a dead end in Scenario I, pass a blocked driveway in Scenario II, and park on a high density parking lot in Scenario III. The illustrated motion plans are computed with BiRRT* and the steering functions CCR^{00}-RS in (a), CCR^{00}-D in (b), and HCR^{00}-RS in (c).

Table 5.5 Benchmark of BiRRT* along with the G^1, G^2, and G^3 continuous steering functions in the three scenarios from Fig. 5.11. The metrics TTFS, number of curvature and curvature rate discontinuities, number of cusps, and final path length are given with mean and standard deviation. Note that for consistency with Footnote 5 (p. 82), every curvature discontinuity is also counted as a curvature rate discontinuity.

scen.	steer. fct.	cont.	TTFS [s]	#curv. discont. [–]	#curv. rate discont. [–]	#cusps [–]	length [m]	succ. [%]
I	RS	G^1	0.01±0.01	6.4±0.8	6.4±0.8	2.0±0.0	40.9±3.5	100
	CC^{00}-RS	G^2	0.02±0.01	0.0±0.0	13.7±3.0	2.0±0.0	42.6±3.1	100
	CCR^{00}-RS	G^3	0.03±0.03	0.0±0.0	0.0±0.0	2.3±0.7	47.2±4.1	100
II	Dubins	G^1	0.17±0.13	14.2±1.8	14.2±1.8	0.0±0.0	34.8±0.1	100
	CC^{00}-D	G^2	0.22±0.22	0.0±0.0	24.7±4.5	0.0±0.0	34.8±0.1	100
	CCR^{00}-D	G^3	2.25±1.70	0.0±0.0	0.0±0.0	0.0±0.0	34.9±0.1	98
III	RS	G^1	0.06±0.05	6.7±1.2	6.7±1.2	3.0±1.0	14.5±4.2	100
	HC^{00}-RS	$G^{2\bullet}$	0.27±0.28	1.8±0.9	10.6±2.6	2.9±1.3	15.7±4.4	100
	HCR^{00}-RS	$G^{3\bullet}$	0.77±0.65	2.1±1.1	2.1±1.1	4.2±1.9	20.7±6.6	100

$^\bullet$between direction switches

As shown in Tab. 5.5 and Fig. 5.12, the curvature rate discontinuities in the path can be eliminated or significantly reduced by combining BiRRT* with the G^3 continuous steering functions. At least between direction switches, these

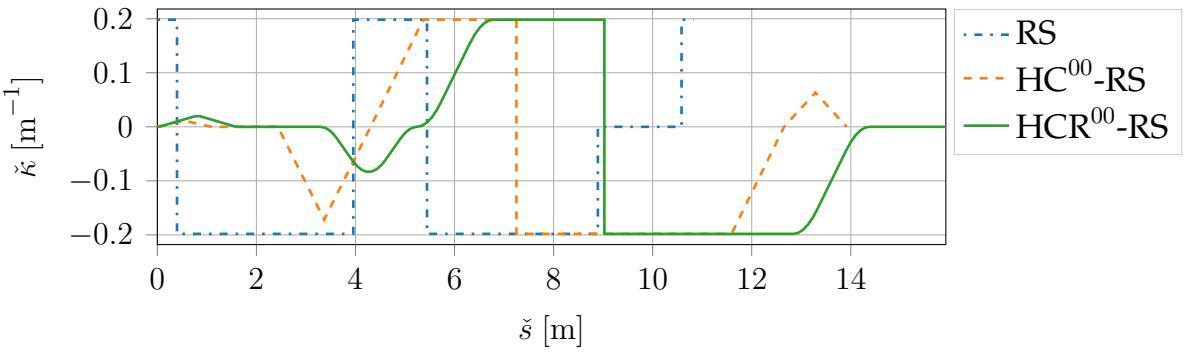


Figure 5.12 Curvature profile of three example paths in Scenario III computed with the steering functions RS, HC^{00}-RS, and HCR^{00}-RS. The smoothness increases from discrete curvatures to a linear and a quadratic profile, where curvature discontinuities are only allowed at cusps.

steering functions bound the maximum curvature, the maximum curvature rate, and the maximum curvature acceleration. While this improves the overall smoothness of the motion plans, two limitations can be observed in Tab. 5.5: (1) the average TTFS increases significantly in highly complex

scenarios, such as Scenario II, leading to a potential decline in success rate, and (2) the length of the paths and the number of cusps might increase. The reasons behind this are the higher computation time of a single G^3 continuous steering procedure (see Tab. 2.7) and the typically higher path length of the corresponding output (see Fig. 2.43(a)). While the former allows BiRRT* to explore less states, the latter implies that the G^3 continuous steering procedures are more likely in collision with the environment than the G^1 and G^2 continuous counterparts. Note, however, that the analyzed limitations play only a minor role in less complex scenarios, such as in Scenario I, where planning with the G^3 continuous steering functions yields high quality paths with only small computational overhead.

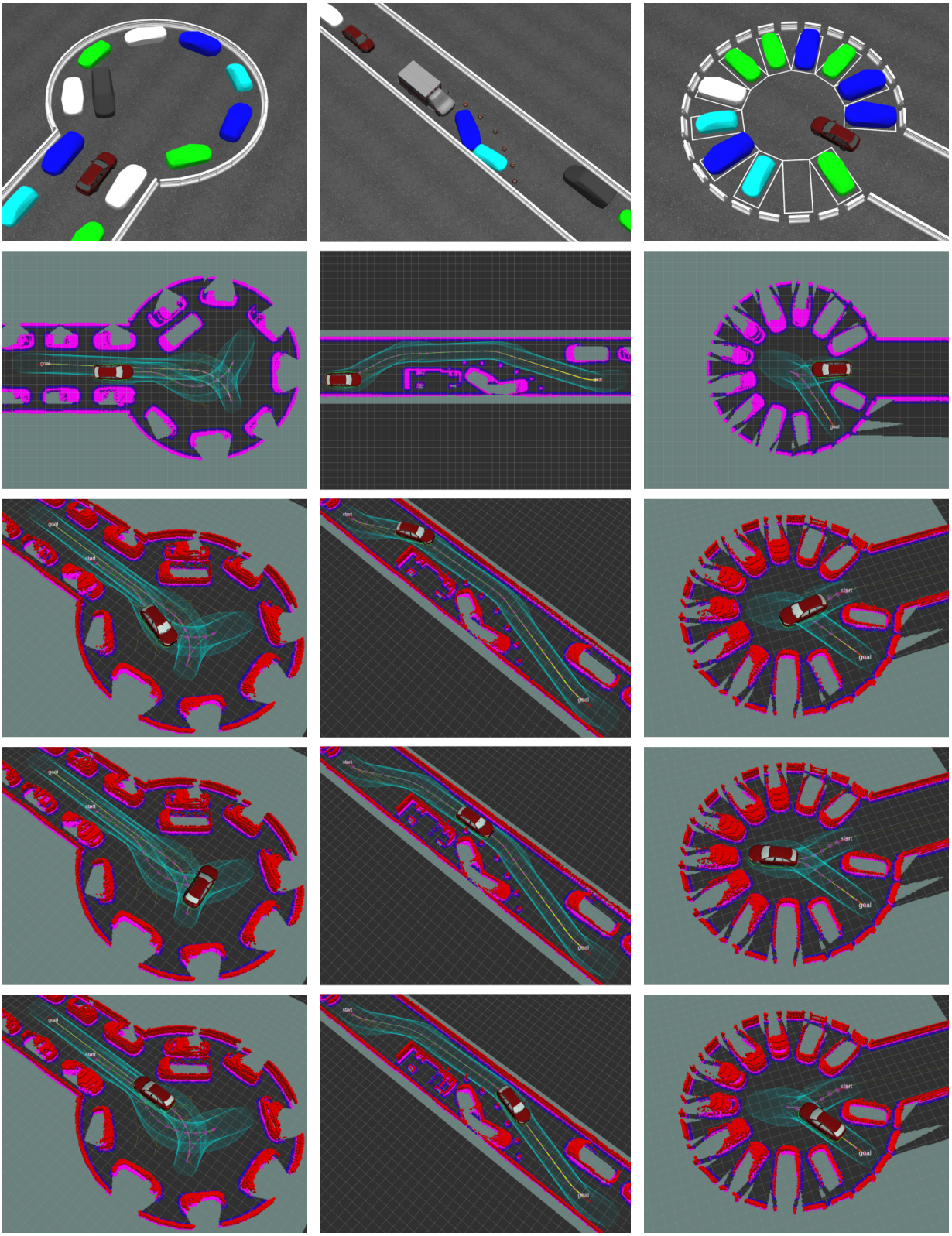
5.3.4 Motion Planning in Gaussian Belief Space

The previous three sections have shown a variety of experiments, where motion plans are computed without explicitly considering the uncertainties of the system. Especially in tight environments, however, neglecting possible disturbances might lead to dangerous situations with high collision probabilities. Therefore, this section explores the benefits and the possible drawbacks of planning in belief space by taking into account the localization and control uncertainty. Note that the perception noise of the LiDAR system is currently neglected, but could also be considered here, e.g. by switching to a probabilistic occupancy grid.

Fig. 5.13 visualizes the three scenarios used to benchmark the belief space planner against two nominal baselines. While Scenario I is already known from the previous section, two novel situations are introduced in Scenario II and III including an accident blocking the road and a circular parking lot.

The environment is represented by an occupancy grid with a resolution of 10 cm and a size of 40 m $\times$ 40 m in Scenario I and III and 50 m $\times$ 10 m in Scenario II. Motion planning in belief space is conducted here on the basis of (5.4)–(5.5) using RRT*, the belief steering functions from Ch. 2, and the concept of beliefprints from Ch. 3. Note that the latter is required to enforce the chance constraint in (5.4i). To do so, the beliefprints are first computed using an icosahedron and the unpadded footprint of the ego-vehicle and then checked for collision with the surrounding obstacles.

An overview of the required parameters for planning in belief space including the confidence P for the computation of the beliefprints can be found in Tab. 5.6.



(a) Scenario I. **(b)** Scenario II. **(c)** Scenario III.

Figure 5.13 Motion planning in Gaussian belief space. The red ego-vehicle is required to execute a multi-point turn at a dead end in Scenario I, pass an accident at the side of the road in Scenario II, and park on a circular parking lot in Scenario III. The illustrated example paths are calculated with RRT* and the belief steering functions BCC^{00}-RS in (a), BCC^{00}-D in (b), and BHC^{00}-RS in (c).

Table 5.6 Additional parameters required for belief space planning. Adapted from [213], © 2018 IEEE.

parameter	symbol	value
controller gain	k_1, k_2, k_3	1.5, 0.36, 1.2
motion noise	α_1, α_2, α_3, α_4	0.2, 0.1, 0.1, 0.3
measurement noise	σ_x, σ_y, σ_θ	0.1 m, 0.1 m, 0.05 rad
confidence	P	97.1 % (three-sigma in three dim.)

The two baselines used to benchmark the belief space planner solve the scenarios above with BiRRT* and the nominal steering functions. While one of the two baselines applies no footprint padding, the other one inflates the footprint by 10 cm to every side serving as a hard safety buffer. Within this context, it has to be noted that applying a constant footprint padding assumes an evenly distributed two-dimensional covariance, which does not change along the path.

A summary of the experimental results including the evaluation of six metrics can be found in Tab. 5.7.

Table 5.7 Comparison of planning in belief space with the two nominal baselines. The letter 'B' is used to distinguish the belief steering functions from the nominal ones. Values with a plus-minus sign are given with mean and standard deviation. Adapted from [213], © 2018 IEEE.

scen.	planner	steer. fct.	footprint	TTFS [s]	#vertices [–]	#cusps [–]	length [m]	CP [%]	succ. [%]
I	BiRRT*	CC^{00}-RS	unpadded	0.01±0.01	117.0±3.3	2.0±0.0	42.0±3.4	3.6±8.2	100
	BiRRT*	CC^{00}-RS	padded	0.02±0.01	118.3±3.7	2.0±0.0	42.6±3.1	1.2±3.3	100
	RRT*	BCC^{00}-RS	unpadded	0.23±0.29	45.1±3.3	2.1±0.5	46.9±3.7	0.1±0.4	100
II	BiRRT*	CC^{00}-D	unpadded	0.05±0.04	236.4±6.9	0.0±0.0	37.2±0.1	9.4±13.5	100
	BiRRT*	CC^{00}-D	padded	0.14±0.13	233.6±17.6	0.0±0.0	37.3±0.1	3.9±6.1	100
	RRT*	BCC^{00}-D	unpadded	3.22±2.43	71.4±9.5	0.0±0.0	37.3±0.2	0.9±1.1	88
III	BiRRT*	HC^{00}-RS	unpadded	0.05±0.06	106.8±4.4	1.4±0.7	18.2±4.7	8.8±20.5	100
	BiRRT*	HC^{00}-RS	padded	0.06±0.09	107.6±4.9	1.5±0.9	18.2±5.0	3.6±9.8	100
	RRT*	BHC^{00}-RS	unpadded	0.93±1.16	50.1±4.2	2.2±1.7	19.1±4.7	0.1±0.4	100

Compared to the previous benchmarks, two additional metrics are computed here, namely the number of vertices in the tree and the collision probability (CP). While the former helps to understand how well the different approaches explore the state space, the latter evaluates the robustness of a

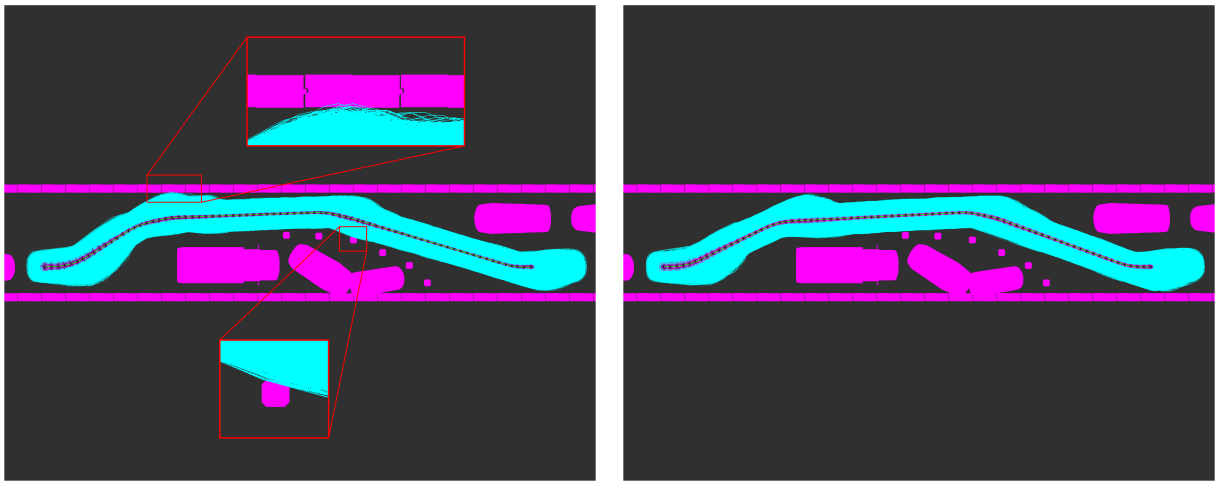
motion plan. As such, the CP is defined as the relative number of 100 Monte Carlo (MC) runs (see Sec. 2.5.1) that collide with the environment.

It can be seen in Tab. 5.7 that planning in Gaussian belief space increases the TTFS and allows to explore on average up to 70 % less vertices than the nominal approaches. The result is a possible increase in path length and in the number of direction switches. Furthermore, the success rate might drop when not enough states can be explored within the given sampling time. The reasons behind these effects are threefold: (1) the belief propagation and the computation of the beliefprints increase the complexity of the planning process leading to less exploration, (2) planning in belief space can only be tackled with the single-tree approach and not with the typically faster two-tree version of the planner, and (3) the nominal baselines also add potentially dangerous states to the tree, which are sorted out by the belief space planner.

The latter can be observed in Tab. 5.7, where only on average up to 0.9 % of the 100 MC runs are in collision when planning was conducted in belief space. Two aspects lead to this result: on the one hand, the chance constraint in (5.4i) enforces a CP below $1 - P$ for every vehicle state, and on the other hand, the cost term in (5.5c) incentivizes the planner to increase the distance to surrounding obstacles. On the contrary, the CP of the nominal motion plans is up to an order of magnitude higher compared to the belief space approach. For instance, the mean CP raises to up to 9.4 % in the nominal case without footprint padding. Adding a hard safety buffer of 10 cm to the footprint helps to reduce this number, but the corresponding CPs are on average still as high as 3.9 %. The reason for this is that the actual covariance is neither two-dimensional nor evenly distributed or constant along the path as assumed by a static footprint padding.

The previous observations are qualitatively highlighted in Fig. 5.14, which augments an example motion plan of the nominal and the belief space approach in Scenario II with 100 MC runs. It can be observed that in the nominal case, up to 9 % of the runs collide with the upper wall or one of the traffic cones although a footprint padding of 10 cm has been applied during planning. In contrast to that, none of the MC runs collide with the environment in the second case when planning is conducted in Gaussian belief space.

In conclusion, belief space planning significantly increases the safety of the motion plans while at the same time raising the computational complexity. It has to be kept in mind that planning in belief space requires a profound understanding of the system's uncertainties as well as a model of the underlying noise. With respect to computation time, a significant speed-up could



(a) Nominal motion plan. **(b)** Belief space motion plan.

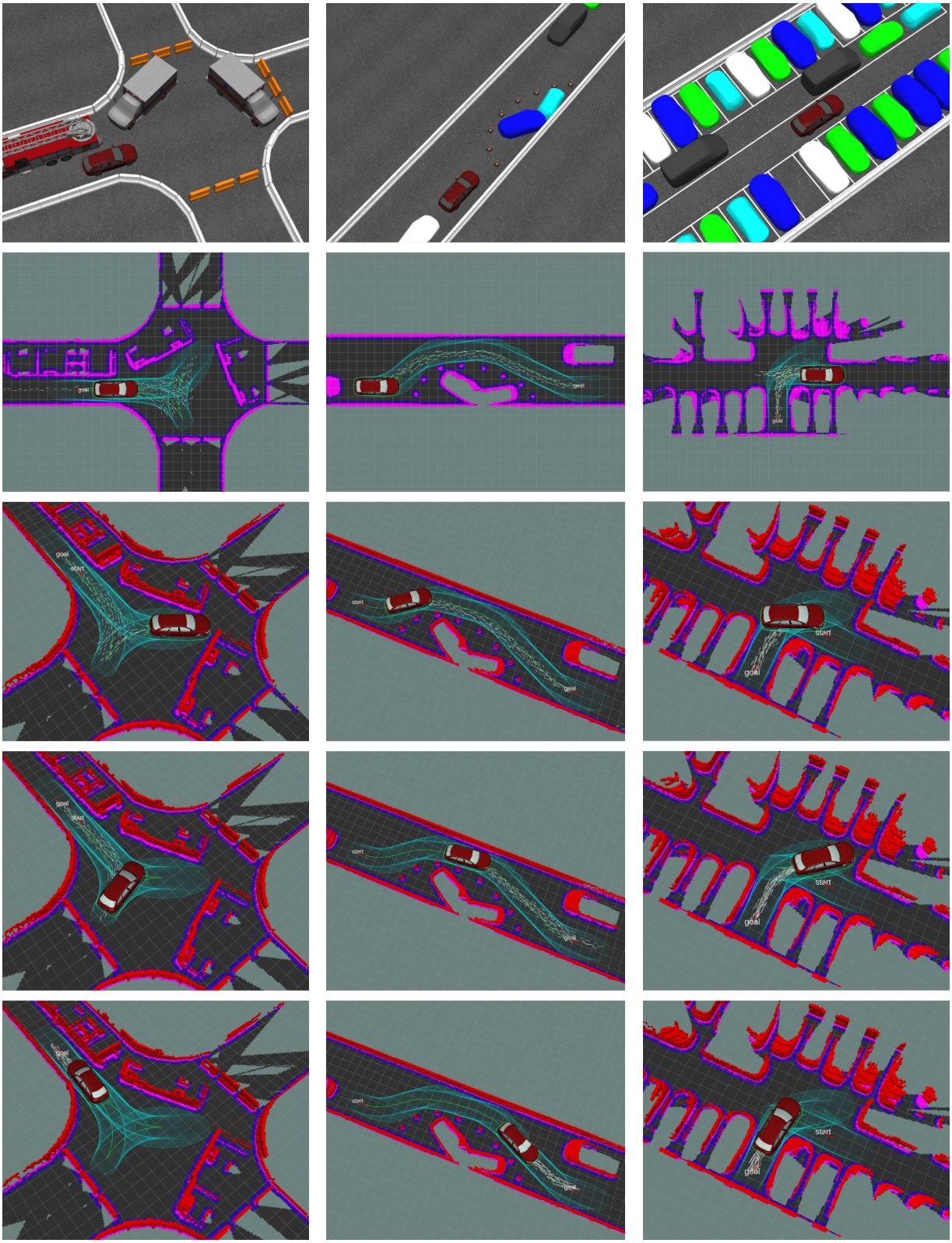
Figure 5.14 Qualitative comparison of two example paths in Scenario II once computed with the nominal baseline and a 10 cm footprint padding (a) and once with the Gaussian belief space planner (b). Both solutions are augmented with 100 MC runs and the corresponding swaths of the robot (cyan). In the image on the left, 9 % of the simulated runs collide with the environment (upper wall or third traffic cone) while none of them are in collision on the right.

be achieved by switching to a non-uniform sampling distribution that adapts to the environment as shown in the next section.

5.3.5 Guided Motion Planning

One of the major drawbacks of sampling-based motion planners are the potentially high computation times and the low convergence rates due to uninformed (uniform) sampling. Especially in narrow passages, such as in Fig. 5.13(b), exhaustive sampling might be required to find one of the relatively few samples that steer the robot towards the goal. In order to overcome such problems, non-uniform sampling distributions can be used that leverage the information about the environment to accelerate the planning process. Within this context, this section evaluates the potential of guiding the sampling-based motion planner BiRRT* with the pose predictions of the CNN introduced in Ch. 4. The corresponding evaluations are conducted on the three automated driving scenarios shown in Fig. 5.15.

The first scenario depicts a blocked intersection that requires the red ego-vehicle to perform a multi-point turn. A narrow passage problem is given in the second scenario, where an evasive maneuver has to be conducted in order



(a) Scenario I. **(b)** Scenario II. **(c)** Scenario III.

Figure 5.15 Guided motion planning in three automated driving scenarios including a blocked intersection (a), a blocked lane due to an accident (b), and a high density parking lot (c). The visualized example paths are computed with HC^{00}-RS steer in (a) and (c) and with CC^{00}-Dubins in (b). The gray arrows on the ground illustrate 200 sampled vehicle poses from the output of the CNN. Adapted from [216], © 2019 IEEE.

to pass the accident at the side of the road. The third scenario finally shows the already known high density parking lot with a perpendicular parking task. Motion planning is carried out in all three scenarios using an occupancy grid with a resolution of 10 cm and a size of 60 m $\times$ 60 m. Notice that the CNN has not seen these environments before as they were barred from the training dataset. Furthermore, they expose the CNN to novel artifacts including traffic cones and various new vehicle shapes.

Guiding BiRRT* with the CNN from Ch. 4 is benchmarked in this section against two baselines. The first one applies uniform sampling similar to the experiments in the previous sections, and the second one uses the OSE heuristic [30] as described in Sec. 4.4 to bias the planner towards the goal. In case BiRRT* is combined with either the CNN or the OSE, non-uniform samples are computed in batches of 100 at a frequency of 4 Hz. These samples are then mixed evenly with uniform ones in order not to violate the theoretical guarantees of the planner.

An overview of the experimental results generated with BiRRT* along with the different sampling distributions can be found in Tab. 5.8.

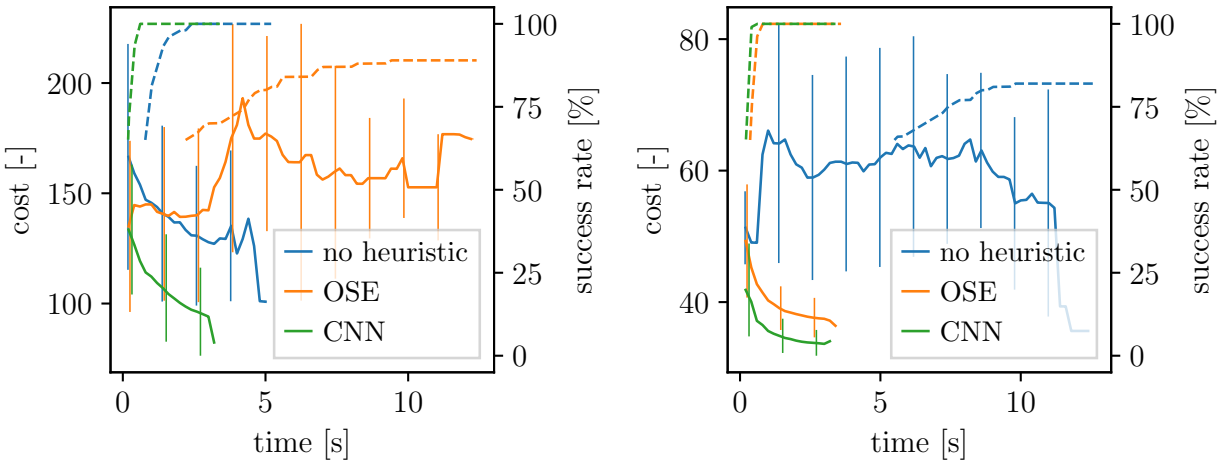
Table 5.8 Guided motion planning benchmark. The results are obtained with BiRRT* and the steering functions HC^{00}-RS in Scenario I and III and CC^{00}-Dubins in Scenario II. Values with a plus-minus sign are given with mean and standard deviation. Adapted from [216], © 2019 IEEE.

scen.	heuristic	pred. [ms]	TTFS [s]	#vertices [−]	#cusps [−]	length [m]	$\text{cost}_{\text{TTFS}}$ [−]	$\text{cost}_{\text{TTFS+3s}}$ [−]	succ. [%]
I	-	-	0.57±0.41	79.2±14.3	4.0±1.2	35.2±4.8	162.0±48.3	124.1±28.2	100
	OSE	127.6	1.54±2.15	139.2±27.6	4.6±1.6	38.7±7.0	160.7±50.4	145.1±42.4	89
	CNN	82.8	0.13±0.07	140.6±7.5	2.2±0.8	28.1±3.9	134.4±37.2	93.4±18.9	100
II	-	-	3.51±2.31	127.0±12.6	0.0±0.0	25.7±0.2	63.4±16.4	60.6±15.9	82
	OSE	17.4	0.12±0.14	187.2±12.0	0.0±0.0	25.7±0.1	50.1±11.2	37.4±2.9	100
	CNN	82.4	0.11±0.03	221.2±8.0	0.0±0.0	25.7±0.1	43.7±9.4	33.6±1.9	100
III	-	-	2.92±2.53	96.1±15.7	3.6±1.5	20.7±14.3	152.4±51.0	148.2±51.9	91
	OSE	19.0	0.02±0.03	154.6±6.0	3.0±1.0	12.8±1.5	143.6±21.0	119.2±16.3	100
	CNN	80.9	0.09±0.02	157.9±4.6	2.0±0.0	13.0±0.6	100.4±23.5	84.0±3.7	100

Three additional metrics are evaluated here, namely the prediction time to generate the batch of heuristic samples, the cost of the initial solution $\text{cost}_{\text{TTFS}}$, and the cost after 3 s of optimization $\text{cost}_{\text{TTFS+3s}}$. The last two metrics are computed according to (5.2) and indicate the quality of the path at two different time steps.

It can be seen in Tab. 5.8 that the samples from the CNN not only help to compute paths with the initially lowest average cost, but also allow to

converge to a lower-cost solution than the other two sampling distributions. This can also be seen in Fig. 5.16, which illustrates the convergence rate in Scenario I and II. The corresponding plot for Scenario III is omitted here for brevity, but can be found in [216].



(a) Scenario I. **(b)** Scenario II.

Figure 5.16 Convergence and success rate during planning. The solid lines indicate the average cost of the three approaches, the error bars the respective standard deviations, and the dotted lines the success rate over time, which is only visualized above 65 % for better readability. Notice that the cost of a single motion plan decreases monotonically with respect to planning time, however, averaging over multiple runs can result in a non-monotonic behavior. Adapted from [216], © 2019 IEEE.

Furthermore, it can be observed in Tab. 5.8 that the predictions of the CNN stabilize the average TTFS to about 100 ms and reduce the corresponding standard deviation to the lowest value of all three approaches. This is highly beneficial from a practical point of view as low latencies allow to react quickly to new situations leading to an overall improvement of the system's performance. Within this context, it has to be noted that the TTFS of the CNN-guided BiRRT* could be further reduced by improving the prediction time of the CNN. The latter currently takes about 81 ms to 83 ms (see Tab. 5.8) and is therefore twice as high as in the benchmark conducted in Sec. 4.4. This is due to the fact that the CNN runs here along with the simulation on an Nvidia Quadro K2200 and not on an Nvidia Titan X as in the previous evaluation.

In addition to low computation times, it is also required in practice to find solutions, if any exist, with a high success rate. Tab. 5.8 shows that the only approach that achieves a success rate of 100 % in all three scenarios is the

one that uses samples from the CNN. While BiRRT* combined with the OSE heuristic also solves all simulation runs in Scenario II and III, its success rate drops by 11 % in Scenario I. The reason for this is that the OSE heuristic reduces the vehicle to a circular holonomic robot and only takes into account the nonholonomic constraints using a cost term [30]. In Scenario I, this fact leads to the problem that the heuristic proposes an immediate turn at the current position. As such a maneuver can, however, not be realized due to the obstacles on both sides (see Fig. 5.15(a)), many of the heuristically generated samples do not help the planner to solve this situation. BiRRT* must therefore rely on the other half of the samples (uniformly generated), which are, however, not enough to solve all of the 100 runs.

5.4 Summary

The effectiveness of the developed methods in sampling-based motion planning is demonstrated in this chapter. A variety of experiments were conducted both in simulation and in the real world on two distinct vehicles with both single and double Ackermann steering. The benchmarks were carried out on a broad set of challenging automated driving scenarios with a focus on maneuvering at low speeds in tight environments. These scenarios require a global solution to the motion planning problem, which in some situations can only be solved by moving forwards and backwards.

In particular, the benchmark in Sec. 5.3.1 shows that the novel steering function HC-RS steer allows to compute smoother paths than RS steer with a higher success rate and less direction switches than CC-RS steer. As a result, HC-RS steer yields a powerful approach for sampling-based motion planning in tight environments by directly enforcing curvature continuity between direction switches.

The benefits of not only steering the front but also the rear wheels is highlighted in Sec. 5.3.2. It is shown that double Ackermann steering is highly beneficial from a motion planning perspective as it improves the overall quality of the computed paths. Furthermore, two real-world experiments are included in this section that demonstrate the effectiveness of the developed motion planner on a vehicle with double Ackermann steering.

Increasing the smoothness of the motion plans to curvature rate continuity using the novel G^3 continuous steering functions is conducted in Sec. 5.3.3. The analysis reveals that this can be realized with only little computational

overhead in less complex scenarios. In extremely tight environments, however, problem-specific sampling distributions would be required to compensate for the increase in computation time and the deterioration in path quality.

The results in Sec. 5.3.4 highlight that planning in Gaussian belief space significantly increases the robustness of the motion plans by bounding the collision probability of every vehicle state. This was realized with the extension of the steering functions to Gaussian belief space along with the novel concept of beliefprints. The additional computational complexity, however, requires heuristics that speed-up the planning process and improve the convergence rate towards a cost-minimizing solution.

An example of such an approach, where a CNN generates proposals for the sampling-based motion planner, was finally benchmarked in Sec. 5.3.5. It is demonstrated that the CNN adapts to various driving situations and helps to stabilize the average TTFS to about 100 ms in the conducted experiments. In comparison to uniform sampling and an A*-based heuristic, the CNN not only enables the generation of motion plans with an initially lower cost, but also improves the convergence to a better solution. It can therefore be concluded that the resulting approach is particularly suitable for global motion planning with real-time constraints.

6 Conclusion and Outlook

This thesis addresses the motion planning problem in automated driving and focuses on the computation of global motion plans in complex tight environments. Although first results in this field date back to at least 1986 [110], coming up with a generic approach that computes high quality solutions in arbitrary scenarios still remains a challenging task. However, such an approach is required for level 4/5 automated driving [185] in order to ensure autonomy in all possible scenarios.

6.1 Conclusion

Motivated by the research questions in Sec. 1.3, several modular components were developed, analyzed, and benchmarked within this thesis in order to improve the effectiveness of existing global motion planners. The key contributions were made in three different fields and are briefly summarized in the following paragraphs.

Steering functions The state of the art, as reviewed in Sec. 1.2.1, shows that steering functions are an important concept in both search-based and sampling-based motion planning. Many of the existing approaches rely on the well-known steering functions Dubins and Reeds-Shepp whose paths are only discrete in curvature and hence typically require a post-smoothening step. An alternative to this is CC steer that enforces curvature continuity along the entire path. The experimental results in Sec. 5.3.1 have demonstrated, however, that planning with CC steer is impractical in tight environments as zero curvature is always enforced at direction switches. Therefore, the novel steering function HC steer was introduced in Sec. 2.3.2 that enforces curvature continuity as the vehicle is moving either forwards or backwards, but allows curvature discontinuities at direction switches. This leads to the fact that HC steer computes smoother paths than RS steer and outperforms CC steer in terms of path length. Especially in tight environments, HC steer

is thus an effective tool to compute directly executable paths with curvature continuity between direction switches (see Sec. 5.3.1 and Sec. 5.3.2).

Even higher degrees of smoothness can be realized by not only enforcing curvature but also curvature rate continuity. For this purpose, the two novel steering functions CCR and HCR steer were introduced in Sec. 2.4.1 and Sec. 2.4.2. In comparison to CC and HC steer, these two steering functions additionally constrain the maximum curvature acceleration along the path allowing to (1) improve the closed-loop tracking performance of the motion controller, (2) reduce the mechanical stress on the steering system, and (3) increase the comfort due to a reduction in lateral jerk. However, enforcing curvature rate continuity increases both the computation time and the path length making it more challenging to find feasible motion plans in tight environments (see Sec. 5.3.3).

Planning under uncertainty Due to the complexity of the motion planning problem, many state-of-the-art approaches neglect the uncertainties of the system. However, safety-critical systems, such as automated vehicles, have to consider these uncertainties in order to guarantee a bounded collision probability. To achieve this, two contributions were made: (1) the previously mentioned steering functions were extended to belief space in which every state of the vehicle is associated with its uncertainty arising from imperfect localization and control (see Sec. 2.5), and (2) two algorithms were proposed that allow to assess whether the collision probability of a Gaussian distributed vehicle state exceeds a user-defined threshold (see Sec. 3.2). Combining both contributions in a motion planner allows to significantly increase the robustness of the motion plans by bounding the collision probability along the computed vehicle motion (see Sec. 5.3.4). However, the additional complexity of planning in belief space requires heuristics that accelerate the planning process and improve the convergence rate.

Sampling distributions Sampling-based motion planners generally provide stronger theoretical guarantees with respect to optimality and completeness than search-based approaches. However, sampling-based planners typically suffer from slow convergence rates if samples are only drawn from a uniform distribution. Therefore, problem-specific sampling distributions are often required in order to guide the motion planner efficiently towards the goal. For this purpose, a data-driven approach was realized in Ch. 4, where a CNN is trained to predict a distribution over future vehicle poses given observations of the environment. Interfacing the CNN with the motion planner BiRRT*

allows to stabilize the time-to-first-solution to about 100 ms in the conducted experiments and additionally improves the convergence rate in comparison to uniform sampling and an A*-based heuristic (see Sec. 5.3.5). Hence, the presented approach is particularly suitable for real-time motion planning in complex tight environments.

All key contributions were not only evaluated individually, but also in combination with the sampling-based motion planner RRT* and its bidirectional extension BiRRT* (see Ch. 5). Finally, it has to be answered whether such an approach can satisfy the initially derived requirements for global motion planning. This question is addressed in Tab. 6.1, where the satisfaction of each of the eight requirements from Sec. 1.1 is qualitatively evaluated with a filled circle. The reasons for the respective choices are briefly discussed in

Table 6.1 Evaluation of the developed motion planner on the basis of the requirements from Tab. 1.1. The filled circles visualize the degree to which each requirement is satisfied.

The motion **planner** is:	The motion **plan** is:
● generic	● collision-free
◕ complete	● smooth
◕ optimal	◕ robust
◕ efficient	● true-to-contour

○ 0 % ◔ 25 % ◑ 50 % ◕ 75 % ● 100 %

the following paragraphs starting with the requirements that are completely satisfied followed by the ones that are (not yet) 100 % fulfilled.

From the broad set of scenarios that were tested in Ch. 5, it can be concluded that the developed motion planner is generic and thus capable to solve arbitrary planning problems. Furthermore, the computed motion plans are collision-free and take into account the actual shape (and not just a rough bounding box) of all objects in the scene. Smoothness is realized by enforcing curvature and curvature rate continuity during planning resulting in directly executable motion plans.

Although RRT* and BiRRT* provide theoretical guarantees with respect to completeness and optimality, both properties can only be guaranteed with an infinite number of samples. Therefore, the two respective requirements in Tab. 6.1 are only considered as partially fulfilled. Furthermore, the efficiency of these two planners (e.g. their convergence rate) highly depends on the

underlying sampling distribution. While uniform sampling has proven to be fairly inefficient, guiding the motion planner with the CNN significantly improves the planning performance. As the CNN, however, contains approximately 2.5 million parameters (resulting in a non-negligible inference time), the efficiency aspect in Tab. 6.1 is only considered as mostly satisfied. Apart from that, the motion planner currently neglects the perception uncertainty, but allows to consider both the localization and the control errors in the computation of motion plans. As the latter two are typically larger than the perception uncertainty (assuming a LiDAR sensor set), the robustness requirement in Tab. 6.1 is classified as mostly but not completely fulfilled.

Nevertheless, it has to be emphasized that the presented implementation is a powerful approach that noticeably advances the state of the art and allows to solve a large variety of challenging motion planning problems.

6.2 Outlook

Based on the previous chapters, four possible future research directions are identified below:

(1) Currently, the motion planner decomposes the planning problem into path planning (spatial dimension) and velocity profiling (temporal dimension), which is also known as path-velocity decomposition [91]. Although this approach has been widely used in the robotics community over the last decades, it is known that such a decomposition can lead to suboptimal solutions especially in (highly) dynamic environments. In order to resolve this problem, planning must be directly conducted in space and time. With the current approach, this could be achieved by extending the presented steering functions from paths to trajectories, where every vehicle state contains both spatial and temporal information. A first step into this direction could be the computation of time-optimal rather than length-optimal solutions to the steering problem in (2.1). In case of the Dubins car, for example, the time-optimal solution is trivial for a start and goal state with zero velocity (shortest path augmented with a longitudinal bang-bang input), but more challenging to compute if arbitrary velocities are allowed at both states.

(2) In order to tackle the complexity of planning simultaneously in space and time, problem-specific sampling distributions are required that also

take into account the temporal dimension of the underlying problem. While [5] shows a possible solution for local motion planning with a fixed time horizon, the duration of the global motion plan is typically not known a priori making novel techniques necessary.

(3) The steering functions in Ch. 2 satisfy the dynamic actuator limits geometrically with the maximum curvature rate $\sigma_{\max}$ and the maximum curvature acceleration $\rho_{\max}$. As these parameters are, however, functions of various variables including the steering angle and the longitudinal velocity, worst case assumptions must be made to satisfy the actuator constraints along the entire path [53, 128]. As a result, the computed solutions in Ch. 2 are longer than the shortest path that can be realized by a system with bounded steering rate and bounded steering acceleration. It remains an open question how this limitation can be resolved without significantly increasing the computation time of the steering functions.

(4) The concept of beliefprints presented in Ch. 3 is currently applied to the ego-vehicle whose footprint is precisely known in advance. In order to transfer this concept to the computation of an obstacle's beliefprint, the presented algorithms must be extended such that the perception uncertainty of the obstacle's shape can be taken into account as well.

Finally, the necessity of a benchmark suite for motion planning cannot be stressed enough here. Similar to [2], a framework is required that allows researchers to objectively compare the performance of different (global) motion planners in predefined automated driving scenarios. Along with a strong open-source culture, such a framework would very likely enable the community to make faster progress in the complex field of motion planning for automated vehicles.

A Appendix

A.1 Robot Motion

The following sections describe the robot's equations of motion on a straight line (Sec. A.1.1), on a circular arc (Sec. A.1.2), on a clothoid (Sec. A.1.3), and on a cubic spiral (Sec. A.1.4). The required partial derivatives for linearization are also listed in the respective sections below.

A.1.1 Straight Line

Given a robot with state $\check{\mathbf{x}}_k = \begin{pmatrix} \check{x}_k & \check{y}_k & \check{\theta}_k \end{pmatrix}^\mathsf{T}$ and input $\check{\mathbf{u}}_k = \begin{pmatrix} \Delta\check{s}_k & \check{\kappa}_k \end{pmatrix}^\mathsf{T}$, where $\Delta\check{s}_k$ denotes the signed arc length and $\check{\kappa}_k$ the curvature at $\begin{pmatrix} \check{x}_k & \check{y}_k \end{pmatrix}$. For $\check{\kappa}_k = 0$, the robot moves on a straight line, which can be described as

$$\check{x}_{k+1} = \check{x}_k + \Delta\check{s}_k \cos(\check{\theta}_k), \tag{A.1a}$$
$$\check{y}_{k+1} = \check{y}_k + \Delta\check{s}_k \sin(\check{\theta}_k), \tag{A.1b}$$
$$\check{\theta}_{k+1} = \check{\theta}_k + \check{\kappa}_k \Delta\check{s}_k. \tag{A.1c}$$

The corresponding partial derivatives with respect to the robot's state $\check{\mathbf{x}}_k$ are given as

$$\frac{\partial \check{x}_{k+1}}{\partial \check{x}_k} = 1, \tag{A.2a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.2b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.2c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.3a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{y}_k} = 1, \tag{A.3b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.3c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{\theta}_k} = -\Delta \check{s}_k \sin(\check{\theta}_k), \tag{A.4a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{\theta}_k} = \Delta \check{s}_k \cos(\check{\theta}_k), \tag{A.4b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\theta}_k} = 1. \tag{A.4c}$$

The partial derivatives with respect to the input $\check{\mathbf{u}}_k$ are derived as

$$\frac{\partial \check{x}_{k+1}}{\partial \Delta \check{s}_k} = \cos(\check{\theta}_k), \tag{A.5a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \Delta \check{s}_k} = \sin(\check{\theta}_k), \tag{A.5b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \Delta \check{s}_k} = \check{\kappa}_k, \tag{A.5c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{\kappa}_k} = 0, \tag{A.6a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{\kappa}_k} = 0, \tag{A.6b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\kappa}_k} = \Delta \check{s}_k. \tag{A.6c}$$

A.1.2 Circular Arc

Considering the same state $\check{\mathbf{x}}_k$ and input $\check{\mathbf{u}}_k$ as in Sec. A.1.1, the robot moves on a circular arc if $\check{\kappa}_k \neq 0$. The corresponding equations of motion are given as

$$\check{x}_{k+1} = \check{x}_k + \frac{1}{\check{\kappa}_k}(-\sin(\check{\theta}_k) + \sin(\check{\theta}_k + \Delta \check{s}_k \check{\kappa}_k)), \tag{A.7a}$$
$$\check{y}_{k+1} = \check{y}_k + \frac{1}{\check{\kappa}_k}(+\cos(\check{\theta}_k) - \cos(\check{\theta}_k + \Delta \check{s}_k \check{\kappa}_k)), \tag{A.7b}$$
$$\check{\theta}_{k+1} = \check{\theta}_k + \check{\kappa}_k \Delta \check{s}_k. \tag{A.7c}$$

The partial derivatives with respect to the robot's state $\check{\mathbf{x}}_k$ can be computed as

$$\frac{\partial \check{x}_{k+1}}{\partial \check{x}_k} = 1, \tag{A.8a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.8b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.8c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.9a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{y}_k} = 1, \tag{A.9b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.9c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{\theta}_k} = \frac{1}{\check{\kappa}_k}(-\cos(\check{\theta}_k) + \cos(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)), \tag{A.10a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{\theta}_k} = \frac{1}{\check{\kappa}_k}(-\sin(\check{\theta}_k) + \sin(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)), \tag{A.10b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\theta}_k} = 1. \tag{A.10c}$$

Deriving the equations of motion with respect to the input $\check{\mathbf{u}}_k$ leads to

$$\frac{\partial \check{x}_{k+1}}{\partial \Delta\check{s}_k} = \cos(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k), \tag{A.11a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \Delta\check{s}_k} = \sin(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k), \tag{A.11b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \Delta\check{s}_k} = \check{\kappa}_k, \tag{A.11c}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{\kappa}_k} = \frac{+\sin(\check{\theta}_k) - \sin(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)}{\check{\kappa}_k^2} + \frac{\Delta\check{s}_k\cos(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)}{\check{\kappa}_k}, \tag{A.12a}$$
$$\frac{\partial \check{y}_{k+1}}{\partial \check{\kappa}_k} = \frac{-\cos(\check{\theta}_k) + \cos(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)}{\check{\kappa}_k^2} + \frac{\Delta\check{s}_k\sin(\check{\theta}_k + \Delta\check{s}_k\check{\kappa}_k)}{\check{\kappa}_k}, \tag{A.12b}$$
$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\kappa}_k} = \Delta\check{s}_k. \tag{A.12c}$$

A.1.3 Clothoid

Moving on a clothoid requires to expand the state of the robot to $\check{\mathbf{x}}_k = \begin{pmatrix}\check{x}_k & \check{y}_k & \check{\theta}_k & \check{\kappa}_k\end{pmatrix}^\mathsf{T}$. The input changes to $\check{\mathbf{u}}_k = \begin{pmatrix}\Delta\check{s}_k & \check{\sigma}_k\end{pmatrix}^\mathsf{T}$, where $\check{\sigma}_k$ describes the curvature rate of the clothoid. The equations of motion can be derived as

$$\begin{aligned}\check{x}_{k+1} = \check{x}_k + \sqrt{\frac{\pi}{|\check{\sigma}_k|}}(&+ \operatorname{sgn}(\Delta\check{s}_k)\cos(k_1)(C_f(k_2) - C_f(k_3)) \\ &- \operatorname{sgn}(\check{\sigma}_k)\sin(k_1)(S_f(k_2) - S_f(k_3))),\end{aligned} \tag{A.13a}$$

$$\begin{aligned}\check{y}_{k+1} = \check{y}_k + \sqrt{\frac{\pi}{|\check{\sigma}_k|}}(&+ \operatorname{sgn}(\Delta\check{s}_k)\sin(k_1)(C_f(k_2) - C_f(k_3)) \\ &+ \operatorname{sgn}(\check{\sigma}_k)\cos(k_1)(S_f(k_2) - S_f(k_3))),\end{aligned} \tag{A.13b}$$

$$\check{\theta}_{k+1} = \check{\theta}_k + \check{\kappa}_k\Delta\check{s}_k + \frac{1}{2}\operatorname{sgn}(\Delta\check{s}_k)\check{\sigma}_k\Delta\check{s}_k^2, \tag{A.13c}$$

$$\check{\kappa}_{k+1} = \check{\kappa}_k + \check{\sigma}_k\left|\Delta\check{s}_k\right|, \tag{A.13d}$$

where the Fresnel integrals [1] are given as

$$C_f(t) = \int_0^t \cos(\frac{\pi}{2}u^2)\ \mathrm{d}u, \tag{A.14a}$$

$$S_f(t) = \int_0^t \sin(\frac{\pi}{2}u^2)\ \mathrm{d}u. \tag{A.14b}$$

Note that (A.14) can be efficiently approximated using e.g. Chebyshev polynomials [122]. The arguments of the Fresnel integrals k_1, k_2, and k_3 in (A.13) are described by

$$k_1 = \check{\theta}_k - \operatorname{sgn}(\Delta\check{s}_k)\frac{\check{\kappa}_k^2}{2\check{\sigma}_k}, \tag{A.15a}$$

$$k_2 = \operatorname{sgn}(\check{\sigma}_k)\frac{\check{\kappa}_k + \check{\sigma}_k\left|\Delta\check{s}_k\right|}{\sqrt{\pi\left|\check{\sigma}_k\right|}}, \tag{A.15b}$$

$$k_3 = \operatorname{sgn}(\check{\sigma}_k)\frac{\check{\kappa}_k}{\sqrt{\pi\left|\check{\sigma}_k\right|}}. \tag{A.15c}$$

The partial derivatives of (A.13) with respect to the state $\check{\mathbf{x}}_k$ can now be calculated according to

$$\frac{\partial\check{x}_{k+1}}{\partial\check{x}_k} = 1, \tag{A.16a}$$

$$\frac{\partial \check{y}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.16b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.16c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \check{x}_k} = 0, \tag{A.16d}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.17a}$$

$$\frac{\partial \check{y}_{k+1}}{\partial \check{y}_k} = 1, \tag{A.17b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.17c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \check{y}_k} = 0, \tag{A.17d}$$

$$\tag{A.17e}$$

$$\begin{aligned}\frac{\partial \check{x}_{k+1}}{\partial \check{\theta}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}(&- \operatorname{sgn}(\Delta\check{s}_k)\sin(k_1)(C_f(k_2) - C_f(k_3)) \\ &- \operatorname{sgn}(\check{\sigma}_k)\cos(k_1)(S_f(k_2) - S_f(k_3))),\end{aligned} \tag{A.18a}$$

$$\begin{aligned}\frac{\partial \check{y}_{k+1}}{\partial \check{\theta}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}(&+ \operatorname{sgn}(\Delta\check{s}_k)\cos(k_1)(C_f(k_2) - C_f(k_3)) \\ &- \operatorname{sgn}(\check{\sigma}_k)\sin(k_1)(S_f(k_2) - S_f(k_3))),\end{aligned} \tag{A.18b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\theta}_k} = 1, \tag{A.18c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \check{\theta}_k} = 0, \tag{A.18d}$$

$$\begin{aligned}\frac{\partial \check{x}_{k+1}}{\partial \check{\kappa}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}\frac{\check{\kappa}_k}{|\check{\sigma}_k|}(&+ \operatorname{sgn}(\check{\sigma}_k)\sin(k_1)(C_f(k_2) - C_f(k_3)) \\ &+ \operatorname{sgn}(\Delta\check{s}_k)\cos(k_1)(S_f(k_2) - S_f(k_3))) \\ &+ \operatorname{sgn}(\Delta\check{s}_k)\frac{1}{\check{\sigma}_k}(\cos(k_4) - \cos(k_5)),\end{aligned} \tag{A.19a}$$

$$\frac{\partial \check{y}_{k+1}}{\partial \check{\kappa}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}\frac{\check{\kappa}_k}{|\check{\sigma}_k|}(-\operatorname{sgn}(\check{\sigma}_k)\cos(k_1)(C_f(k_2)-C_f(k_3)) + \operatorname{sgn}(\Delta\check{s}_k)\sin(k_1)(S_f(k_2)-S_f(k_3))) + \operatorname{sgn}(\Delta\check{s}_k)\frac{1}{\check{\sigma}_k}(\sin(k_4)-\sin(k_5)), \tag{A.19b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\kappa}_k} = \Delta\check{s}_k, \tag{A.19c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \check{\kappa}_k} = 1, \tag{A.19d}$$

where k_4 and k_5 are given as

$$k_4 = k_1 + \operatorname{sgn}(\Delta\check{s}_k)\operatorname{sgn}(\check{\sigma}_k)\frac{\pi}{2}k_2^2, \tag{A.20a}$$

$$k_5 = k_1 + \operatorname{sgn}(\Delta\check{s}_k)\operatorname{sgn}(\check{\sigma}_k)\frac{\pi}{2}k_3^2. \tag{A.20b}$$

The derivation of the equations of motion in (A.13) with respect to the input $\check{\mathbf{u}}_k$ results in

$$\frac{\partial \check{x}_{k+1}}{\partial \Delta\check{s}_k} = \cos(k_4), \tag{A.21a}$$

$$\frac{\partial \check{y}_{k+1}}{\partial \Delta\check{s}_k} = \sin(k_4), \tag{A.21b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \Delta\check{s}_k} = \check{\kappa}_k + \check{\sigma}_k\left|\Delta\check{s}_k\right|, \tag{A.21c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \Delta\check{s}_k} = \operatorname{sgn}(\Delta\check{s}_k)\check{\sigma}_k, \tag{A.21d}$$

$$\frac{\partial \check{x}_{k+1}}{\partial \check{\sigma}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}(-\operatorname{sgn}(\Delta\check{s}_k)(\frac{1}{2\check{\sigma}_k}\cos(k_1)+k_6\sin(k_1))(C_f(k_2)-C_f(k_3)) + \operatorname{sgn}(\check{\sigma}_k)(\frac{1}{2\check{\sigma}_k}\sin(k_1)-k_6\cos(k_1))(S_f(k_2)-S_f(k_3)) + \operatorname{sgn}(\Delta\check{s}_k)(k_7\cos(k_4)-k_8\cos(k_5))), \tag{A.22a}$$

$$\frac{\partial \check{y}_{k+1}}{\partial \check{\sigma}_k} = \sqrt{\frac{\pi}{|\check{\sigma}_k|}}\Big(- \operatorname{sgn}(\Delta\check{s}_k)\Big(\frac{1}{2\check{\sigma}_k}\sin(k_1) - k_6\cos(k_1)\Big)\big(C_f(k_2) - C_f(k_3)\big) - \operatorname{sgn}(\check{\sigma}_k)\Big(\frac{1}{2\check{\sigma}_k}\cos(k_1) + k_6\sin(k_1)\Big)\big(S_f(k_2) - S_f(k_3)\big) + \operatorname{sgn}(\Delta\check{s}_k)(k_7\sin(k_4) - k_8\sin(k_5))\Big), \tag{A.22b}$$

$$\frac{\partial \check{\theta}_{k+1}}{\partial \check{\sigma}_k} = \frac{1}{2}\operatorname{sgn}(\Delta\check{s}_k)\Delta\check{s}_k^2, \tag{A.22c}$$

$$\frac{\partial \check{\kappa}_{k+1}}{\partial \check{\sigma}_k} = |\Delta\check{s}_k|\,, \tag{A.22d}$$

where k_6, k_7, and k_8 are given as

$$k_6 = \frac{\partial k_1}{\partial \check{\sigma}_k} = \operatorname{sgn}(\Delta\check{s}_k)\frac{\check{\kappa}_k^2}{2\check{\sigma}_k^2}, \tag{A.23a}$$

$$k_7 = \frac{\partial k_2}{\partial \check{\sigma}_k} = -\frac{1}{2\sqrt{\pi\,|\check{\sigma}_k|}}\left(\frac{\check{\kappa}_k}{|\check{\sigma}_k|} - \operatorname{sgn}(\check{\sigma}_k)\,|\Delta\check{s}_k|\right), \tag{A.23b}$$

$$k_8 = \frac{\partial k_3}{\partial \check{\sigma}_k} = -\frac{1}{2\sqrt{\pi\,|\check{\sigma}_k|}}\frac{\check{\kappa}_k}{|\check{\sigma}_k|}. \tag{A.23c}$$

A.1.4 Cubic Spiral

The robot's state on a cubic spiral is described by $\check{\mathbf{x}}_k = \begin{pmatrix}\check{x}_k & \check{y}_k & \check{\theta}_k & \check{\kappa}_k & \check{\sigma}_k\end{pmatrix}^\mathsf{T}$. Its input is given by $\check{\mathbf{u}}_k = \begin{pmatrix}\Delta\check{s}_k & \check{\rho}_k\end{pmatrix}^\mathsf{T}$, where $\check{\rho}_k$ denotes the second derivative of curvature at the current position $\begin{pmatrix}\check{x}_k & \check{y}_k\end{pmatrix}$. The equations of motion can be derived as

$$\check{x}_{k+1} = \check{x}_k + \int_0^{\Delta\check{s}_k} \cos(\check{\theta}_{k+1})\ \mathrm{d}s, \tag{A.24a}$$

$$\check{y}_{k+1} = \check{y}_k + \int_0^{\Delta\check{s}_k} \sin(\check{\theta}_{k+1})\ \mathrm{d}s, \tag{A.24b}$$

$$\check{\theta}_{k+1} = \check{\theta}_k + \check{\kappa}_k\Delta\check{s}_k + \frac{1}{2}\operatorname{sgn}(\Delta\check{s}_k)\check{\sigma}_k\Delta\check{s}_k^2 + \frac{1}{6}\check{\rho}_k\Delta\check{s}_k^3, \tag{A.24c}$$

$$\check{\kappa}_{k+1} = \check{\kappa}_k + \check{\sigma}_k\,|\Delta\check{s}_k| + \frac{1}{2}\check{\rho}_k\Delta\check{s}_k^2, \tag{A.24d}$$

$$\check{\sigma}_{k+1} = \check{\sigma}_k + \check{\rho}_k\,|\Delta\check{s}_k|\,. \tag{A.24e}$$

Efficient techniques, such as Gauss-Legendre quadrature [1], exist in order to evaluate the integrals in (A.24a)–(A.24b). The partial derivatives of (A.24) with respect to $\check{\mathbf{x}}_k$ and $\check{\mathbf{u}}_k$ are omitted here for brevity.

A.2 Hybrid Curvature Candidate Paths

This section details the computation of the candidate paths for HC^{00}-RS steer on the basis of the families given in Tab. 2.3. The focus lies on the illustration of the tangency conditions as well as on the existence conditions of every family. The notation is adopted from Sec. 2.3.2, where $i \in \{1, 2, 3, 4\}$ and $j \in \{5, 6, 7, 8\}$ correspond to one of the four circles at the start and at the goal state, respectively (see Fig. 2.19). Furthermore, the variable d is used below to describe the Euclidean distance between the centers of the visualized start and goal circles. Notice that an implementation of the candidate paths presented in this section can be found in [208].

A.2.1 Family CSC

The family CSC has to be divided into the two subfamilies $CeSC$ and $CiSC$. The difference lies in the position of the tangent, which once runs parallel to the line connecting both circle centers (external tangent) and once crosses that line (internal tangent).

A.2.1.1 $CeSC$

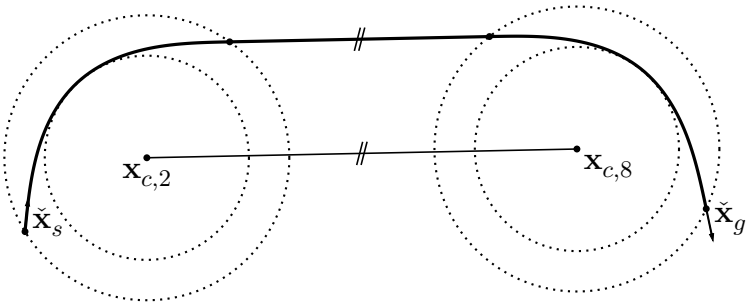


Figure A.1 $CeSC$ candidate path.

The existence conditions of the family $CeSC$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.25a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.25b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 2r \sin(\mu). \tag{A.25c}$$

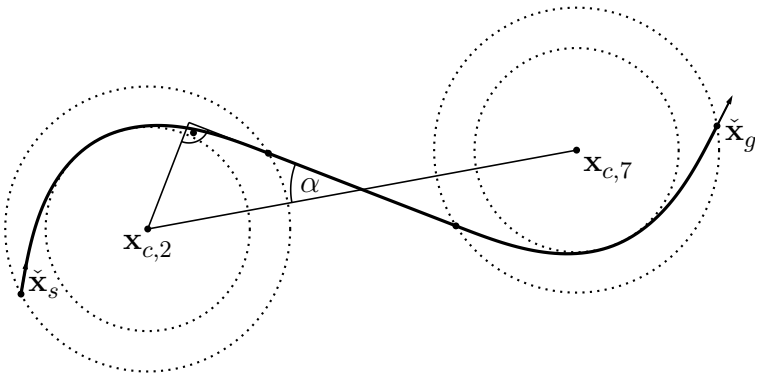


Figure A.2 $CiSC$ candidate path.

A.2.1.2 $CiSC$

The angle α in Fig. A.2 is obtained by

$$\alpha = \arcsin\left(\frac{r\cos(\mu)}{d/2}\right), \tag{A.26}$$

and the existence conditions of the family $CiSC$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.27a}$$
$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.27b}$$
$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 2r. \tag{A.27c}$$

A.2.2 Family CCC

The height h in Fig. A.3 is computed using the Pythagorean theorem as

$$h = \sqrt{4r^2 - d^2/4}, \tag{A.28}$$

and the existence conditions of the family CCC can be derived as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.29a}$$
$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.29b}$$
$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\leq} 4r. \tag{A.29c}$$

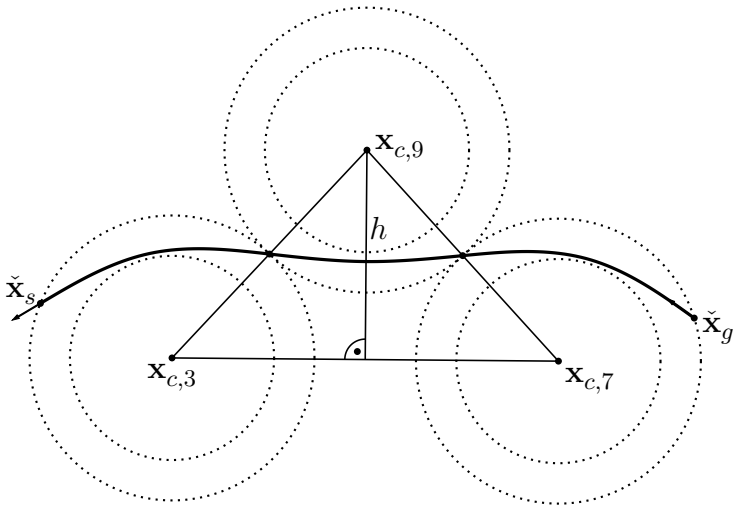


Figure A.3 CCC candidate path.

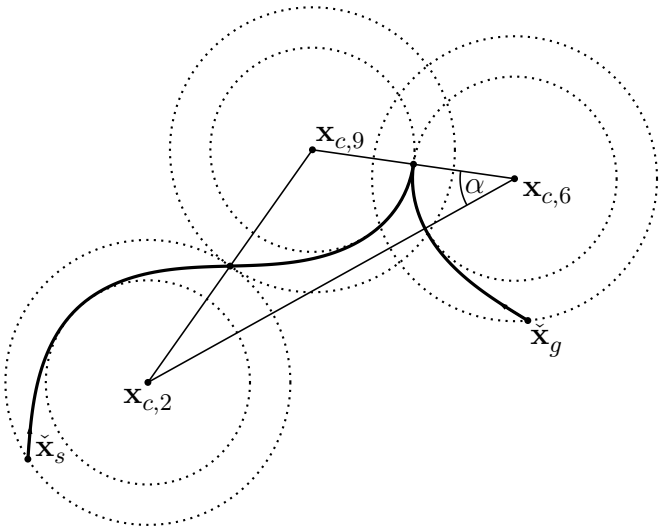


Figure A.4 $CC|C$ candidate path.

A.2.3 Family $CC|C$

The angle α in Fig. A.4 is obtained using the law of cosines by

$$\alpha = \arccos\left(\frac{4(\kappa_{\max}^{-2} - r^2) + d^2}{4\kappa_{\max}^{-1} d}\right), \tag{A.30}$$

and the existence conditions of the family $CC|C$ are given as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.31a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.31b}$$

$$2(r - \kappa_{\max}^{-1}) \overset{!}{\leq} \left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\leq} 2(r + \kappa_{\max}^{-1}). \tag{A.31c}$$

A.2.4 Family $C|CC$

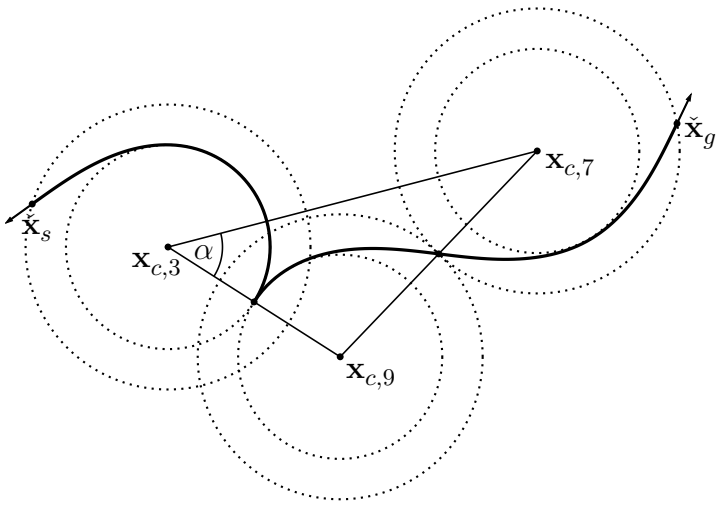


Figure A.5 $C|CC$ candidate path.

The angle α in Fig. A.5 is computed according to (A.30) and the existence conditions of the family $C|CC$ are identical to the ones in (A.31).

A.2.5 Family $C|C|C$

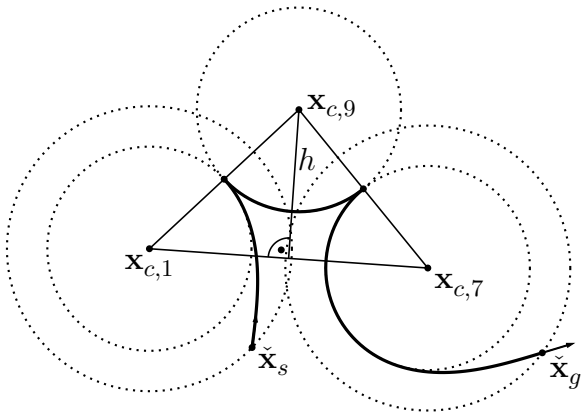


Figure A.6 $C|C|C$ candidate path.

The height h in Fig. A.6 is computed using the Pythagorean theorem as

$$h = \sqrt{4\kappa_{\max}^{-2} - d^2/4}, \tag{A.32}$$

and the existence conditions of the family $C|C|C$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.33a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.33b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\leq} 4\kappa_{\max}^{-1}. \tag{A.33c}$$

A.2.6 Family $CSC|C$

The family $CSC|C$ has to be divided into the two subfamilies $CeSC|C$ and $CiSC|C$ similar to CSC in Sec. A.2.1.

A.2.6.1 $CeSC|C$

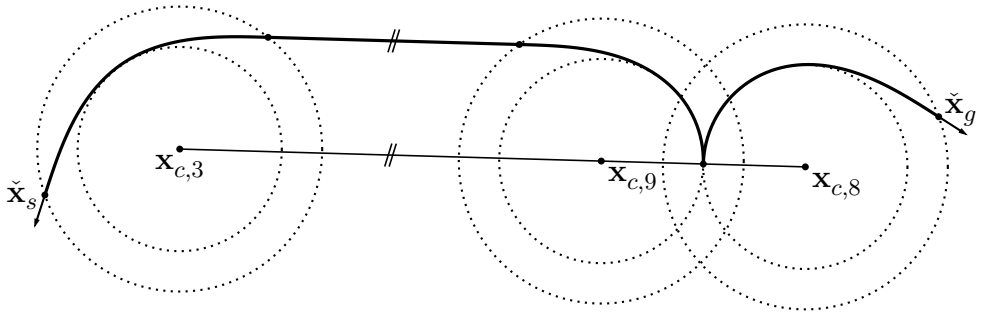


Figure A.7 $CeSC|C$ candidate path.

The existence conditions of the family $CeSC|C$ can be derived as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.34a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.34b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 2(\kappa_{\max}^{-1} + r\sin(\mu)). \tag{A.34c}$$

A.2.6.2 $CiSC|C$

The angle α in Fig. A.8 is obtained by

$$\alpha = \arcsin\left(\frac{2r\cos(\mu)}{d}\right), \tag{A.35}$$

and the height h by

$$h = 2\kappa_{\max}^{-1}\sin(\alpha). \tag{A.36}$$

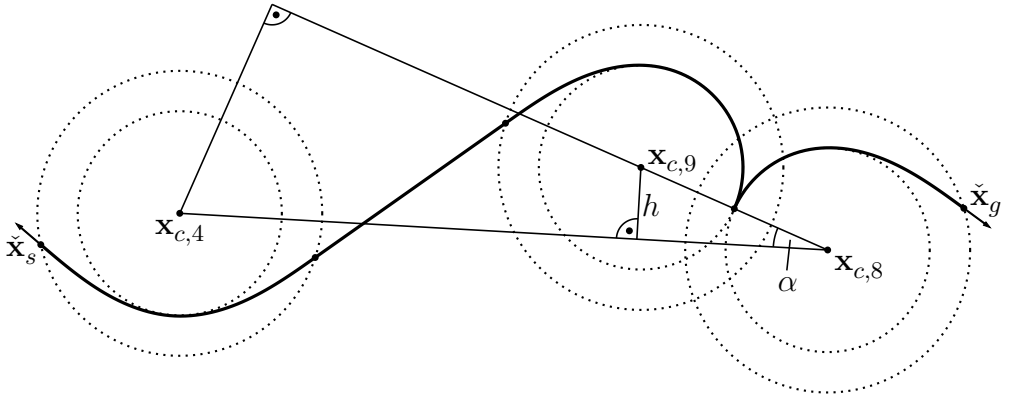


Figure A.8 $CiSC|C$ candidate path.

The existence conditions of the family $CiSC|C$ can be derived as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.37a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.37b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} \sqrt{(2r\cos(\mu))^2 + (2(r\sin(\mu) + \kappa_{\max}^{-1}))^2}. \tag{A.37c}$$

A.2.7 Family $C|CSC$

The family $C|CSC$ has to be divided into the two subfamilies $C|CeSC$ and $C|CiSC$ similar to CSC in Sec. A.2.1.

A.2.7.1 $C|CeSC$

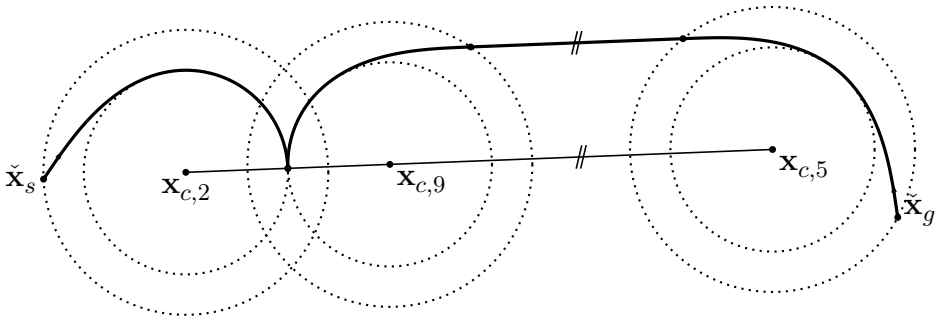


Figure A.9 $C|CeSC$ candidate path.

The existence conditions of the family $C|CeSC$ are identical to the ones in (A.34).

A.2.7.2 $C|CiSC$

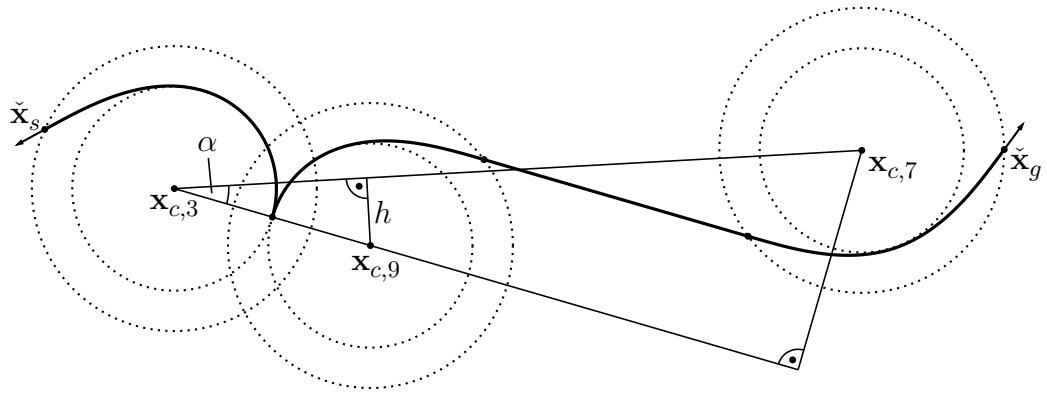


Figure A.10 $C|CiSC$ candidate path.

The variables α and h in Fig. A.10 are computed according to (A.35)–(A.36), and the existence conditions of the family $C|CiSC$ are identical to the ones in (A.37).

A.2.8 Family $CC|CC$

The family $CC|CC$ also has to be divided into two cases as illustrated below.

A.2.8.1 Case 1

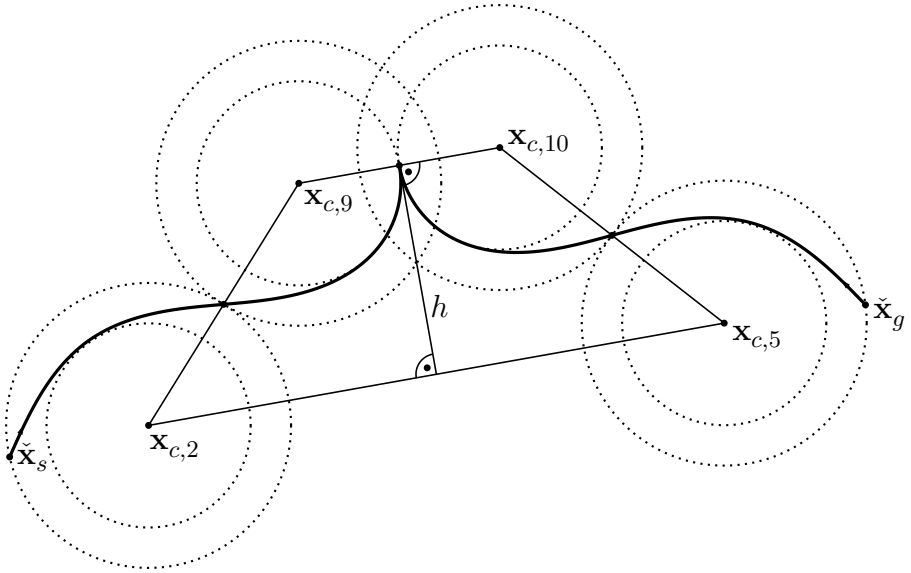


Figure A.11 $CC|CC$ candidate path (first case).

The height h in Fig. A.11 can be computed by

$$h = \sqrt{4r^2 - (d/2 - \kappa_{\max}^{-1})^2}, \tag{A.38}$$

and the corresponding existence conditions are given as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.39a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.39b}$$

$$\left\| \mathbf{x}_{c,j} - \mathbf{x}_{c,i} \right\|_2 \overset{!}{\leq} 4r + 2\kappa_{\max}^{-1}. \tag{A.39c}$$

A.2.8.2 Case 2

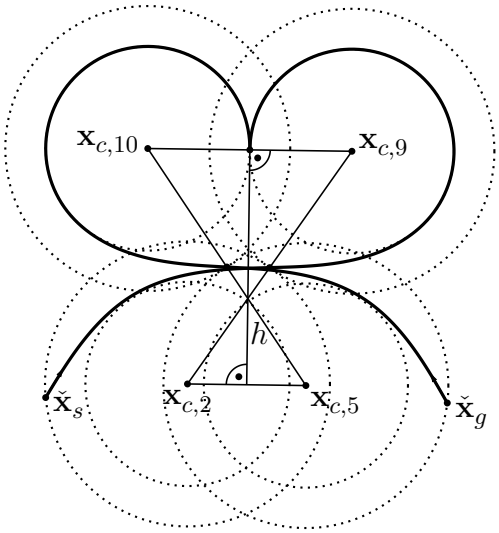


Figure A.12 $CC|CC$ candidate path (second case).

The height h in Fig. A.12, which goes all the way from the line connecting $\mathbf{x}_{c,2}$ and $\mathbf{x}_{c,5}$ to the line connecting $\mathbf{x}_{c,9}$ and $\mathbf{x}_{c,10}$, can be derived as

$$h = \sqrt{4r^2 - (d/2 + \kappa_{\max}^{-1})^2}, \tag{A.40}$$

and the corresponding existence conditions are given by

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.41a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.41b}$$

$$\left\| \mathbf{x}_{c,j} - \mathbf{x}_{c,i} \right\|_2 \overset{!}{\leq} 4r - 2\kappa_{\max}^{-1}. \tag{A.41c}$$

A.2.9 Family $C|CC|C$

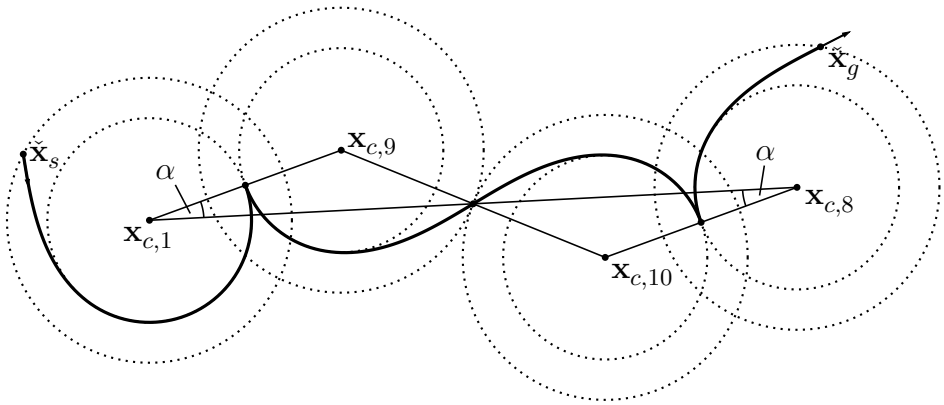


Figure A.13 $C|CC|C$ candidate path.

The angle α in Fig. A.13 is obtained using the law of cosines by

$$\alpha = \arccos\left(\frac{-r^2 + 4\kappa_{\max}^{-2} + d^2/4}{2\kappa_{\max}^{-1} d}\right), \tag{A.42}$$

and the existence conditions of the family $C|CC|C$ can be derived as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.43a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.43b}$$

$$-2r + 4\kappa_{\max}^{-1} \overset{!}{\leq} \left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\leq} 2r + 4\kappa_{\max}^{-1}. \tag{A.43c}$$

A.2.10 Family $C|CSC|C$

The family $C|CSC|C$ has to be divided into the two subfamilies $C|CeSC|C$ and $C|CiSC|C$ similar to CSC in Sec. A.2.1.

A.2.10.1 $C|CeSC|C$

The existence conditions of the family $C|CeSC|C$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.44a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.44b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 2r\sin(\mu) + 4\kappa_{\max}^{-1}. \tag{A.44c}$$

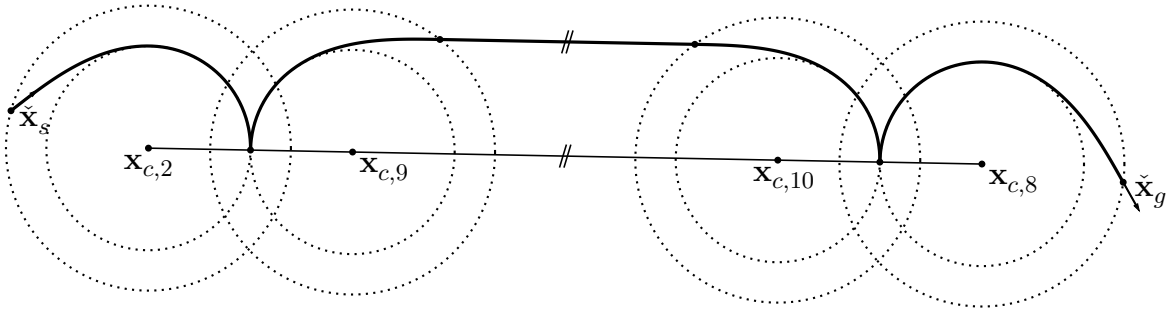


Figure A.14 $C|CeSC|C$ candidate path.

A.2.10.2 $C|CiSC|C$

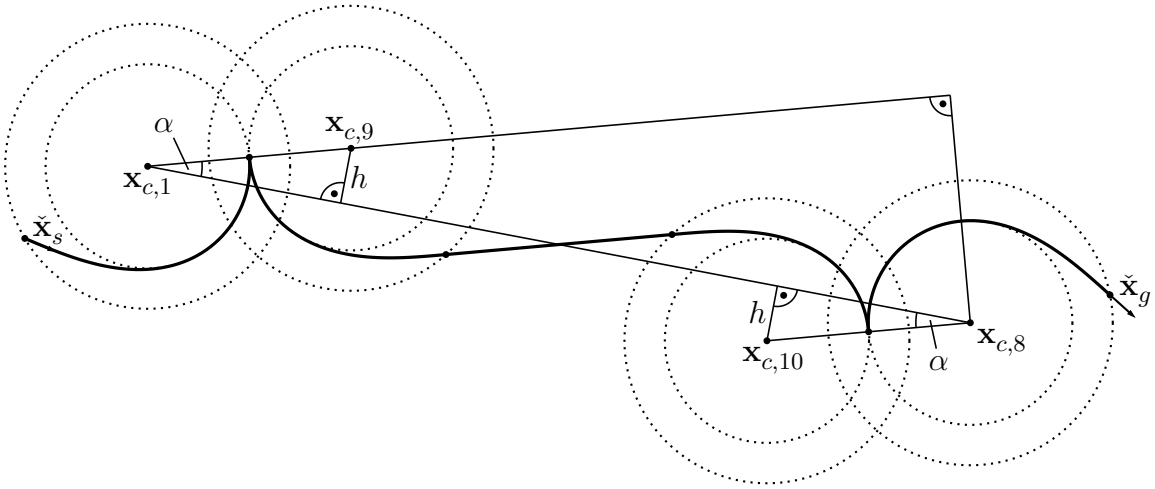


Figure A.15 $C|CiSC|C$ candidate path.

The variables α and h in Fig. A.15 are computed according to (A.35)–(A.36), and the existence conditions of the family $C|CiSC|C$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.45a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.45b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} \sqrt{\left(2r\cos(\mu)\right)^2 + \left(2r\sin(\mu) + 4\kappa_{\max}^{-1}\right)^2}. \tag{A.45c}$$

A.2.11 Family $CS|C$

The family $CS|C$ has to be divided into the two subfamilies $CeS|C$ and $CiS|C$ similar to CSC in Sec. A.2.1.

A.2.11.1 $CeS|C$

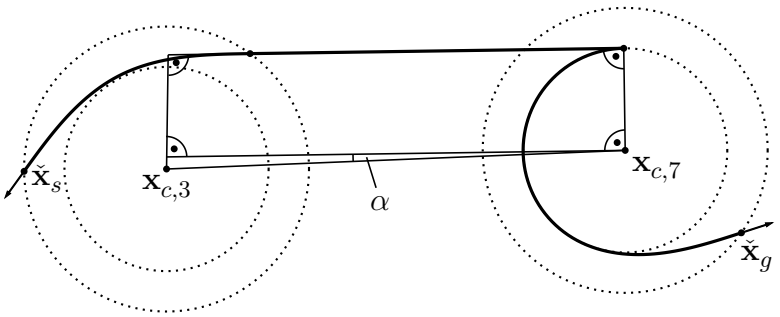


Figure A.16 $CeS|C$ candidate path.

The angle α in Fig. A.16 is obtained by

$$\alpha = \arcsin\left(\frac{r\cos(\mu) - \kappa_{\max}^{-1}}{d}\right), \tag{A.46}$$

and the existence conditions of the family $CeS|C$ can be derived as

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.47a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.47b}$$

$$d \overset{!}{\geq} \sqrt{(r\sin(\mu))^2 + (r\cos(\mu) - \kappa_{\max}^{-1})^2}. \tag{A.47c}$$

A.2.11.2 $CiS|C$

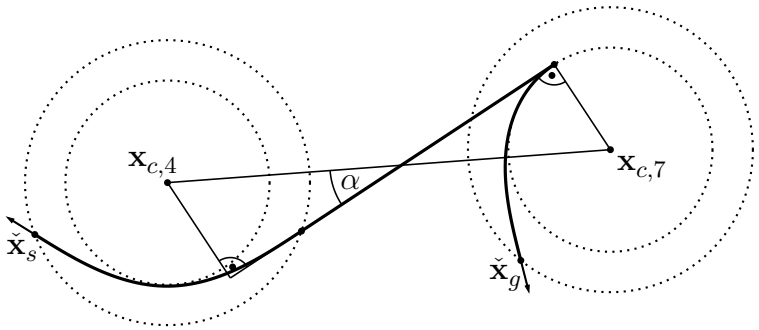


Figure A.17 $CiS|C$ candidate path.

The angle α in Fig. A.17 is given by

$$\alpha = \arcsin\left(\frac{r\cos(\mu) + \kappa_{\max}^{-1}}{d}\right), \tag{A.48}$$

and the existence conditions of the family $CiS|C$ by

$$g_i \cdot g_j \overset{!}{=} +1, \tag{A.49a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.49b}$$

$$d \overset{!}{\geq} \sqrt{(r\sin(\mu))^2 + (r\cos(\mu) + \kappa_{\max}^{-1})^2}. \tag{A.49c}$$

A.2.12 Family $C|SC$

The family $C|SC$ has to be divided into the two subfamilies $C|eSC$ and $C|iSC$ similar to CSC in Sec. A.2.1.

A.2.12.1 $C|eSC$

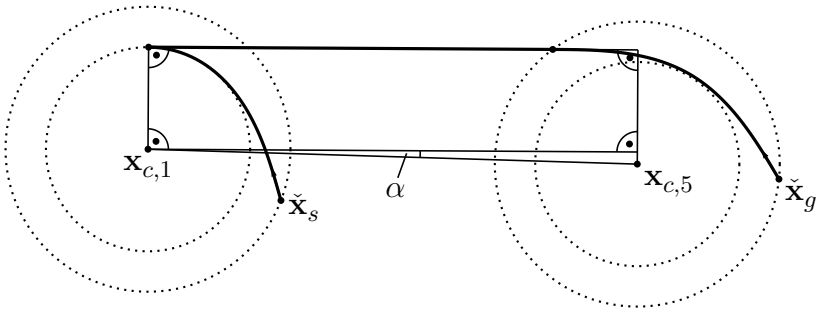


Figure A.18 $C|eSC$ candidate path.

The variable α in Fig. A.18 is computed according to (A.46), and the existence conditions of the family $C|eSC$ are identical to the ones in (A.47).

A.2.12.2 $C|iSC$

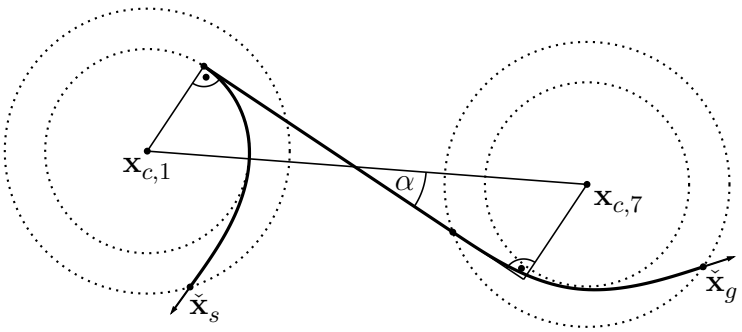


Figure A.19 $C|iSC$ candidate path.

The variable α in Fig. A.19 is computed according to (A.48), and the existence conditions of the family $C|iSC$ are identical to the ones in (A.49).

A.2.13 Family $C|S|C$

The family $C|S|C$ has to be divided into the two subfamilies $C|eS|C$ and $C|iS|C$ similar to CSC in Sec. A.2.1.

A.2.13.1 $C|eS|C$

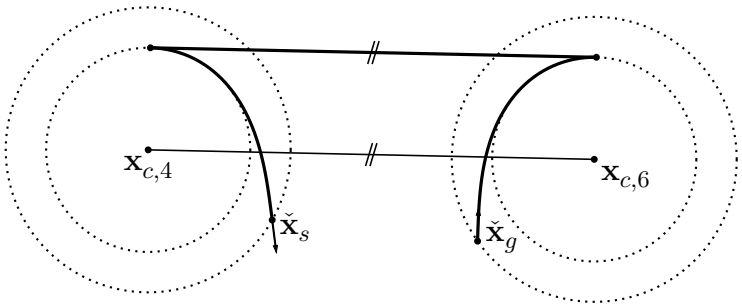


Figure A.20 $C|eS|C$ candidate path.

The existence conditions of the family $C|eS|C$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.50a}$$

$$t_i \cdot t_j \overset{!}{=} +1, \tag{A.50b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{\geq} 0. \tag{A.50c}$$

A.2.13.2 $C|iS|C$

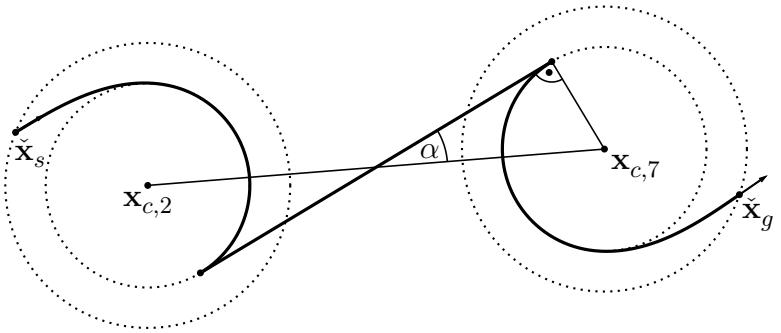


Figure A.21 $C|iS|C$ candidate path.

The angle α in Fig. A.21 is obtained by

$$\alpha = \arcsin\left(\frac{\kappa_{\max}^{-1}}{d/2}\right), \tag{A.51}$$

and the existence conditions of the family $C|iS|C$ are given as

$$g_i \cdot g_j \overset{!}{=} -1, \tag{A.52a}$$

$$t_i \cdot t_j \overset{!}{=} -1, \tag{A.52b}$$

$$\left\|\mathbf{x}_{c,j} - \mathbf{x}_{c,i}\right\|_2 \overset{!}{>} 2\kappa_{\max}^{-1}. \tag{A.52c}$$

Bibliography

[1] **M. Abramowitz and I. Stegun**. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. Applied Mathematics Series. U.S. Dept. of Commerce, National Bureau of Standards, 1972.

[2] **M. Althoff, M. Koschi, and S. Manzinger**. *CommonRoad: Composable Benchmarks for Motion Planning on Roads*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2017, pp. 719–726.

[3] **B. Axelrod, L. P. Kaelbling, and T. Lozano-Pérez**. *Provably Safe Robot Navigation with Obstacle Uncertainty*. In: *Robotics: Science and Systems XIII*. 2017.

[4] **V. Badrinarayanan, A. Kendall, and R. Cipolla**. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. In: *arXiv preprint arXiv:1511.00561* (2015).

[5] **M. Bansal, A. Krizhevsky, and A. Ogale**. *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst*. In: *arXiv preprint arXiv:1812.03079* (2018).

[6] **R. H. Bartels, J. C. Beatty, and B. A. Barsky**. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1995.

[7] **U. Baumann et al.** *Predicting Ego-Vehicle Paths from Environmental Observations with a Deep Neural Network*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2018, pp. 4709–4716.

[8] **J. van den Berg et al.** *LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information*. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 895–913.

[9] **D. Bertsimas and J. N. Tsitsiklis**. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[10] **C. G. L. Bianco, A. Piazzi, and M. Romano**. *Smooth Motion Generation for Unicycle Mobile Robots Via Dynamic Path Inversion*. In: *IEEE Transactions on Robotics* 20.5 (2004), pp. 884–891.

[11] **M. Bilodeau and D. Brenner**. *Theory of Multivariate Statistics*. Springer, 1999.

[12] **L. Blackmore, H. Li, and B. Williams**. *A Probabilistic Approach to Optimal Robust Path Planning with Obstacles*. In: *American Control Conference*. IEEE. 2006, pp. 2831–2837.

[13] **A. Blake et al.** *Efficient Computation of Collision Probabilities for Safe Motion Planning*. In: *arXiv preprint arXiv:1804.05384* (2018).

[14] **J.-D. Boissonnat, A. Cerezo, and J. Leblond**. *A note on shortest paths in the plane subject to a constraint on the derivative of the curvature*. Tech. rep. 2160. Institut National de Recherche en Informatique et en Automatique, 1994.

[15] **M. Bojarski et al.** *End to End Learning for Self-Driving Cars*. In: *arXiv preprint arXiv:1604.07316* (2016).

[16] **M. Bolle et al.** *Early level 4/5 automation by restriction of the use-case*. In: *17. Internationales Stuttgarter Symposium*. 2017, pp. 531–545.

[17] **S. D. Bopardikar et al.** *Robust belief space planning under intermittent sensing via a maximum eigenvalue-based bound*. In: *The International Journal of Robotics Research* 35.13 (2016), pp. 1609–1626.

[18] *Bosch and Daimler: San José targeted to become pilot city for an automated on-demand ride-hailing service*. Robert Bosch GmbH. 2018. URL: https://www.bosch-presse.de/pressportal/de/en/. (visited on 2019/03/12).

[19] **H.-H. Braess**. *The Intelligent Vehicle on the Intelligent Road – What did PROMETHEUS Achieve?* In: *5. Internationales Stuttgarter Symposium*. 2003, pp. 608–627.

[20] **J. E. Bresenham**. *Algorithm for computer control of a digital plotter*. In: *IBM Systems Journal* 4.1 (1965), pp. 25–30.

[21] **R. W. Brockett**. *Control Theory and Singular Riemannian Geometry*. In: *New Directions in Applied Mathematics*. Springer, 1982, pp. 11–27.

[22] **A. Broggi et al.** *The ARGO Autonomous Vehicle's Vision and Control Systems*. In: *International Journal of Intelligent Control and Systems* 3.4 (1999), pp. 409–441.

[23] **A. Bry and N. Roy**. *Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 723–730.

[24] **M. Buehler, K. Iagnemma, and S. Singh**. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009.

[25] **G. Calafiore and M. C. Campi**. *Sampled Convex Programs and Probabilistically Robust Design*. In: *Probabilistic and Randomized Methods for Design under Uncertainty*. Springer, 2006, pp. 161–188.

[26] **L. Caltagirone et al.** *LIDAR-based Driving Path Generation Using Fully Convolutional Neural Networks*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2017, pp. 573–578.

[27] **J. Canny**. *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[28] **C. Chen**. *Motion Planning for Nonholonomic Vehicles with Space Exploration Guided Heuristic Search*. PhD thesis. Technische Universität München, 2016.

[29] **C. Chen**. *Optimal Path for a Car-like Robot to Reach a Given Straight Line*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2018, pp. 2270–2276.

[30] **C. Chen et al.** *Path Planning with Orientation-Aware Space Exploration Guided Heuristic Search for Autonomous Parking and Maneuvering*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2015, pp. 1148–1153.

[31] **C. Chen et al.** *Motion Planning under Perception and Control Uncertainties with Space Exploration Guided Heuristic Search*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2017, pp. 712–718.

[32] **H. Chernoff**. *Locally Optimal Designs for Estimating Parameters*. In: *The Annals of Mathematical Statistics* 24.4 (1953), pp. 586–602.

[33] **M. Chester et al.** *Parking Infrastructure: A Constraint on or Opportunity for Urban Redevelopment? A Study of Los Angeles County Parking Supply and Growth*. In: *Journal of the American Planning Association* 81.4 (2015), pp. 268–286.

[34] **S. Choudhury et al.** *Data-driven planning via imitation learning*. In: *The International Journal of Robotics Research* 37.13-14 (2018), pp. 1632–1672.

[35] **T. Christopher**. *Analysis of Dynamic Scenes: Application to Driving Assistance*. PhD thesis. Institut National Polytechnique de Grenoble, 2009.

[36] **F. Codevilla et al.** *End-to-end Driving via Conditional Imitation Learning*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2018, pp. 4693–4700.

[37] **H. Coxeter**. *Regular Polytopes*. Courier Corporation, 2012.

[38] **E. Degtiariova-Kostova and V. Kostov**. *Irregularity of Optimal Trajectories in a Control Problem for a Car-like Robot*. Tech. rep. 3411. Institut National de Recherche en Informatique et en Automatique, 1998.

[39] **F. Dillen**. *The Classification of Hypersurfaces of a Euclidean Space with Parallel Higher Order Fundamental Form*. In: *Mathematische Zeitschrift* 203.1 (1990), pp. 635–643.

[40] **D. Dolgov et al.** *Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments*. In: *The International Journal of Robotics Research* 29.5 (2010), pp. 485–501.

[41] *Driverless Auto, Guided By Radio, Navigates Street*. The Washington Herald. Aug. 6, 1921, p. 5.

[42] **N. E. Du Toit and J. W. Burdick**. *Probabilistic Collision Checking With Chance Constraints*. In: *IEEE Transactions on Robotics* 27.4 (2011), pp. 809–815.

[43] **L. E. Dubins**. *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents*. In: *American Journal of Mathematics* 79.3 (1957), pp. 497–516.

[44] **A. Elfes**. *Using Occupancy Grids for Mobile Robot Perception and Navigation*. In: *Computer* 22.6 (1989), pp. 46–57.

[45] **C. Ericson**. *Real-Time Collision Detection*. CRC Press, 2004.

[46] **D. Fassbender, B. C. Heinrich, and H.-J. Wuensche**. *Motion Planning for Autonomous Vehicles in Highly Constrained Urban Environments*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2016, pp. 4708–4713.

[47] **D. Ferguson and M. Likhachev**. *Efficiently Using Cost Maps For Planning Complex Maneuvers*. In: *IEEE International Conference on Robotics and Automation, Workshop on Planning with Cost Maps*. IEEE. 2008.

[48] **D. Ferguson, M. Likhachev, and A. Stentz**. *A Guide to Heuristic-based Path Planning*. In: *International Conference on Automated Planning and Scheduling, Workshop on Planning under Uncertainty for Autonomous Systems*. 2005.

[49] **C. Fernandes, L. Gurvits, and Z. X. Li**. *A Variational Approach to Optimal Nonholonomic Motion Planning*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 1991, pp. 680–685.

[50] **M. Fliess et al.** *Flatness and Defect of Nonlinear Systems: Introductory Theory and Examples*. In: *International Journal of Control* 61.6 (1995), pp. 1327–1361.

[51] **J. D. Foley et al.** *Computer Graphics: Principles and Practice*. Addison-Wesley, 1996.

[52] **T. Fraichard and J.-M. Ahuactzin**. *Smooth Path Planning for Cars*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2001, pp. 3722–3727.

[53] **T. Fraichard and A. Scheuer**. *From Reeds and Shepp's to Continuous-Curvature Paths*. In: *IEEE Transactions on Robotics* 20.6 (2004), pp. 1025–1035.

[54] **C. Fulgenzi et al.** *Risk based motion planning and navigation in uncertain dynamic environment*. Tech. rep. Institut National de Recherche en Informatique et en Automatique, 2010.

[55] **J. D. Gammell et al.** *Batch Informed Trees (BIT*): Sampling-based Optimal Planning via the Heuristically Guided Search of Implicit Random Geometric Graphs*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2015, pp. 3067–3074.

[56] **J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot**. *Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2997–3004.

[57] **T. Gawron and M. Michałek**. *A G^3-Continuous Extend Procedure for Path Planning of Mobile Robots with Limited Motion Curvature and State Constraints*. In: *Applied Sciences* 8.11 (2018).

[58] *Gazebo*. Open Source Robotics Foundation. URL: http://gazebosim.org. (visited on 2018/12/30).

[59] **J. E. Gentle**. *Computational Statistics*. Springer, 2009.

[60] **F. Ghilardelli, G. Lini, and A. Piazzi**. *Path Generation Using η^4-Splines for a Truck and Trailer Vehicle*. In: *IEEE Transactions on Automation Science and Engineering* 11.1 (2014), pp. 187–203.

[61] **P. K. Ghosh**. *A Unified Computational Framework for Minkowski Operations*. In: *Computers & Graphics* 17.4 (1993), pp. 357–378.

[62] **E. G. Gilbert, D. W. Johnson, and S. S. Keerthi**. *A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space*. In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203.

[63] **C. Gläser et al.** *Environment Perception for Inner-City Driver Assistance and Highly-Automated Driving*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2014, pp. 1270–1275.

[64] *Global Status Report on Road Safety 2018*. World Health Organization. 2018.

[65] **G. H. Golub and C. F. Van Loan**. *Matrix Computations*. Johns Hopkins University Press, 1996.

[66] **D. González et al.** *A Review of Motion Planning Techniques for Automated Vehicles*. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145.

[67] **I. Goodfellow et al.** *Generative Adversarial Nets*. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[68] **S. Grodde**. *Ein lokaler Trajektorienplaner für das automatische Einparken*. Tech. rep. Universität Erlangen-Nürnberg, Lehrstuhl für Regelungstechnik, 2005.

[69] **G. Guennebaud, B. Jacob, et al.** *Eigen v3*. 2010. URL: http://eigen.tuxfamily.org. (visited on 2019/01/17).

[70] **B. Gutjahr, L. Gröll, and M. Werling**. *Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC*. In: *IEEE Transactions on Intelligent Transportation Systems* 18.6 (2017), pp. 1586–1595.

[71] **J. Hardy and M. Campbell**. *Contingency Planning Over Probabilistic Obstacle Predictions for Autonomous Road Vehicles*. In: *IEEE Transactions on Robotics* 29.4 (2013), pp. 913–929.

[72] **P. E. Hart et al.** *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.

[73] **T. M. Howard and A. Kelly**. *Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots*. In: *The International Journal of Robotics Research* 26.2 (2007), pp. 141–166.

[74] **C. Hubschneider et al.** *Adding Navigation to the Equation: Turning Decisions for End-to-End Vehicle Control*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2017, pp. 307–314.

[75] **C. Hubschneider et al.** *Integrating End-to-End Learned Steering into Probabilistic Autonomous Driving*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2017, pp. 2109–2115.

[76] **B. Ichter et al.** *Real-Time Stochastic Kinodynamic Motion Planning via Multiobjective Search on GPUs*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2017, pp. 5019–5026.

[77] **B. Ichter, J. Harrison, and M. Pavone**. *Learning Sampling Distributions for Robot Motion Planning*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2018, pp. 7087–7094.

[78] **B. Ichter and M. Pavone**. *Robot Motion Planning in Learned Latent Spaces*. In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2407–2414.

[79] *Intra-Logistics with Integrated Automatic Deployment (ILIAD) Project*. 2018. URL: https://twitter.com/iliad_project. (visited on 2018/12/05).

[80] **S. Ioffe and C. Szegedy**. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. In: *International Conference on Machine Learning*. 2015, pp. 448–456.

[81] **S. R. Jammalamadaka and A. Sengupta**. *Topics in Circular Statistics*. Vol. 5. World Scientific Publishing, 2001.

[82] **L. Janson et al.** *Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions*. In: *The International Journal of Robotics Research* 34.7 (2015), pp. 883–921.

[83] **L. Janson, B. Ichter, and M. Pavone**. *Deterministic sampling-based motion planning: Optimality, complexity, and performance*. In: *The International Journal of Robotics Research* 37.1 (2018), pp. 46–61.

[84] **L. Janson, E. Schmerling, and M. Pavone**. *Monte Carlo Motion Planning for Robot Trajectory Optimization Under Uncertainty*. In: *Robotics Research*. Springer, 2018, pp. 343–361.

[85] **P. Jeevan et al.** *Realizing Autonomous Valet Parking with Automotive Grade Sensors*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 3824–3829.

[86] **M. Jordan and A. Perez**. *Optimal Bidirectional Rapidly-Exploring Random Trees*. Tech. rep. 2013-021. Massachusetts Institute of Technology, 2013.

[87] **S. J. Julier and J. K. Uhlmann**. *Unscented Filtering and Nonlinear Estimation*. In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422.

[88] **T. Kanade, C. Thorpe, and W. Whittaker**. *Autonomous Land Vehicle Project at CMU*. In: *ACM Conference on Computer Science*. ACM. 1986, pp. 71–80.

[89] **Y. Kanayama and B. I. Hartman**. *Smooth Local Path Planning for Autonomous Vehicles*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 1989, pp. 1265–1270.

[90] **H. Kano and H. Fujioka**. *B-Spline Trajectory Planning with Curvature Constraint*. In: *American Control Conference*. IEEE. 2018, pp. 1963–1968.

[91] **K. Kant and S. W. Zucker**. *Toward Efficient Trajectory Planning: The Path-Velocity Decomposition*. In: *The International Journal of Robotics Research* 5.3 (1986), pp. 72–89.

[92] **S. Karaman et al.** *Anytime Motion Planning using the RRT**. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1478–1483.

[93] **S. Karaman and E. Frazzoli**. *Incremental Sampling-based Algorithms for Optimal Motion Planning*. In: *Robotics Science and Systems VI*. 2010.

[94] **S. Karaman and E. Frazzoli**. *Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods*. In: *IEEE Conference on Decision and Control*. IEEE. 2010, pp. 7681–7687.

[95] **C. Katrakazas et al.** *Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions*. In: *Transportation Research Part C: Emerging Technologies* 60 (2015), pp. 416–442.

[96] **A. Kelly and B. Nagy**. *Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control*. In: *The International Journal of Robotics Research* 22.7-8 (2003), pp. 583–601.

[97] **H. K. Khalil**. *Nonlinear Systems*. Vol. 3. Prentice Hall, 2002.

[98] **B. Kim, L. P. Kaelbling, and T. Lozano-Pérez**. *Guiding Search in Continuous State-Action Spaces by Learning an Action Sampler from Off-target Search Experience*. In: *AAAI Conference on Artificial Intelligence*. AAAI. 2018, pp. 6509–6516.

[99] **D. P. Kingma and J. L. Ba**. *Adam: A Method for Stochastic Optimization*. In: *International Conference on Learning Representations*. 2015.

[100] **S. Klemm et al.** *RRT*-Connect: Faster, Asymptotically Optimal Motion Planning*. In: *IEEE International Conference on Robotics and Biomimetics*. IEEE. 2015, pp. 1670–1677.

[101] **S. Koenig and M. Likhachev**. *D* Lite*. In: *AAAI Conference on Artificial Intelligence*. AAAI. 2002, pp. 476–483.

[102] **K. Komoriya and K. Tanie**. *Trajectory Design and Control of a Wheel-type Mobile Robot Using B-spline Curve*. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems*. IEEE. 1989, pp. 398–405.

[103] **J. Krafcik**. *Waymo One: The next step on our self-driving journey*. Waymo. 2018. URL: https://medium.com/waymo. (visited on 2019/03/19).

[104] **T. Kunz, A. Thomaz, and H. Christensen**. *Hierarchical Rejection Sampling for Informed Kinodynamic Planning in High-Dimensional Spaces*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2016, pp. 89–96.

[105] **Y.-L. Kuo, A. Barbu, and B. Katz**. *Deep sequential models for sampling-based planning*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2018, pp. 6490–6497.

[106] **A. Kushleyev and M. Likhachev**. *Time-bounded Lattice for Efficient Planning in Dynamic Environments*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 1662–1668.

[107] **R. Kusumoto et al.** *Informed Information Theoretic Model Predictive Control*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2019.

[108] **H. Kwakernaak and R. Sivan**. *Linear Optimal Control Systems*. Wiley-Interscience, 1972.

[109] **A. Lambert et al.** *A Fast Monte Carlo Algorithm for Collision Probability Estimation*. In: *IEEE International Conference on Control, Automation, Robotics and Vision*. IEEE. 2008, pp. 406–411.

[110] **J.-P. Laumond**. *Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints*. In: *International Conference on Intelligent Autonomous Systems*. 1986, pp. 346–354.

[111] **J.-P. Laumond et al.** *A Motion Planner for Nonholonomic Mobile Robots*. In: *IEEE Transactions on Robotics and Automation* 10.5 (1994), pp. 577–593.

[112] **J.-P. Laumond et al.** *Robot Motion Planning and Control*. Springer, 1998.

[113] **J.-P. Laumond, S. Sekhavat, and F. Lamiraux**. *Guidelines in Nonholonomic Motion Planning for Mobile Robots*. In: *Robot Motion Planning and Control*. Springer, 1998, pp. 1–53.

[114] **S. M. LaValle**. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. 98-11. Iowa State University, 1998.

[115] **S. M. LaValle**. *Planning Algorithms*. Cambridge University Press, 2006.

[116] **S. M. LaValle and J. J. Kuffner Jr**. *Randomized Kinodynamic Planning*. In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400.

[117] **A. Lee et al.** *Sigma Hulls for Gaussian Belief Space Planning for Imprecise Articulated Robots amid Obstacles*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 5660–5667.

[118] **Y. Li, Z. Littlefield, and K. E. Bekris**. *Asymptotically optimal sampling-based kinodynamic planning*. In: *The International Journal of Robotics Research* 35.5 (2016), pp. 528–564.

[119] **M. Likhachev et al.** *Anytime Dynamic A*: An Anytime, Replanning Algorithm*. In: *International Conference on Automated Planning and Scheduling*. 2005, pp. 262–271.

[120] **M. Likhachev and D. Ferguson**. *Planning Long Dynamically-Feasible Maneuvers for Autonomous Vehicles*. In: *The International Journal of Robotics Research* 28.8 (2009), pp. 933–945.

[121] **B. D. Luders, S. Karaman, and J. P. How**. *Robust Sampling-based Motion Planning with Asymptotic Optimality Guarantees*. In: *AIAA Guidance, Navigation, and Control Conference*. AIAA. 2013.

[122] **Y. L. Luke**. *Mathematical Functions and their Approximations*. Academic Press, 1975.

[123] **R. Madaan et al.** *Learning Adaptive Sampling Distributions for Motion Planning by Self-Imitation*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Machine Learning in Robot Motion Planning*. IEEE. 2018.

[124] **E. Marder-Eppstein, D. V. Lu, and D. Hershberger**. *costmap_2d*. 2012. URL: https://wiki.ros.org/costmap_2d. (visited on 2019/10/03).

[125] **M. McNaughton**. *Parallel Algorithms for Real-time Motion Planning*. PhD thesis. Carnegie Mellon University, 2011.

[126] **I. Miller et al.** *Cornell University's 2005 DARPA Grand Challenge Entry*. In: *Journal of Field Robotics* 23.8 (2006), pp. 625–652.

[127] **M. Montemerlo et al.** *Junior: The Stanford Entry in the Urban Challenge*. In: *Journal of Field Robotics* 25.9 (2008), pp. 569–597.

[128] **B. Müller, J. Deutscher, and S. Grodde**. *Continuous Curvature Trajectory Design and Feedforward Control for Parking a Car*. In: *IEEE Transactions on Control Systems Technology* 15.3 (2007), pp. 541–553.

[129] **R. M. Murray and S. S. Sastry**. *Nonholonomic Motion Planning: Steering Using Sinusoids*. In: *IEEE Transactions on Automatic Control* 38.5 (1993), pp. 700–716.

[130] **V. Nair and G. E. Hinton**. *Rectified Linear Units Improve Restricted Boltzmann Machines*. In: *International Conference on Machine Learning*. 2010, pp. 807–814.

[131] **R. Oliveira et al.** *Trajectory Generation using Sharpness Continuous Dubins-like Paths with Applications in Control of Heavy Duty Vehicles*. In: *arXiv preprint arXiv:1801.08995v1* (2018).

[132] **R. Oliveira et al.** *Trajectory Generation using Sharpness Continuous Dubins-like Paths with Applications in Control of Heavy Duty Vehicles*. In: *European Control Conference*. IEEE. 2018, pp. 935–940.

[133] **B. Paden et al.** *A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles*. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55.

[134] **C. Park, J. S. Park, and D. Manocha**. *Fast and Bounded Probabilistic Collision Detection for High-DOF Trajectory Planning in Dynamic Environments*. In: *IEEE Transactions on Automation Science and Engineering* 15.3 (2018), pp. 980–991.

[135] **J. S. Park, C. Park, and D. Manocha**. *Efficient Probabilistic Collision Detection for Non-Convex Shapes*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2017, pp. 1944–1951.

[136] **G. Parlangeli and G. Indiveri**. *Dubins inspired 2D smooth paths with bounded curvature and curvature derivative*. In: *IFAC Symposium on Intelligent Autonomous Vehicles*. IFAC. 2010, pp. 252–257.

[137] **S. Patil, J. van den Berg, and R. Alterovitz**. *Estimating Probability of Collision for Safe Motion Planning under Gaussian Motion and Sensing Uncertainty*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3238–3244.

[138] **N. Pérez-Higueras, F. Caballero, and L. Merino**. *Learning Human-Aware Path Planning with Fully Convolutional Networks*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2018, pp. 5897–5902.

[139] **J. Petereit**. *Adaptive State $\times$ Time Lattices: A Contribution to Mobile Robot Motion Planning in Unstructured Dynamic Environments*. PhD thesis. Karlsruher Institut für Technologie, 2017.

[140] **J. Petereit, T. Emter, and C. W. Frey**. *Safe Mobile Robot Motion Planning for Waypoint Sequences in a Dynamic Environment*. In: *IEEE International Conference on Industrial Technology*. IEEE. 2013, pp. 181–186.

[141] **A. Piazzi et al.** *Quintic G^2-Splines for the Iterative Steering of Vision-Based Autonomous Vehicles*. In: *IEEE Transactions on Intelligent Transportation Systems* 3.1 (2002), pp. 27–36.

[142] **A. Piazzi et al.** *η^3-Splines for the Smooth Path Generation of Wheeled Mobile Robots*. In: *IEEE Transactions on Robotics* 23.5 (2007), pp. 1089–1095.

[143] **M. Pivtoraiko et al.** *Differentially Constrained Mobile Robot Motion Planning in State Lattices*. In: *Journal of Field Robotics* 26.3 (2009), pp. 308–333.

[144] **M. Pivtoraiko and A. Kelly**. *Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2005, pp. 3231–3237.

[145] **R. Platt Jr. et al.** *Belief space planning assuming maximum likelihood observations*. In: *Robotics: Science and Systems VI*. 2010.

[146] **P. Polack et al.** *The Kinematic Bicycle Model: a Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?* In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2017, pp. 812–818.

[147] **D. A. Pomerleau**. *ALVINN: An Autonomous Land Vehicle In a Neural Network*. In: *Advances in Neural Information Processing Systems*. 1989, pp. 305–313.

[148] **F. P. Preparata and S. J. Hong**. *Convex Hulls of Finite Sets of Points in Two and Three Dimensions*. In: *Communications of the ACM* 20.2 (1977), pp. 87–93.

[149] **A. H. Qureshi, M. J. Bency, and M. C. Yip**. *Motion Planning Networks*. In: *arXiv preprint arXiv:1806.05767* (2018).

[150] **A. H. Qureshi and M. C. Yip**. *Deeply Informed Neural Sampling for Robot Motion Planning*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2018, pp. 6582–6588.

[151] **R. Rajamani**. *Vehicle Dynamics and Control*. Springer, 2006.

[152] **J. Reeds and L. Shepp**. *Optimal paths for a car that goes both forwards and backwards*. In: *Pacific Journal of Mathematics* 145.2 (1990), pp. 367–393.

[153] **J. H. Reif**. *Complexity of the Mover's Problem and Generalizations*. In: *IEEE Annual Symposium on Foundations of Computer Science*. IEEE. 1979, pp. 421–427.

[154] **J. Reuter**. *Mobile Robots Trajectories with Continuously Differentiable Curvature: An Optimal Control Approach*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 1998, pp. 38–43.

[155] *Robot Operating System (ROS)*. Open Source Robotics Foundation. URL: https://ros.org. (visited on 2018/12/30).

[156] **C. Rösmann, F. Hoffmann, and T. Bertram**. *Kinodynamic Trajectory Optimization and Control for Car-Like Robots*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2017, pp. 5681–5686.

[157] **M. Rufli and R. Siegwart**. *On the Application of the D* Search Algorithm to Time-Based Planning on Lattice Graphs*. In: *European Conference on Mobile Robotics*. 2009, pp. 105–110.

[158] **A. Scheuer**. *Planification de chemins à courbure continue pour robot mobile nonholonome*. PhD thesis. Institut National Polytechnique de Grenoble, 1998.

[159] **A. Scheuer and T. Fraichard**. *Planning Continuous-Curvature Paths for Car-Like Robots*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 1996, pp. 1304–1311.

[160] **A. Scheuer and T. Fraichard**. *Collision-Free and Continuous-Curvature Path Planning for Car-Like Robots*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 1997, pp. 867–873.

[161] **A. Scheuer and T. Fraichard**. *Continuous-Curvature Path Planning for Car-Like Vehicles*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 1997, pp. 997–1003.

[162] **A. Scheuer and C. Laugier**. *Planning Sub-Optimal and Continuous-Curvature Paths for Car-Like Robots*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 1998, pp. 25–31.

[163] **G. Schildbach and F. Borrelli**. *A Dynamic Programming Approach for Nonholonomic Vehicle Maneuvering in Tight Environments*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2016, pp. 151–156.

[164] **E. Schmerling, L. Janson, and M. Pavone**. *Optimal Sampling-Based Motion Planning under Differential Constraints: the Driftless Case*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2015, pp. 2368–2375.

[165] **E. Schmerling and M. Pavone**. *Evaluating Trajectory Collision Probability through Adaptive Importance Sampling for Safe Motion Planning*. In: *Robotics: Science and Systems VIII*. 2017.

[166] **R. Schneider**. *Convex Bodies: The Brunn–Minkowski Theory*. Cambridge University Press, 1993.

[167] **W. Schwarting, J. Alonso-Mora, and D. Rus**. *Planning and Decision-Making for Autonomous Vehicles*. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210.

[168] **U. Schwesinger et al.** *Automated Valet Parking and Charging for e-Mobility - Results of the V-Charge Project*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2016, pp. 157–164.

[169] *Science: Radio Auto*. Time Magazine. Aug. 10, 1925.

[170] **S. Sekhavat and J.-P. Laumond**. *Topological Property for Collision-Free Nonholonomic Motion Planning: The Case of Sinusoidal Inputs for Chained Form Systems*. In: *IEEE Transactions on Robotics and Automation* 14.5 (1998), pp. 671–680.

[171] **T. Shan and B. Englot**. *Belief Roadmap Search: Advances in Optimal and Efficient Planning Under Uncertainty*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2017, pp. 5318–5325.

[172] **D. H. Shin and S. Singh**. *Path Generation for Robot Vehicles Using Composite Clothoid Segments*. Tech. rep. 90-312. Canergie Mellon University, 1990.

[173] **D. C. Shoup**. *Cruising for parking*. In: *Transport Policy* 13.6 (2006), pp. 479–486.

[174] **B. Siciliano and O. Khatib**. *Springer Handbook of Robotics*. Springer, 2016.

[175] **C. Siedentop et al.** *Path-Planning for Autonomous Parking with Dubins Curves*. In: *Uni-DAS e.V. Workshop Fahrerassistenzsysteme*. 2015, pp. 11–18.

[176] **D. Silver et al.** *The Predictron: End-To-End Learning and Planning*. In: *International Conference on Machine Learning*. 2017, pp. 3191–3199.

[177] **S. Skogestad and I. Postlethwaite**. *Multivariable Feedback Control: Analysis and design*. Vol. 2. Wiley, 2001.

[178] **A. Srinivas et al.** *Universal Planning Networks*. In: *International Conference on Machine Learning*. 2018, pp. 4739–4748.

[179] **I. A. Şucan, M. Moll, and L. E. Kavraki**. *The Open Motion Planning Library*. In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82. http://ompl.kavrakilab.org.

[180] **W. Sun et al.** *Safe Motion Planning for Imprecise Robotic Manipulators by Minimizing Probability of Collision*. In: *Robotics Research*. Springer, 2016, pp. 685–701.

[181] **H. J. Sussmann**. *The Markov-Dubins problem with angular acceleration control*. In: *IEEE Conference on Decision and Control*. IEEE. 1997, pp. 2639–2643.

[182] **H. J. Sussmann and G. Tang**. *Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control*. Tech. rep. 91-10. Rutgers University, 1991.

[183] **A. Tamar et al.** *Learning from the Hindsight Plan - Episodic MPC Improvement*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2017, pp. 336–343.

[184] **G. Tanzmeister et al.** *Efficient Evaluation of Collisions and Costs on Grid Maps for Autonomous Vehicle Motion Planning*. In: *IEEE Transactions on Intelligent Transportation Systems* 15.5 (2014), pp. 2249–2260.

[185] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE International. Standard J3016_201806. 2018.

[186] **S. Thrun et al.** *Probabilistic Robotics*. MIT Press, 2005.

[187] **D. Tilbury, R. M. Murray, and S. S. Sastry**. *Trajectory Generation for the N-Trailer Problem Using Goursat Normal Form*. In: *IEEE Transactions on Automatic Control* 40.5 (1995), pp. 802–819.

[188] **C. Urmson et al.** *Autonomous Driving in Urban Environments: Boss and the Urban Challenge*. In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.

[189] **A. Vatavu et al.** *Environment Estimation with Dynamic Grid Maps and Self-Localizing Tracklets*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2018, pp. 3370–3377.

[190] *Vision Zero on the move*. The Swedish Transport Administration. 2015.

[191] **M. P. Vitus and C. J. Tomlin**. *Closed-Loop Belief Space Planning for Linear, Gaussian Systems*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 2152–2159.

[192] **H. Vorobieva et al.** *Automatic Parallel Parking in Tiny Spots: Path Planning and Control*. In: *IEEE Transactions on Intelligent Transportation Systems* 16.1 (2015), pp. 396–410.

[193] **M. Werling et al.** *Optimal trajectories for time-critical street scenarios using discretized terminal manifolds*. In: *The International Journal of Robotics Research* 31.3 (2012), pp. 346–359.

[194] **S. Wirges et al.** *Object Detection and Classification in Occupancy Grid Maps using Deep Convolutional Networks*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2018, pp. 3530–3535.

[195] **C. Xie et al.** *Toward Asymptotically Optimal Motion Planning for Kinodynamic Systems using a Two-Point Boundary Value Problem Solver*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2015, pp. 4187–4194.

[196] **W. Xu et al.** *Motion Planning under Uncertainty for On-Road Autonomous Driving*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2014, pp. 2507–2512.

[197] **Y. Yang and O. Brock**. *Adapting the Sampling Distribution in PRM Planners Based on an Approximated Medial Axis*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2004, pp. 4405–4410.

[198] **A. Yershova et al.** *Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 3856–3861.

[199] **D. Yi et al.** *Generalizing Informed Sampling for Asymptotically-Optimal Sampling-Based Kinodynamic Planning via Markov Chain Monte Carlo*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2018, pp. 7063–7070.

[200] **C. Zhang, J. Huh, and D. D. Lee**. *Learning Implicit Sampling Distributions for Motion Planning*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2018, pp. 3654–3661.

[201] **X. Zhang et al.** *Autonomous Parking using Optimization-Based Collision Avoidance*. In: *IEEE Conference on Decision and Control*. IEEE. 2018, pp. 4327–4332.

[202] **X. Zhang, A. Liniger, and F. Borrelli**. *Optimization-Based Collision Avoidance*. In: *arXiv preprint arXiv:1711.03449* (2017).

[203] **J. Ziegler et al.** *Making Bertha Drive – An Autonomous Journey on a Historic Route*. In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20.

[204] **J. Ziegler et al.** *Trajectory Planning for Bertha - a Local, Continuous Method*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2014, pp. 450–457.

[205] **J. Ziegler and C. Stiller**. *Fast Collision Checking for Intelligent Vehicle Motion Planning*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2010, pp. 518–522.

[206] **J. Ziegler, M. Werling, and J. Schröder**. *Navigating car-like Robots in unstructured Environments using an Obstacle sensitive Cost Function*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2008, pp. 787–791.

[207] **M. Zucker, J. Kuffner, and J. A. Bagnell**. *Adaptive workspace biasing for sampling based planners*. In: *IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 3757–3762.

List of Publications

[208] **H. Banzhaf**. *Steering Functions for Car-Like Robots*. 2017. URL: https://github.com/hbanzhaf/steering_functions. (visited on 2018/08/22).

[209] **H. Banzhaf, F. Niewels, and J. M. Zöllner**. *Automated Driving in Dense Scenarios – A Motion Planning Perspective*. In: *International VDI Conference on Automated Driving*. VDI. 2018.

[210] **H. Banzhaf, F. Quedenfeld, D. Nienhüser, S. Knoop, and J. M. Zöllner**. *High Density Valet Parking Using k-Deques in Driveways*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2017, pp. 1413–1420. © 2017 IEEE.

[211] **H. Banzhaf, L. Palmieri, D. Nienhüser, T. Schamm, S. Knoop, and J. M. Zöllner**. *Hybrid Curvature Steer: A Novel Extend Function for Sampling-Based Nonholonomic Motion Planning in Tight Environments*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2017, pp. 2239–2246. © 2017 IEEE.

[212] **H. Banzhaf, D. Nienhüser, S. Knoop, and J. M. Zöllner**. *The Future of Parking: A Survey on Automated Valet Parking with an Outlook on High Density Parking*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2017, pp. 1827–1834. © 2017 IEEE.

[213] **H. Banzhaf, M. Dolgov, J. Stellet, and J. M. Zöllner**. *From Footprints to Beliefprints: Motion Planning under Uncertainty for Maneuvering Automated Vehicles in Dense Scenarios*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2018, pp. 1680–1687. © 2018 IEEE.

[214] **H. Banzhaf, N. Berinpanathan, D. Nienhüser, and J. M. Zöllner**. *From G^2 to G^3 Continuity: Continuous Curvature Rate Steering Functions for Sampling-Based Nonholonomic Motion Planning*. In: *IEEE Intelligent Vehicles Symposium*. IEEE. 2018, pp. 326–333. © 2018 IEEE.

[215] **H. Banzhaf, P. Sanzenbacher, U. Baumann, and J. M. Zöllner**. *Learning to Predict Ego-Vehicle Poses for Sampling-Based Nonholonomic Motion Planning*. In: *arXiv preprint arXiv:1812.01127* (2018).

[216] **H. Banzhaf, P. Sanzenbacher, U. Baumann, and J. M. Zöllner**. *Learning to Predict Ego-Vehicle Poses for Sampling-Based Nonholonomic Motion Planning*. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1053–1060. © 2019 IEEE.

[217] **U. Baumann, Y.-Y. Huang, C. Gläser, M. Herman, H. Banzhaf, and J. M. Zöllner**. *Classifying Road Intersections using Transfer-Learning on a Deep Neural Network*. In: *IEEE International Conference on Intelligent Transportation Systems*. IEEE. 2018, pp. 683–690. © 2018 IEEE.

[218] **M. Dolgov and H. Banzhaf**. *Verfahren und Vorrichtung zum Prüfen einer Trajektorie für ein Fahrzeug*. Patent Application No. DE102018220581. Nov. 2018.

List of Awards

[219] **H. Banzhaf, M. Dolgov, J. Stellet, and J. M. Zöllner**. *From Footprints to Beliefprints: Motion Planning under Uncertainty for Maneuvering Automated Vehicles in Dense Scenarios*. In: *IEEE International Conference on Intelligent Transportation Systems*. Best Paper Award. 2018.

[220] **H. Banzhaf, N. Berinpanathan, D. Nienhüser, and J. M. Zöllner**. *From G^2 to G^3 Continuity: Continuous Curvature Rate Steering Functions for Sampling-Based Nonholonomic Motion Planning*. In: *IEEE Intelligent Vehicles Symposium*. Third Prize Best Application Paper Award. 2018.