Motion Planning for Autonomous Vehicles in Partially Observable Environments

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

Ömer Şahin Taş, M.Sc.

Tag der mündlichen Prüfung: 21.06.2022

Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr. Mykel Kochenderfer

Abstract

Uncertainties ranging from sensor noise to unobservable intentions of other traffic participants accumulate in the data processing pipeline in autonomous driving, resulting in incomplete or even misinterpreted environment representations. This frequently leads motion planning algorithms to plan motions of conservative driving style.

This dissertation develops two motion planners that compensate the deficiencies from preceding modules by exploiting reaction capabilities of a vehicle. It first presents a thorough analysis on the source and classification of uncertainties, and highlights the properties of an ideal motion planner. It subsequently delves into methods for modeling uncertainties and quantifying information. These are used to define a mathematical model that renders driving objectives together with constraints which ensure safety. The resulting planning problem is solved in real-time, in two distinct ways: first, with nonlinear optimization, and secondly, by framing it as a partially observable Markov decision process (POMDP) and approximating the solution with sampling. The planner utilizing nonlinear optimization considers multiple maneuvers jointly and yields a motion profile that is shaped by their individual probability. It maintains safety by ensuring the feasibility of a chance-constrained fallback option. In this way, the formulation inherently postpones maneuver decisions to a later time, whenever the current uncertainty is too high. On the other hand, planning with the POMDP framework focuses on improving sampling efficiency in Monte Carlo planning. First, it defines information rewards that guide samples to more rewarding actions. It employs a general heuristic to help sampling for the reward-shaped problem. Secondly, it exploits the continuity in the reward structure for action selection and achieves significant performance improvements especially for a higher number of actions. Evaluations show that planning with nonlinear optimization demonstrates great success both in driving experiments and in simulation studies, whereas planning with the POMDP framework is particularly well-suited for modeling complex interactions.

Kurzfassung

Unsicherheiten, welche aus Sensorrauschen oder nicht beobachtbaren Manöverintentionen anderer Verkehrsteilnehmer resultieren, akkumulieren sich in der Datenverarbeitungskette eines autonomen Fahrzeugs und führen zu einer unvollständigen oder fehlinterpretierten Umfeldrepräsentation. Dadurch weisen Bewegungsplaner in vielen Fällen ein konservatives Verhalten auf.

Diese Dissertation entwickelt zwei Bewegungsplaner, welche die Defizite der vorgelagerten Verarbeitungsmodule durch Ausnutzung der Reaktionsfähigkeit des Fahrzeugs kompensieren. Diese Arbeit präsentiert zuerst eine ausgiebige Analyse über die Ursachen und Klassifikation der Unsicherheiten und zeigt die Eigenschaften eines idealen Bewegungsplaners auf. Anschließend befasst sie sich mit der mathematischen Modellierung der Fahrziele sowie den Randbedingungen, welche die Sicherheit gewährleisten. Das resultierende Planungsproblem wird mit zwei unterschiedlichen Methoden in Echtzeit gelöst: Zuerst mit nichtlinearer Optimierung und danach, indem es als teilweise beobachtbarer Markov-Entscheidungsprozess (POMDP) formuliert und die Lösung mit Stichproben angenähert wird. Der auf nichtlinearer Optimierung basierende Planer betrachtet mehrere Manöveroptionen mit individuellen Auftrittswahrscheinlichkeiten und berechnet daraus ein Bewegungsprofil. Er garantiert Sicherheit, indem er die Realisierbarkeit einer zufallsbeschränkten Rückfalloption gewährleistet. Der Beitrag zum POMDP-Framework konzentriert sich auf die Verbesserung der Stichprobeneffizienz in der Monte-Carlo-Planung. Erstens werden Informationsbelohnungen definiert, welche die Stichproben zu Aktionen führen, die eine höhere Belohnung ergeben. Dabei wird die Auswahl der Stichproben für das reward-shaped Problem durch die Verwendung einer allgemeinen Heuristik verbessert. Zweitens wird die Kontinuität in der Reward-Struktur für die Aktionsauswahl ausgenutzt und dadurch signifikante Leistungsverbesserungen erzielt. Evaluierungen zeigen, dass mit diesen Planern große Erfolge in Fahrversuchen und Simulationsstudien mit komplexen Interaktionsmodellen erreicht werden.

Acknowledgments

The journey for this dissertation began in May 2011, after I had the opportunity to meet Prof. Dr.-Ing. Christoph Stiller's autonomous driving team AnnieWAY in the Netherlands. They inspired me to pursue my own master's at KIT, and specialize in the topics Prof. Stiller's institute was offering. I not only sharpened my German skills and received a scholarship for my intended master's studies in Germany, but passed the curriculum taught in German, and finished my master's thesis research under the supervision of Prof. Stiller's researchers. I was then fortunate enough to continue my research for a Ph.D. in the same group. I can now say, with the utmost pride and excitement, completing my research for this Ph.D. has been life changing. I sincerely thank Prof. Stiller, not only for providing the necessary mentorship for my studies, but for an amazing opportunity and atmosphere which inspired me in my own scientific career, and for teaching me how research can be managed and organized sustainably.

Prof. Dr. Mykel Kochenderfer's works on decision making substantially shaped the focus of research on planning. Our initial contact in late summer 2018 led me on such an authentic and wholesome journey to better get to know him in person. I cannot express how patient and motivating he was to constantly answer my questions, accepting to be my co-advisor, and devoting his valuable time to attend my Ph.D. defense.

I am equally indebted to Dr. Martin Lauer for being immensely helpful and sharing his valuable expertise whenever it was needed. His guidance has undoubtedly enabled and motivated me to begin certain topics with familiarity, and end with mastery.

A big thanks to all my colleagues for the few amazing years. I want to especially thank Dr.-Ing. Ole Salscheider for his never-ending support in any topic that picked at my brain. He was always ready for a discussion, to lend a hand, or serve as a required replacement for any task. A further special thanks to my colleagues Dr.-Ing. Fabian Poggenhans, Dr.-Ing. Johannes Beck, and Florian

Wirth for the countless fruitful discussions and unrestricted support over the years. During my time researching for this Ph.D. I also supervised several students and I want to highlight the curiosity of Felix Hauser, Johannes Fischer, and Maximilian Beck. Our discussions led me to question and develop my ideas further, and led to motivating factors that helped me delve deeper into certain topics.

Two years after I started as a research scientist, I was assigned the role of department manager to Prof. Stiller's team at the FZI Research Center for Information Technology. Besides my research on planning and control for autonomous systems, I learned how to tackle bureaucracy and have gained vast layers of experience on time-varying, partially observable systems. At this point, I would like to thank our division manager Dr. Alexander Viehl and our secretary Sonja Göttl for their guidance and help on tackling everything related to bureaucracy. I appreciate the support of my deputy department manager Dr.-Ing. Hendrik Königshof for his friendship and for bearing with my quirky German grammar questions over the years. I am more than fortunate to manage the activities of an excellent team of researchers at FZI: Denis, Marc, Fabian, Ole, Sascha, Tilman, Max, Piotr, Julius, Annika, Mohit, Frank, Kevin, Etienne, Julian, and Hendrik. For the constructive discussions on any topic around autonomous driving, I am grateful for the colleagues at KIT: Eike, Claudio, Jan, Jannik, Christoph, Sven, and Christian. A big thanks to you all.

This dissertation would not be complete without mentioning my dear friends Arda, Burak, Sarper, Joe, and my parents and sister for their support and encouragement throughout my studies. Finally, I thank my partner Cemre for her support and patience while completing this dissertation. You have always been the sunshine of my life since the day we met.

Karlsruhe, January 2022 *Ömer Şahin Taş*

Table Of Contents

Abstract i

Kurzfassung iii

Acknowledgments v

Acronyms and Symbols xi

1 Introduction 1
1.1 Motivation and Objectives 3
1.2 Contributions 4
1.3 Outline of This Work 5

2 Fundamentals of Planning under Uncertainty for Autonomous Vehicles 7
2.1 Environment Modeling for Probabilistic Planning 7
2.1.1 World, Environment, Scene and Situation 7
2.1.2 Map-guided Scene Understanding 9
2.1.3 Situation Prediction 12
2.2 Behavior and Motion Planning 13
2.2.1 Fundamentals 13
2.2.2 Properties of Planning Algorithms 16
2.2.3 Objectives of Planners 17
2.3 Uncertainties a Planner Must Tackle 18
2.4 Planning Approaches that Consider Uncertainty 19
2.4.1 Sampling-Based Planners 19
2.4.2 Numerical Optimization Based Planners 19
2.4.3 Planning with the POMDP Framework 21
2.4.4 Machine Learned Planners 22

3 Modeling Uncertain Information 23
3.1 Parametric Approaches 23
3.1.1 Gaussian Distribution 23
3.1.2 Mixture Distributions 26
3.2 Nonparametric Methods 27
3.2.1 Sampling Algorithms 27
3.2.2 Kernel Density Estimation 29
3.3 Quantifying Information 30
3.3.1 Variance of Estimates 31
3.3.2 Entropy 31

4 Optimal Proactive Planning 33
4.1 Optimal Motion Planning 33
4.1.1 Planning in Receding Horizon 33
4.1.2 Vehicle Models 36
4.1.3 Optimality Criteria and Cost Functional 39
4.2 Definitions of Terms Related with Safety 44
4.3 Methods to Maintain Safety 45
4.3.1 Time-To-X Metrics 45
4.3.2 Collision Detection in Workspace 46
4.3.3 Chance Constraints 47
4.3.4 Responsibility Sensitive Safety 49
4.3.5 Verification Approaches 49
4.3.6 Falsification Approaches 51
4.3.7 Others 52
4.4 Co-planning of Probabilistic Proactive Z-Plans 53
4.4.1 Alternative Fallback Plans and Present Uncertainties 54
4.4.2 Proactive Planning for Ordinary Driving 56
4.4.3 Proactive Interaction with Other Participants 59

5 Planning with Numerical Optimization 63
5.1 Nonlinear Optimization 63
5.2 Local Descent 64
5.3 Constrained Optimization 65
5.3.1 Penalty Methods 65
5.3.2 Lagrange Multipliers and Duality 66
5.4 Solvers for Constrained Nonlinear Problems 68
5.4.1 Active-Set Methods 68
5.4.2 Interior-Point Methods 69
5.5 Calculating Derivatives 70
5.6 Motion Planning as an Optimization Problem 72
5.6.1 Decision-Theoretic MPC 72
5.6.2 Model Predictive Contouring Control 75
5.6.3 Decision-Theoretic MPCC 76

6 Planning with Sequential Decision Making 79
6.1 Sequential Decision Making 79
6.1.1 Markov Decision Processes 80
6.1.2 Partially Observable Markov Decision Processes . . 82
6.2 Sequential State Estimation 84
6.2.1 Particle Filter 85
6.2.2 Unweighted Particle Filter 86
6.3 Solving MDPs with State Uncertainty in Real-Time 87
6.3.1 Monte Carlo Tree Search 88
6.3.2 Multi-Armed Bandits 89
6.3.3 Partially Observable Monte Carlo Planning 93
6.3.4 POMDP Algorithms with Continuous Spaces 95
6.3.5 Active Information Gathering with Reward Shaping . 97
6.3.6 Information Particle Filter Tree 98
6.4 Motion Planning with the POMDP Framework 102
6.4.1 State, Action and Observation Spaces 102
6.4.2 Transition Model 104
6.4.3 Observation Model 106
6.4.4 Reward Model 109
6.4.5 Rollout Policy and Belief Initialization 110

7 Evaluation **111**
7.1 Planning with Numerical Optimization 111
7.1.1 Planning in Fully Observable Environments 112
7.1.2 Changes in Driving Goals and Constraints 115
7.1.3 Noncompliant Traffic Participants 116
7.1.4 Imperfect Perception 117
7.1.5 Postponing Decisions While Ensuring Safety 121
7.2 Planning with the POMDP Framework 122
7.2.1 Intersection Crossing in Real-World Scenarios . . . 122
7.2.2 Intersection Crossing with Active Information Gathering 127
7.2.3 Exploiting Lipschitz Continuity of Rewards 133
7.3 A Comparison of the Proposed Planning Frameworks and Discussions 139

8 Conclusion & Future Directions **143**

References **147**

A Appendix **181**
A.1 Appendix on Modeling Uncertain Information 181
A.1.1 Truncated Gaussian Distribution 181
A.1.2 Further Information Measures 182
A.2 Appendix on Mathematical Models 183
A.2.1 Intelligent Driver Model 183
A.2.2 Runge-Kutta Integration 184
A.2.3 A General and Adaptive Loss Function 184
A.3 Additional Results 185
A.3.1 Active Information Gathering 185
A.3.2 Lipschitz Continuity of Rewards 188
A.4 Parameter Values Used in the Evaluations 188

Acronyms and Symbols

Acronyms

CDF	Cumulative Density Function
DPW	Double Progressive Widening
GCDC	Grand Cooperative Driving Challenge
ICS	Inevitable Collision States
IDM	Intelligent Driver Model
IPFT	Information Particle Filter Tree
IPM	Interior Point Method
KDE	Kernel density estimation
KKT	Karush–Kuhn–Tucker
KL	Kullback-Leibler
KLD	Kullback-Leibler Divergence
MAB	Multi-Armed Bandits
MAE	Mean Absolute Error
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MP	Merge Point
MPC	Model Predictive Control

MPCC	Model Predictive Contouring Control
PDF	Probability Density Function
PF	Particle Filter
PFT	Particle Filter Tree
POMCP	Partially Observable Monte Carlo Planning
POMCPOW	POMCP with Observation Widening
POMDP	Partially Observable Markov Decision Process
POSLB	Pareto Optimal Sampling for Lipschitz Bandits
POSLB-V	Pareto Optimal Sampling for Lipschitz Bandits with Variances
PW	Progressive Widening
ROS	Robot Operating System
SAE	Society of Automotive Engineers
SQP	Sequential Quadratic Programming
TUVG	Truncated Univariate Gaussian Distribution
UCB	Upper Confidence Bound
UCB-V	Upper Confidence Bound with Variances
UVG	Univariate Gaussian Distribution
V2V	Vehicle-to-Vehicle

Operators and Notations

v	Variable
$\boldsymbol{v}$	Vector variable
$\boldsymbol{V}$	Matrix or tensor variable
F	Scalar-valued function
$\mathbf{f}, \mathbf{F}$	Vector-, matrix- or tensor-valued function

1 Introduction

Intelligent vehicle research has made tremendous progress within the last three decades [Ben+14]. By reducing the impact of human errors which count for 90% of the major rationale of roadway accidents [BDJ01, p. 245], driver assistance systems have considerably increased traffic safety and driving comfort (see Figure 1.1). The success of driver assistance systems in mitigating the severity of roadway accidents has shifted the research focus from driver assistance applications to realizing autonomous driving, where further improvements in driving safety and overall traffic flow efficiency is possible [De +14].

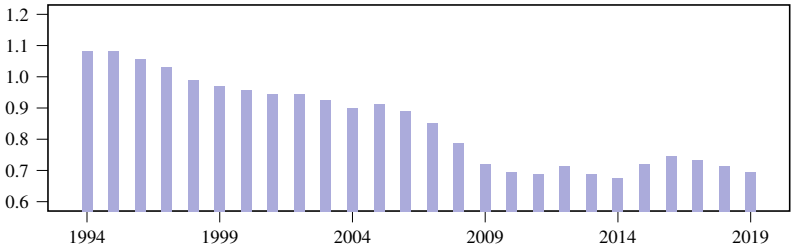


Figure 1.1: Fatality rates per 100 million kilometers traveled between 1994 and 2019 from the NHTSA Fatality Analysis Reporting System Encyclopedia [FAR21]. At the end of the 1990s, driver assistance systems are introduced and shortly after became mandatory. The next big leap in fatality rate mitigation can be expected with the further widespread introduction of autonomous vehicles active on the road.

Realizing autonomous driving is challenging, and cannot be achieved in a single step. For this reason, autonomous driving systems are classified by the realized level of automation. Figure 1.2 shows the widely accepted taxonomy made by the Society of Automotive Engineers (SAE) [SAE21]. Whereas *driver support* can be achieved with single sensors and automation subsystems, complexity increases substantially for *automated driving*, and holistic information processing that considers all environment information with their respective uncertainties becomes indispensable [Taş+16].

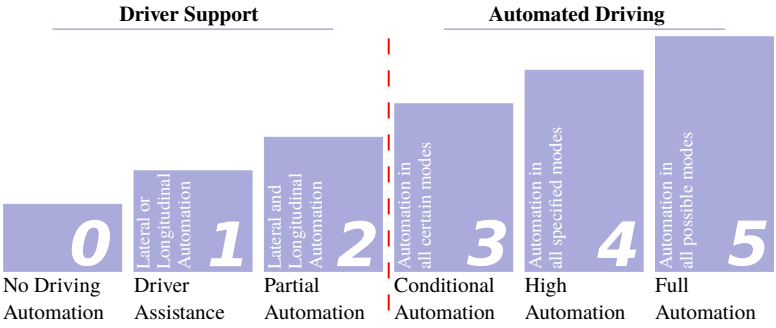


Figure 1.2: The taxonomy made by the SAE to classify automation levels of an autonomous vehicle [SAE21]. After the red dashed line complexity increases considerably.

In a layer-function-based classification, information processing of an autonomous vehicle can be divided into three layers and several modules, as depicted in Figure 1.3. The Sensor layer is the lowest layer of the architecture where the system retrieves information on its environment with various sensing modalities. The Data Fusion layer has access to the whole sensor data stream and aims to combine strengths of different types of sensors. Its output is an environment model that typically contains probabilistic features such as existence probability and state variance of objects, estimated driving corridor, and also the ego motion derived from subsequent measurements. The Understanding and Planning layer comprises Scene Understanding, Situation Prediction, Behavioral Planning and Motion Planning modules. During holistic processing, these modules sequentially enrich the environment model by additional interpretations and conclusions. The output of this layer is a motion profile transmitted to the vehicle controllers [Taş+16; Taş+17].

Research efforts to realize autonomous driving mostly take place in the latter two layers [Yur+20]. Despite diverse and advanced sensor setups, perception and fusion algorithms lack robust and reliable operation, especially in adverse weather and light conditions. Scene understanding algorithms can partly recognize the relation between objects, whereas prediction algorithms can provide a reliable prediction only for a short time horizon. Finally, planner algorithms can only operate with a defensive behavior under such an uncertainty, hence cannot exploit possible interactions between traffic participants.

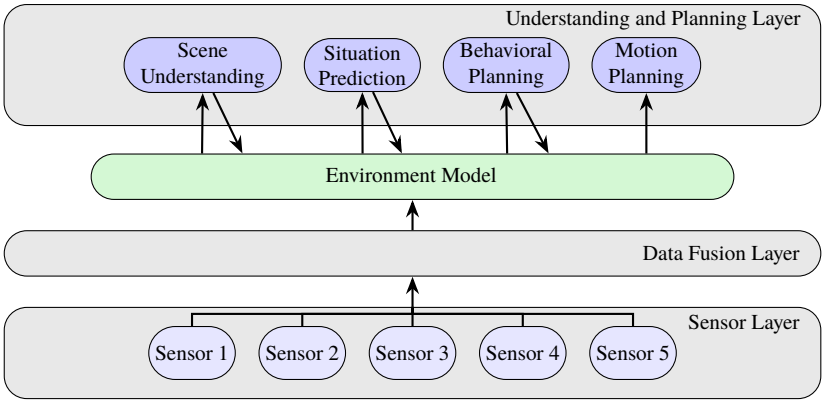


Figure 1.3: Illustration of an autonomous vehicle system architecture [Taş+17].

1.1 Motivation and Objectives

Ranging from sensor to planning layer, the uncertainties in the environment model accumulate. Despite the progress in research, environment perception will never yield perfect results in all possible outdoor conditions, nor will understanding and prediction modules ever reveal relations between objects and driving intentions completely. Even in the case of intervehicular communication, where vehicles broadcast their state and driving intentions, disputes will potentially occur. It is the duty of the planning module to plan safe yet comfortable motion under these uncertainties.

A planner to observe uncertainties in the environment requires quantification and classification of these. For this, the definition of a coherent environment model that serves as an input to the planner is necessary. This model should contain information on the confidence of measurements, current and future free space by yielding available route and driving options, traffic regulations, as well as visible and occluded areas. The task of the planner is then to plan comfortable proactive motion plans by considering the probabilistic information in the environment model. By exploiting available motion margins, it must react appropriately against unexpected or rule-violating behaviors of other traffic participants by computing a fallback motion. Furthermore, it must be able to execute action that maximizes its understanding of the environment.

The objective of this thesis is to develop a novel motion planner that accomplishes the driving task even in the case of impaired perception and partial observability. Therefore, this work first aims to define an environment model from the planning perspective, together with the uncertainties it contains. Secondly, it presents the features of the optimal motion and methods to quantify risk to remain collision-free even in the worst-case evolution of the current traffic situation. Subsequently, it models and solves the resulting problem using two distinct approaches: direct methods and sampling-based methods. Finally, it evaluates the strengths and weaknesses of both approaches and concludes by highlighting achieved objectives and provides future research directions.

1.2 Contributions

The contributions of this thesis are as follows:

- Information gathering in motion planning for autonomous vehicles. This work shows that a motion planner can gather information either passively, by postponing maneuver decisions to a later time, or actively, by executing dedicated actions that maximize the vehicle's information on the environment.
- Maneuver neutral motion planning with numerical optimization. Conventional planners impose the homotopy class of the motion to be selected beforehand. In cases where maneuver intentions of another vehicle are unclear, or the scene model contains phantom objects, this can lead to defensive behavior. This work considers available maneuvers by integrating their parameters into a nonlinear optimization problem and planning a motion that jointly reflects the probabilities of those maneuvers. The planner inherently blends discrete maneuver decisions with continuous optimization variables, and in this way, postpones a maneuver decision whenever the current uncertainty is too high.
- A numerical optimization based proactive planner with safety constraints. While existing planning approaches that attempt to guarantee safety make restrictive assumptions about maneuver intentions and sensor error characteristics, this planner propagates these uncertainties to its fallback motion plan. By utilizing chance constraints directly for the fallback

motion, it remains collision-free in the worst-case evolution of a scene without acting overly conservative.

- Planning with sequential decision making. Formulating the aforementioned planning problem as a decision process allows great flexibility in modeling over numerical optimization based methods. This comes at the cost of higher computational complexity. This work utilizes a Monte Carlo method with importance sampling to solve the problem efficiently.
- A POMDP solver that can solve problems with arbitrary belief-dependent rewards. This work presents a novel POMDP solver that simulates particle-based belief trajectories in Monte Carlo tree search (MCTS). This allows augmenting the objective with expected information gain, which is particularly suitable for solving large problems, where information gathering is an essential part of the optimal policy. The solver employs a general heuristic to tackle sampling requirements of the reward-shaped problem.
- Exploiting continuity in rewards for improved sampling efficiency. Vanilla MCTS methods do not make any assumption on the reward profile of the underlying problem. In contrast, this work shows that in motion planning for autonomous driving, similar actions yield similar rewards. By making a Lipschitz continuity assumption on the outcomes of actions, the MCTS is guided to more promising actions, eventually improving the sampling efficiency for planning with a high number of actions.
- Proven real-time application. Presented algorithms demonstrate high efficiency and reliability on continuous vehicle tests in cooperative platooning, lane change, and intersection crossing scenarios as well as in close-to-application simulations with real data.

1.3 Outline of This Work

Motion planning operates on outputs on current vehicle environment. **Chapter 2** starts with defining the environment model and introducing the fundamentals of motion planning. It presents properties of planning algorithms and certain requirements planners should ideally satisfy. Before reviewing existing motion planning approaches that tackle uncertainties, it elaborates on the types of uncertainties.

Modeling uncertainties and quantification of information are vital for motion planning under uncertainty. **Chapter 3** offers a crash-course on both parametric and nonparametric approaches for these. The approaches serve as the foundation for the planning algorithms presented in the following chapters.

Chapter 4 dives into the mathematical formulation for motion planning under uncertainty. It describes vehicle models and proposes approaches to apply driving objectives to a planning algorithm. It then introduces methods to maintain safety and discusses how safe proactive motion can be planned.

The next two sections present methods to solve the planning problem in real-time. **Chapter 5** frames motion planning as a numerical optimization problem. It presents an approach and a tool to solve the problem efficiently. Once the required algorithms are introduced, it recapitulates the objective function and constraints that resemble the aforementioned driving objectives. The formulation here utilizes parametric methods for uncertainty modeling.

Another approach for motion planning under uncertainty is to frame it as a sequential decision-making problem. While partially observable Markov decision processes (POMDPs) allow great flexibility in modeling, large POMDPs can only be solved with sampling-based approximations in real-time. **Chapter 6** starts with presenting decision processes, and subsequently introduces sampling-based POMDP solver algorithms. These solvers employ nonparametric uncertainty models, which qualify very well. This chapter further proposes an algorithm that exploits the problem structure in motion planning and considerably speeds up the runtimes. The chapter concludes with presenting the models on which the POMDP solver operates.

Chapter 7 demonstrates results for both numerical optimization and POMDP framework formulation. For numerical optimization based planning, it shows results from real test drives ranging from fully observable environments, such as cooperative lane change or intersection crossing, to scenarios where other vehicles might not comply with the traffic rules. Further analysis on imperfect perception and prediction are evaluated with simulation experiments with real data. Planning with the POMDP framework that covers intersection crossing and active information gathering is also evaluated with simulation studies. Once the results are presented, both planning approaches are compared and discussed.

Finally, **Chapter 8** highlights the main contributions of this work and presents an outlook for future research.

2 Fundamentals of Planning under Uncertainty for Autonomous Vehicles

Autonomous vehicles must operate under uncertain knowledge of their environment. There are various ways to model this uncertainty and to solve the resulting problem. This chapter first describes existing definitions on environment model and subsequently presents a coherent representation that allows for the consideration of uncertainties. It subsequently presents fundamentals and characteristic properties of motion planning algorithms, before analyzing uncertainties present in an environment model. The chapter concludes by reviewing existing solution approaches that tackle the planning under uncertainty problem.

2.1 Environment Modeling for Probabilistic Planning

Environment modeling incorporates substantiating the terms world, environment, state, and situation. This work closely follows the verbal definitions proposed by [Ulb+15] and links them with their respecting mathematical proposals defined by [AB+20], but by following the Partially Observable Stochastic Game formulation [Kuh53]. After clarifying these definitions, the processing of the environment model is introduced.

2.1.1 World, Environment, Scene and Situation

For any time $t \in \mathbb{N}$ an autonomous vehicle operates in a portion of *world* $\mathcal{W} \subseteq \mathcal{W}_\forall$ that contains a set of agents $\mathcal{V} = \{\vartheta_0, \ldots, \vartheta_{k_\mathcal{W}}\}$, with $k_\mathcal{W} \in \mathbb{N}_0$. An agent $\vartheta_i \in \mathcal{V}$ can be a member of any class, such as pedestrian, cyclist,

car etc., and has a *physical state* $\boldsymbol{x}_i^{(t)}$ and an *internal state* $\boldsymbol{\chi}_i^{(t)}$. The physical state typically consists of Cartesian coordinates, heading, and speed, i.e. $\boldsymbol{x}_i^{(t)} = [x_i^{(t)}, y_i^{(t)}, \psi_i^{(t)}, v_i^{(t)}]^\mathsf{T} \in \mathbb{R}^4$. The first three components are sometimes referred to as 2D-pose. The internal state of an agent can be modeled as $\boldsymbol{\chi}_i^{(t)} = [r_i^{(t)}, \theta_i^{(t)}]^\mathsf{T}$, where $r_i^{(t)}$ is its route and $\theta_i^{(t)}$ is its maneuver intention. While physical state can simply be referred to as *state* in planning context, prediction algorithms usually refer to physical state and internal state together as state $\boldsymbol{s}_i^{(t)} = [\boldsymbol{x}_i^{(t)}, \boldsymbol{\chi}_i^{(t)}]^\mathsf{T}$.

Sensors of an automated vehicle perceive physical states of the agents in its surroundings. These measurements are filtered and tracked in fusion layer prior to the creation of an environment model. Tracking allows matching objects along time, yielding $\boldsymbol{o}_i^{(t-\tau:t)}$, where $\tau \leq t, \tau \in \mathbb{N}$ is the tracked past history time steps. Fused sensor measurements, i.e. *observations*, of an agent at time t can be denoted with $\boldsymbol{o}_i^{(t)} \in \mathbb{R}^4$. The uncertainty in the observations are frequently approximated by a Gaussian distribution. While $\boldsymbol{x}$ of an agent can typically be observed by another agent with some measurement noise, $\boldsymbol{\chi}$ is hidden. Therefore, the environment of such a modeling is called *partially observable*.

An *environment* $\mathcal{E}_i$ for the portion of world $\mathcal{W}$ at time t from the viewpoint of agent i can be represented as $\mathcal{E}_i(\mathcal{W}, t) \coloneqq (\boldsymbol{s}, \Psi, \Omega, \mathcal{V}_\mathcal{E})$, where $\boldsymbol{s}$ is its own state, $\Psi \subset \mathbb{R}^2$ is its visible area bounded by a polygon, Ω is the map and map-related information of $\mathcal{W}$ including contextual information such as states of traffic lights, and $\mathcal{V}_\mathcal{E} = \{\vartheta_i\}_{i=1}^{k_\mathcal{E}}$ a set of observed agents, with $k_\mathcal{E} \leq k_\mathcal{W}, k_\mathcal{E} \in \mathbb{N}$. An observed agent ϑ_i at time t can be represented by another agent as $[\boldsymbol{o}_i^{(t)}, \hat{\boldsymbol{\chi}}_i^{(t)}]^\mathsf{T}$. The state of the ego vehicle together with the states of observed agents is occasionally represented as $\boldsymbol{s}^{(t)}$. Because the true state can only be estimated, it can be represented as a probability distribution over state space. Such a representation is called *belief* $\boldsymbol{b}^{(t)}$.

There are various definitions for the terms *scene* and *situation*. This work follows the definition of Ulbrich et al., who provide a thorough and systematic analysis. According to them, scene is a snapshot of the environment that enriches it with map-related information, self-representations such as skills, and field-of-view together with the relationship of objects [Ulb+15]. Therefore, a scene model $\mathcal{S}_i$ from the viewpoint of agent i can be expressed as a tuple $\mathcal{S}_i(\mathcal{W}, t) \coloneqq (\mathcal{E}_i, r_i, \omega(\Psi_i), \omega(\mathcal{V}_i))$. $\omega(\cdot)$ is a function that entails map related operations and adds attributes obtained from traffic regulations.

Where a scene associates the pose of the objects relative to a map and describes the relationship between them, a situation represents the future evolution of the scene for time horizon of t_h with selected and augmented information based on transient and permanent goals [Ulb+15]. Therefore, the dynamic properties of objects are more prominent. Following the previously introduced notation, we denote the situation from the perspective of an observer i as $\mathcal{P}_i := \varphi(\mathcal{S}_i)$, where prediction φ is a function $\varphi : \boldsymbol{o}_i^{(t-\tau:t)} \mapsto \boldsymbol{x}_i^{(t:t+t_\mathrm{h})}$ and $\varphi : \mathbb{R}^{4\times(\tau+1)} \to \mathbb{R}^{4\times(t_\mathrm{h}+1)}$.

2.1.2 Map-guided Scene Understanding

Maps are essential in motion planning. Independent of whether they are read statically from a database, or are created dynamically during a drive, they store information on route options, lane topology, traffic rules, and potential conflict areas. These are necessary for relating perceived information and understanding the scene, which eventually eases the prediction of scene evolution [Pet+13].

Maps consist of individual elements. Lanes, for example, consist of atomic sections on which neither traffic rules nor the topology changes. Such sections are called *lanelets* [BZS14]. A sequence of individual lanelets along a route are called a *lanelet sequence* [Pog+18]. Separate lanelet sequences can cross and merge with other lanelets in traffic scenes such as intersections, roundabouts, and acceleration lanes. The relation between individual lanelets and traffic rules are typically handled by graphs [PJ20].

The Frenet-Serret frame defines coordinates along a given line by (s, d), where s is the longitudinal and d is the lateral component. Coordinates in Cartesian frame can be transformed to the Frenet frame with

$$(s, d) = {}^{\mathcal{F}}_{\mathcal{C}}M\,(x, y), \tag{2.1}$$

and back with

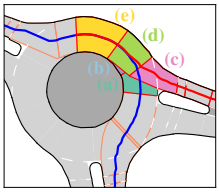
$$(x, y) = {}^{\mathcal{C}}_{\mathcal{F}}M\,(s, d). \tag{2.2}$$

It should be noted that this transformation is not bijective [Taş14].

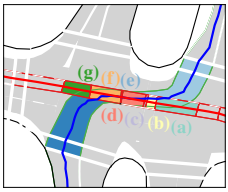
When poses of vehicles are mapped to lanelet centerlines, their coordinates in the Frenet frame can be aligned using the same reference. This is done by calculating the relative distances to the intersection point of the routes along

the Frenet frame. Availability of such relative positions together with traffic rules allows analyzing route and driving intentions of other vehicles. If an approaching vehicle has alternative routes that cross the route of the ego vehicle, and it is unclear which one it will follow, the route with the closest intersection point is taken as the reference. Since unreferenced coordinates in the Frenet frame are not considered useful for this work, these transformations are assumed to be aligned.

The calculation of intersection points and relative coordinates can in some circumstances be inaccurate and therefore misleading. In merge scenes, where merging lanelet geometries have small curvatures, the intersection point is typically far ahead of the point where lanelet conflict begins. In such cases, if vehicle intentions and safety criteria are applied with respect to the intersection point, inaccurate results are obtained (see Figure 2.1). A solution for this problem is to use the first lanelet in the sequence that has a conflict with the other sequence, and subsequently to calculate intersection points of their right vs. left borderline combinations. The resulting point provides precise results for relative distance calculations.



(a) Lanelet (a) conflicts with (d) and (c). Lanelet (b) conflicts with (d) and (c). Both sequences (ab) and (cd) merge at (e).



(b) Conflicts of lanelet with many lanelets, while the first conflicting lanelet (a) is the one to be used for intersection analysis.

Figure 2.1: The lanelets of a lanelet map used in practice. Multiple conflicting lanelets that in reality resemble a single merging lanelet sequence. Blue line is the centerline of the reference lanelet sequence. Querying the first intersecting lanelet along the oncoming red line returns inaccurate results. Higher accuracy is obtained by calculating intersection along lanelet boundaries and projecting the intersection point on the respective centerline.

A further source of inaccuracy can lie in the calculation of distances to the intersection points. The Euclidean distance between the current position of the vehicles and the intersection point can be used for this task. Even though this is a straightforward and fast approach in intersections such as roundabouts and U-turns, or in cases where the vehicle maneuvers in the close vicinity of

the ego vehicle, it will cause significant errors. For this reason, distances are calculated in Frenet frame.

An exemplary traffic scene is depicted in Figure 2.2. The beginning of the route centerline of the ego vehicle is set as the initial point in Frenet frame, i.e. $(s, d) = (0, 0)$. The green vehicle may turn right or left. This will lead to intersection points of $*$ and $\star$, where the former is closer. The position of the green vehicle is calculated as $s_{\text{green}} = s_{\text{curr}} + s_1 - s_2$. It is clear that the green vehicle's route will either cross or merge with the ego-route, however, it is unclear for the red vehicle. Still, as it may cross the lanelets of the blue vehicle, it has to be considered. The distance interval it crosses the ego-route corresponds to the arc distance between $\oplus$ and $\oplus$. The relative longitudinal coordinates on Frenet frame are depicted on a vertical line to the right of the figure. The beginning of intersections are marked with a dot, having the color of the respective vehicle and continuing as a line until the point where the interaction ends. Whereas the interval is bounded for the red vehicle, it is unbounded for the green vehicle, as it may remain on the ego-route.

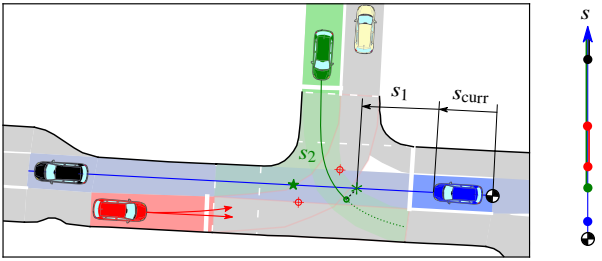


Figure 2.2: A scene with relative distances to the ego vehicle, which is depicted in blue. Positions of the vehicles projected onto the Frenet-coordinate frame are depicted on the right. The green vehicle may remain on the route of the ego vehicle, whereas the red vehicle will cross only for some distance interval.

Traffic rules together with relative coordinates allow for refining the environment model by filtering out trivial agents that do not have any influence on the decision and the motion of the ego vehicles. Agents that are relevant for the ego vehicle are called as scene objects and are denoted with $\mathcal{V}_{\mathcal{S}}$, where $\mathcal{V}_{\mathcal{S}} \subseteq \mathcal{V}_{\mathcal{E}}$. In Figure 2.2, the red, green, and black vehicles are scene objects, though the beige vehicle is not.

Apart from agent-related analysis, a scene model contains traffic signs and self-skills including its visible field, as introduced before. Traffic signs, except driving corridor markings, are effective from a particular longitudinal position onwards, and can therefore be fairly mapped to arc length coordinates, eventually easing analysis on interactions and traffic rule compliance. The region outside the visible field is unknown for the autonomous vehicle, and it may contain other agents. As in the case of tracked vehicles, mapping borders of the visible field over the drivable area to corridor centerlines allows reasoning-out maneuver intentions of eventual objects in coherence with the traffic rules.

2.1.3 Situation Prediction

Situation prediction aims to estimate the future motion of the traffic participants. While estimating this, it shall reflect the increasing uncertainty in the predicted time horizon t_h in a multi-modal fashion, while considering the physical bounds and traffic regulations. The situational awareness encoded into the scene model by identifying relations and interactions among traffic participants and traffic rules serves as a basis for this.

Predicting motion of participants involves predicting their maneuvers, which typically encompasses two interrelated classification problems: route prediction and maneuver intention prediction. The route an agent chooses depends on the map topology and the accompanying traffic rules. For a given route, an agent i may choose among the discrete intention classes $\theta_i^{(t)} \in \{\mathit{traverse}, \mathit{yield}, \mathit{lane\text{-}change}, \mathit{stop}\}$. Whereas the intention *yield* is with respect to agent j, i.e. $\theta_{(i|j)}^{(t)}$, the others are not relative to any agent.

Prediction at junctions is a challenging task having an underlying multi-agent prediction problem. The motion of every agent is connected, since every single alternative maneuver affects other participants. Furthermore, the number of maneuver alternatives, i.e. *homotopy classes*, grows exponentially with the number of participants and number of route options [Taş14; Ben+15; PKI15; AD17; PLG21]. In cases where multiple maneuver alternatives are possible, accurately predicting homotopy classes is more important than predicting motion profile in that homotopy class. This is an arduous task, especially for agents like cyclists and motorbikes.

The complexity in predicting other traffic participants motion and the opportunity to divide this into subproblems have led to the utilization of a variety of methods. An overview on prediction algorithms is provided in [LVL14]. This work classifies algorithms based on the semantics used and serves as an intuitive introduction to the topic. A recent paper surveys a broad spectrum of the works on driver behavior models and organizes them based on the underlying algorithm, chosen models, and solved tasks [BDK20]. Some of the most recent prediction algorithms such as the DESIRE Network [Lee+17], the Multipath [Cha+19], the TPNet [Fan+20], and the CoverNet [Pha+20] take top view image data as an input and predict the future motion with a CNN.

A motion planner should ideally be able to predict the evolution of the current scenario based upon its chosen actions. For this, it requires a transition model of its environment. Although there are established methods to predict the future of a situation, many cannot be utilized for such a purpose. One such method is to employ a motion planner for each of the agents in the environment. However, querying motion planners can be computationally expensive, especially if alternative maneuver options must be evaluated. Furthermore, it must identify the objectives of such a motion planner with obtained measurements, further increasing its complexity. A common way to have a simple estimate is to utilize the Intelligent Driver Model (IDM) [THH00], which mimics human driving behavior for follow and free driving scenarios, see Appendix A.2.1. Another alternative would be to use the Foresighted Driver Model [EDK15].

2.2 Behavior and Motion Planning

The operation of a planner as well as the evaluation of planners require presenting several definitions and approximations. Once the fundamentals are introduced, several essential properties a planner algorithm may have are presented. These fundamentals and properties serve as a foundation in the rest of this work.

2.2.1 Fundamentals

An autonomous vehicle, once it has perceived and interpreted its environment, must navigate through it to reach its goal while maximizing its preferences and

anticipating moving agents and traffic rules. This objective can be expressed as finding an optimal action sequence that minimizes a cost functional subject to constraints

$$\underset{\boldsymbol{a}}{\text{minimize}} \qquad J(\boldsymbol{a}) = V\left(\boldsymbol{x}(t_\mathrm{f})\right) + \int_{t_0}^{t_\mathrm{f}} L(\boldsymbol{x}(t), \boldsymbol{a}(t))\,\mathrm{d}t \tag{2.3a}$$

$$\text{subject to} \qquad \mathbf{g}\left(\boldsymbol{x}(t), \boldsymbol{a}(t)\right) = \mathbf{0}, \quad \forall t \in [t_0, t_\mathrm{f}] \tag{2.3b}$$

$$\mathbf{h}\left(\boldsymbol{x}(t), \boldsymbol{a}(t)\right) \leq \mathbf{0}, \quad \forall t \in [t_0, t_\mathrm{f}] \tag{2.3c}$$

where $\boldsymbol{a}(t) \in \mathbb{R}^{n_{\boldsymbol{a}}}$ is the *control input trajectory*, $\boldsymbol{x}(t) \in \mathbb{R}^{n_{\boldsymbol{x}}}$ is the *state trajectory*, $\mathbf{g}$ is the equality and $\mathbf{h}$ is the inequality constraints. The cost functional in Equation (2.4a) is given in Bolza-Form and therefore consists of a terminal cost V and a running cost L. The running costs sum up in the time interval for which the planning is done, i.e. $[t_0, t_\mathrm{f}]$.

In driving, the arrival time t_f is usually not known beforehand. Applications that treat the arrival time as an additional optimization parameter require an additional boundary condition called the *transversality condition*. While driving in traffic, the arrival time is free and the time interval becomes $[t_0, \infty)$. According to Barbălat's Lemma, as the integral in the Lagrangian term must be finite, $\lim_{t\to\infty} \boldsymbol{x}(t) = \boldsymbol{x}_\mathrm{goal}$ holds. This asymptotic stability allows reducing the cost functional to its Lagrangian term. Furthermore, except for parking applications, a goal state $\boldsymbol{x}_\mathrm{goal}$ is not defined at all, and therefore the problem in Equation (2.3) is not bound to final conditions.

In practice, the planning problem is solved for a finite time horizon $t_\mathrm{h} < \infty$. By using Bellman's principle of optimality, it can be shown that the asymptotic stability holds for sufficiently big horizons $t_\mathrm{h} \in [\Delta t, \infty)$. Constraints on the problem, which reflect vehicle dynamics or external factors, do not allow following an analytical solution but require a numerical one. This can be done with colocation methods, by approximating the control and the state trajectory along t_h by a number of sampling points, or *colocation points*, $\{t_i = t_0 + i t_\mathrm{s},\ 0 \leq i < n\}$, where t_s is the sampling interval and n is the number of sampling points. Whereas control inputs are set piecewise constant at every colocation point, variables that depend on the control input are integrated using Euler or Runge-Kutta methods, see Appendix A.2.2.

The resulting discretized planning problem has the form

$$\underset{\boldsymbol{a}}{\text{minimize}} \qquad J(\boldsymbol{a}) = \sum_{i=0}^{n-1} L(\boldsymbol{x}_i, \boldsymbol{a}_i) \tag{2.4a}$$

$$\text{subject to} \qquad \mathbf{g}(\boldsymbol{x}_i, \boldsymbol{a}_i) = \mathbf{0}, \quad i = 0, \dots, n-1 \tag{2.4b}$$

$$\mathbf{h}(\boldsymbol{x}_i, \boldsymbol{a}_i) \leq \mathbf{0}, \quad i = 0, \dots, n-1 \tag{2.4c}$$

facilitating the use of various optimization algorithms. Independent of the cost function, constraints, and the chosen optimization algorithm, the minimizer is called the *optimal control input* $\boldsymbol{a}^*$ and the corresponding state trajectory is simply called as the *optimal trajectory*, or the *optimal motion* $\boldsymbol{x}^*$

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} J(\boldsymbol{a}) \tag{2.5a}$$

$$\boldsymbol{x}^* = F(\boldsymbol{x}_0, \boldsymbol{a}^*). \tag{2.5b}$$

Changing environment as well as limited sensor information require planning in continuous-fashion on the basis of new information. Because of accumulating uncertainty along time, and limited computational capacity, motion is calculated repeatedly for a limited time horizon. The resulting problem is often still too complex to solve in high frequency in order to meet the requirements of stability. Therefore, the planning problem of autonomous driving is often subdivided into three submodules: behavior planning, motion planning, and motion control. High level decisions including maneuver decisions are made by the *behavior planning* or *decision-making* module, whereas the calculation of a collision-free trajectory for a maneuver decision is done by the *motion planning* or *trajectory planning* module. The tracking of planned control inputs can be done by a low-level controller, which operates in very high frequencies. Such a controller module is called *motion control*. Although the separation of modules seems reasonable at first, it is an inferior approach to a planning that handles the duties of all three submodules, as exploiting the full maneuver potential is not possible due to the simplifications made while assigning tasks to submodules.

A further measure to cope with the aforementioned complexities in real-time is to divide the motion planning problem into path-planning and velocity planning problems. In contrast to motion, a *path* defines the course of poses without specifying their time and therefore, it does not entail any information on velocity.

With path-velocity decomposition, first a path is found and afterwards a velocity profile is sought for that path [KZ86]. This allows for low-frequency path planning and higher frequency velocity planning against dynamic factors. Even though this decomposition considerably reduces complexity, by holding the path fixed for some interval, it restricts lateral capabilities.

In motion planning, path-time diagrams are frequently used for analyzing maneuvers and interactions. The center of the driving corridor is taken as the reference path, and the relative intersection point of each agent is mapped as such (see Figure 2.2). Because of the dimensionality reduction, it does not capture information on lateral terms. There are two alternative forms of representing the path-time diagram. In early motion planning works, where a robot has to traverse some area without time constraints, the path is laid on the x-axis and the optimal time is inspected. In real-time applications where the dynamicity of the agent is in the foreground, such as autonomous driving, the x-axis is taken as time and the progress along it is mapped onto the y-axis.

2.2.2 Properties of Planning Algorithms

Motion planning algorithms are classified according to several properties.

Online/Offline – If an algorithm can find a solution in runtime, it is said to be *online*. Only online algorithms can handle changes in the environment and inaccuracies in the execution of the planned path [RN16, p. 421].

Completeness – If an algorithm can find a solution whenever a solution exists, or indicates inexistence of a solution otherwise, the algorithm is called *complete*. With increasing search space dimension, this requirement is weakened by its convenient subforms: *resolution* completeness and *probabilistic* completeness. They indicate that the algorithm will be able to find a solution if the discretization is fine enough, and the probability of finding the solution goes to one as the number of iterations go to infinity, respectively [LaV06, p. 185].

Anytime – If an algorithm can yield a feasible output immediately after its execution and gradually improves this with the number of iterations executed, the algorithm is said to be *anytime* capable. Therefore, such algorithms have a reasonable output quality at any interruption [RN16, p. 1063].

Global/Local – The environment can create multiple distinct maneuver options each having an optimum. Whereas many optima are in their local neighborhood, only one of them is the global one. Depending on if the planner can find the global one, or will find a local one, it is classified either as *global* or *local* algorithm [Cor+09, p. 424; Ben+15].

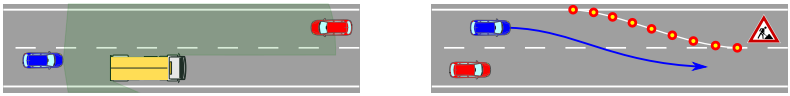
An autonomous vehicle motion planning algorithm must be online, probabilistic complete, and ideally, anytime capable and global.

2.2.3 Objectives of Planners

The most important properties a motion planning algorithm can have are already presented. Apart from these algorithmic properties, there are task-based objectives a motion planner shall satisfy. These objectives can be categorized as 1) utility, 2) comfort, 3) risk, 4) interaction, 5) information gathering.

Utility concerns task execution and completion in the sense of reaching the destination in a collision-free way. It includes aspects like holding vehicle constraints and ensuring fail-safe planning. While utility is the primary target to achieve, maintaining a *comfortable* ride is the secondary target. It is strongly related with proper *risk* handling and *interaction* capabilities of the planner. Risk encompasses aspects like limited quality in sensor measurements, visible field, or intentions of perceived traffic participants. Evaluation of the current risk results in risk-sensitive behavior. Interactive capabilities are highly dependent on uncertainties, but in fact, they represent a different characteristic. They enable the planner to follow social norms by demonstrating courtesy to other participants or planning transparent motion. Besides these commonly accepted requirements, a motion planner should perform *active* information gathering actions to maximize its knowledge about the environment, without entering risky situations. Slight lateral offsetting for visibility maximization and targeted steering and acceleration actions to reveal the potentially ambiguous intention of vehicles that the ego vehicle interacts with fall in the category of active information gathering, see Figure 2.3. If a motion planner postpones some of its decisions by leveraging the fact that more information will be available in the future, then it is said to perform *passive* information gathering.

The explanation on requirements implies that all categories are connected to each other and uncertainties play a vital role in satisfying these requirements.



(a) Slight lateral offsetting for visibility maximization.

(b) Acceleration and steering actions to reveal the intention of others.

Figure 2.3: Scenarios that benefit from active information gathering.

2.3 Uncertainties a Planner Must Tackle

Optimal planning in fully observable environments is a solved problem [Gon+15; Pad+16]. The challenge that remains is maintaining the robust and reliable operation of the vehicle in all possible environment conditions. Proper uncertainty modeling and handling facilitates robust operation, and increases reliability indirectly.

Classification of Uncertainties

Sources of uncertainties are briefly explained in Section 2.1. They typically result from incomplete information which is either completely missing, such as obscurity of the future, or contain random information, such as noisy measurements. Apart from these statistical uncertainties, they can have an underlying systematic structure, such as the correlation between individual state variables or multi-modalities.

These uncertainties result in three categories with which a planner has to deal with. The *uncertainty in physical state* covers the uncertainty in position and velocity values of the agents. The *uncertainty in prediction* encompasses the uncertainties in route alternatives, intentions, and their motion profiles. It must be underlined that the uncertainty in route and maneuver intention pose the most important source of uncertainty for motion planing, as a wrong estimation can trigger maneuver decisions that potentially end up with crashes. The *uncertainty in existence* reflects the uncertainty of an object to exist, either inside or beyond the visible area. Existence of objects inside the field-of-view is concerned with the topic of phantom detections. Whereas the information in the regions beyond the field-of-view is missing, it can partly be clarified with the information available [TS20].

2.4 Planning Approaches that Consider Uncertainty

Motion planning for autonomous vehicles is a broadly studied topic. An overview on notable works are presented in surveys [Gon+15; Pad+16]. The vast majority of approaches focus either on path planning, or motion planning in fully observable environments. However, the real challenge lies in planning safe, interactive motion despite imperfect perception and incomplete knowledge. Several distinguishing works that propose planning algorithms in such settings are presented next. They are classified by their underlying technique.

2.4.1 Sampling-Based Planners

Early examples of motion planners utilize either randomized sample-based or lattice-based search algorithms. Succeeding works enhance these with probability constraints. Luders et al. introduce chance constraints to the rapidly-exploring random tree (RRT) algorithm [LKH10]. A succeeding work extends it with motion patterns learned from Gaussian processes and applies it for driving tasks [Aou+13]. Besides introducing chance constraints into RRT, swapping state with belief in RRT* algorithm is proven to be effective for motion planning in tight environments [Ban+18]. Lattice search, which selects state lattice samples from a precomputed graph of feasible maneuvers, is also extended to consider probability constraints. A recent work creates quintic Bézier curves on an occupancy grid world and considers the uncertainties in localization [Art+20].

2.4.2 Numerical Optimization Based Planners

Numerical optimization has always been a favorable technique for motion planning. Safety in the presence of uncertainties is either ensured by robust or by stochastic control techniques. Robust techniques result in overly conservative motion plans and are therefore not preferred. The majority of works that follow stochastic techniques model uncertainties with Gaussian distribution, and employ chance constraints to maintain safety. In this way, a closed-form solution for a given confidence level is obtained.

Feedback control based chance-constrained convex programs are introduced in [VT11; VT13; LKK15]. Integration of chance constraints in model predictive control for motion planning is presented in [Car+14]. Additional to chance constraints, Gaussian processes are proven to be effective to model nonlinearities and to add them as a residual noise [HKZ19]. Independent of whether a planning algorithm considers uncertainties or not, a common need is the presence of a reference contour that can guide the planner. Applications of model predictive contouring control (MPCC) for autonomous vehicle motion planning arises as a convenient solution [Sch+17; Lin18]. The MPCC formulation has shown to be advantageous for non-Gaussian chance-constrained planning as well [WJW20].

Although chance constraints limit risk, they do not ensure safety. A common way is to predict all future states of traffic participants and verify the presence of a collision-free fallback maneuver. Pek et al. co-planned fallback maneuvers while simultaneously planning the desired one. However, the future state prediction does not involve limited visibility or uncertain localization of the ego vehicle [PA20]. A later work extended this approach with chance constraints [Brü+21]. Taş et al. introduced co-planning of fallback maneuvers under limited visibility and in the presence of noncompliant traffic participants in an earlier work. The fallback maneuver is restricted to emergency braking and is introduced to the numerical optimization based motion planner in the form of chance constraints [TS18]. Alsterda et al. co-planned fallback plans by using a separate cost function for the fallback, and tackled uncertainties in road surface friction [ABG19].

The effect of limited visibility and algorithms that aim to maximize visibility while following ordinary driving goals is occasionally studied. An optimization-based motion planner that maximizes the visible field of the vehicle is presented in [And+19a]. Similar to visibility, another important source of uncertainty results from driving intentions. Sadigh et al. modeled other traffic participants as underactuated systems and presented a gradient-based approach that can actively gather information on their internal states and eventually manipulate them [Sad+16].

Optimization-based approaches are frequently used when planning in a centralized fashion [Qia+16]. As autonomous driving will not benefit from centralized planning in the near to mid-future, it is out of scope of this work.

2.4.3 Planning with the POMDP Framework

Formulating motion planning as a POMDP has significant advantages over numerical optimization based approaches: they can operate with distributions of arbitrary forms, they are not trapped in local minima, and they can perform active information gathering. Previously mentioned optimization-based work that actively gathers information explicitly states that a POMDP formulation is superior to their approach [Sad+16]. Motion planning with the POMDP framework is becoming increasingly popular with the advent of online POMDP solvers and deep learning based predictors.

Solving POMDPs becomes computationally tractable only under certain assumptions. Many works discretize the state, action, and observation spaces to tackle this. Early works use POMDPs only for behavior planning, which typically includes several states and few actions [UM13]. Others require pre-training at every new scene encountered and are therefore not scalable [BGD14]. Sampling-based POMDP solvers enable planning algorithms to operate in previously unseen environments in an online fashion [LKK16; Hub+18]. Nevertheless, the number of available actions remains as rudimentary decisions, such as change lane, accelerate, or decelerate. Planning with the POMDP formulation also allows tackling limited visibility [Bou+18; Hub+19].

Recent works focus on improving the tree structure of sampling-based POMDPs and thereby increase the efficiency and scalability. Several online algorithms perform better than regular solvers in settings among which driving fits into [SK18]. Some works utilize these algorithms in further driving scenarios and demonstrate success [SK20; Bey+21]. Others focus on tailoring search for the traffic scenario. Naumann exploited action independence under clear precedence [Nau21]. Wray et al. addressed scalability challenges of planning with the POMDP framework in real-world applications and defined arbitration points for targeted action selection. They demonstrated scalability of their approach with driving experiments under limited visibility [Wra+21].

Another recent work uses iterative linear-quadratic Gaussian control in Gaussian belief space to solve a game-theoretic continuous POMDP [Sch+21]. Compared to sampling-based POMDP algorithms, which have exponential complexity in the planning horizon, it has a linear complexity. It can solve a quadratic game with runtimes comparable to model predictive control (MPC) planners. However, it converges to the locally optimal solution in belief space.

2.4.4 Machine Learned Planners

The breakthrough in neural networks has revolutionized the way intelligent systems operate. Autonomous driving has benefited from the representational power of deep neural networks ranging from perception to prediction. The presence of simulation environments and an ever-growing amount of sensor data have recently enabled the deployment of reinforcement learning or deep learning based planning algorithms.

There are various ways to benefit from learning in planning. Some planners use POMDP solvers [Som+13; Luo+19] and utilize deep reinforcement learning [Mir+16] to learn the future returns of actions and show promising results [PK19]. The problem of learning only expected values, which can strongly impair the planner performance and endanger safety, has been successfully solved by applying distributional reinforcement learning [Kam+21].

Other works motivate from end-to-end learning and aim to output control commands. One early work utilizes recurrent neural networks and feeds this with a mid-level representation to output the most likely motion [BKO19]. Another one creates a set of motion plans, and utilizes the learned framework and then chooses among these [Zen+19; CSU21]. Creating multi-modal and interactive motion plans has also recently become possible [Sal+20; Iva+20]. A notable drawback of these deep learning based neural planners is both the amount of examples they require, and missing guarantees on safety. Even though these approaches are quite formative in their direct application to motion planning, they can be combined with classical approaches serving them as a safety net.

3 Modeling Uncertain Information

Planning under uncertainty requires modeling uncertain information as *random variables*, i.e. the value of which cannot be predicted with certainty because its possible outcome depends on a random phenomenon. Even though the value of a random variable is not known a priori, the probability of a value it may take can be modeled and estimated. A function that assigns a probability for each random outcome is called the *probability distribution* of a variable. States and measurements are typically modeled as probability distributions over continuous spaces.

There are two major approaches for uncertainty modeling: parametric and nonparametric approaches. This chapter first presents both approaches and then continues with introducing the notion of information.

3.1 Parametric Approaches

Parametric approaches express probability distributions with a constant number of parameters. This allows closed form solutions, which are favorable in many applications.

3.1.1 Gaussian Distribution

The most common distribution in continuous spaces is the *Gaussian* or *normal* distribution [Lyo14]. Following the central limit theorem, under certain mild assumptions, the normalized sum of independent random variables approximately follows the Gaussian distribution $\mathcal{N}$ as the number of variables tends to infinity.

Measurements of multiple random variables can be modeled with *multivariate* Gaussian distribution. For a d-dimensional random variable $\boldsymbol{x}$, the probability density function (PDF) is

$$P(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}} \cdot \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{3.1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is a positive definite covariance matrix. Gaussian distributions have the nice property that an addition, subtraction, or multiplication of two Gaussian distributions result in a new Gaussian distribution, if normalized appropriately.

A special case of the Gaussian distribution is the *univariate* Gaussian distribution (UVG), where the random variable has a size of one. The density of a univariate Gaussian distribution is obtained by scaling it to the *standard* Gaussian distribution $\mathcal{N}(0,1)$ and calculating the PDF $\phi(x)$ or cumulative density function (CDF) $\Phi(x)$ of the standard Gaussian distribution. There are computationally efficient numerical approximations for these expressions [Pre+92, p. 213]. Therefore, it is desirable to model uncertainties with univariate Gaussian distributions in real-time applications.

Position or velocity measurements are, however, two-dimensional, resulting in a *bivariate* Gaussian distribution. If x and y denote the axes of the Cartesian frame, the mean vector and the covariance matrix are defined as

$$\boldsymbol{\mu} = \begin{bmatrix}\mu_{\mathrm{x}}\\ \mu_{\mathrm{y}}\end{bmatrix},\ \boldsymbol{\Sigma} = \begin{bmatrix}\sigma_{\mathrm{x}}^2 & \rho\sigma_{\mathrm{x}}\sigma_{\mathrm{y}}\\ \rho\sigma_{\mathrm{x}}\sigma_{\mathrm{y}} & \sigma_{\mathrm{y}}^2\end{bmatrix} \tag{3.2}$$

where ρ is the correlation coefficient between x and y components, and $\sigma_{\mathrm{x}} > 0$ and $\sigma_{\mathrm{y}} > 0$ are standard deviations in the axes of the coordinate frame.

In many cases, the standard deviation for a specific direction is of interest. For measurement $\boldsymbol{x}$ modeled with $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a linear transformation of a rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ results in

$$\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{R}\boldsymbol{\mu} + \boldsymbol{t}, \boldsymbol{R}\boldsymbol{\Sigma}\boldsymbol{R}^{\mathsf{T}}), \tag{3.3}$$

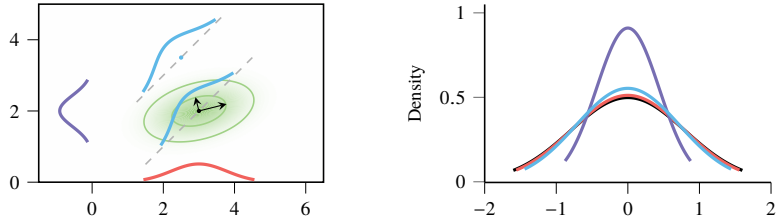
which is a new multivariate Gaussian distribution. A clear result of the equation above is that the covariance matrix is invariant to translation. This is an intuitive result, as the covariance matrix defines the correlation between axis directions.

Measurements $\hat{\boldsymbol{x}}$ are frequently modeled as their expected value $\boldsymbol{x} = \mathbb{E}[\hat{\boldsymbol{x}}]$ with an additive white noise $\zeta \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, i.e. $\hat{\boldsymbol{x}} = \boldsymbol{x} + \zeta$. The expected value can be used to define the center of the rotation, in which the transformation defined in Equation (3.3) reduces to an additive translation of the mean value.

A covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ can be projected onto a single direction by defining a vector $\boldsymbol{u} \in \mathbb{R}^d$ of unit length. The projection results in a UVG with

$$\sigma_{\boldsymbol{u}}^2 = \lambda_{\boldsymbol{u}}(\boldsymbol{\Sigma}) = \boldsymbol{u}\boldsymbol{\Sigma}\boldsymbol{u}^\mathsf{T}. \tag{3.4}$$

This allows decoding the variance in a specific direction, see Figure 3.1.



(a) Distributions. (b) Densities.

Figure 3.1: Eigenvectors and various projections of a bivariate Gaussian distribution $\mathcal{N}(\left[\begin{smallmatrix}3\\2\end{smallmatrix}\right], \left[\begin{smallmatrix}0.61 & 0.12\\0.12 & 0.19\end{smallmatrix}\right])$. The projections onto x and y axes are shown in red and purple, whereas the projection on lines of 45° angles are shown in light blue. Despite the translation, the shape of the distribution remains the same. In the right subfigure, the individual densities are depicted in their respecting color. The black density in the back corresponds to the biggest variance.

A covariance matrix can be rotated such that ρ becomes zero, the covariance matrix becomes diagonal, and hence, the spread is maximized in one direction. The vectors that point to the principal directions of the matrix together with their magnitude can be calculated with principal component analysis. This can be used to define an upper bound of the variance for every possible rotation.

Gaussian distributions assign a nonzero probability value to every value in the interval $(-\infty, \infty)$, i.e. they have *infinite support*. In many cases, including sensor measurements, both a lower and an upper bound can be specified. The *truncated* Gaussian distribution allows for bounding the support of the Gaussian distribution, while preserving its desirable general features [Bur14]. Density calculation of a truncated univariate Gaussian is provided in Appendix A.1.1

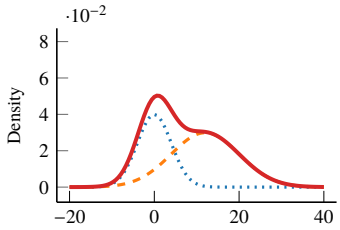
3.1.2 Mixture Distributions

Random variables in some cases can only be poorly represented unimodal. For instance, the distribution of the longitudinal position of a vehicle is typically characterized by a multi-modal distribution, the number of modalities depending on the number of its maneuver alternatives. Such a distribution can be modeled as a linear combination of basic distributions, as a *mixture distribution*.

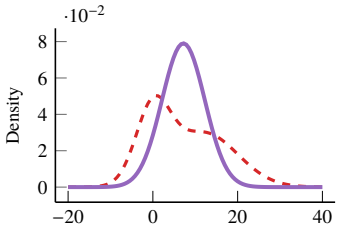
In cases where the underlying components are Gaussian, the density is given by

$$P(x \mid \mu_1, \sigma_1, w_1, \ldots, \mu_n, \sigma_n, w_n) = \sum_{i=1}^{n} w_i \mathcal{N}(x \mid \mu_i, \sigma_i) \tag{3.5}$$

where w_i are *mixing coefficients* such that $\sum_{i=1}^{n} w_i = 1$ and $w_i \geq 0$. Figure 3.2 depicts a mixture distribution with Gaussian components.



(a) A mixture of $\mathcal{N}(0, 4)$ and $\mathcal{N}(12, 8)$ with weights $w_1 = 0.4$ and $w_2 = 0.6$, respectively.



(b) An approximation of the mixture distribution with a single $\mathcal{N}(7.20, 5.05)$.

Figure 3.2: A Gaussian mixture distribution that consists of two Gaussian distributions. An approximation with a single UVG yields poor results.

Mixture distributions enable representing complex densities with basic components. However, if the observations do not have the form of a unimodal distribution, the inference of the mixture distribution becomes a challenging task. One way is to formulate it as a joint probability distribution of observed and latent variables, and match them with likelihood maximization algorithms [Bis06, p. 430]. However, these methods are based on local optimization algorithms, and therefore, are prone to errors in case of singularities and local minima.

3.2 Nonparametric Methods

An exact inference for most of the probabilistic models in real-world applications is computationally intractable. A remedy is to use a finite number of random samples to find an approximate result. Algorithms of this kind are known as *Monte Carlo* methods.

Monte Carlo methods are utilized either for generating random samples from a given probability distribution, or estimating the expected value of a function given a probability distribution with randomly created samples [RC04, p. 79]. Because they operate with samples and do not make any assumption on the form of the distribution, they are nonparametric and have a very high representation power. An important property of Monte Carlo estimation is that the obtained accuracy depends only on the variance of the probability distribution and not on its dimensionality. Therefore, it is possible to reliably estimate the distribution with only a small number of independent samples [Mac03, p. 358]. However, obtaining independent samples is not easy. For this reason, increasing the number of samples typically results in better accuracy [Bis06, p. 524]. But, this comes at the cost of increased computational complexity.

3.2.1 Sampling Algorithms

There are many distributions from which direct sampling is not possible. However, if the functional form of their density is known, sampling can be done by utilizing distributions, which are more easily sampled. Let $P(x)$ be such a complex *target* distribution. If the functional form of its density is known, such that

$$P(x) = \widetilde{P}(x)/z \tag{3.6}$$

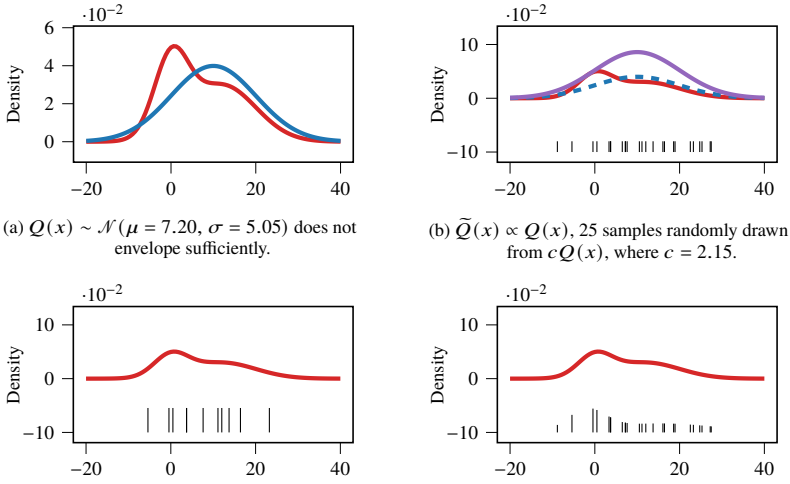
where z is the normalizing constant, sampling can be done by using a *proposal* distribution $Q(x)$ which is more easily sampled (see Figure 3.3a).

Rejection Sampling

The idea in rejection sampling, also known as *accept-reject* sampling, is to generate a sample point $o_i \sim Q(x)$ and then to perform uniform sampling

$u_i \sim \mathcal{U}([0, cQ(o_i)])$ for that point, independently. If $u_i \leq \widetilde{P}(o_i)$, the sample point is accepted, otherwise it is rejected. The constant c is a scaling factor that ensures $cQ(o_i) \geq \widetilde{P}(o_i)$ for all values of x (see Figures 3.3b and 3.3c).

The acceptance probability of samples is $\widetilde{P}(x)/cQ(x)$. Therefore, rejection sampling works best, if $Q(x)$ is a good approximation to $P(x)$. With increasing dimension, c grows exponentially. For this reason, rejection sampling counts as an inefficient algorithm for sampling in bigger dimensions than one or two [Bis06, p. 532].



(a) $Q(x) \sim \mathcal{N}(\mu = 7.20, \sigma = 5.05)$ does not envelope sufficiently.

(b) $\widetilde{Q}(x) \propto Q(x)$, 25 samples randomly drawn from $cQ(x)$, where $c = 2.15$.

(c) 56% of the samples are rejected.

(d) All samples are retained and weighted.

Figure 3.3: Comparison between rejection sampling and importance sampling.

Importance Sampling

In applications where the goal is the evaluation of expected values, rejection sampling suffers from a major problem. In high dimensional problems, particularly in cases where the density is confined in small regions, only a tiny amount of particles is retained. Instead of comparing the samples against a criterion and in case rejecting them, they can be evaluated with respect to their *importance*.

Importance sampling calculates the importance of samples from $o_i \sim Q(x)$ by

$$\widetilde{w}_i = \frac{P(o_i)}{Q(o_i)}. \tag{3.7}$$

After the importance of all samples have been calculated, the particle weights are normalized $w_i = \widetilde{w}_i / \sum_{j=1}^{n} \widetilde{w}_j$, where n is the number of samples (see Figure 3.3d). The expected value of a function $F(x)$ with respect to $P(x)$ can be calculated from the particle set $\{(o_i, w_i)\}_{i=1}^{n}$ as

$$\mathbb{E}[F] = \int F(x)P(x)\,\mathrm{d}x \simeq \sum_{i=1}^{n} w_i F(o_i). \tag{3.8}$$

With $n \to \infty$ this approximation converges to the real expectation [CD02].

Likewise in rejection sampling, importance sampling depends on how well $Q(x)$ matches $P(x)$. But retaining them allows for efficient recycling, especially while processing sequential data [Mac03, p. 103]. Increasing dimensions pose a problem on importance sampling, by causing some weights to dominate others and reducing the number of effective samples. But, the so-called *degeneracy* problem can be alleviated with decent resampling procedures in sequential processing [Aru+02].

3.2.2 Kernel Density Estimation

Sampling algorithms represent continuous probability distributions with discrete, and a finite number of samples. In many applications, it is not sufficient to represent densities with particles. A continuous representation of the density is often required. *Kernel density estimation* (KDE) is an efficient approach for extracting density from a set of samples.

KDE performs regression of samples to so-called kernel functions K and weights the samples according to their distance from the kernel center. For weighted samples $\{(\boldsymbol{o}_i, w_i)\}_{i=1}^{n}$ in d-dimensional space, the density is given by

$$\hat{F}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i K_{\boldsymbol{H}}(\boldsymbol{x} - \boldsymbol{o}_i) \tag{3.9}$$

where K_H is the scaled kernel

$$K_{\boldsymbol{H}}(\boldsymbol{x}) = |\boldsymbol{H}|^{-1/2} K\left(\boldsymbol{H}^{-1/2}\boldsymbol{x}\right) \tag{3.10}$$

and $\boldsymbol{H} \in \mathbb{R}^{(d\times d)}$ is the symmetric, positive definite *smoothing matrix* or *window width matrix* [Sil86, p. 76; Gra18, p. 35]. A kernel function must have its maximum at 0, be symmetric around this value, and must integrate to one. There are many functions that satisfy these properties. However, compared with its parametric form, the width of the kernel has a bigger impact on the fit. Whereas wide kernels cause underfitting by introducing excessive smoothing, narrow kernels make them too responsive to the individual points and cause overfitting.

A common choice for the kernel function is the zero-mean Gaussian distribution

$$K(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{x}\right). \tag{3.11}$$

For windowing $\boldsymbol{H}$, a diagonal matrix is chosen to control smoothing in every dimension independently. The optimal smoothing factor for Gaussian kernels, or the *bandwidth*, is provided by *Silverman's rule of thumb*

$$\sqrt{\boldsymbol{H}_{ii}} = \left(\frac{4}{d+2}\right)^{\frac{1}{d+4}} \hat{\sigma}_i n^{-\frac{1}{d+4}}, \quad i = 1, \ldots, d, \tag{3.12}$$

where $\hat{\sigma}_i$ is the estimated standard deviation in the i^{th} dimension. In the univariate cases this reduces to $\sigma_{\text{opt}} \approx 1.06\hat{\sigma}n^{-1/5}$ [Sil86, p. 87]. Even though Silverman's rule of thumb is popular for choosing bandwidths, it performs suboptimally if the underlying density is not Gaussian [Sil86, p. 45].

3.3 Quantifying Information

Any random variable that can take multiple, distinct values can be characterized by the piece of *information* it carries. In any specific realization, increasing number of possible outcomes reduces the information content of the realization. Likewise, in case of random variables, the realization of a random variable given a probability reflects the information content of that realization. Hence, the value of information can be seen as *"the amount of uncertainty that would*

be removed upon the realization of a random variable". This suggests that the value of information is inversely proportional to the present uncertainty.

3.3.1 Variance of Estimates

The direct relation between the amount of information and uncertainty can raise the idea to use arguments used in parametric representations of probability distributions for information quantification. In case the uncertainty estimation is unbiased and is modeled as a Gaussian distribution, functions mapping covariance matrices into scalars can be used for quantifying information. They either calculate the norm, or the trace, or the area of the corresponding ellipsoid of the covariance matrix [BV04, p. 387]. Even though these are computationally cheap and allow to quantify the amount of information [Mih+02], they can only be used for uncertainty models of unimodal Gaussian densities.

3.3.2 Entropy

Hartley points out that the logarithm is the best measure to calculate the amount of information a realization carries [Har28]. Building upon his work, Shannon defines the properties an information measure should carry and defines the notion of *entropy* H of a discrete random variable as

$$H(x) = \mathbb{E}\left[-\log P(x)\right] = -\sum_{x} P(x) \log P(x) \tag{3.13}$$

which satisfies all those properties [Sha48].

Entropy calculation adopts the convention $0 \log(0) = 0$, and hence $H(x) = 0$ if $P(x = i) = 0$ for all $i \in \mathbb{N} \setminus \{k\}$ and $P(x = k) = 1$. This is highly intuitive, as adding outcomes of variables with zero probability, or deterministic variables in general, does not carry any information and shall not change the entropy.

Several further measures are discussed in Appendix A.1.2. In tasks where gathering information and reducing the uncertainty are aimed, the choice of the information measure is not decisive. Therefore, entropy serves as a simple yet reliable information measure.

4 Optimal Proactive Planning

The planning module has to tackle the uncertainties that accumulate up to motion planning. The planner must process these uncertainties, and accordingly plan proactive motion profiles while simultaneously attempting to maximize its driving goals. This chapter starts with presenting a mathematical formulation of driving goals and vehicle models. It subsequently argues how to maintain safety in planning by utilizing the aforementioned uncertainty models. It derives the required conditions to ensure safety which will be integrated into the solver algorithms in the following chapters.

4.1 Optimal Motion Planning

The settings a vehicle has to operate in, as well as approaches for modeling uncertainties are presented in the previous chapters. Accordingly, the objective of planning that is already defined in Section 2.2 is now presented in detail.

4.1.1 Planning in Receding Horizon

Available computational resources and changing environment conditions require iterative planning in a receding horizon fashion for a finite time. The motions of the agents in the environment are predicted for the predefined time horizon and the control inputs that optimize the state and control requirements of the ego vehicle are calculated repeatedly using a model.

The time-discretized planning problem is already presented in Equation (2.4). A planning horizon t_h with a sampling interval of t_s can be approximated with n colocation points with $t_\mathrm{h} = n \times t_\mathrm{s}$. However, the number of free *trajectory support points* $[\boldsymbol{x}_i, \boldsymbol{a}_i]$ is not equal to n, as during computation t_c a number of

control inputs n_{pin} must be kept fixed for temporal stability. The time duration they are *pinned* t_{pin} must satisfy

$$t_{\text{c}} \leq t_{\text{pin}}, \tag{4.1}$$

which results in

$$n_{\text{pin}} = \lceil t_{\text{pin}}/t_{\text{s}} \rceil. \tag{4.2}$$

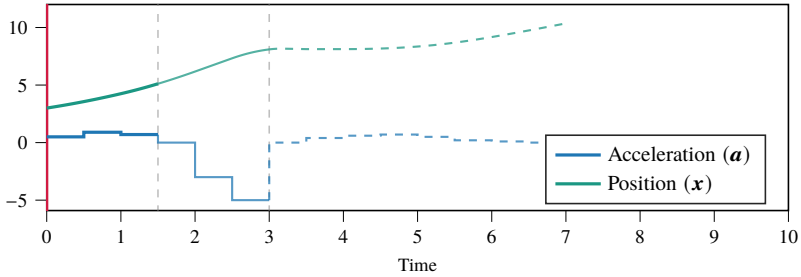
The time calculations introduced so far are intuitive and simple. However, they do not incorporate delays in perception, and assume that an environment model is present at planning time t_0. In reality, the environment information has a delay of t_{p}. For this reason, the time at which planning is performed t_0 is set to a time in the past $t_0 = t_{\text{now}} - t_{\text{p}}$, and therefore,

$$t_{\text{c}} + t_{\text{p}} \leq t_{\text{pin}} \tag{4.3}$$

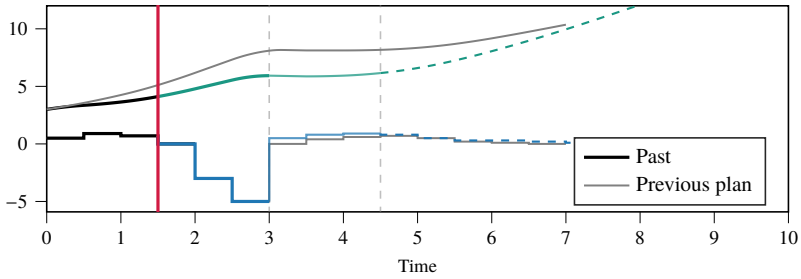
must hold. For brevity, the following definitions assume the presence of an up-to-date environment model at t_0 and set $t_0 = 0$. Furthermore, it must be underlined that the execution of a planned motion is typically subject to time delays, and in some cases it might not be sufficient to cover these delays in feedback fashion. If the delays are not negligible and have a systematic behavior, either the vehicle model must include a model of the time-delay, or alternatively, the planned trajectory support points can be shifted in time and the control reference can be applied earlier.

A planner computes a motion profile for the time interval $(t_0, t_{\text{h}}]$, while it can optimize only the time interval $(t_{\text{pin}}, t_{\text{h}}]$. The interval $(t_{\text{pin}}, 2t_{\text{pin}}]$ is set fixed in the next replanning step, whereas the rest will be recomputed in the subsequent planning instances. Since the actions planned for this time interval are irreversible, following the safety definition of Petti et al., the vehicle must ensure that any state in this time interval is not an *inevitable collision state*, i.e. does not lead to a collision at a time after this interval [PF05; TS18].

Figure 4.1 depicts an online algorithm performing planning in a receding horizon. The execution of the motion planned in Figure 4.1a results in the profile depicted in Figure 4.1b. Due to some modeling errors, the vehicle is at another state than expected. The control profile $t \in (t_{\text{pin}}, 2t_{\text{pin}}]$ of the first planning instance is still kept fixed for temporal consistency.



(a) Planning performed at $t_0 = 0.0$. Acceleration at $t \in [t_0, t_{\text{pin}}]$ is pinned, and $t \in (t_{\text{pin}}, 2t_{\text{pin}}]$ is will be kept fixed during the next replanning.



(b) Replanning performed at $t_0 = 1.5$. The gray line depicts the state profile under the previous control profile, whereas the black one depicts the driven motion. The offset between them depicts the modeling errors.

Figure 4.1: Planning in receding horizon. The time of planning is denoted with a red vertical line. Planned control inputs, i.e. acceleration, is depicted in blue and the resulting state, i.e. position, is depicted in green. Due to modeling errors, the control inputs planned at the top figure did not bring the vehicle into the desired state. The planner alters the motion profile $t \in (t_{\text{pin}}, t_{\text{h}}]$ to correct this error.

The discretized motion profile, the *parameter vector* $\boldsymbol{p} \in \mathbb{R}^{(n+1)\times(n_{\boldsymbol{x}}+n_{\boldsymbol{a}})}$, of the motion planning problem subject to these time interval definitions correspond to

$$\boldsymbol{p} = [\underbrace{\boldsymbol{x}_0, \boldsymbol{a}_0}_{\text{measurement}}, \underbrace{\boldsymbol{x}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{x}_{\text{pin}}, \boldsymbol{a}_{\text{pin}}}_{\text{pinned}}, \underbrace{\boldsymbol{x}_{\text{pin}+1}, \boldsymbol{a}_{\text{pin}+1}, \ldots, \boldsymbol{x}_{2\text{pin}}, \boldsymbol{a}_{2\text{pin}}}_{\text{executed next}},$$
$$\underbrace{\boldsymbol{x}_{2\text{pin}+1}, \boldsymbol{a}_{2\text{pin}+1}, \ldots, \boldsymbol{x}_n, \boldsymbol{a}_n}_{\text{will be replanned}}]^{\mathsf{T}}. \tag{4.4}$$

Pinning of the terms in $\boldsymbol{p}$ can be done both by treating the pinned as *non-variables*, in case the chosen solver allows this, or by entering these terms as *variables* and applying additional equality constraints to fix their values. Another approach is to trim the trajectory and keep the values from index (pin) on. This approach is not preferred in work as including the "pinned" terms inside $\boldsymbol{p}$ eases both understanding the "executed next" terms and an inspection of eventual non-uniform time delays in environment perception t_p. Furthermore, it presents the general form, and thereby is valid for planners that rely on differential flat vehicle models.

4.1.2 Vehicle Models

Motion planning must observe the dynamics of the vehicle. Any trajectory support point must satisfy

$$\dot{\boldsymbol{x}} = \mathbf{g}(\boldsymbol{x}, \boldsymbol{a})$$

where $\mathbf{g}$ is a nonlinear equation reflecting the motion of the vehicle. Vehicles are subject to nonholonomic constraints, indicating that certain values of state variables depend on their time derivatives as the underlying constraints limiting the state variables cannot be fully integrated. This results in complex vehicle equations, eventually having a significant impact on the planner performance.

There is a wide range of models available, starting from very rough estimates for behavior planning tasks to very precise models used for driving at the limits of handling [GGG20].

Point Mass Model

The simplest approach to model the motion of the vehicle is to assume it in the form of a point mass. This model neglects yaw dynamics, and the states are obtained by two integrators. The states and controls are defined as

$$\boldsymbol{x} = [x, y, v_\mathrm{x}, v_\mathrm{y}]^\mathsf{T} \tag{4.5a}$$

$$\boldsymbol{a} = [a_\mathrm{x}, a_\mathrm{y}]^\mathsf{T}. \tag{4.5b}$$

The dynamics of the vehicle are modeled with the friction circle

$$\sqrt{a_x^2 + a_y^2} \leq \gamma g\,, \tag{4.6}$$

where γ is the friction coefficient and g is the gravitational acceleration.

The point mass model is linear and can be calculated easily. However, the heading angle, which is indispensable for collision avoidance, can only be estimated from successive states. Furthermore, it neglects yaw dynamics and is therefore bound to big errors even at moderate speeds.

Kinematic Bicycle Model

Yaw dynamics can be integrated into the calculations by utilizing the kinematic bicycle model. The model is *kinematic*, as it does not consider tire forces and ignores the longitudinal and lateral slip for both tires. The vehicle motion is described only by geometric equations, see Figure 4.2.

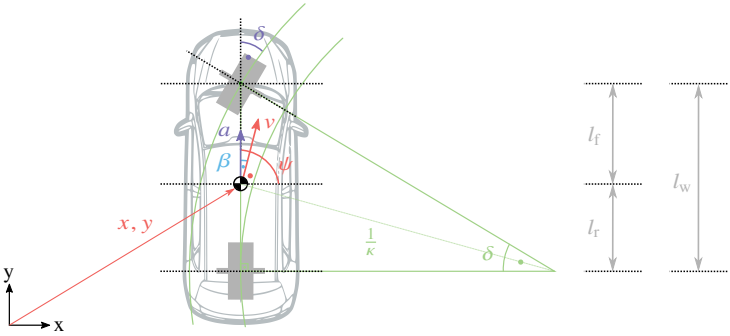


Figure 4.2: Geometry behind the kinematic bicycle model.

The states and controls are defined as

$$\boldsymbol{x} = [x, y, \psi, v]^\mathsf{T} \tag{4.7a}$$

$$\boldsymbol{a} = [\delta, a]^\mathsf{T}. \tag{4.7b}$$

Motion equations can be given as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos(\psi+\beta) \\ v\sin(\psi+\beta) \\ \frac{v}{l_\mathrm{w}}\tan(\delta)\cos(\beta) \\ a \end{bmatrix}, \tag{4.8a}$$

with side-slip angle

$$\beta = \arctan\left(\frac{l_\mathrm{r}}{l_\mathrm{w}}\tan(\delta)\right), \tag{4.8b}$$

which is a function of the steering angle δ, as the dynamics of the tires are ignored [Raj11, p. 24]. The dynamic limits of the vehicle are represented by two constraints on acceleration in longitudinal and lateral directions

$$a_\mathrm{lon}^- \leq a \leq a_\mathrm{lon}^+, \tag{4.9a}$$
$$a_\mathrm{lat}^- \leq a_\mathrm{lat} \leq a_\mathrm{lat}^+, \tag{4.9b}$$

where the lateral acceleration is obtained as

$$a_\mathrm{lat} = \left(\frac{v^2}{l_\mathrm{w}}\right)\tan(\delta)\cos(\beta). \tag{4.10}$$

At speeds lower than 30 km/h the side-slip angle β is negligible, and the vehicle model becomes linear. Particularly, the vehicle model becomes *differentially flat*, allowing all states and variables to be expressed with time derivatives of *flat outputs*

$$x, y, \psi, v, \delta, a \longleftrightarrow x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}. \tag{4.11}$$

This eases the application of constraints regarding the vehicle model considerably, [Fli+95; Cho+05, p. 448; Zie+14b; Taş+18]. The differentially flat kinematic bicycle model is sometimes referred to as the *continuous-steering car* [LaV06, p. 614].

The kinematic bicycle model can yield instantaneous changes of steering angle. To ensure continuous changes, either the rate of change can be constrained and penalized, or a low-pass behavior can be applied.

4.1.3 Optimality Criteria and Cost Functional

The primary objective of a motion planner is to plan a comfortable and safe ride that leads to the desired end state in a reasonable amount of time. Whereas safety is an aspect that needs to be satisfied at all times, comfort and travel-time are contradicting objectives that need to be balanced. These objectives can be defined as *value* and *range* cost terms of the states and control variables building up the running costs L in Equation (2.4). Value cost for a variable x can be defined as

$$J_{\text{val}}(x) = w_{\text{val},\,x} \, \|x^* - x\|^2, \tag{4.12}$$

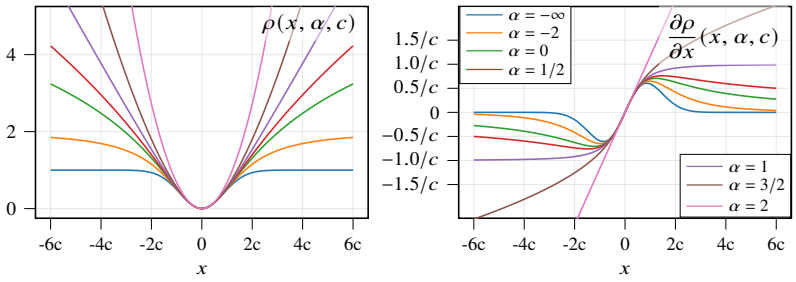
where x^* is the desired value, and $w_{(\cdot)}$ is the corresponding weighting factor. Likewise, in cases where there is not a single value of an optimum, a range cost constituting a tolerance band from a lower bound x^- until an upper bound x^+ can be defined

$$J_{\text{ran}}(x) = \begin{cases} w_{\text{ran},\,x} \, \|x - x^+\|^2 & x^+ < x, \\ 0 & x^- \leq x \leq x^+, \\ w_{\text{ran},\,x} \, \|x^- - x\|^2 & x < x^-. \end{cases} \tag{4.13}$$

The range cost definition includes conditional statements. The utilization of such statements is not straightforward in parametric approaches. If such conditional statements can be utilized, an *asymmetric* cost around an optimal value x^* can be defined. This is especially useful, when, for example, travel speed is considered. Speeds higher than the allowed maximum speed are rarely acceptable, and hence, can be punished with quadratic loss. Traveling at slower speeds is often an ordinary behavior, such as in dense traffic, and therefore can be penalized with a tolerant loss, such as *Cauchy loss*, see Figure 4.3. Equation (4.14) presents such an asymmetric loss

$$J_{\text{asym}}(x) = \begin{cases} w_{\text{asym},x} \, \|x - x^*\|^2 & x \geq x^*, \\ w_{\text{asym},x} \, \log\left(1 + \|x - x^*\|^2\right) & x < x^*. \end{cases} \tag{4.14}$$

While using a tolerant loss function in cost, it is important to observe that the gradient does not diminish in the operating value interval unintentionally.



(a) Loss function.

(b) Derivative of the loss function

Figure 4.3: A general and adaptive loss function $\rho(x, \alpha, c)$ [Bar19], see Appendix A.2.3. $\alpha = 0$ corresponds to Cauchy loss and $\alpha = 2$ corresponds to quadratic loss. Both are symmetric in the vicinity of $x = 0$.

The running costs of the planning problem can be defined as

$$L(\boldsymbol{x}_0, \boldsymbol{a}) = J_{\text{total}} = J_{\text{driving}} + J_{\text{comfort}} \tag{4.15}$$

with

$$J_{\text{driving}}(\boldsymbol{x}) = J_{\text{asym}}(\boldsymbol{x}_v) \tag{4.16}$$

and

$$\begin{aligned} J_{\text{comfort}}(\boldsymbol{x}, \boldsymbol{a}) = J_{\text{val}}(\boldsymbol{a}_a) + J_{\text{val}}(\boldsymbol{a}_{\text{lat}}) + J_{\text{val}}(\boldsymbol{j}) \\ + J_{\text{ran}}(\boldsymbol{a}_a) + J_{\text{ran}}(\boldsymbol{a}_{\text{lat}}) + J_{\text{ran}}(\boldsymbol{j}), \end{aligned} \tag{4.17}$$

where $\boldsymbol{x}_v$ represents the speed values in the state vector, $\boldsymbol{a}_a$ represents the longitudinal acceleration values in the control vector, $\boldsymbol{a}_{\text{lat}}$ represents the lateral accelerations calculated using Equation (4.10), and $\boldsymbol{j}$ is the jerk values calculated from longitudinal acceleration values by using finite differences. The lateral acceleration is penalized separately from the longitudinal component to control the lateral maneuvering behavior of the vehicle. The use of jerk terms serves for further smoothing of the control inputs and is optional. The optimal values of all the comfort terms are zero, whereas for the driving goal, the desired speed v_{des} can be taken as 90% of the speed limit. The individual weighting factors of the cost terms can be tuned manually or learned from driving data. The range terms in the second row mimic naturalistic driving style.

Homotopy-Class-Free Planning

A comfortable, optimal motion can easily be planned with the equations above in free driving. However, in the presence of other agents, optimal planning becomes a challenging task. In an intersection scenario, where the ego vehicle has the right-of-way, the maneuver intention of the oncoming vehicle still might be unclear: it could perform an aggressive maneuver to cross the intersection first, eventually letting its behavior be classified as *noncompliant* (see Figure 4.4). Conventional planning approaches use the most likely maneuver of the oncoming vehicle and plan an appropriate safe motion profile [Taş14; Ben+15; PLG21]. However, due to current uncertainty, the most likely maneuver cannot always be identified with high confidence. In circumstances, the ego vehicle might even identify the true maneuver as slightly more unlikely, and in turn execute a suboptimal behavior.

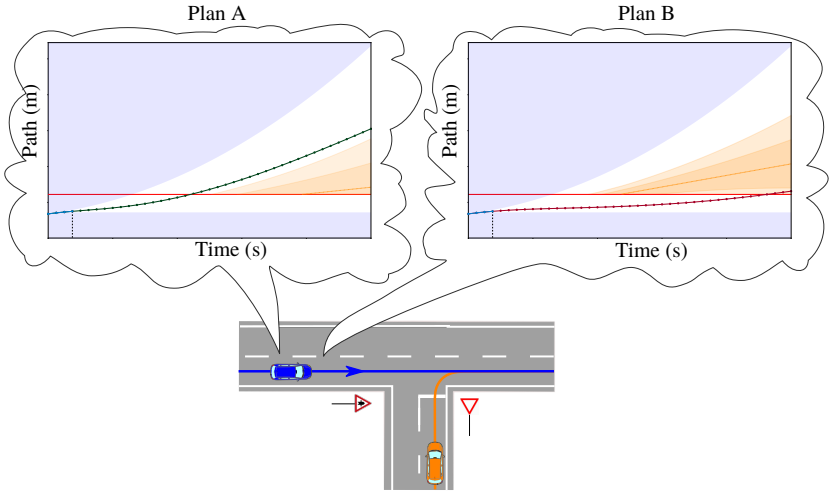


Figure 4.4: An intersection scenario where the ego vehicle (blue) has the right-of-way. It is unclear if the orange vehicle will yield to it. The ego vehicle has two maneuver alternatives depending on the intention of the orange one: Plan A and Plan B. The blue areas illustrate the unreachable regions by the ego vehicle. The orange areas correspond to the predicted position of the orange vehicle, which is modeled with a truncated univariate Gaussian distribution. The red line indicates the point at which the driving corridors intersect. The motion plans for each case are depicted in green and red.

A motion planner should ideally consider the confidence in maneuver prediction while planning a motion profile. For this, the parameter vector of the planning problem is extended to consider both maneuver options simultaneously. Next, because the parameter values are defined by the cost function and applied constraints, cost terms of individual maneuvers are weighted with their probabilities. The resulting motion profile is *homotopy-class-free* and its maneuver is *neutral* [THS18].

The parameter vector $\boldsymbol{p}$ of such a motion is

$$\boldsymbol{p} = [\ \overbrace{\boldsymbol{x}_0, \boldsymbol{a}_0}^{\text{measurement}}\ , \overbrace{\boldsymbol{x}_1, \boldsymbol{a}_1, \dots, \boldsymbol{x}_{\text{pin}}, \boldsymbol{a}_{\text{pin}}}^{\text{pinned}}, \overbrace{\underbrace{\boldsymbol{x}_{\text{pin}+1}, \boldsymbol{a}_{\text{pin}+1}, \dots, \boldsymbol{x}_{2\text{pin}}, \boldsymbol{a}_{2\text{pin}}}_{\text{C}}}^{\text{shared \& executed next}}, \underbrace{\boldsymbol{x}_{2\text{pin}+1}, \boldsymbol{a}_{2\text{pin}+1}, \dots, \boldsymbol{x}_n, \boldsymbol{a}_n}_{\text{CA}}, \underbrace{\boldsymbol{x}_{n+1}, \boldsymbol{a}_{n+1}, \dots \boldsymbol{x}_{2n-2\text{pin}}, \boldsymbol{a}_{2n-2\text{pin}}}_{\text{CB}}]^{\mathsf{T}}. \tag{4.18}$$

Figure 4.5 illustrates a comparison between a homotopy-class-bound and a homotopy-class-free motion profiles. The underbraces in Equation (4.18) denote to which maneuver segments in Figure 4.5 the parameter vector elements correspond. Given $\boldsymbol{p}$, parameter vectors of maneuvers A and B with neutral component C can be defined as

$$\boldsymbol{p}^{\star}_{\text{A}} = \left[\boldsymbol{p}_{0:\text{pin}}, \boldsymbol{p}_{\text{pin}+1:2\text{pin}}, \boldsymbol{p}_{2\text{pin}+1:n}\right]^{\mathsf{T}}, \tag{4.19a}$$

$$\boldsymbol{p}^{\star}_{\text{B}} = \left[\boldsymbol{p}_{0:\text{pin}}, \boldsymbol{p}_{\text{pin}+1:2\text{pin}}, \boldsymbol{p}_{n+1:2n-2\text{pin}}\right]^{\mathsf{T}}. \tag{4.19b}$$

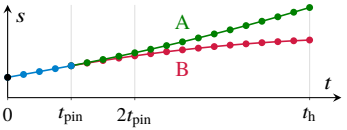
The cost of the neutral plan is

$$J^{\star}_{\text{total}} = w_{\text{A}}\, J_{\text{total}}\left(\boldsymbol{p}^{\star}_{\text{A}}\right) + w_{\text{B}}\, J_{\text{total}}\left(\boldsymbol{p}^{\star}_{\text{B}}\right), \tag{4.20}$$

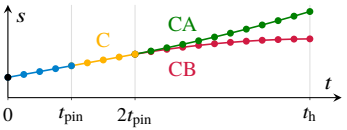
where

$$w_m = F_w(p_m) \tag{4.21}$$

is the weighting factor, which is a function of the probability of the specific maneuver p_m. It should be underlined that both parameter vectors $\boldsymbol{p}^{\star}_{\text{A}}$ and $\boldsymbol{p}^{\star}_{\text{B}}$ are subject to the same constraints as $\boldsymbol{p}$ of a conventional approach defined in Equation (4.4).



(a) Conventional planners decide on the maneuver option, selecting either A or B.



(b) Homotopy-class-free planning can plan a neutral maneuver C.

Figure 4.5: Motion profile comparison of homotopy-class-bound and homotopy-class-free planning on path-time diagram. $\mathbf{s}_{\text{B},i} \preceq \mathbf{s}_{\text{C},i} \preceq \mathbf{s}_{\text{A},i}$, $\forall i \in \{n_{\text{pin}}+1, \ldots, 2n_{\text{pin}}\}$.

Homotopy-class-free planning creates motion plans that do not necessarily trigger harsh braking. The plans have low *transparency* and in this way, the ego vehicle's reckoning with an unfavorable maneuver of the oncoming vehicle cannot be easily revealed. While such scenarios would result in immediate defensive behaviors in conventional planners, the defensiveness and the maneuver transitions of homotopy-class-free planning is controlled with the function F_w.

The presented planning scheme requires $(n-2n_{\text{pin}})\times(n_{\boldsymbol{x}}+n_{\boldsymbol{a}})$ more parameters per extra maneuver considered. However, the number of alternative maneuver options can be reduced, and usually two competing options remain. In cases where there is a single maneuver available, such as in straight driving, the weight factor of the parameters reserved for the second maneuver is set to zero and its constraints are set to non-binding values.

Executing neutral plans may result in unsafe situations. Ensuring safe, neutral plans will be presented in the next chapter, after tackling safety in the next sections.

Uncertainties in Motion Profiles

Apart from the uncertainty in maneuver intentions of other participants, consideration of uncertainties in their motion profiles with additional cost terms gives the planner an estimate on their current and future distribution, eventually improving comfort. Its formulation will be covered in the next chapter.

Changes in Driving Objectives and Constraints

In some cases, driving objectives and bounds of the constraints acting on the motion planner change upon newly arrived information. This causes jerky motions despite the comfort terms defined in Equation (4.17). In case the changes are not safety-critical, countermeasures can be taken. A remedy is to dynamically adapt the weighting factors and rebalance the importance of comfort terms. This can require excessive parameter tuning, unless they are learned from driving data. A more practical approach is to *gradually* activate a cost term reflecting the constraint into the optimization problem or change the desired value. Driving experiments show that even a linear activation or deactivation of constraints results in very smooth maneuvers [Taş+18].

4.2 Definitions of Terms Related with Safety

Robustness, reliability, criticality, risk, and safety are related terms for safety analysis. Although being closely related, they have different foci and clarifying these is essential for understanding the contributions of this work.

Robustness is the ability to adapt and operate properly under a variety of conditions, including invalid inputs or adverse environment conditions [Taş+16]. Whereas *reliability* is the ability of a system to function for a specific period of time under the predefined operating conditions [VAK10, p. 220]. Therefore, reliability is concerned with endurance of systems and is out of scope of this work.

Other frequently confused terms are *criticality* and *risk*. Criticality is a degree of the impact a case or a fault has on the operation of the system [Ger91, p. 55]. In contrary, risk is not directly focused on the integrity of the system. It reflects the possibility that an undesired situation occurs after choosing actions under uncertainty [ISO09].

A motion with zero risk is classified as *safe*. However, the presence of other traffic participants hinder achieving zero risk. This is where robustness comes into play. A robust planning algorithm can handle critical scenarios without causing increased risk. Hence, modeling and quantification of uncertainties are central in evaluating risk, safety, and robustness.

4.3 Methods to Maintain Safety

The uncertainty of the environment information and planning homotopy-class-free maneuvers require delicate safety-handling. There are various approaches to maintain safety under uncertainty, each having their own strengths and weaknesses.

4.3.1 Time-To-X Metrics

A widely accepted metric for quantifying current risk and evaluating safety are the time-to-x (TTX) metrics, where "x" is a chosen event. The event can be a collision (TTC), the last possible reaction (TTR), braking (TTB), an intersection beginning (TTI) etc. A detailed overview on time metrics are presented in [HSK06]. Since TTX metrics are independent of relative distance and speed, early traffic flow and road safety studies utilized them [MB01]. Many pre-crash and collision avoidance systems [ZAG06; Hil07] and later situation prediction modules, such as in Hidden Markov models [SWA16], used them for an analysis in time domain. Recent studies on deep reinforcement learning exploit their invariance of distance and speed, and employed them in their frameworks [Kur+21].

TTX metrics generally suffer from two major problems for autonomous driving applications. First, they are calculated only in one dimension; i.e. the longitudinal direction. Even though there are works that overcome this deficiency by generalizing them into the Cartesian space [War+14], these generalizations have not found a widespread application. The second one is, even though sensor data is modeled probabilistically, these metrics are deterministic in their raw-form and therefore not robust to uncertain information. Berthelot et al. use Monte Carlo methods to model TTC as a probability distribution [Ber+11; Ber+12]. More recent works by Stellet et al. focus on deriving closed-form expressions of the probability distributions and utilize them in emergency braking systems [Ste+15; Ste+16]. Despite these extensions, these metrics lack incorporating interactions of other traffic participants and do not cover maneuver capabilities of the vehicles. Therefore, they stick to the current measurements and can coarsely cover the future development.

4.3.2 Collision Detection in Workspace

A common approach to maintain safety is to ensure that the agent's *workspace* $\mathbb{W}$, i.e. the space in which the agent is located, does not overlap with that of dynamic or static obstacles. Many planning algorithms, independent of if they are sampling-based or not, rely on these types of safety checks. They employ a set of geometric primitives to approximate the shape of the vehicle or obstacles and perform collision checks over the planning horizon.

The most rudimentary approach is to model objects with polygons. However, this approach introduces $d_{\mathbb{W}}$ additional constraints per polygon-vertex, where $d_{\mathbb{W}} = 2$ in the two-dimensional Cartesian space. Unfortunately, polygon-based constraints introduce non-differentiable points, which precludes their utilization in gradient-based methods [Sch+14]. A more efficient approach is to decompose the obstacle shapes into circles and to check candidate motions against collisions [ZS10]. Such circle primitives can also be integrated into optimization-based methods, by analytically deriving continuous circle-to-circle or circle-to-polygon distances [Zie+14a].

An elegant approach for performing collision checks under uncertainty is to use elliptic primitives. Since uncertainties are frequently modeled as bivariate Gaussian distributions and the covariance matrix of it is an ellipse in general, these checks can be directly linked with confidence levels. The intersection of ellipses can be used for obtaining the risk of a collision [IPM13]. However, the resulting calculation is not suitable for planning with gradient-based approaches. An alternative is to approximate one of the areas described with an ellipse by circles with equal radii. For a circle located at (x_o, y_o) with radius r to be outside a region described by an ellipse centered at (x, y) with a semi-major axis and semi-minor axis length of σ_{x} and σ_{y}, it must satisfy

$$\sqrt{\left(\frac{x_o - x}{\sigma_{\mathrm{x}} + \delta}\right)^2 + \left(\frac{y_o - y}{\sigma_{\mathrm{y}} + \delta}\right)^2} > 1, \tag{4.22}$$

where δ is an offset that ensures the enlarged ellipse encapsulates the Minkowski sum of the circle and the ellipse. δ is chosen slightly larger than the radius of the circle, i.e. $\delta = r + \epsilon$ [Bri+19].

Apart from these geometric primitives, a further option is to consider the objects in workspace-time-space. The volume swept by the trajectory is put into a tree-tree query for collision checks. The approach serves as a fast collision checker for sampling-based methods and scales well with increasing number of obstacles [SSF15].

4.3.3 Chance Constraints

In case the objects in a workspace are represented with probability distributions, previously introduced deterministic safety checks are not sufficient. If $\boldsymbol{x}$ is a vector of variables, $\boldsymbol{\zeta}$ is a vector of random variables, and $\mathbf{h}$ is a set of functions, chance constraints can be defined as

$$\Pr\left(\mathbf{h}(\boldsymbol{x}, \boldsymbol{\zeta}) \leq \mathbf{0}\right) \geq 1 - \alpha \tag{4.23}$$

with α being the prescribed risk level [CC63]. Constraints of this form are intractable for arbitrary probability distributions, unless a sampling-based approach is used. Even in cases where $\mathbf{h}$ is affine in $\boldsymbol{x}$ and $\boldsymbol{\zeta}$, the feasible solution set can still be non-convex [SN99]. Several approaches deal with the non-convexity. One approach is to use a convex bounding function [NS07; VT13], such as rectangles [DKH15] or spherical sectors [HD15]. Another approach is to model the uncertainties with known distributions and to reformulate the probabilistic constraints into deterministic, convex ones [CE06; BLW06].

If the chance constraint is defined by n line segments and the random variables $\boldsymbol{\zeta}$ are Gaussian distributed, the variables with additive Gaussian noise $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\zeta}$ can be translated with $\boldsymbol{t}$ and projected to the normal vector $\boldsymbol{u}$ such that they are perpendicular to the constraint. The chance constraint can then be defined as

$$\Pr(\boldsymbol{u}\hat{\boldsymbol{x}} - \boldsymbol{t} \leq \mathbf{0}) \geq 1 - \alpha. \tag{4.24}$$

This constraint can be expressed as conjunction of the constraints

$$\bigwedge_{i=0,\ldots,n-1} \Pr(\boldsymbol{u}_i\hat{\boldsymbol{x}}_i - \boldsymbol{t}_i \leq 0) \geq 1 - \alpha,$$

which in turn can be expressed for individual univariate Gaussian distributions by using Equation (3.4) as

$$\boldsymbol{u}_i \boldsymbol{x}_i + \sqrt{\boldsymbol{u}_i \boldsymbol{\Sigma}_i \boldsymbol{u}_i^\mathsf{T}} \cdot \phi^{-1}(1-\alpha) \leq \boldsymbol{t}_i \tag{4.25}$$

where ϕ^{-1} is

$$\phi^{-1}(\gamma) = \sqrt{2}\mathrm{erf}^{-1}(2\gamma - 1). \tag{4.26}$$

This results in a convex constraint for $\gamma \geq 0.5$. An alternative is to use ellipsoidal relaxation on joint chance constraints [Van04, p. 44]. This corresponds to taking the quantile function of χ^2-distribution as ϕ^{-1}. However, as proven with the comparison made by Vitus et al., this method can lead to highly conservative results [VT11].

There are many applications of this approach in motion planning for autonomous driving [LKH10; VT11; Car+14; LKK15]. In case the mean and the variance of the distribution is not known, they can be estimated from sampled data and subsequently chance constraints can be satisfied [CC06; LK21].

Analogous to constraining the $(1-\alpha)$-quantile and limiting the *Value-at-Risk* (VaR), the expected risk in this quantile can be constrained as well. This is called *Conditional Value-at-Risk* (CVaR), *Expected Shortfall*, or *Tail VaR* [Art+99]. CVaR, which can be interpreted as "how bad is bad", has better properties than VaR, such as being convex when optimized over α [RU+00]. It has gained attention in risk-sensitive optimal policies [CG14]. A recent work compares the risk metrics and argued on the axioms they must satisfy in robotics applications [MP17].

In some works, the joint cumulative distribution of two multivariate Gaussian distributions, i.e. the PDF of position or pose of two vehicles, is directly calculated. The work presented in [Cam+14] relies on the online computation of two bivariate Gaussian distributions defined in [Dre78]. Even though the paper states that the computation is fast, it does not mention the total runtime of the proposed risk assessment algorithm. The work [HBC14] follows a similar approach for calculating pose overlaps, but employs Monte Carlo sampling to obtain the probability of a collision.

4.3.4 Responsibility Sensitive Safety

The approaches presented so far serve as tools to calculate the risk level of a traffic situation and to evaluate safety. Driving in traffic requires multi-agent interaction. There are rules that impose proper actions and behavior the drivers must follow. *Responsibility-Sensitive-Safety* (RSS) provides a comprehensive analysis of risky situations in driving and formalizes an interpretation of the rules accompanied by mathematical models. It assures safety if the dynamics of all vehicles do not exceed a given set of parameterized limits [SSS17].

RSS is proposed as a separate, interpretable model that acts supplementary to the motion planner. Thus, it provides high flexibility for the choice of the planner. Since it is designed for safety assurance, a planner should ideally never require the RSS-module to intervene and make a correction to assure safety.

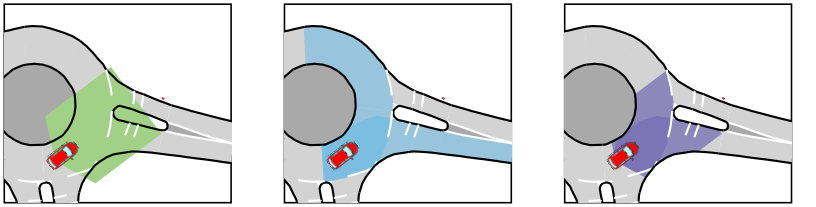
In practice, RSS suffers from two major problems. First, it requires around 40 parameters which include non-scalar values, hence making the analysis particularly complicated. A recent work studies the expectations for certain driving scenarios these parameters are required to meet [Nau+21]. Nevertheless, generalization remains an open-topic for further research. A second problem is again related to measurement uncertainties. Safe distances and braking maneuvers utilized in RSS do not incorporate the uncertainty involved in the environment model. It should be underlined that RSS is able to provide safety guarantees only for an appropriate choice of parameters, which is quite difficult in practice.

4.3.5 Verification Approaches

Formal verification techniques count as the most complete safety assurance methods. These approaches construct a mathematical model of the *system*. By utilizing formal logic and probabilistic model checking, they can exhaustively examine if the safety conditions are fulfilled and eventually ensure that a system is safe. A major problem with these approaches is that they scale poorly to systems with large and complex state spaces [Lee+20]. Therefore, it is not feasible to express the complex environment while driving.

A convenient way to perform formal verification is to employ set-based over-approximations for the reachability of the agents in the environment.

Althoff et al. showed suitability of the approach for utilization in hybrid systems, such as in driving, where the agents are subject to both discrete decisions and continuous driving dynamics [ASB07a; ASB07b]. Starting from an initial but deterministic state, the approach calculates the set of states the system can reach for all possible inputs for a certain prediction horizon. Safety is afterwards checked against overlaps with reachable sets of distinct traffic participants.



(a) Acceleration-based occupancy. Calculated with limits of acceleration values obtained from circle-of-forces.

(b) Lane-following occupancy. Obtained by using road network and the longitudinal reachability limited by speed regulations.

(c) Result of occupancy calculation. Obtained from the intersection of three distinct occupancy calculations.

Figure 4.6: Occupancy calculated for the red vehicle traveling at 7.5 m/s on a roundabout. The safe-distance occupancy of cars on adjacent lanes is not included [AM16]. The results can be used as truncation bounds of the truncated univariate Gaussian distribution to represent the predicted position of the red vehicle [THS18].

A major disadvantage with reachable set-based verification is the quick growing of over-approximations over time, resulting in large areas of unsafe regions. This eventually leads to overly conservative motion profiles. Magdici et al. proposed to ensure the existence of a fallback maneuver until the next planning instance with reachable sets, and called this approach as *fail-safe* motion planning. In case the existence of a safe motion cannot be ensured in the next planning instance, then the previously calculated safe one is selected. In their work, they decoupled planning of optimal motion from fallback plans [MA16].

Integration of fallback maneuver calculation into optimal planning was first proposed and applied by Taş et al. In their work, they argued on safe motion planning and inevitable collision states (ICS) of Petti et al. (see Section 4.1.1), tackled time-delays, and ensured the existence of a fallback maneuver by employing constraints in the time interval $t \in [t_0, 2t_{\text{pin}})$ while optimizing the motion profile. By treating position and speed measurements with additive Gaussian noise, they have defined a chance constraint on the full-braking

distance into the nonlinear optimization problem and accounted for the present uncertainty. As a result of limited visibility and time-delays, to prevent a potential increase in damage, the fallback maneuver is restrained to braking only [TS18]. Shortly after, Pek et al. followed the same approach independently by arguing on ICS and proposing to integrate fallback maneuver calculation as a constraint into optimization-based planning. Different from the aforementioned approach, by adopting a formulation that decouples longitudinal and lateral planning, they could consider swerve maneuvers as fallback [PA18]. However, unlike the aforementioned work, they did not argue on limited visibility, time-delays, and uncertainties. In subsequent publications they conducted driving experiments, and proved the efficiency and generality of their approach.

Fallback maneuvers without reachable set prediction can be categorized under scenario stochastic MPC, as done by Alsterda et al. in their Contingency MPC framework. The framework focuses on autonomous vehicle control on slippery roads and plans with a fallback for decreased road surface friction [ABG19].

Safety verification is becoming increasingly popular in autonomous driving. Besides motion prediction and planning, it has found application in threat assessment [AFS12; SKA18], and in arbitrators for decision making [Orz+21].

4.3.6 Falsification Approaches

Falsification approaches address the main deficit of verification approaches: they scale large and complex state spaces at the expense of completeness. By searching for examples that violate safety conditions, these approaches are not bound to specifications set in verification approaches and can reveal implementation mistakes. Therefore, they cannot ensure the absence of failures.

Recent works employ motion planning approaches for a targeted search from an initial state towards failure states. An application for hybrid systems is proposed by utilizing an RRT algorithm [Dre+15]. A subsequent work uses this approach in adaptive cruise control systems [Kos+19]. A more recent work, *Adaptive Stress Testing* argues on the poor applicability of RRTs for falsification in large state spaces with hidden states and unknown disturbance, and proposes to find the most likely path to a failure event [Lee+20]. Although falsification approaches are promising, they can only be used for offline testing and validation, and not as a safety module in motion planning.

4.3.7 Others

Some other methods in literature cannot be classified under the aforementioned approaches. Albeit they aim to maintain safety, they can only provide an estimate of risky situations, or serve as a heuristic to reduce the risk. Nevertheless, it is important to briefly present some of these.

Eidehall et al. highlight that TTX-metrics are limited to deterministic values and do not incorporate stochastic future evolution of their values. Therefore, they employ Monte Carlo sampling onto the vehicle models and estimate different threat metrics [EP08]. An original idea that inspects the *freedom* or *margin* of control is presented in [CPI14]. The approach inspects the control freedom among homotopic maneuver options and evaluates which one allows the largest amount of freedom.

An approach to mitigate risk while driving is to define a function obtained by extracting a risk value from a combination of threat metrics from driven trajectories [DE14]. By using such a *risk map*, a vehicle can follow a behavior that optimizes its driving goals while minimizing the risk. This approach can, however, lead to unnatural actions while trying to minimize the risk, if not engineered appropriately. The authors state that faster intersection crossing can be a risk minimizing behavior in circumstances. This statement implies that their approach fails to decently handle the uncertainties and yield a human-like behavior. The same applies to a dozen of works that penalize the time spent at intersections and similar crash-likely areas. Furthermore, it is hard to identify such areas precisely (see Figure 2.1).

Other map-based risk analyses aim to reason about presence of traffic participants in occluded areas. The first work that focuses on this task utilizes particle-based reasoning for occlusions [YVJ19]. They propagate particles along a driving direction from visible areas to occluded areas over time and reason out potential participants. A later work by Wang et al. inspects safety by using reachable sets for uncertainty propagation [WBS21]. Nevertheless, these works basically join disconnected visible areas. This operation is only advantageous when the visible range is big and the occlusions are relatively small. Eventually, these are rather occupancy prediction methods than an original method to maintain safety.

There are works that suggest dealing with safety from a system-wide perspective. Whereas early works merely monitor the continuous operation and the progress of the system components, and initiate a reset or hard stop upon any anomaly [BFD08; Kam+08; Kim+12], later works additionally evaluate the discrepancies among the information in system components [MM15]. All of these works show promising results in application by initiating recoveries or safe-stops. A later work of Taş et al. presents the RobustSENSE-architecture, which monitors individual components such as sensors, fusion, or scene understanding algorithms with performance measures. The performance measures are evaluated component-wise, module-wise, and system-wide, eventually degrading the system and thereby indirectly increasing robustness of algorithms and maintaining safety. The proposed performance assessment system shows successful results in test drives, in which phantom objects are synthetically injected into the perception pipeline. The system can infer the anomaly and the central Bayesian network can reason out the detection of a phantom object [Taş+17]. Even though system-wide assessment like presented above increase robustness considerably, they cannot provide any guarantee on safety.

4.4 Co-planning of Probabilistic Proactive Z-Plans

The recapitulation provided in the previous section highlights two major takeaways for maintaining safety in receding horizon planning: 1) consideration of uncertainties requires utilization of chance constraints 2) guaranteeing safety while not being overly conservative requires employing verification approaches that are based on inevitable collision states (ICS). The first work that addresses the need to combine them is presented by Taş et al., in which these conditions are integrated into the motion planner in the form of constraints. In this way, a probabilistic and proactive fallback motion is co-planned [TS18].

Co-planning of such a fallback plan is not straightforward. This section first presents alternative fallback plans and then reveals why a certain type of fallback plan should be preferred over its alternative. Subsequently, uncertainties in the execution of the fallback plan are analyzed and modeled. Since the uncertainties present in the environment model have a direct impact on safety, constraints that must be satisfied while using the fallback plan are presented next.

4.4.1 Alternative Fallback Plans and Present Uncertainties

Alternative fallback plans can be divided into two classes: full-braking maneuvers and swerve maneuvers. As studied by Schmidt, full-braking at speeds lower than roughly 40 km/h is more advantageous, whereas swerving at higher speeds, which may involve combined steering and braking, becomes more desirable for collision avoidance with a single object [Sch14, p. 24]. However, there are two problems with swerving for collision avoidance under uncertainty. Firstly, for Gaussian error propagation and uncertainty modeling, vehicle models are frequently linearized. This can lead to big modeling errors if maneuvers at handling limits are executed. Secondly and more importantly, the visible field might not be sufficient to cover the required area for the execution of swerve maneuvers. Therefore, this work chooses full-braking as fallback and calls such a maneuver as *Z-plan*, implying the last option the vehicle can execute.

Modeling Braking Distance as a Univariate Gaussian Distribution

In case speed v and the maximal deceleration a^- values are deterministic, the braking distance s_{brake} can be calculated as

$$s_{\text{brake}} = F_{\text{brake}}(v, a^-) = \frac{v^2}{2a^-}. \tag{4.27}$$

As introduced in Section 3.1.1, speed v as well as position $\boldsymbol{x}$ values are bound to uncertainties, which are typically modeled as univariate and bivariate Gaussian distributions, respectively. Therefore, this equation is not sufficient for calculating the braking distance under uncertainty.

Speed measurements can be modeled with an additive white Gaussian noise

$$\hat{v} = v + \zeta_v \quad \text{where} \quad \zeta_v \sim \mathcal{N}\left(0, \sigma_v^2\right). \tag{4.28}$$

The square operation on Equation (4.27) on additive Gaussian noise results in an additive Gamma distribution, which does not have a quantile function that can be solved in closed-form or be approximated in real-time. The error on a function can be calculated by linearizing it using first-order Taylor series expansion

$$F(\boldsymbol{x}) \approx F\left(\boldsymbol{x}^{(0)}\right) + \nabla F\big|_{\boldsymbol{x}^{(0)}} \left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right). \tag{4.29}$$

As $F\left(\boldsymbol{x}^{(0)}\right)$ is a constant, it does not contribute to the error. Assuming the variables in the vector $\boldsymbol{x}$ are stochastically independent, the function's variance is approximated as

$$\sigma_F^2 \approx \left(\frac{\partial F}{\partial \boldsymbol{x}_0}\right)^2 \sigma_{\boldsymbol{x}_0}^2 + \ldots + \left(\frac{\partial F}{\partial \boldsymbol{x}_{n-1}}\right)^2 \sigma_{\boldsymbol{x}_{n-1}}^2. \tag{4.30}$$

If the formulation above is applied on Equation (4.27)

$$\begin{aligned}\sigma_{\text{brake}}^2 &\approx \left(\frac{\partial F_{\text{brake}}}{\partial v}\right)^2 \sigma_v^2 + \left(\frac{\partial F_{\text{brake}}}{\partial a^-}\right)^2 \sigma_{a^-}^2 \\ &\approx \left(\frac{v}{a^-}\right)^2 \sigma_v^2 + \left(-\frac{v^2}{2(a^-)^2}\right)^2 \sigma_{a^-}^2.\end{aligned} \tag{4.31}$$

Braking distance can now be modeled as an expected value in braking distance with an additive white Gaussian noise ζ_{brake}

$$\hat{s}_{\text{brake}} = s_{\text{brake}} + \zeta_{\text{brake}} \quad \text{where} \quad \zeta_{\text{brake}} \sim \mathcal{N}(0, \sigma_{\text{brake}}^2). \tag{4.32}$$

The exact value of a^- depends on both the friction coefficient between the road and the tire, and the current lateral acceleration. Therefore, even when the vehicle is following its own route and not performing any swerve maneuver, the maximum braking deceleration might vary. It is a design and experimentation choice, whether to select a conservative deterministic value for a^-, or to estimate it, and eventually model it with additional Gaussian noise.

Modeling Stop Position as a Univariate Gaussian Distribution

The position of any agent can be modeled as a bivariate variable in Cartesian x and y coordinates with white Gaussian noise

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \zeta_{\boldsymbol{x}} \quad \text{where} \quad \zeta_{\boldsymbol{x}} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{x}}\right). \tag{4.33}$$

The braking distance is previously modeled as univariate along the center of the driving corridor. Because position is modeled as a bivariate variable with additive Gaussian noise, it must be projected onto the driving corridor to obtain the uncertainty in stop position.

The projection of the expected value onto the centerline can be done by calculating the shortest normal distance to the center of the driving corridor, which is modeled as a polyline (see Section 2.1.2). The variance along s can be approximated as $\sigma_s^2 = \lambda_{\text{lon}}(\boldsymbol{\Sigma}_{\boldsymbol{x}})$, as presented in Equation (3.4). The longitudinal position $\hat{s}$ is expressed as

$$\hat{s} = s + \zeta_s \quad \text{where} \quad \zeta_s \sim \mathcal{N}\left(0, \sigma_s^2\right). \tag{4.34}$$

Once the longitudinal position is calculated, the stop position $\hat{s}_{\text{stop}}$ is obtained by adding Equation (4.32) and Equation (4.34)

$$\begin{aligned} \hat{s}_{\text{stop}} &= \hat{s} + \hat{s}_{\text{brake}} \\ &= s + \zeta_s + s_{\text{brake}} + \zeta_{\text{brake}} \end{aligned} \tag{4.35}$$

The expected value of the stop position by using Equation (4.27) becomes

$$s_{\text{stop}} = s + \frac{v^2}{2a^-}, \tag{4.36}$$

Assuming that braking distance and current longitudinal position are not correlated, the variance in the stop position is calculated using Equation (4.30) as

$$\sigma_{\text{stop}}^2 = \sigma_s^2 + \sigma_{\text{brake}}^2. \tag{4.37}$$

This yields

$$\hat{s}_{\text{stop}} = F_{\text{stop}}(\boldsymbol{x}, v, a^-) = s_{\text{stop}} + \zeta_{\text{stop}} \quad \text{where} \quad \zeta_{\text{stop}} \sim \mathcal{N}\left(0, \sigma_{\text{stop}}^2\right). \tag{4.38}$$

4.4.2 Proactive Planning for Ordinary Driving

The previous subsection elaborates on emergency braking as a Z-plan under uncertainty. Because the longitudinal stop position can be modeled with a univariate Gaussian error around the linearized current state, it can be used to define a chance constraint (see Section 4.3.3) in real-time applications.

In ordinary driving scenarios, no interaction with other traffic participants is required, see Figure 4.7. Depending on the existence of other agents in the driving corridor, there are two different forms of the chance constraint.

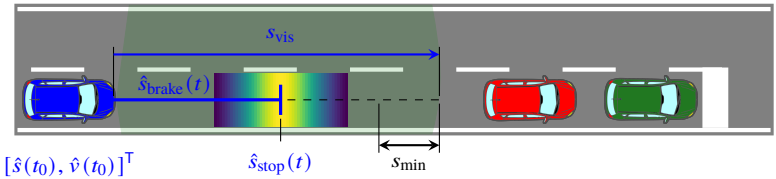


Figure 4.7: An ordinary driving scenario with limited visible area. The uncertainty in the braking distance of the blue vehicle is modeled as a Gaussian distribution.

Limited Visibility

In cases where there is not any agent within the visible field along the driving corridor, the chance constraint on safety is written only for the visibility condition, by using Equation (4.38) in Equation (4.23)

$$\Pr\left(F_{\text{stop}}\left(\boldsymbol{x}, v, a^{-}\right) - s_{\text{vis}} - s_{\text{min}} \leq 0\right) \geq 1 - \alpha, \tag{4.39}$$

or simply

$$\Pr\left(\hat{s}_{\text{stop}} - s_{\text{vis}} - s_{\text{min}} \leq 0\right) \geq 1 - \alpha, \tag{4.40}$$

where s_{vis} and s_{min} correspond to visible free distance and standstill distance, respectively.

In receding horizon planning, for a motion plan to be considered as safe, it must ensure the existence of the Z-plan in the time-interval $t \in [t_0, 2t_{\text{pin}})$, until the next replanning is completed, as shown in Section 4.1.1 and Section 4.3.5. By using time discretization, the constraint presented above can be applied as conjunction of constraints at each sampling instance i

$$\bigwedge_{i=0,\ldots,2n_{\text{pin}}} \Pr\left(\hat{s}_{\text{stop},i} - s_{\text{vis}} + s_{\text{min}} \leq 0\right) \geq 1 - \alpha. \tag{4.41}$$

These nonlinear constraints can be integrated into the motion planner by using a computationally efficient approximation of quantile function of the univariate Gaussian distribution.

Agents with Uncertain Physical States

Position and velocity measurements of agents inside the visible field are represented with additive Gaussian noise. Projecting these values onto the centerline enables calculating the distribution of their braking distances and stop positions, as done for the ego vehicle. Considering the situation depicted in Figure 4.7, the longitudinal position and speed of the vehicle k driving closest in front can be modeled as $\hat{s}_k = s_k + \zeta_{s,k}$ where $\zeta_{s,k} \sim \mathcal{N}(0, \sigma^2_{s,k})$, and $\hat{v}_k = v_k + \zeta_{v,k}$ where $\zeta_{v,k} \sim \mathcal{N}(0, \sigma^2_{v,k})$. The worst case in such a scenario is when the vehicle k applies full-braking immediately after the environment information is processed. Therefore, the difference in the stop positions Δs_{stop} of both vehicles become crucial. However, as stop positions are modeled with additive noise, Δs_{stop} is bound to noise as well. Therefore,

$$\Delta s_{\text{stop}} = F_{\text{stop}}\left(\boldsymbol{x}, v, a^-\right) - F_{\text{stop},k}\left(\boldsymbol{x}_k, v_k, a_k^-\right). \tag{4.42}$$

Because the additive noise in stop positions is Gaussian

$$\Delta s_{\text{stop}} \approx \hat{s}_{\Delta\text{stop}} = s_{\Delta\text{stop}} + \zeta_{\Delta\text{stop}}, \tag{4.43}$$

where

$$s_{\Delta\text{stop}} = s_{\text{stop}} - s_{\text{stop},k} \tag{4.44}$$

and

$$\zeta_{\Delta\text{stop}} \sim \mathcal{N}\left(0, \sigma^2_{\Delta\text{stop}}\right), \tag{4.45}$$

such that

$$\sigma^2_{\Delta\text{stop}} = \sigma^2_{\text{stop}} + \sigma^2_{\text{stop},k}. \tag{4.46}$$

The chance constraint can now be expressed as

$$\Pr\left(\Delta\hat{s}_{\text{stop}} + s_{\min} \leq 0\right) \geq 1 - \alpha. \tag{4.47}$$

Because of the safety condition, this constraint is applied as a conjunction of chance constraints

$$\bigwedge_{i=0,\ldots,2n_{\text{pin}}} \Pr\left(\hat{s}_{\Delta\text{stop},i} + s_{\min} \leq 0\right) \geq 1 - \alpha. \tag{4.48}$$

4.4.3 Proactive Interaction with Other Participants

The fact that prediction modules can never identify the maneuver intentions with certainty makes proactive planning indispensable while in interaction with other participants. Such scenarios occur at intersections and while overtaking. Even though the structure of these seem to be entirely different at first, they are described by the same formulations, but with different parameters. This work picks intersection scenarios for analyzing proactive planning.

Intersections pose two distinct classes of problems for a vehicle that is approaching to an intersection:

1. the ego vehicle has to yield to oncoming vehicles, but it has a limited visible field
2. the ego vehicle has the right-of-way, but the oncoming vehicle might not comply with the traffic rules and not yield

Motion planning for both of these classes heavily depends on the visible range and the acceptable braking distance on the merging routes given this range.

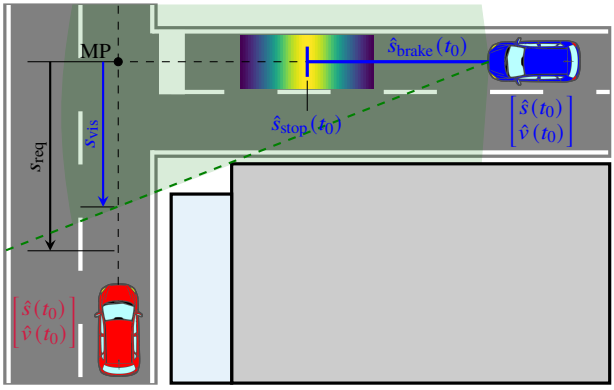


Figure 4.8: An intersection under limited visibility. For illustration purposes, the point where the routes cross, MP – merge point, is shown in the center of both corridors. See Figure 2.2 for a more realistic corridor and MP example.

Intersection Crossing with Limited Visibility

Intersections can pose a difficult problem on the planner if visibility is hindered by the buildings or obstacles around. Cases where the ego vehicle has to yield to other vehicles is a rather simple problem: the ego vehicle must only ensure it can stop before the intersection, while its visible distance along the intersecting route (s_{vis}) is shorter than the distance required for an oncoming vehicle to pass (s_{req}). This condition can be introduced as a constraint into the motion planner.

A visualization of such a braking constraint in path–time–speed diagram is depicted in Figure 4.9. The full-braking motion must remain below the red-depicted surface, *surface-of-no-return*, to be considered as safe. The planner can recover motion profiles that exceed the surface after $2t_{\text{pin}}$, the blue-depicted surface, as that part of the motion profile will be recalculated in the next planning time. Points on the red surface correspond to *point-of-no-return*, a term which is used in other works, such as in [Gin+08; Cam+14], that tackle safety for a single motion profile. It should be underlined that a motion exceeding this surface can still end up with a collision-free state, as this depends on the presence as well as the motion of other traffic participants.

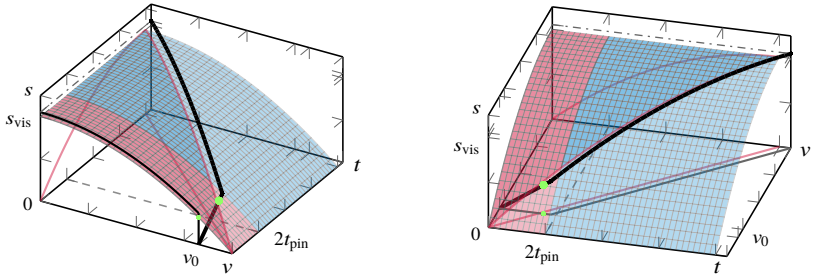


Figure 4.9: Safety inspected in speed–time–path ($v - t - s$) space. Any motion below the red-depicted surface-of-no-return can come to a full stop before s_{vis}. The red line is the full-braking motion from the highest safe initial speed. The black line represents an unsafe motion as it leaves the red surface at the green point, at the point-of-no-return.

Considering the situation depicted in Figure 4.8, because of the full stop position constraint for $s_{\text{vis}} < s_{\text{req}}$, the ego vehicle will decelerate until $s_{\text{vis}} \geq s_{\text{req}}$. If s_{vis} is still not sufficient at the beginning of the intersection zone for any reason, depending on the scenario, the ego vehicle can creep into the intersection to avoid any deadlock. However, creeping behavior can be undesired.

In order to avoid such an overly conservative creeping behavior, planning at intersections can be modeled such that the ego vehicle enforces a comfortable braking deceleration to the oncoming vehicle, even though the latter has the right-of-way. The reaction of an oncoming vehicle to the ego vehicle can be modeled with a comfortable deceleration a^{-}_{cft}. A pessimistic prediction of the ego vehicle's motion from the perspective of an oncoming vehicle is that the ego vehicle drives with constant speed to the intersection, ignoring the traffic rules. Given a fixed value of a^{-}_{cft} and v_{ego}, the time required for the oncoming vehicle traveling at speed v_k to decelerate to the speed of the ego vehicle is

$$t_{\text{dec}} = \frac{v_k - v_{\text{ego}}}{|a^{-}_{\text{cft}}|}. \tag{4.49}$$

The distance required for ego vehicle to *see* inside this route can be obtained as

$$s_{\text{req}} = v_k\, t_{\text{dec}} + \frac{1}{2}\, a^{-}_{\text{cft}}\, t^2_{\text{dec}} + v^{(t_0)}_{\text{ego}}\, t_{\text{hw}}, \tag{4.50}$$

where t_{hw} is the minimum time headway, which is set to 2.0 seconds by traffic rules. By applying the value of t_{hw} and Equation (4.49),

$$s_{\text{req}} = S_{\text{req}}(v^{(t_0)}_{\text{ego}}, v_k, a^{-}_{\text{cft}}) = v_k \left(\frac{v_k - v^{(t_0)}_{\text{ego}}}{|a^{-}_{\text{cft}}|} \right) + \frac{1}{2}\, a^{-}_{\text{cft}} \left(\frac{v_k - v^{(t_0)}_{\text{ego}}}{|a^{-}_{\text{cft}}|} \right)^2 + 2v^{(t_0)}_{\text{ego}}. \tag{4.51}$$

If there are no vehicles within the visible field, the maximum allowed speed for that route v^{+} can be taken for v_k.

While $s_{\text{vis}} < S_{\text{req}}(v^{(t_0)}_{\text{ego}}, v^{+}, a^{-}_{\text{cft}})$, the ego vehicle must hold the constraint

$$\Pr\left(\hat{s}_{\text{stop}} - s_{\text{MP}} + s_{\text{min}} \leq 0\right) \geq 1 - \alpha. \tag{4.52}$$

Because of the safety condition in receding horizon planning, this constraint must be satisfied for the first $2n_{\text{pin}}$ indices of the motion

$$\bigwedge_{i=0,\ldots,2n_{\text{pin}}} \Pr\left(\hat{s}_{\text{stop},i} - s_{\text{MP}} + s_{\text{min}} \leq 0\right) \geq 1 - \alpha. \tag{4.53}$$

Once $S_{\text{req}}(v_{\text{ego}}^{(t_0)}, v^+, a_{\text{cft}}^-) < s_{\text{vis}}$, the ego vehicle passes the intersection according to the traffic rules, i.e. perform a full stop, if there is a stop sign.

The reaction enforced by the ego vehicle onto the oncoming vehicle does not necessarily have to be as big as a_{cft}^-. It attains smaller values, if $s_{\text{req}} < s_{\text{vis}}$. This motivates the notion of *politeness* η, which can be defined by using insights from MOBIL [KTH07], an interaction-aware extension of IDM, as

$$\eta = F_\eta(a) = \frac{a_{\text{cft}}^- - a_k}{a_{\text{cft}}^-}. \tag{4.54}$$

Intersection Crossing with Noncompliant Traffic Participants

In intersection scenarios, even if the ego vehicle has the right-of-way, it should not enter into an unsignalized intersection recklessly, assuming other vehicles are compliant. Albeit the ego vehicle may have zero responsibility in the accident, it would still damage the brittle reputation of autonomous driving. For this reason, planning algorithms should ideally tackle traffic rule-violating behaviors of other participants.

Integrating such a feature can easily be done by utilizing the formulations presented above. At an intersection, if a detected vehicle cannot brake comfortably to decelerate to the speed of the ego vehicle and hold the required minimum time headway, it is considered as noncompliant. This is checked by using Equation (4.51)

$$s_k + S_{\text{req}}(v_{\text{ego}}^{(t_0)}, v_k^{(t_0)}, a_{\text{cft}}^-) \overset{?}{\leq} s_{\text{MP}}. \tag{4.55}$$

At an intersection, while $s_{\text{vis}} < S_{\text{req}}(v_{\text{ego}}^{(t_0)}, v^+, a^-)$, the ego vehicle continues driving to the intersection while holding the constraint defined in Equation (4.53). After $S_{\text{req}}(v_{\text{ego}}^{(t_0)}, v^+, a^-) < s_{\text{vis}}$, it checks if there are any vehicles approaching, and if none, it deactivates the constraint mentioned above. If the oncoming vehicle is noncompliant, the ego vehicle initiates a give-way maneuver. Otherwise, if the difference in the sides of Equation (4.51) increase over subsequent times, the vehicle is considered as compliant and the constraint is deactivated.

5 Planning with Numerical Optimization

The ideal definition of the objective function, the chance constraints, as well as the structure of the parameter vector establish a complex problem, for which finding the optimal solution is difficult. Moreover, online planning imposes real-time requirements, which pose an additional challenge. Nonlinear optimization arises as a prevalent tool to meet these requirements.

This chapter starts with nonlinear optimization methods. Since the solution and modularity in gradient-based optimization depend on differentiation, methods for calculating derivatives are inspected next. This chapter finally presents a novel nonlinear optimization based motion planner that employs the previously defined objective function and safety concepts.

5.1 Nonlinear Optimization

An optimization problem can be defined as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & F(\boldsymbol{x}) && \text{(5.1a)} \\ \text{subject to} \quad & \mathbf{g}(\boldsymbol{x}) = \mathbf{0} && \text{(5.1b)} \\ & \mathbf{h}(\boldsymbol{x}) \leq \mathbf{0}, && \text{(5.1c)} \end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are the optimization parameters, $F : \mathbb{R}^n \to \mathbb{R}$ is the objective function, $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ are the equality constraints, and $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p$ are the inequality constraints. In nonlinear problems, at least either of the constraint functions or the objective function is nonlinear.

5.2 Local Descent

Nonlinear optimization algorithms benefit from continuous differentiability of the objective and constraint functions, facilitating the search for an infimum. Depending on the degree of continuous differentiability, they take a step toward to the minimum from a feasible initial guess, by using a first- or second-order Taylor approximation on the local cost.

Quadratic approximation of a function $F(\boldsymbol{x})$ around $\boldsymbol{x}^{(i)}$ is

$$F(\boldsymbol{x}) \approx F(\boldsymbol{x}^{(i)}) + \nabla F(\boldsymbol{x}^{(i)})^\mathsf{T}(\boldsymbol{x} - \boldsymbol{x}^{(i)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(i)})^\mathsf{T}\nabla^2 F(\boldsymbol{x}^{(i)})(\boldsymbol{x} - \boldsymbol{x}^{(i)}). \tag{5.2}$$

For a descent direction $\boldsymbol{d}^{(i)}$ from $\boldsymbol{x}^{(i)}$ this Taylor approximation becomes

$$F(\boldsymbol{x}^{(i)} + \boldsymbol{d}^{(i)}) \approx F(\boldsymbol{x}^{(i)}) + \nabla F(\boldsymbol{x}^{(i)})^\mathsf{T}\boldsymbol{d}^{(i)} + \frac{1}{2}\boldsymbol{d}^{(i)\mathsf{T}}\nabla^2 F(\boldsymbol{x}^{(i)})\boldsymbol{d}^{(i)}. \tag{5.3}$$

The *Newton's method* obtains the descent direction by setting the derivative of this Taylor approximation to zero

$$\boldsymbol{d}^{(i)} = -\left[\nabla^2 F(\boldsymbol{x}^{(i)})\right]^{-1}\nabla F(\boldsymbol{x}^{(i)}). \tag{5.4}$$

Hence, an optimum $\boldsymbol{x}^*$ must have $\nabla F(\boldsymbol{x}^*) = \mathbf{0}$ and a positive semi-definite $\nabla^2 F(\boldsymbol{x}^*)$. The first condition is known as the first-order necessary condition, and the both conditions together are known as the second order necessary conditions. If $\nabla^2 F(\boldsymbol{x}^*)$ is positive definite instead, then $\boldsymbol{x}^*$ is a strict local optimum of F. This is then known as the second-order sufficient conditions.

The quadratic approximation around a point can be insufficient, and therefore, optimization algorithms operate with iterative steps. There are two strategies for moving from $\boldsymbol{x}^{(i)}$ to $\boldsymbol{x}^{(i+1)}$ and they both follow the same fundamental principle of decreasing the function value at each iteration.

Line search strategy selects a step factor α in the descent direction

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \alpha^{(i)}\boldsymbol{d}^{(i)}. \tag{5.5}$$

The computational complexity of calculating the *exact* value of α is high. A more efficient approach is to approximate it, and use the saved computational

capacity for performing more iterations. A sufficient reduction in the function value is guaranteed by Armijo and curvature conditions, known collectively as Wolfe conditions [NW06, p. 33; KW19, p. 60].

Trust-region strategy constructs a quadratic model about F in the local region of $\boldsymbol{x}^{(i)}$. Depending on the reliability of the local model, they increase the radius of the trust region and then select the direction that reduces the function value at most. Its operation is in reverse order if compared with line search strategies. Findings in this work confirm that both strategies exhibit comparable performance [GNS09, p. 394].

5.3 Constrained Optimization

The constraints defined in Equation (5.1) is incorporated into the solution method by converting the constrained optimization problem into an unconstrained one. There are two ways to accomplish this.

5.3.1 Penalty Methods

The simplest way to consider constraints is to add them with a weighting term into the objective function whenever they are violated

$$\underset{\boldsymbol{x}}{\text{minimize}} \;\; F(\boldsymbol{x}) + F_{\text{penalty}}(\mathbf{g}(\boldsymbol{x}), \mathbf{h}(\boldsymbol{x})). \tag{5.6}$$

F_{penalty} is a *penalty* or *loss* function that inspects the violation and reflects it with an additional scaled cost with a *penalty factor* $\boldsymbol{\mu}$. Loss functions available in machine learning domain, such as soft L1-loss, Huber loss, Cauchy loss, arctan loss, quadratic loss, or even sigmoid functions can be used as a penalty. It is an internal task of the optimization algorithm to adapt the penalty factor to prevent constraint violations. It starts with a small factor value and increases in subsequent iterations to ensure feasibility. These methods suffer from a major problem: depending on the penalty function, there is no guarantee on feasibility of the solution. Especially during the initial iterations, the solution can violate the constraints or have poor accuracy. Furthermore, as the penalty factor increases, the gradients can behold sharp discontinuities.

Barrier Methods

Feasibility can be ensured by choosing a penalty function, the value of which increases to infinity as the optimization parameters approach the constraints. Such a function acts as a *barrier* and when started from feasible parameter values, an *interior point*, these methods ensure feasibility. Logarithmic functions are commonly used for this

$$\underset{\boldsymbol{x}}{\text{minimize}} \;\; F(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}} \log(-\mathbf{h}(\boldsymbol{x})). \tag{5.7}$$

5.3.2 Lagrange Multipliers and Duality

A local minimum of a function $F(\boldsymbol{x})$ subject to an equality constraint $G(\boldsymbol{x}) = 0$ must have aligned gradient vectors[1]. The method of Lagrange multipliers relates this condition to a *Lagrange multiplier* $\zeta \in \mathbb{R}$, which compensates the scales of the gradients

$$\nabla F(\boldsymbol{x}) - \zeta \nabla G(\boldsymbol{x}) = 0. \tag{5.8}$$

Joseph-Louis Lagrange noticed that the left-hand sides of this function and the equality constraint correspond to the partial derivative of the function

$$\Lambda(\boldsymbol{x}, \zeta) = F(\boldsymbol{x}) - \zeta G(\boldsymbol{x}), \tag{5.9}$$

which was later named after him as the *Lagrangian*. Solving $\nabla \Lambda(\boldsymbol{x}, \zeta) = \mathbf{0}$ returns stationary points, which can be a minimum or a saddle point.

In contrast to equality constraints, inequality constraints do not *bind* the solution, i.e. they are not active, for all values. In such cases, the Lagrange multiplier of an inequality constraint ξ is set to zero. In other cases, the minimum of $F(\boldsymbol{x})$ constrained by only a single inequality constraint is

$$\underset{\boldsymbol{x}}{\text{minimize}} \; \underset{\xi \geq 0}{\text{maximize}} \; \Lambda(\boldsymbol{x}, \xi). \tag{5.10}$$

This reformulation is known as the *primal* problem.

[1] An illustration can be found in [NW06, p. 308; Zie17, p. 54; KW19, p.172].

These formulations can be generalized to the problem defined in Equation (5.1). The Lagrangian becomes

$$\Lambda(\boldsymbol{x}, \boldsymbol{\zeta}, \boldsymbol{\xi}) = F(\boldsymbol{x}) - \boldsymbol{\zeta}^{\mathsf{T}}\mathbf{g}(\boldsymbol{x}) - \boldsymbol{\xi}^{\mathsf{T}}\mathbf{h}(\boldsymbol{x}) \tag{5.11}$$

where $\boldsymbol{\zeta} \in \mathbb{R}^m$, and $\boldsymbol{\xi} \in \mathbb{R}^p$ are Lagrange multipliers for equality constraints and inequality constraints. The stationary points of the corresponding primal problem can be found by the Karush–Kuhn–Tucker (KKT) conditions

$$\nabla F(\boldsymbol{x}^*) - \boldsymbol{\zeta}^{\mathsf{T}}\nabla\mathbf{g}(\boldsymbol{x}^*) - \boldsymbol{\xi}^{\mathsf{T}}\nabla\mathbf{h}(\boldsymbol{x}^*) = \mathbf{0} \tag{5.12}$$

$$\mathbf{g}(\boldsymbol{x}^*) = \mathbf{0} \tag{5.13a}$$

$$\mathbf{h}(\boldsymbol{x}^*) \leq \mathbf{0} \tag{5.13b}$$

$$\boldsymbol{\xi} \geq \mathbf{0} \tag{5.14}$$

$$\boldsymbol{\xi}^{\mathsf{T}}\mathbf{h}(\boldsymbol{x}^*) = \mathbf{0} \tag{5.15}$$

which imply stationarity, feasibility (Equations (5.13a) and (5.13b)), dual feasibility, and complementary slackness, respectively.

The primal problem becomes

$$\underset{\boldsymbol{x}}{\text{minimize}} \; \underset{\boldsymbol{\zeta}, \boldsymbol{\xi} \geq \mathbf{0}}{\text{maximize}} \; \Lambda(\boldsymbol{x}, \boldsymbol{\zeta}, \boldsymbol{\xi}) \tag{5.16}$$

and its *dual* form reverses the order of extremum operations

$$\underset{\boldsymbol{\zeta}, \boldsymbol{\xi} \geq \mathbf{0}}{\text{maximize}} \; \underset{\boldsymbol{x}}{\text{minimize}} \; \Lambda(\boldsymbol{x}, \boldsymbol{\zeta}, \boldsymbol{\xi}). \tag{5.17}$$

The dual problem is concave and solving it is often easier than the primal one. Under some conditions, the difference between primal and dual solution is zero, which is called *strong duality* and otherwise, the dual problem defines a lower bound, which is called *weak duality*. One condition that ensures strong duality requires the existence of an interior point that is feasible [BV04, p. 225], whereas another condition requires the gradients of the active constraints to be linearly independent [NW06, p. 320].

5.4 Solvers for Constrained Nonlinear Problems

There are two methods for solving constrained nonlinear optimization problems with feasibility guarantees: *active-set methods* and *interior-point methods*.

5.4.1 Active-Set Methods

Active set methods rely on identifying the subset of inequality constraints that are binding the solution for a given iteration. These constraints, which necessarily are linearly independent due to strong duality, are used to construct a subproblem, whereas the non-binding ones are discarded.

The sequential quadratic programming (SQP) iteratively approximates the nonlinear optimization problem in Equation (5.1) by subproblems. At each iterate $(\boldsymbol{x}^{(i)}, \boldsymbol{\zeta}^{(i)})$ it applies Newton's method to the KKT conditions and solves the resulting Taylor approximation of the change in the Lagrangian function with linearized constraints for the search direction $\boldsymbol{d}$

$$\underset{\boldsymbol{d}}{\text{minimize}} \quad \frac{1}{2}\boldsymbol{d}^{\mathsf{T}}\nabla^2_{\boldsymbol{x},\boldsymbol{x}}\Lambda(\boldsymbol{x}^{(i)}, \boldsymbol{\zeta}^{(i)}, \boldsymbol{\xi}^{(i)})\boldsymbol{d} + \nabla_{\boldsymbol{x}} F(\boldsymbol{x}^{(i)})^{\mathsf{T}}\boldsymbol{d} \tag{5.18a}$$

$$\text{subject to} \quad \nabla_{\boldsymbol{x}}\mathbf{g}(\boldsymbol{x}^{(i)})^{\mathsf{T}}\boldsymbol{d} + \mathbf{g}(\boldsymbol{x}^{(i)}) = \mathbf{0} \tag{5.18b}$$

$$\nabla_{\boldsymbol{x}}\mathbf{h}(\boldsymbol{x}^{(i)})^{\mathsf{T}}\boldsymbol{d} + \mathbf{h}(\boldsymbol{x}^{(i)}) \leq \mathbf{0}. \tag{5.18c}$$

By using the solution $\boldsymbol{d}$, it defines the new iterate until convergence. If the initial guess is close enough to the minimum, and the guess on the active constraints do not change over sequential iterations, the SQP methods act like a Newton method and show quadratic convergence [NW06, p.533].

SQP methods became the standard method for nonlinear optimization in early 1980s. Motion planning has already benefited from its strength. Ziegler used an SQP solver [GI83] for motion planning for the Bertha-Benz Memorial Route [Zie+14a]. Taş used another SQP solver [Kra88; Joh14] for planning multiple maneuvers for the same problem setting [Taş14].

5.4.2 Interior-Point Methods

Interior point methods (IPM) are the newest class of nonlinear optimization methods and are the strongest competitor to SQP methods [Wri05]. These methods replace inequality constraints with equality constraint by introducing slack variables $\boldsymbol{l} \in \mathbb{R}^m$, $\boldsymbol{l} \geq \mathbf{0}$ and utilize a logarithmic barrier term. The resulting optimization problem has the form

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{l}}{\text{minimize}} \quad & F(\boldsymbol{x}) - \boldsymbol{\mu}^{\mathsf{T}} \log \boldsymbol{l} && \text{(5.19a)} \\ \text{subject to} \quad & \mathbf{g}(\boldsymbol{x}) = \mathbf{0} && \text{(5.19b)} \\ & \mathbf{h}(\boldsymbol{x}) - \boldsymbol{l} = \mathbf{0}. && \text{(5.19c)} \end{aligned}$$

Because of the barrier term, an optimal solution is obtained as $\boldsymbol{\mu} \to \mathbf{0}$, which is always a feasible, interior point. Therefore, during optimization, the value of $\boldsymbol{\mu}$ is continuously decreased until KKT conditions for an equality constrained problem are met. Most of the IPMs follow the continuation (or homotopy) approach and solve the primal-dual problem [NW06, p. 570].

The terms *IPMs* and *barrier methods* are frequently used interchangeably. The difference lies in the slack variables IPMs utilize. Whereas barrier methods require starting from an interior point, slack variables allow IPMs to start from any point, including infeasible ones.

In large problems IPMs are generally faster than SQP methods. However, slack variables introduced with constraints increase the computational cost, making them perform worse than SQP methods in problems with a high number of constraints. Since they do not apply Taylor's approximation on KKT conditions like SQP methods, IPMs converge to a solution with higher accuracy.

The IPM solver Ipopt is known as the most powerful and robust nonlinear optimization algorithm available [WB06]. It is a primal-dual algorithm and features a "filter" line search method, a second-order correction method for the step factor selection, and inertia correction capabilities which enable its robust reputation. Its filter line search tackles the optimization problem as a bi-objective optimization problem of reducing the objective value and satisfying the constraints. Whenever it cannot find a feasible line search step factor, it enters into restoration mode and minimizes the constraint violation. With its non-commercial license it has made IPMs widespread.

5.5 Calculating Derivatives

Newton's method requires computation of gradient and Hessians at each iteration. There are four alternative ways to compute these: 1) manually deriving analytic expressions, 2) numerical differentiation with finite differences, 3) symbolic differentiation in computer math engine, 4) automatic differentiation, see Table 5.1.

Derivative Method	Accuracy	Speed	Setup Time	Error Safe
Analytical Derivatives	+++	++	-	-
Numeric Differentiation	-	-	+++	+++
Symbolic Differentiation	+++	+	+	++
Automatic Differentiation	+++	+	++	++
Hybrid Auto. Differentiation	+++	+++	++	++

Table 5.1: Comparison of various differentiation methods based on [Gif+18; Neu+16]. Appropriate handling of analytical derivative expressions can yield results as fast as hybrid automatic differentiation.

Analytical derivatives are time-consuming and error-prone. Finite differences arise as an easy way to automate this process. They evaluate the function at points in the vicinity of point-of-interest. However, they yield inaccurate results due to floating point precision errors and are also pretty slow. Symbolic math engines, by calculating the derivatives as expressions, are not bound to accuracy problems. Even though literature on automatic differentiation claims that they suffer from *expression swell*, modern computer software can simplify the resulting derivative expressions considerably. Nevertheless, they suffer from two major problems that limit its applicability for research purposes. First, derivative expressions are often required to be manually brought into the system, as the current computer software that benefits from symbolic math engine cannot be automatically embedded into compiled source code. Although this does not appear as a problem at first glance, for a continuously changing software of new variables, parameter lengths, and function definitions, it becomes increasingly harder to maintain. The second problem is concerning its flexibility. Similarly for analytic derivatives, conditional statements that depend on the specific numerical values require elaborate handling.

Automatic differentiation follows a different strategy for calculating derivatives. It focuses on numerical evaluation of derivatives rather than their symbolic forms. Central to automatic differentiation is the chain-rule. The chain-rule enables decomposing the derivative of the given expression into derivatives of its elementary components. Hence, a given expression is first decomposed into elementary operations and elementary functions, and their partial derivatives are kept in memory. The trace of such elementary operations is represented by an *expression graph*. For a given numerical value, the derivative of each elementary block is calculated and propagated to the next block.

There are two modes of automatic differentiation, depending on the direction derivatives are propagated: *forward mode* and *reverse mode*. In forward mode, chain-rule to each elementary block in forward direction are applied. This requires one forward pass with function value and derivative calculation for every input direction. In reverse mode, on the other hand, derivatives are calculated from output to input, requiring backwards propagation with the number of output parameters. However, in order to record the dependencies in the expression graph, a forward pass with the function value prior to backwards pass is required. This increases the computational overhead. For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, reverse mode automatic differentiation performs faster when $n \gg m$. This comes with the cost of increased memory requirements [Bay+18].

Forward mode automatic differentiation can be implemented by using *dual numbers*. Dual numbers have the form $v + \dot{v}\epsilon$, where $v, \dot{v} \in \mathbb{R}$ and ϵ is an abstract quantity such that $\epsilon^2 = 0$ and $\epsilon \neq 0$. Evaluating the dual number on an expression yields an expression of the form $f(v + \dot{v}\epsilon) = f(v) + f'(v)\dot{v}\epsilon$, from which the derivative of an input is obtained by setting $\epsilon = 1$. The straightforward way of implementing reverse mode automatic differentiation is *operator overloading*, which essentially overloads existing elementary functions and operators, as done in forward mode by using dual numbers. The other way is *source-code transformation*, but it is becoming increasingly less popular [Bay+18].

Automatic differentiation is an error-safe technique allowing a great flexibility. It provides derivatives at machine precision and have a small factor of computational overhead. Available software libraries deploy complex techniques to improve robustness against corner-cases and performance, such as retaping, checkpointing, and region-based memory management [Mar19].

In the programming language C++, the most powerful automatic differentiation library is CppAD [Bel21]. Although the performance of a library strongly depends on the problem at hand, it shows the best performance in several benchmarks. It employs sparse-matrix methods to reduce the storage and computational costs. Another library in C++ that features sparse-matrix methods is CasADi [And+19b]. Even though the benchmarks suggest it has inferior performance, motion planning experiments conducted in this work show comparable performance with CppAD. The main deficit of CasADi is that it cannot directly be integrated into existing software and requires a complete rewrite. In the beginning of this section, practical limitations of using symbolic derivatives was highlighted. The library CppADCodeGen, which is built on top of CppAD, performs *"hybrid automatic differentiation"* [Joã09]. It uses operator-overloading and produces optimized source-code. The source code can be statically compiled and linked dynamically, or used in just-in-time fashion. In this way, it yields the highest performance and is especially useful for motion planning for rigid body dynamics [Gif+18; Neu+16].

5.6 Motion Planning as an Optimization Problem

The uncertainties a motion planner must tackle, the mathematical formulation of driving goals and vehicle models, methods to maintain safety, and numerical optimization based solver algorithms are already presented. This section combines these models and methods, and presents a novel motion planner.

5.6.1 Decision-Theoretic MPC

Planning in receding horizon can be done efficiently by treating it as a numerical optimization problem and solving it repeatedly as the environment changes with time. Planning method of this type is called *model predictive control* (MPC).

Homotopy-class-free planning, introduced in Section 4.1, postpones the decision at the time of planning and performs passive information gathering. Herewith, it suffers from a major problem: postponing decisions may end with more but conflicting information and can cause ambiguity, eventually resulting in unsafe motion.

On the other side, proactive Z-plans, as introduced in Section 4.4, lead to defensive and transparent maneuver plans, which are prone to exploitation by other vehicles. A remedy is to set the planning algorithm to postpone its full-braking Z-plan until the very last moment. However, this may result in uncomfortable motion. Moreover, the criteria on rule-violation in Equation (4.55) is based on deterministic thresholds, delivering a binary result on the rule-compliance of the other vehicle. A smooth transition utilizing continuous probabilistic measure should be preferred for comfort reasons.

The strengths of one approach cover the weaknesses of the other, highlighting the benefits of the hybrid method *decision-theoretic MPC*. It allows integrating various rule-compliance detection methods directly into the planning, and allows smooth transitions between maneuvers. As the motion of the first and the second maneuver options have fallbacks, any motion that lies between these maneuvers will have a feasible fallback motion. For this reason, decision-theoretic MPC is capable of planning proactive, optimal motion while not behaving overcautious.

Phantom Object Treatment

The benefit of using decision-theoretic MPC is easily illustrated in case of dubious object detections. Despite utilizing multiple sensors with distinct modalities and temporal filtering [DSW15; Ric+19], the environment information can still contain ambiguity. This leaves two options: 1) suppressing the existence probability to zero 2) completing the existence probability to one, resulting in a false-positive (phantom) detection.

Suppressing the existence can potentially cause an accident, as in the case of Uber's accident on March 18, 2018[2]. Therefore, false positive object detections are more preferred over suppressing, in order to avoid any severe consequences. However, transmitting this to the motion planner will cause braking in the absence of an object in reality. Decision-theoretic MPC can gently tackle such cases.

[2] The accident is known as the "death of Elaine Herzberg", and became the first pedestrian killed by an autonomous car. She was pushing a bicycle across a road in Arizona, US, when the system of the car failed to recognize her.

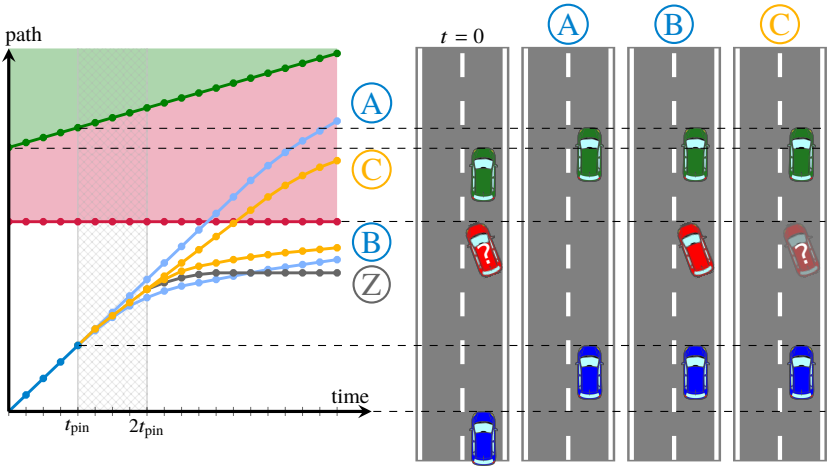


Figure 5.1: A situation where the existence of the red vehicle is unclear. The left-hand side shows the corresponding path–time diagram. Conventional motion planners choose between maneuvers A and B. With decision-theoretic MPC, a neutral maneuver C, the profile of which depends on the existence probability of the red vehicle, is planned. The hatched time interval on the path-time diagram will be driven next.

An exemplary scenario where the ego vehicle has a potential phantom detection is illustrated in Figure 5.1. The ego vehicle will either suppress the existence probability of the red vehicle and plan maneuver A, or will treat it as a real object and plan maneuver B, both while holding the fallback maneuver Z, as presented in Section 4.4.2. A more optimal approach is to plan a maneuver-neutral motion by weighting the maneuvers A and B with detector probabilities and the existence probability of the red vehicle, as presented in [TS20]. This formulation allows the planner to react to the phantom object smoothly, while ensuring fallback braking maneuver Z against the worst-case evolution of the scenario.

5.6.2 Model Predictive Contouring Control

The longitudinal position s along a route is essential in analyzing starting point of traffic rules effectiveness and play a vital role in calculating time-to-intersection, as clarified in the preceding chapters. Therefore, the motion planner must repeatedly calculate its *progress* along its route centerline. This is not only computationally unfavorable, but can further lead to convergence and stability problems, especially in problems with conflicting objectives. The *model predictive contouring control* (MPCC), which includes progress in its state variables, can be used as an effective solution for this.

MPCC is originally defined as a path following control algorithm for machine tools [LMG10], and after demonstrated success in autonomous racing [LDM15], it has gained popularity for motion planning applications. Several later works enhanced this approach for planning under uncertainty [Sch+17], or utilized Gaussian process regression to learn model uncertainties [HKZ19].

MPCC essentially defines cost terms and constraints to minimize lateral and longitudinal tracking errors. During initialization, it first defines the reference Cartesian coordinates and the corresponding yaw angle $\left(x^{\text{ref}}(s_i), y^{\text{ref}}(s_i), \psi^{\text{ref}}(s_i)\right)$ for a desired longitudinal position, or progress, s_i. It subsequently calculates the longitudinal and the lateral error in a coordinate frame that is aligned with the tangent at the reference coordinates

$$e_{\text{lon}} = -(x^{\text{ref}} - x)\sin(\psi^{\text{ref}}) + (y^{\text{ref}} - y)\cos(\psi^{\text{ref}}) \tag{5.20a}$$

$$e_{\text{lat}} = (x^{\text{ref}} - x)\cos(\psi^{\text{ref}}) + (y^{\text{ref}} - y)\sin(\psi^{\text{ref}}), \tag{5.20b}$$

as depicted in Figure 5.2. In addition to these error terms, it defines error on the projected speed on the reference contour $\dot{s}$. For this purpose, it extends the state vector $\boldsymbol{s}$ and control vector $\boldsymbol{a}$ with progress and projected speed as $\boldsymbol{s}_{\text{mpcc}} = [\boldsymbol{s}, s]^\top$ and $\boldsymbol{a}_{\text{mpcc}} = [\boldsymbol{a}, \dot{s}]^\top$, respectively. Using the notation presented in Equation (4.12) the contouring objective is

$$J_{\text{mpcc}}(\boldsymbol{x}, \boldsymbol{a}) = J_{\text{val}}(e_{\text{lon}}) + J_{\text{val}}(e_{\text{lat}}) + J_{\text{val}}(\dot{s}) \tag{5.21}$$

and the constraints on these individual error terms are defined by defining and upper and lower bounds.

5.6.3 Decision-Theoretic MPCC

The presented decision-theoretic MPC framework performs multiple operations with respect to a centerline. The MPCC framework contains the progress variable in its state definition and therefore eases the optimization problem by providing convergence and stability advantages. Therefore, this work applies the path formulation of MPCC on decision-theoretic MPC. The objectives and constraints are now outlined and applied on the MPCC formulation.

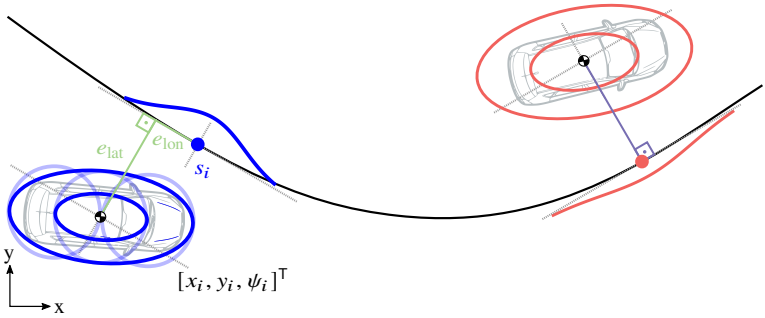


Figure 5.2: Illustration of the MPCC formulation together with uncertainty projection.

The parameter vector of the homotopy-class-free motion planning problem with two distinct maneuver options is

$$\boldsymbol{p} = [\boldsymbol{p}_{\text{pinned}}, \boldsymbol{p}_{\text{C}}, \boldsymbol{p}_{\text{CA}}, \boldsymbol{p}_{\text{CB}}]^\mathsf{T}, \tag{5.22}$$

the same as Equation (4.18). The individual maneuvers, A and B, are used together to calculate the total cost. The parameters that constitute these maneuvers are

$$\boldsymbol{p}_{\text{A}}^\star = \left[\boldsymbol{p}_{\text{pinned}}, \boldsymbol{p}_{\text{C}}, \boldsymbol{p}_{\text{CA}}\right]^\mathsf{T}, \tag{5.23a}$$

$$\boldsymbol{p}_{\text{B}}^\star = \left[\boldsymbol{p}_{\text{pinned}}, \boldsymbol{p}_{\text{C}}, \boldsymbol{p}_{\text{CB}}\right]^\mathsf{T}, \tag{5.23b}$$

respectively.

The presented planning approach relies on ICS and applies safety constraints only for the immediate horizon that will be set fixed in the next replanning. The safety constraints consist of circle-to-ellipse workspace constraints and

constraints on braking distance. As described in Section 4.3.2, the ego vehicle is approximated with three circles, whereas the agents are modeled with ellipses. The braking constraints are for ensuring the validity of fallback maneuvers and thereby the inexistence of inevitable collision states.

The contour defined in the MPCC formulation serves as an ideal reference to project uncertainties and calculate stop positions, as required by the approach presented in Section 4.4.1. In order to consider uncertainties of longer planning horizons, collision soft-constraints are applied on the entire horizon. The collision risk is calculated by projecting the position of the other agent, which is modeled as $\hat{\boldsymbol{x}}_o = \boldsymbol{x}_o + \boldsymbol{\zeta}_0$ where $\boldsymbol{\zeta}_o \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_o)$, onto the reference contour. The mean values of the projection s_o and d_o are obtained by using Equation (2.1) and the variances by using Equation (3.4). The collision risk is obtained as

$$J_{\text{collision}}(\boldsymbol{x}) = w_{\text{coll}} \left(\left(1 + \operatorname{erf}\left(\frac{(s - s_o)}{\sqrt{2\lambda_{\text{lon}}(\boldsymbol{\Sigma}_o)}} \right) \right) \left(1 + \operatorname{erf}\left(\frac{(d - d_o)}{\sqrt{2\lambda_{\text{lat}}(\boldsymbol{\Sigma}_o)}} \right) \right) \right). \tag{5.24}$$

In this way, collision risk is modeled by using univariate distributions. Even though safety in the immediate horizon is ensured by hard-constraints, such additional cost terms yield better convergence properties. The constraints applied on the motion planning are summarized on Table 5.2.

Constraint	Equation	$\boldsymbol{p}_{\text{pinned}}$	$\boldsymbol{p}_{\text{shared}}$	$\boldsymbol{p}_{\text{maneuver,1}}$	$\boldsymbol{p}_{\text{maneuver,2}}$
Kinematic veh. model	(4.8)	✓	✓	✓	✓
Lon. acceleration	(4.9b)	✓	✓	✓	✓
Lat. acceleration	(4.9a)	✓	✓	✓	✓
Lon. contouring error	(5.20a)	✓	✓	✓	✓
Lat. contouring error	(5.20b)	✓	✓	✓	✓
Circle-ellipse constr.	(4.22)	✓	✓		
Braking chance constr.	(4.47) or (4.52)	✓	✓		
Collision soft-constr.	(5.24)	✓	✓	✓	✓

Table 5.2: Active constraints on the parameters of the decision-theoretic MPCC.

The cost of an individual maneuver option is calculated as

$$J_{\text{total}}(\boldsymbol{x}, \boldsymbol{a}) = J_{\text{driving}} + J_{\text{comfort}} + J_{\text{mpcc}} + J_{\text{collision}}, \tag{5.25}$$

where the projected speed summand in J_{mpcc} is calculated with the asymmetric loss $J_{\text{asym}}(\dot{s})$ defined in Equation (4.14) instead of $J_{\text{val}}(\dot{s})$, as done in Equation (4.16).

The cost of a planned motion is

$$J^{\star}_{\text{total}} = w_{\text{A}}\, J_{\text{total}}\left(\boldsymbol{p}^{\star}_{\text{A}}\right) + w_{\text{B}}\, J_{\text{total}}\left(\boldsymbol{p}^{\star}_{\text{B}}\right), \tag{5.26}$$

where the weight factors w_{A} and w_{B} are linearly dependent to the respective maneuver probability.

The presented planning scheme assumes for the ego vehicle a constant uncertainty along the planning horizon. Even though this is in contrast to stochastic MPC frameworks [Wei09], this is a reasonable assumption. The operating conditions and uncertainties remain unchanged for most of the time, and even if confidence deteriorates, the vehicle is able to execute a fallback maneuver.

The motion planner, due to the structure of the parameter vector and respective constraints, does not have band-diagonal gradient or Hessian matrices. Therefore, this planning scheme especially benefits from sparse-matrix methods. The motion planner further utilizes the smooth objective change feature described in Section 4.1.3. While replanning, the initialization is obtained by filling the missing data at the end of the previous solution with extrapolation.

6 Planning with Sequential Decision Making

Motion planning is essentially a decision making under uncertainty problem and can be framed as a *partially observable Markov decision process* (POMDP). Although POMDPs allow great flexibility in modeling, finding exact solutions of them is generally computationally intractable. In real-time applications, it is common to approximate the solution with sampling-based approaches.

This chapter starts with introducing sequential decision making and the fundamentals of decision processes. It subsequently presents a sampling-based tree-search approach to tackle decision problems. This tree-search approach utilizes multi-armed bandits for action selection, which are suboptimal for motion planning. Hence, alternative multi-armed bandit algorithms are proposed next. This chapter then continues with presenting an algorithm that employs the aforementioned tree-search to solve a POMDP problem. However, this very established algorithm proves to be inefficient for continuous action and observation spaces. It further cannot consider belief-dependent information rewards. Thus, a new algorithm to overcome these deficits is introduced. The chapter finally presents the modeling to frame the motion planning for autonomous driving task as a POMDP.

6.1 Sequential Decision Making

Decision processes model an agent's decisions for changes in its environment with the goal of optimizing its *utility* u. Whereas in model predictive control the objective is typically to minimize the costs over the horizon, in sequential decision making *rewards* r over a sequence of actions are maximized. In the sense of their objective, they can be treated as identical. The strength of sequential decision making and its main difference with model predictive control

comes from the flexibility of the models that can be integrated. While numerical optimization based planning requires objective and constraint functions to be smooth and differentiable, it allows for complex, arbitrary models. Further, in contrast to numerical optimization based methods, utility maximization can be done over nested models without any additional effort.

6.1.1 Markov Decision Processes

Decision making in stochastic environments with fully observable states can be framed as a *Markov decision process* (MDP) under the assumption that state transitions are *Markovian*, i.e. the next state depends only on the current state and action, and not on their priors. An MDP is characterized by the tuple $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ is the action space, $T(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a})$ is the transition model which defines the distribution of the successor state $\boldsymbol{s}'$ after taking an action $\boldsymbol{a}$ from state $\boldsymbol{s}$, $R(\boldsymbol{s}, \boldsymbol{a})$ is the reward model that represents the reward r obtained while executing an action $\boldsymbol{a}$ from state $\boldsymbol{s}$, and $\gamma \in [0, 1)$ is the discount factor for future rewards ensuring a finite cumulative reward, as long as the rewards are finite. In general, higher discounts lead to shorter planning horizons resulting in *myopic* behavior. The probabilistic graphical model of an MDP is given in Figure 6.1a.

The action selected at time t is determined by the *policy* π. Because of the Markov assumption, the policy of an MDP is a function of the current state

$$\boldsymbol{a} = \pi^{(t)}(\boldsymbol{s}). \tag{6.1}$$

In infinite horizon MDPs, in which the transitions and rewards are stationary, policies can be modeled stationary as well [Koc15, p. 79].

Executing the policy π in state $\boldsymbol{s}$ returns an expected cumulative reward under the policy π. This is called the *state-value function* or *value function* and for infinite horizon problems it is defined as

$$U^{\pi}(\boldsymbol{s}) \coloneqq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R\left(\boldsymbol{s}^{(t)}, \boldsymbol{a} = \pi(\boldsymbol{s}^{(t)})\right) \;\middle|\; \boldsymbol{s}^{(0)} = \boldsymbol{s}\right] \tag{6.2}$$

$$= R\left(\boldsymbol{s}, \pi(\boldsymbol{s})\right) + \gamma \sum_{\boldsymbol{s}'} T\left(\boldsymbol{s}' \mid \boldsymbol{s}, \pi(\boldsymbol{s})\right) U^{\pi}\left(\boldsymbol{s}'\right). \tag{6.3}$$

In state $\boldsymbol{s}$ the expected cumulative reward after taking an action $\boldsymbol{a}$ and following the policy π afterwards is called *action-value function* or *Q-value function* and for infinite horizons problems it is defined as

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) \coloneqq \mathbb{E}\left[R\left(\boldsymbol{s}, \boldsymbol{a}\right) + \sum_{t=1}^{\infty} \gamma^t R\left(\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)} = \pi(\boldsymbol{s}^{(t)})\right)\right] \tag{6.4}$$

$$= R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'} T\left(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}\right) U^{\pi}(\boldsymbol{s}'). \tag{6.5}$$

Different policies can be compared by their expected utility. A *rational* agent will always choose the optimal policy

$$\pi^* = \arg\max_{\pi} U^{\pi}(\boldsymbol{s}) \tag{6.6}$$

yielding the highest expected utility. One way to obtain the optimal policy is to use the value iteration algorithm, which recursively applies the *Bellman update*

$$U_{k+1}(\boldsymbol{s}) \coloneqq \max_{\boldsymbol{a}} \left(R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'} T\left(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}\right) U_k\left(\boldsymbol{s}'\right)\right), \tag{6.7}$$

to improve the value function. The Bellman update can be operationalized in form of $B : \mathbb{R}^{|\boldsymbol{s}|} \mapsto \mathbb{R}^{|\boldsymbol{s}|}$. It can be shown that such an operator is a γ-contraction for $0 \leq \gamma < 1$ with respect to ℓ_∞-norm, guaranteeing that successive operations of B will converge to a unique point, which is the optimal utility U^* for Equation (6.7). Once U^* is known, π^* can be extracted by using

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \left(R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}'} T\left(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}\right) U^*\left(\boldsymbol{s}'\right)\right). \tag{6.8}$$

The existence of U^* for a policy π implies the existence of Q^*. Therefore,

$$U^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}). \tag{6.9}$$

and

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a}). \tag{6.10}$$

In this way, exact solutions of decision problems in finite spaces can be found.

6.1.2 Partially Observable Markov Decision Processes

In many sequential decision-making problems the system state is not observable to the agent. Such decision problems can be framed as a POMDP, which is essentially an MDP with a model of possible observations, see Figure 6.1b. Therefore, a POMDP extends an MDP with an observation space $\mathbb{O}$ and an observation model $Z(\boldsymbol{o}^{(t)} \mid \boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t-1)})$ that gives the probability of observing $\boldsymbol{o}$ after executing the action $\boldsymbol{a}$ and arriving at state $\boldsymbol{s}$. Like transition models and rewards in MDPs, observation models are modeled as time-invariant.

A POMDP infers the current state from the *history* of observations and actions $\boldsymbol{h}^{(t)} = \left(\boldsymbol{o}^{(0:t)}, \boldsymbol{a}^{(0:t-1)}\right)$. Therefore, the decisions of a POMDP do not depend only on the current observation, but on the whole history. However, the full history trace of observations and actions can become very long, eventually making its maintenance impractical. Instead, the history can be summarized by belief $\boldsymbol{b}$ over the state space $\mathbb{S}$ [Åst65; PGT06]. Starting from an initial belief $\boldsymbol{b}^{(0)}$, the belief of being in state $\boldsymbol{s}$ is obtained by the posterior distribution

$$\boldsymbol{b}(\boldsymbol{s}) = P(\boldsymbol{s} \mid \boldsymbol{h}, \boldsymbol{b}^{(0)}). \tag{6.11}$$

Because of the *equivalence of belief and history*, a policy in a POMDP maps current belief to an action

$$\boldsymbol{a} = \pi(\boldsymbol{b}). \tag{6.12}$$

A POMDP can be viewed as an MDP that operates on *belief-states*, the so-called *belief* MDP, see Figure 6.1c. For this, belief dependent reward and transition models are required. Such reward function can be defined as

$$\rho(\boldsymbol{b}, \boldsymbol{a}) = \sum_{\boldsymbol{s}} R(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{b}(\boldsymbol{s}) \tag{6.13}$$

which essentially is the expected cumulative reward for all possible belief-states. The state transition of a belief-state MDP $P(\boldsymbol{b}' \mid \boldsymbol{b}, \boldsymbol{a})$ can be approximated by taking advantage of the problem structure, as will be shown in the next section.

The value function and the action-value functions for the belief MDP become

$$U^{\pi}(\boldsymbol{b}) \coloneqq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \rho\left(\boldsymbol{b}^{(t)}, \boldsymbol{a}^{(t)} = \pi(\boldsymbol{b}^{(t)})\right) \;\middle|\; \boldsymbol{b}^{(0)} = \boldsymbol{b}\right] \tag{6.14}$$

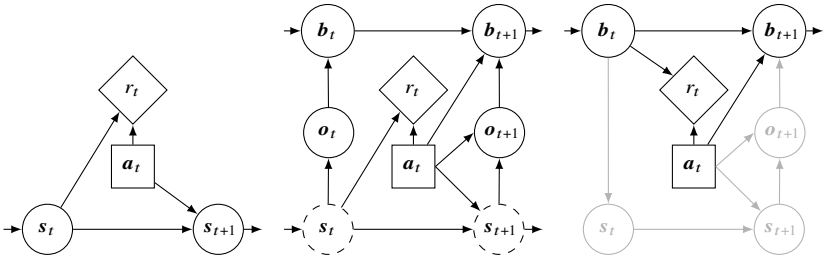
and

$$Q^{\pi}(\boldsymbol{b}, \boldsymbol{a}) \coloneqq \mathbb{E}\left[\rho\left(\boldsymbol{b}, \boldsymbol{a}\right) + \sum_{t=1}^{\infty} \gamma^t \rho\left(\boldsymbol{b}^{(t)}, \boldsymbol{a}^{(t)} = \pi(\boldsymbol{b}^{(t)})\right)\right] \tag{6.15}$$

$$= \rho\left(\boldsymbol{b}, \boldsymbol{a}\right) + \gamma \sum_{\boldsymbol{b}'} P\left(\boldsymbol{b}' \mid \boldsymbol{b}, \boldsymbol{a}\right) U^{\pi}(\boldsymbol{b}') \tag{6.16}$$

respectively. It can be shown that $\pi^*(\boldsymbol{b})$ for this belief MDP is also the optimal policy of the underlying POMDP [RN16, p. 672].

Reformulating a POMDP as a belief MDP does not necessarily simplify the problem. Since a belief is a probability distribution, even if the state space is finite containing $|\mathbb{S}|$ states, the corresponding belief is defined over $(|\mathbb{S}| - 1)$-dimensional continuous *belief space* $\mathbb{B}$. This phenomenon is called the *curse of dimensionality*. A further problem is the *curse of history* that describes the exponential growth in the number of distinct possible action-observation histories along the planning horizon. Additional complexity arises from the calculation of belief dependent Equation (6.14) and Equation (6.15). Therefore, compared to an MDP with finite number of states, a POMDP resembles a much harder problem to solve [PT87].



(a) MDP diagram. (b) POMDP diagram. (c) Belief MDP diagram.

Figure 6.1: Comparison of probabilistic graphical models of an MDP, a POMDP, and a belief MDP.

6.2 Sequential State Estimation

A belief MDP must perform belief updates upon new actions and observations. Given observation and transition models, the next belief can be calculated for discrete state, observation and action spaces as

$$\begin{aligned}
\boldsymbol{b}'(\boldsymbol{s}') &= P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}, \boldsymbol{o}) = P(\boldsymbol{s}^{(t+1)} \mid \boldsymbol{b}^{(t)}, \boldsymbol{a}^{(t)}, \boldsymbol{o}^{(t+1)}) \\
&= \frac{P(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{b}, \boldsymbol{a}) P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a})}{P(\boldsymbol{o} \mid \boldsymbol{b}, \boldsymbol{a})} \\
&= \frac{P(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{b}, \boldsymbol{a}) \sum_{\boldsymbol{s}} P(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{b}, \boldsymbol{a}) P(\boldsymbol{s} \mid \boldsymbol{b}, \boldsymbol{a})}{P(\boldsymbol{o} \mid \boldsymbol{b}, \boldsymbol{a})} \\
&= \frac{P(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{a}) \sum_{\boldsymbol{s}} P(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) P(\boldsymbol{s})}{P(\boldsymbol{o} \mid \boldsymbol{b}, \boldsymbol{a})} \\
&= \eta \, Z(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{a}) \sum_{\boldsymbol{s}} T(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \boldsymbol{b}(\boldsymbol{s}),
\end{aligned} \tag{6.17}$$

where the normalization factor η is

$$\begin{aligned}
\eta^{-1} &= P(\boldsymbol{o} \mid \boldsymbol{b}, \boldsymbol{a}) \\
&= \sum_{\boldsymbol{s}'} P(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{b}, \boldsymbol{a}) P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}) \\
&= \sum_{\boldsymbol{s}'} Z(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{a}) \sum_{\boldsymbol{s}} T(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \boldsymbol{b}(\boldsymbol{s}).
\end{aligned} \tag{6.18}$$

Equation (6.17) corresponds to the general *Bayes filter* algorithm for finite state spaces. Due to multiplication and sum, the equation can only be calculated under strong assumptions on the shape of underlying distributions. Chapter 3 has demonstrated the advantages of using non-parametric approaches for uncertainty modeling.

The elements of the problem structure used during belief update in a belief MDP is illustrated in light gray in Figure 6.1c.

6.2.1 Particle Filter

Particle filters (PF) approximate the belief with a set of weighted state samples $\{(\boldsymbol{s}_i, w_i)\}_{i=1}^n$ [Aru+02; DGA00]. The particle approximation of the current belief given the information *prior* to this time is

$$\boldsymbol{b}(\boldsymbol{s}) = P(\boldsymbol{s} \mid \boldsymbol{h}, \boldsymbol{b}^{(0)}) \approx \hat{\boldsymbol{b}}(\boldsymbol{s}) = \sum_{i=1}^{n} w_i \delta(\boldsymbol{s} - \boldsymbol{s}_i). \tag{6.19}$$

The *predicted* density of the state after taking the action $\boldsymbol{a}$ can be obtained by using the transition model

$$P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}) \approx \sum_{i=1}^{n} w_i T(\boldsymbol{s}' \mid \boldsymbol{s}_i, \boldsymbol{a}). \tag{6.20}$$

The predicted density can be approximated by a set of particles $\boldsymbol{s}_i'$ sampled from the density $T(\boldsymbol{s}' \mid \boldsymbol{s}_i, \boldsymbol{a})$

$$P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}) \approx \sum_{i=1}^{n} w_i \delta(\boldsymbol{s}' - \boldsymbol{s}_i'). \tag{6.21}$$

After receiving an observation $\boldsymbol{o}$, the approximated density can be updated with the new measurement

$$P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}, \boldsymbol{o}) \approx \eta \sum_{i=1}^{n} w_i Z(\boldsymbol{o} \mid \boldsymbol{s}', \boldsymbol{a}) T(\boldsymbol{s}' \mid \boldsymbol{s}_i, \boldsymbol{a}), \tag{6.22}$$

where η is the normalization factor. This *posterior* distribution weights w_i with the observation likelihood $\widetilde{w}_i' = w_i Z(\boldsymbol{o} \mid \boldsymbol{s}_i', \boldsymbol{a})$, and afterwards normalizes them $w_i' = \widetilde{w}_i' / \sum_n \widetilde{w}_n'$ to account for the normalization factor η, which was also presented in Equation (6.17)). The particle approximation of the posterior density becomes

$$\boldsymbol{b}'(\boldsymbol{s}') = P(\boldsymbol{s}' \mid \boldsymbol{b}, \boldsymbol{a}, \boldsymbol{o}) \approx \hat{\boldsymbol{b}}'(\boldsymbol{s}') = \sum_{i=1}^{n} w_i' \delta(\boldsymbol{s}' - \boldsymbol{s}_i'). \tag{6.23}$$

In this way, the importance sampling method presented in Section 3.2.1 is applied to sequential state estimation problem.

This sampling procedure often suffers from the particle degeneracy problem, which describes the case where only a small portion of the weights have a non-negligible weight, as introduced in Section 3.2.1. A common solution is to track the effective sample size, which is a function of the variance of the weights and the number of particles, and resample new particles with reset particle weights. In some cases, resampling can result in repeated selection of a few particles, i.e. *particle deprivation* or *particle impoverishment*. This can be avoided by adding noise to the particle set to reduce such an overfitting [TBF05, p. 113]. The resulting algorithm is called the *generic PF* or *sequential importance resampling (SIR) PF* and is outlined in Algorithm 6.1.

Algorithm 6.1: Sequential Importance Resampling Particle Filter

Data: $\hat{\boldsymbol{b}} = \{\boldsymbol{s}_i, w_i\}_{i=1}^n, \boldsymbol{a}, \boldsymbol{o}$
Result: $\hat{\boldsymbol{b}}' = \{\boldsymbol{s}_i', w_i'\}_{i=1}^n$
$\boldsymbol{b}' \leftarrow \emptyset$
for $i = 1, \ldots, n$ **do**
  $\boldsymbol{s}_i' \sim T(\boldsymbol{s}' \mid \boldsymbol{s}_i, \boldsymbol{a})$ ▷ *sample from predicted density*
  $\widetilde{w}_i' = w_i Z(\boldsymbol{o} \mid \boldsymbol{s}_i', \boldsymbol{a})$ ▷ *measurement update*
end
$w_i' = \widetilde{w}_i' / \sum_n \widetilde{w}_n'$ ▷ *normalize all weights*
if resampling necessary **then**
  `Resample`$(\{\boldsymbol{s}_i', w_i'\}_{i=1}^n)$ ▷ *substitute particles & reset weights*
end
if noise necessary **then**
  `Noise`$(\{\boldsymbol{s}_i', w_i'\}_{i=1}^n)$ ▷ *replace some particles by random particles*
end
$\boldsymbol{b}' \leftarrow \{\boldsymbol{s}_i', w_i'\}$

6.2.2 Unweighted Particle Filter

The generic PF requires querying transition probability and observation likelihood at specific points. In large problems the transition and observation can be modeled with generative models $\mathcal{G}$. These operate in a black box fashion and do not provide the density of transition and observation models, as required in the generic particle filter. The unweighted PF estimates belief by performing

rejection sampling and therefore requires only samples from these black box models. Its operation is described in Algorithm 6.2. Like generic PF, it benefits from resampling and noising strategies [SV10].

Algorithm 6.2: Particle Filter with Rejection

```
1 b' ← ∅
2 for i = 1, ..., |b| do
3 |   s ← Random(b)                          ▷ pick random state
4 |   repeat
5 |   |   (s', o') ~ G(s, a)                 ▷ sample with generative model
6 |   until o' = o                           ▷ sample matches actual observation
7 |   b' ← b' + s'
8 end
```

6.3 Solving Markov Decision Processes with State Uncertainty in Real-Time

Solving a POMDP requires optimizing over a sequence of actions and observations to consider a very large number of possible outcomes. This comes at the cost of high computational complexity, as clarified in Section 6.1.2. The value iteration algorithm for POMDPs computes the value over the complete belief space, independent of how likely a belief-state is reached. Some solvers exploit the fact that the optimal value function for a finite horizon POMDP is piecewise linear and convex, and approximate it with vectors [SS73]. Other solvers define bounds on value function [Hau97], evaluate the value function at certain points [PGT03], or employ heuristics to choose the beliefs to update [SS04]. In large problems all of these approaches are ineffective and result in suboptimal policies. An efficient approach is to compute policy for the current belief, and upon any change in the belief, to replan it.

6.3.1 Monte Carlo Tree Search

Monte Carlo tree search (MCTS), used for solving MDPs, incrementally builds a search tree of states $\boldsymbol{s}$ representing nodes, and actions $\boldsymbol{a}$ representing edges until an end criterion is met [Bro+12]. Every node contains a Q-value $Q(\boldsymbol{s}, \boldsymbol{a})$ that is estimated by the mean return of all successor states $\boldsymbol{s}'$ in which action $\boldsymbol{a}$ was selected, and a visitation count $N(\boldsymbol{s}, \boldsymbol{a})$ for each action, and an overall count $N(\boldsymbol{s}) = \sum_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{a})$.

The MCTS algorithm proceeds in four steps. In *simulation* phase it starts from the initial node and selects actions according to the tree policy. The simulation procedure continues until it reaches a node that is not in the tree yet. The tree is *expanded* by that node and a *rollout* is performed to estimate the value of that node. The policy used in the rollout allows for incorporating domain-knowledge into MCTS and thereby can result in faster convergence. In the final step, *backup*, the parent nodes are updated from bottom-up with the gathered information. Recursive application of these four steps of MCTS improves Q-value estimates iteratively, and as soon as the end criterion is met, the action yielding the highest Q-value is selected. Sequential decision-making problems can in this way be solved in anytime-fashion.

In MCTS every action corresponds to a reward distribution that is unknown while selecting actions. The vanilla MCTS algorithm samples actions from uniform distribution resulting in *flat* Monte Carlo trees. Such a sampling is highly suboptimal and results in slow convergence. A much more efficient way is to utilize information from previous simulations. But in this case, the action that is believed to be optimal might be overestimated due to its relative high number of samples. Choosing suboptimal actions can potentially reveal higher rewards. This resembles the *exploration vs. exploitation dilemma*. To tackle this, Kocsis et al. formulated action selection at nodes as a multi-armed bandit (MAB) problem and employed the Upper Confidence Bound (UCB) MAB algorithm to refine action selection. They renamed their algorithm as Upper Confidence Bound Applied to Trees (UCT) [KS06]. In this way, search for the optimal action is guided to regions where higher rewards are likely.

6.3.2 Multi-Armed Bandits

Bandit problems are a class of a sequential decision problem, where the algorithm picks one arm $\boldsymbol{a}$ from some set of arms $|\mathcal{A}| = k$ in every round $t \in \{1, \ldots, T\}$ with the goal of maximizing the rewards it collects over T rounds [Sli19]. The reward of each arm is a distribution that is initially unknown to the algorithm. These reward distributions are modeled independent and identically distributed, and stationary. Since the distributions are unknown, a maximization of reward requires trading-off between the exploration of unknown arms and exploitation of good arms. There are many MAB algorithms to tackle this problem.

Arms of MABs correspond to actions, and rewards to Q-values in decision processes. In MCTS, a new bandit round is executed upon visiting a state, therefore $t = N(\boldsymbol{s})$. Likewise, $N(\boldsymbol{s}, \boldsymbol{a})$ denotes the number of time an arm is played. The following subsections present bandit equations for the MCTS-context.

Upper Confidence Bound

The most widely used MAB in decision making is the Upper Confidence Bound (UCB) algorithm [ACF02]. The algorithm calculates the reward of an arm by augmenting its average reward by an exploration bonus that is highest for rarely tried actions

$$Q^{\oplus}_{\mathrm{UCB}}(\boldsymbol{s}, \boldsymbol{a}) = Q(\boldsymbol{s}, \boldsymbol{a}) + c \sqrt{\frac{2 \log N(\boldsymbol{s})}{N(\boldsymbol{s}, \boldsymbol{a})}}. \tag{6.24}$$

The first term is the current Q-value for an arm $\boldsymbol{a}$ and therefore, resembles exploitation. The second term is related to the upper confidence interval of the average reward serves for exploration. The parameter $c \in \mathbb{R}^+$ in the second term is the *exploration constant* that trades-off exploitation and exploration. The operation of UCB MAB is presented in Algorithm 6.3.

The UCB algorithm owes its widespread application to the thorough theoretical analysis on its regret, i.e. the cumulative difference between optimal and actual reward. The expected regret of UCB is logarithmic in the number of bandit

Algorithm 6.3: Upper Confidence Bound (UCB)

```
function UCB(s):
    if N(s) ≤ k then
        return Random({a : N(s, a) = 0})          ▷ try all arms at least once
    else
        return arg max_{a∈A} Q⊕_UCB(s, a)         ▷ choose the most promising arm
    end
```

rounds. UCB can be seen as a subclass of KL-UCB algorithm, KL short for Kullback-Leibler, that is adapted for Gaussian distributed rewards [GC11].

Upper Confidence Bound with Variances

The UCB algorithm assumes the variance of the rewards of arms as equal and pre-definable, which might be invalid in real applications. Replacing $\sigma^2(\boldsymbol{a})$ with empirical variance $\hat{\sigma}^2_N(\boldsymbol{a})$ of the different arms in the KL-UCB algorithm leads to poor results. The UCB-V algorithm properly utilizes empirical variance during UCB action selection [AMS07]. The arm to be selected is given as

$$Q^{\oplus}_{\text{UCBV}}(\boldsymbol{s},\boldsymbol{a}) = Q(\boldsymbol{s},\boldsymbol{a}) + \sqrt{\frac{2\hat{\sigma}^2_N(\boldsymbol{a})\log N(\boldsymbol{s})}{N(\boldsymbol{s},\boldsymbol{a})}} + \frac{3c\log N(\boldsymbol{s})}{N(\boldsymbol{s},\boldsymbol{a})}. \tag{6.25}$$

With growing $N(\boldsymbol{s},\boldsymbol{a})$, the influence of the exploration constant, modeled with the third term, diminishes and the estimated variances are more strongly considered.

Pareto Optimal Sampling for Lipschitz Bandits

The UCB bandit algorithm does not define any relation between the rewards of the arms. Therefore, the reward of an arm is independent of the reward of another. This is in contrast to continuous dynamical systems, where similar actions return similar outcomes. The Q-value function may, however, have discontinuities depending on the selected reward function. In motion planning, collisions, for example, cause jumps for some action values. However, as presented before,

motion planning is bound to uncertainties. Despite discontinuous reward definition, present uncertainties result in more scattered Q-value profiles around the discontinuity, as inspected in Chapter 7. This scatter allows for Lipschitz continuity assumption on the Q-value profile. If a bandit algorithm exploits the Lipschitz continuity, much faster convergence to the optimal solution can be obtained.

The Pareto Optimal Sampling for Lipschitz Bandits (POSLB) algorithm assumes that the expected rewards is a Lipschitz continuous function over the discrete bandit arms. By utilizing Lipschitz continuity, it derives a regret lower bound for Lipschitz continuous rewards. This allows to guide the bandit to more rewarding arms and to improve sampling efficiency [Mag18, p. 21].

The expected reward function follows

$$|Q(\boldsymbol{s}, \boldsymbol{a}) - Q(\boldsymbol{s}, \boldsymbol{a}')| \leq \mathscr{L} \, |\boldsymbol{a} - \boldsymbol{a}'|$$

for all pairs of arms $(\boldsymbol{a}, \boldsymbol{a}')$ and a Lipschitz constant $\mathscr{L}$ that is known in beforehand. POSLB in its original work is derived for arbitrary reward distributions. Like UCB, rewards can be modeled with Gaussian distributions of equal variance. Algorithm 6.4 presents POSLB for Gaussian distributed rewards.

In the first k rounds POSLB chooses all arms once. After all arms have been chosen once, it determines the *leader* arm for that round according to Equation (6.24). Subsequently, it looks into different arms and picks the arm for which the difference between the actual estimated reward $Q(\boldsymbol{s}, \boldsymbol{a}')$ and the expected reward $\lambda_N(\boldsymbol{a}, \boldsymbol{a}')$ weighted by their visit count is the smallest. The vector λ_N can be interpreted as the *most confusing* reward vector that would make a suboptimal arm the optimal one [Mag18, p. 17]. Therefore, it explores suboptimal arms based on how *well* they satisfy the Lipschitz constraint. The POSLB algorithm integrates Lipschitz continuity assumption with the parameter $\mathscr{L}$ and in lieu of an increased computational complexity of $O(|\mathcal{A}|^2)$ compared to UCB.

Algorithm 6.4: POSLB for Gaussian distributed rewards

function $\lambda_N(\boldsymbol{a}, \boldsymbol{a}')$**:**
  return $\max\left(Q^{\oplus}_{\mathrm{UCB}}(\boldsymbol{s}, \boldsymbol{a}^*_N) - \mathscr{L}\,|\boldsymbol{a} - \boldsymbol{a}'|,\ Q(\boldsymbol{s}, \boldsymbol{a}')\right)$
function $F_N(\boldsymbol{a}, \boldsymbol{a}^*_N)$**:**
  if $\boldsymbol{a} \neq \boldsymbol{a}^*_N$ **then**
    return $\sum_{\boldsymbol{a}' \in \mathcal{A}} N(\boldsymbol{s}, \boldsymbol{a}') \left(Q(\boldsymbol{s}, \boldsymbol{a}') - \lambda_N(\boldsymbol{a}, \boldsymbol{a}')\right)^2 (2\sigma^2)^{(-1)}$
  else
    return $N(\boldsymbol{s}, \boldsymbol{a}) \left(Q(\boldsymbol{s}, \boldsymbol{a}) - Q^{\oplus}_{\mathrm{UCB}}(\boldsymbol{s}, \boldsymbol{a}^*_N)\right)^2 (2\sigma^2)^{(-1)}$
  end
function `POSLB`$(\boldsymbol{s})$**:**
  if $N(\boldsymbol{s}) \leq k$ **then**
    return `Random`$(\{\boldsymbol{a} : N(\boldsymbol{s}, \boldsymbol{a}) = 0\})$ ▷ *try all arms at least once*
  else
    $\boldsymbol{a}^*_N \leftarrow \arg\max_{\boldsymbol{a} \in \mathcal{A}} Q^{\oplus}_{\mathrm{UCB}}(\boldsymbol{s}, \boldsymbol{a})$ ▷ *choose the leader*
    return $\arg\max_{\boldsymbol{a} \in \mathcal{A}} \left(\log N(\boldsymbol{s}) - F_N(\boldsymbol{a}, \boldsymbol{a}^*_N))\right)$ ▷ *explore suboptimal arm*
  end

Pareto Optimal Sampling for Lipschitz Bandits with Variances

POSLB does not employ estimated variances in action selection. The same approach while deriving UCB-V from KL-UCB can be followed to integrate estimated variances. Equating KL-UCB and UCB-V and solving for the variance parameter leads to

$$\sigma^2_N(\boldsymbol{a}) = \sigma^2 = \frac{N(\boldsymbol{s}, \boldsymbol{a})}{2 \log N(\boldsymbol{s})} \left(\sqrt{\frac{2\hat{\sigma}^2_N(\boldsymbol{a}) \log N(\boldsymbol{s})}{N(\boldsymbol{s}, \boldsymbol{a})}} + \frac{3c \log N(\boldsymbol{s})}{N(\boldsymbol{s}, \boldsymbol{a})} \right)^2 .$$

This can be integrated into to the POSLB algorithm and the resulting bandit is called the POSLB-V algorithm [THL21]. In contrast to UCB, UCB-V, and POSLB, the POSLB-V does not have any regret guarantees and serves as a naïve way to integrate estimated variances.

6.3.3 Partially Observable Monte Carlo Planning

The MCTS algorithm can only solve problems with fully observable states. To account for partial observability, the search tree can be branched in both actions and observations, i.e. history. The resulting structure consists of belief nodes instead of state nodes, and therefore is called a *belief tree*. Because a belief node branches in actions and observations, it has a two-layered inner structure: *action node* and *observation node*. The inner structure consists of a value at the top of the node and Q-values for every action. The Q-values are connected to the next *child* belief nodes with observations.

The Partially Observable Monte Carlo Planning (POMCP) algorithm constructs a search tree of actions and observations [SV10]. It samples state particles from the current belief and estimates the value of belief nodes in the tree with Monte Carlo simulation. Silver et al. show that this approach converges to the optimal value function [SV10].

Simulating states instead of history brings the utmost advantage of breaking the curse of dimensionality and of history. In this way, POMCP can operate with a generative model $\mathcal{G}$ that serves as the transition model T, the observation model Z, and the reward model R, with which it samples successor states, observations, and rewards. Hence, it does not require explicit probability distributions for T and Z. This allows the POMCP to solve large problems, i.e. 10^{56} states and 4 actions and 1024 observations, efficiently [SV10].

The operation of POMCP follows the four steps of MCTS and is listed in Algorithm 6.5. The algorithm starts with initiating an empty history $\boldsymbol{h}$ and iteratively calls the `simulate` function after sampling a state $\boldsymbol{s}$ from the current belief b. In `simulate` function the particle propagates through the belief tree. Every belief node stores its value $V(\boldsymbol{h})$ and its visit count $N(\boldsymbol{h})$ as well as the Q-values of its actions $\boldsymbol{a}$ together with the count of how often individual actions are selected $N(\boldsymbol{h}, \boldsymbol{a})$. If a value node is not visited yet, it is initialized by assigning N_{init} and V_{init} on Line 27 and Line 28, respectively. Visit counts and Q-values for every available action are initialized likewise. Subsequently, the value of the node is estimated by calling the `rollout` function before the simulation terminates. If the value node has already been visited before, the algorithm selects an action $\boldsymbol{a}$ by utilizing a multi-armed bandit. This is denoted with the `bandit` function in Line 34. In the next line, the algorithm employs a black box generative model $\mathcal{G}(\boldsymbol{s}, \boldsymbol{a})$ to sample the next state $\boldsymbol{s}'$, the observation

$\boldsymbol{o}$, and the reward r. This step corresponds to sampling $\boldsymbol{s}' \sim T(\boldsymbol{s}, \boldsymbol{a})$ and calculating $r \leftarrow R(\boldsymbol{s}, \boldsymbol{a})$, and then sampling $\boldsymbol{o} \sim Z(\boldsymbol{s}')$. If the new state $\boldsymbol{s}'$ is the terminal state, i.e. the goal state or a collision state, the current reward is assigned as Q-value estimate $\hat{Q}$ and the simulation terminates. Otherwise, the new action and observation are added to history and `simulate` is called recursively. After simulation has terminated, the backup step of MCTS is performed. The values and Q-values of all nodes starting from leaf nodes and continuing to the root of the tree are updated. This backpropagation is done by adding the discounted value of the child node to the immediate reward, as shown in Line 41. The resulting Q-value estimate $\hat{Q}$ is then averaged with the prior Q-values to calculate the current value Q inside the function `updateQ`. This function uses incremental statistics to incorporate the difference of a current Q-value estimate from the average, Δ. Updating with incremental statistics allows for memory-friendly averaging. Finally, the function `rollout` can be seen as a counterpart of `simulation` utilizing a lightweight and optimistic action selection strategy to serve as value heuristics. In contrast to `simulation`, it does not seed any new nodes. The `rollout` terminates after a tree depth of $d_{\max}$ is reached. Once the `solve` procedure ends, the agent picks the action with the highest Q-value at the root, receives a real observation and updates its belief with an unweighted particle filter. The new belief serves as the new root of the tree and the rest of the tree is pruned.

In backpropagation, as the number of visits of an action node increases, the accuracy of the Q-value estimate increases as well. Therefore, more recent value estimates should be considered with increasing weight, compared with earlier simulations. The learning rate $\alpha(n) \in [0, 1]$ which has the generalized form $\alpha(n) = 1/n^{\omega}$ allows to control this. The original POMCP chooses the learning rate exponent ω as 1, and therefore weights all estimates equally. A study shows that the optimal learning rate is achieved for $\omega \approx 0.77$ [EM03]. This can be integrated into the POMCP algorithm by revising Line 22 as $Q(\boldsymbol{h}, \boldsymbol{a}) \leftarrow Q(\boldsymbol{h}, \boldsymbol{a}) + \alpha(n)\Delta$.

Some bandits proposed in the Section 6.3.2 employ estimated variances. Maintenance of estimated variance estimates can be performed in an incremental fashion as well [Fin09]. For this the `updateQ` function must be extended by an additional line $\sigma^2(\boldsymbol{h}, \boldsymbol{a}) \leftarrow \big(1 - \alpha(n)\big)\big(\sigma^2(\boldsymbol{h}, \boldsymbol{a}) + \alpha(n)\Delta^2\big)$, where α is the aforementioned learning rate.

Algorithm 6.5: POMCP

```
1  function solve(b):
2      h ← ∅
3      repeat
4          s ~ b
5          simulate(h, s, d_max)
6      until timeout
7      a* ← arg max_{a∈A} Q(h, a)
8      return a*

9  function rollout(h, s, d):
10     a ← π_rollout(h);
11     (s', o, r) ~ G(s, a)
12     if s' is terminal or d = 0 then
13         return r
14     else
15         h' ← h ∪ {a, o}
16         return r + γ·
               rollout(h', s', d − 1)

17 function updateQ(h, a, Q̂):
18     N(h) ← N(h) + 1
19     N(h, a) ← N(h, a) + 1
20     n ← N(h, a)
21     Δ ← Q̂ − Q(h, a)
22     Q(h, a) ← Q(h, a) + Δ/n

23 function updateV(h):
24     V(h) ← max Q(h, a)

25 function simulate(h, s, d):
26     if h ∉ Tree then
27         N(h) ← N_init
28         V(h) ← V_init
29         for a ∈ A do
30             N(h, a) ← N_init
31             Q(h, a) ← Q_init
32         Q̂ ← rollout(s, d)
33     else
34         a ← bandit(h, A)
35         (s', o, r) ~ G(s, a)
36         if s' is terminal then
37             Q̂ ← r
38         else
39             h' ← h ∪ {a, o}
40             simulate(h', s', d − 1)
41             Q̂ ← r + γ· childV(h, a)
42     updateQ(h, a, Q̂)
43     updateV(h)
```

6.3.4 POMDP Algorithms with Continuous Spaces

The POMCP algorithm can solve problems with discrete actions, observations, and states. In robotic applications, the agent's actions can reasonably be discretized, whereas discrete states or observations pose stronger restrictions, as these are continuous by nature. Since the probability of sampling the same real number from a continuous distribution is zero, POMCP cannot operate with continuous states and observations. Therefore, their discretization is indispensable.

A naïve approach to tackle this problem is to use clustering. A more efficient approach is to use progressive widening (PW) [Cou+11]. PW limits the number

of child nodes of a node in the N^{th} visit to kN^{α}, where $k > 0$ and $\alpha \in (0, 1)$ are hyperparameters. This approach has originally been applied to the action spaces and was found to be highly effective. The pseudocode of the algorithm with an incorporated bandit is presented in Algorithm 6.6. $C(\cdot)$ denotes the children of a node, and the hyperparameters of PW are augmented with the subscript $\boldsymbol{a}$ to indicate that they are the parameters of action variables.

Algorithm 6.6: Action Progressive Widening (PW)

```
1 function actionProgWiden(h):
2     if |C(h)| ≤ k_a N(h)^{α_a} then
3         a ← nextAction(h)                    ▷ generate new action
4         C(h) ← C(h) ∪ {a}                    ▷ expand action nodes
5     return bandit(h, C(h))                   ▷ pick action from the available set
```

Sunberg et al. proposed to employ PW for solving POMDPs on continuous spaces and presented two algorithms [SK18]. In one of their algorithms, they employ action PW and observation PW on POMCP and call the resulting algorithm POMCPOW. POMCPOW samples single particles and uses weighted particle collections to approximate the belief at every node. In every `simulate` call, it adds a new particle weighted with its observation probability to the node. In recursive `simulate` calls, the sampled new state particles from the parent belief enrich the particle approximation, resulting in a more refined belief as the number of weighted particles increases. However, since only a single new particle is added to the node, the change in the belief between successive calls is typically not significant. The second algorithm samples m weighted state particles in the tree instead of single particles. In this way it simulates whole beliefs in the tree and performs weighted particle filter updates together with simulated observations to approximate the belief at successor nodes. The algorithm is called Particle Filter Tree Double PW (PFT-DPW).

The algorithms are fundamentally different from each other, as the POMCPOW simulates *state trajectories* and the PFT-DPW simulates approximated *belief trajectories*. Simulating belief trajectories allows defining rewards that are a function of the belief. Furthermore, whereas POMCPOW simulates a particle until a new observation causes a new child node to be added into the tree, PFT-DPW adds a new node for every simulated particle set. Nodes preceding the new node are not enriched with new particle set, and they reuse the previous

particles to perform the particle filter update. As a new node is added for every m particle, the tree of PFT-DPW is roughly m-times shallower than POMCPOW, causing a shorter-sighted planning for fixed computation time.

POMCPOW and PFT-DPW require a function to query the likelihood of given observations. The requirement on such an explicit model restricts the use of generative models, which counts as one of the novel features of POMCP. However, in robotic applications it is possible to derive an approximate model that works reasonably well. Therefore, these algorithms remain effective in practice [SK18].

6.3.5 Active Information Gathering with Reward Shaping

Many problems, such as motion planning, benefit from executing actions to disambiguate the current situation, as making optimal decisions heavily depends on the information available, see Section 2.2.3. POMDPs, by implicitly gathering information, arise as an effective approach for these problems. However, especially in large problems or in problems with sparse rewards, a sampling-based POMDP solver may require to explore many suboptimal actions until finding the promising ones. One way to accelerate this is to explicitly guide the planner to large future rewards. This can be done by augmenting the original reward with a term that reflects the expected information gain.

Augmenting the reward function with an additional term can result in a different policy than intended. Policy invariant *reward shaping* can be realized by Potential-Based Reward Shaping [NHR99; Eck+16]. This method *shapes* the reward by augmenting it with the potential difference in successive beliefs to guide the agent for long term rewards. A priori information about the problem at hand can be encoded by a proper choice of the potential function.

Reward shaping is often criticized in the machine learning community due to its modification based on domain-specific knowledge. However, collecting information and thus reducing uncertainty is domain independent, and is often a part of the optimal policy. It is straightforward to quantify information by entropy, as introduced in Section 3.3. The negative of entropy fulfills the requirements of a potential in reward shaping and is therefore a general heuristic for information gathering tasks [Eck+16].

The information gain over successive time steps can be quantified by discounted information gain $\mathcal{I}_\gamma = H(\boldsymbol{b}) - \gamma H(\boldsymbol{b}')$. Hence, the shaped reward function over successive beliefs can be given as

$$\rho(\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{b}') = \sum_{\boldsymbol{s}} R(\boldsymbol{s}, \boldsymbol{a})\boldsymbol{b}(\boldsymbol{s}) + w_{\mathcal{I}}\mathcal{I}_\gamma(\boldsymbol{b}, \boldsymbol{b}') \tag{6.26}$$

where $w_{\mathcal{I}}$ serves as a weighting factor to trade off mean state-based rewards against information gathering. A problem with such a reward formulation corresponds to a ρPOMDP [Ara+10].

6.3.6 Information Particle Filter Tree

ρPOMDPs are characterized by their belief-based rewards. Since the reward calculation depends on belief, these problems cannot be solved straightforwardly. Available algorithms require discrete spaces with piecewise linear and convex reward functions and run offline [Ara+10]. Fischer and Taş adapt the PFT-DPW algorithm and develop a novel online solver called Information Particle Filter Tree (IPFT), that can efficiently deal with arbitrary belief-based rewards [FT20].

Like PFT, IPFT approximates the belief by a weighted particle set $\boldsymbol{b} \approx \hat{\boldsymbol{b}}_m = \{\boldsymbol{s}_i, w_i\}_{i=1}^m$ of fixed sized m and simulates belief trajectories in the search tree. Apart from this similarity, it differs from PFT in several ways. First, it integrates information measures in the reward calculation and hence guides the search to more informative beliefs. While entropy evaluation can be done easily at discrete points $\boldsymbol{s}_i$, a continuous belief estimation is required to operate on continuous spaces. Kernel density estimation (KDE), as presented in Section 3.2.2, arises as a solution for this. Given KDE-based belief approximation $\hat{\boldsymbol{b}}_{\text{KDE}}$, differential entropy can be estimated as

$$H(\boldsymbol{b}) \approx \hat{H}(\hat{\boldsymbol{b}}_{\text{KDE}}) = -\sum_{i=1}^{m} w_i \log \hat{\boldsymbol{b}}_{\text{KDE}}(\boldsymbol{s}_i). \tag{6.27}$$

It should be underlined that in higher dimensions a KDE-based entropy estimation may require many particles and the estimation can become computationally expensive, as its complexity is $O(m^2 d)$, where d is the dimension of the continuous state space [FT20].

A second difference of IPFT to PFT is in its operation. IPFT operates more like POMCPOW than PFT-DPW. Whereas PFT-DPW stores sampled particles in the nodes and reuses them when the node is revisited, IPFT samples new particles in every visit. This ensures a valid Monte Carlo estimation and leads to better representation of the belief. The belief update requires sampling an observation from the existing observation child, as reusing a single sampled observation repeatedly would result in an invalid approximation of the belief. Compared with POMCPOW, IPFT suffers from a shallow structure resulting from sampling particle sets instead of single particles, alike PFT. The tree depth must cautiously be observed in applications.

The operation of the IPFT is given in Algorithm 6.7. The outer structure is omitted as it is identical to that of POMCP, which is presented in Algorithm 6.5. The only change in the outer structure is a single line in the `solve` procedure: it samples a small particle set of m particles from the bigger particle set representing the current belief $\hat{\boldsymbol{b}}_m \sim \boldsymbol{b}$ and passes this to the `simulation` function. In simulation, it first checks if the maximum depth is reached, and – if not – proceeds with action selection according to `actionProgWiden`, as already shown in Algorithm 6.6. Afterwards, it samples a new state $\boldsymbol{s}$ from the particle approximation of the belief $\boldsymbol{b}$. By using the transition model it samples the successor state $\boldsymbol{s}' \sim T(\cdot \mid \boldsymbol{s}, \boldsymbol{a})$ for the chosen action and then samples the corresponding observation $\boldsymbol{o} \sim Z(\cdot \mid \boldsymbol{s}', \boldsymbol{a})$. A proper belief update, and hence information gathering, can only be integrated into the solver by means of repeatedly sampled new observation, as indicated in the preceding paragraph. The algorithm proceeds with calculating the posterior belief using a particle filter update in Line 7. Subsequently, it calculates the belief-dependent reward and updates the history with the new action and the sampled observation. Next, it checks the observation widening criteria in Line 10, similar to POMCPOW. It either adds the sampled observation as a child node to the tree and increments the observation counter M in case of discretized observations before calling the `rollout` function, or it picks a child observation from the previously sampled observations with the probability to the counter M to traverse the tree. In case of a continuous observation space, observations are picked with uniform distribution. The search continues in the next tree depth by calling the function `simulate`. Once the simulation terminates, the node statistics are updated with the backpropagated new information. The operation of IPFT is illustrated in Figure 6.2.

Algorithm 6.7: Information Particle Filter Tree

function `simulate`$(\boldsymbol{h}, \boldsymbol{b}, d)$**:**
  if $d = 0$ **then**
    $Q = 0$
  else
    $\boldsymbol{a} \leftarrow$ `actionProgWiden`$(\boldsymbol{h})$
    $\boldsymbol{o} \leftarrow$ sample $\boldsymbol{s}$ from $\boldsymbol{b}$, generate $\boldsymbol{o}$ from $(\boldsymbol{s}, \boldsymbol{a})$
    $\boldsymbol{b}' \leftarrow \mathcal{G}_{\mathrm{PF}(m)}(\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{o})$ ▷ *belief update*
    $r \leftarrow \rho(\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{b}')$
    $\boldsymbol{h}' \leftarrow \boldsymbol{h} \cup \{\boldsymbol{a}, \boldsymbol{o}\}$
    if $|C(\boldsymbol{h}, \boldsymbol{a})| \leq k_{\boldsymbol{o}} N(\boldsymbol{h}, \boldsymbol{a})^{\alpha_{\boldsymbol{o}}}$ **then**
      $C(\boldsymbol{h}, \boldsymbol{a}) \leftarrow C(\boldsymbol{h}, \boldsymbol{a}) \cup \{\boldsymbol{o}\}$ ▷ *observ. widening*
      $M(\boldsymbol{h}, \boldsymbol{a}, \boldsymbol{o}) \leftarrow M(\boldsymbol{h}, \boldsymbol{a}, \boldsymbol{o}) + 1$
      $\hat{Q} \leftarrow r + \gamma \cdot$ `rollout`$(\boldsymbol{h}', \boldsymbol{b}', d-1)$
    else
      $\boldsymbol{o}^\star \leftarrow \boldsymbol{o}^\star \in C(\boldsymbol{h}, \boldsymbol{a})$ w.p. $\frac{M(\boldsymbol{h}, \boldsymbol{a}, \boldsymbol{o}^\star)}{\sum_{\boldsymbol{o}^\dagger} M(\boldsymbol{h}, \boldsymbol{a}, \boldsymbol{o}^\dagger)}$ ▷ *pick child observ.*
      $\hat{Q} \leftarrow r + \gamma \cdot$ `simulate`$(\boldsymbol{h}', \boldsymbol{b}', d-1)$
  `updateQ`$(\boldsymbol{h}, \boldsymbol{a}, \hat{Q})$
  `updateV`$(\boldsymbol{h})$

IPFT can work with generative models to sample states and observations, and therefore, does not require specifying transition and observation models for sampling. However, for importance calculation in weighted particle filtering it requires an explicit model of the observation likelihood. For n runs with particle set of size m, it has a computational complexity of $O(ndm)$ where d is the depth of the tree.

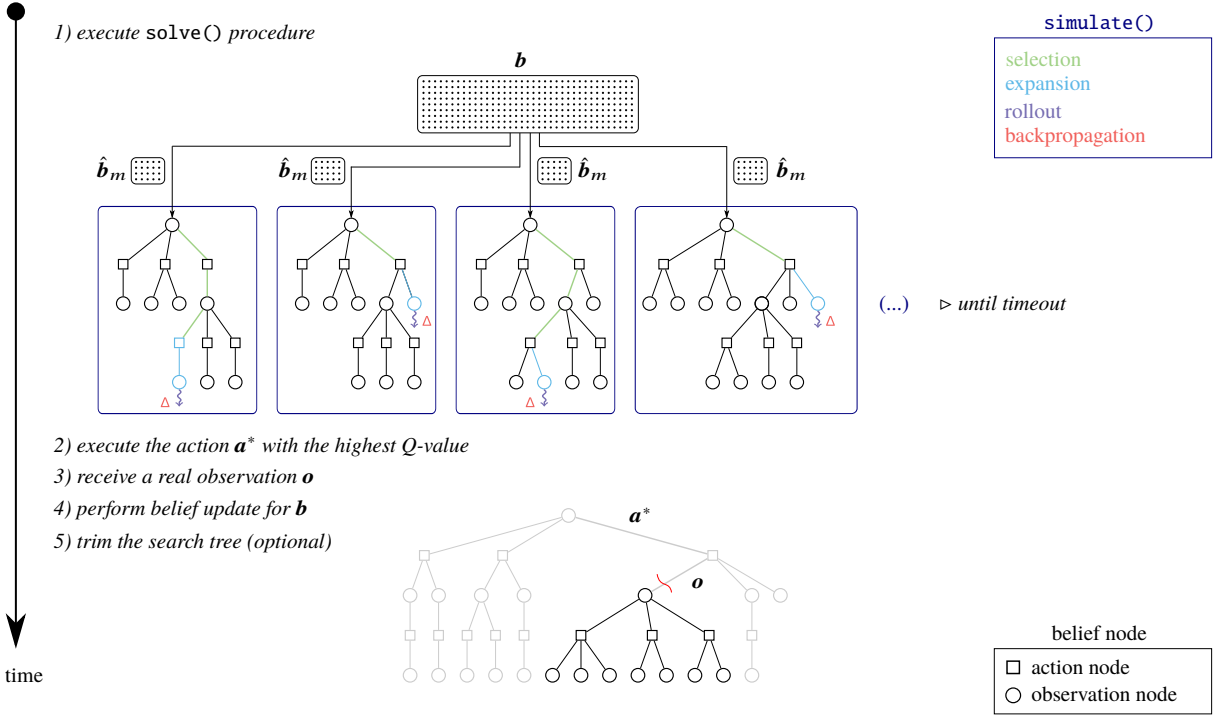


Figure 6.2: Operation steps of IPFT. In the `solve()` procedure the algorithm iteratively samples a weighted particle set $\hat{b}_m$ and calls the `simulate()` procedure until timeout. Subsequently, it executes the steps numbered from 2) to 5), where the latter is optional and by trimming the search tree from the red wavy line, it prevents building a search tree from scratch.

6.4 Motion Planning with the POMDP Framework

Preceding sections introduced models required by a POMDP, and presented online solution algorithms for solving POMDPs. This section presents how motion planning can be framed as a POMDP. The elements of the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{O}, T, Z, R, \gamma)$, which characterizes the POMDP, as well as the initialization of belief are defined below.

6.4.1 State, Action and Observation Spaces

The environment in which an autonomous vehicle operates is presented in Section 2.1. As a simplifying assumption, agents $\mathcal{V}$ in the environment are restricted to the members of classes that traverse the road network, i.e. vehicles. Therefore, route intentions of the vehicles can be matched with their available routes in the road network.

State Space

Matching route intentions with the road network allows defining position of the vehicles along the lane-referenced Frenet-coordinate frame $\mathcal{F}$, as done by [Hub+18]. However, for traversing medium to long distances, this is not a feasible approach. POMDPs perform belief updates which require the desired value of its variables to be preserved. As a vehicle drives, the lane sections – or lanelets – to which the vehicle is matched, change, as presented in Figure 2.1. For this reason, if coordinates are given in the Frenet frame, either the current longitudinal position of a vehicle must comprise all the driven lanelets or the reference point of all samples must be realigned. This is undoubtedly an infeasible approach. A more convenient way is to use the Cartesian coordinate system.

The state of the ego vehicle is defined by its Cartesian coordinates and speed

$$\boldsymbol{s}_{\vartheta_0} = [x_0, y_0, v_0]^\mathsf{T}. \tag{6.28}$$

Whereas the route and maneuver intentions of the ego vehicle are known and do not change over time, they are unknown for other vehicles. These variables are together introduced as the internal state χ in Section 2.1.1. The modeling in this work reduces available route intentions to the first set of options ahead. It does not consider the effect of different maneuver intentions for a given route. However, these can be integrated into the state representation by coupling them with acceleration and the route intention, as done in the situation prediction work in [Sch21]. Alternatively, they can be indirectly resembled with a desired travel speed in sampling [Bha+21]. Thus, the state of the vehicle k is represented as

$$\boldsymbol{s}_{\vartheta_k} = [x_k, y_k, v_k, r_k]^\mathsf{T}. \tag{6.29}$$

The state of the POMDP $\boldsymbol{s} \in \mathbb{S}$ contains the states of all vehicles in the scene

$$\boldsymbol{s} = \left[\boldsymbol{s}_{\vartheta_0}, \boldsymbol{s}_{\vartheta_1}, \ldots, \boldsymbol{s}_{\vartheta_k}\right]^\mathsf{T}. \tag{6.30}$$

This combined state definition is chosen to cover the interactions between the ego vehicle and other vehicles.

The state representation models route intentions as a stationary variable. As a result, the available route options must remain unchanged for a belief update. However, routes are defined by lanelets and with the movement of the vehicle new lanelets can appear as entirely new route options. Matching them with the previous ones is done by using Gale–Shapley algorithm [GS62]. If any change is detected, all route samples r_k are resampled.

Action Space

The road network further enables a simplification that reduces the computational complexity. Instead of planning motion in a driving corridor, velocity along a predefined and optionally optimized path can be planned. This can be seen as a type of path-velocity decomposition (see Page 16), reducing the number of control inputs $\boldsymbol{a}$ to acceleration only, resulting in a one-dimensional problem.

IPFT is capable of selecting actions from a continuous domain. Nevertheless, the action space $\mathbb{A}$ can be discretized into n values by creating an equidistant spaced action set $|\mathcal{A}| = n$ in the comfortable acceleration range $a^-_{\text{cft}} \leq a \leq a^+_{\text{cft}}$.

Observation Space

Observations of states $\boldsymbol{o} \in \mathbb{O}$ are defined similarly to states themselves

$$\boldsymbol{o} = \left[\boldsymbol{o}_{\vartheta_0}, \boldsymbol{o}_{\vartheta_1}, \ldots, \boldsymbol{o}_{\vartheta_k}\right]^\mathsf{T}, \tag{6.31}$$

where $\boldsymbol{o}_{\vartheta_k}$ is the observation of the k^{th} vehicle. As the state of the ego vehicle $\boldsymbol{o}_{\vartheta_0}$ is fully observable, components of its observation matches the true state

$$\boldsymbol{o}_{\vartheta_0} = [x_0, y_0, v_0]^\mathsf{T}. \tag{6.32}$$

In contrast, the route intention of other vehicles cannot be observed. Instead, their yaw angles ψ can be perceived. Therefore, their observation is defined as

$$\boldsymbol{o}_{\vartheta_k} = [x_k, y_k, v_k, \psi_k]^\mathsf{T}. \tag{6.33}$$

6.4.2 Transition Model

Probabilistic state transitions of vehicles can be modeled efficiently along the route they are traversing. This is done by mapping the vehicle positions to the Frenet frame $\mathcal{F}$ by using the transformations introduced in Section 2.1.2

$$(s, d) = {}^{\mathcal{F}}_{C}M\,(x, y), \tag{6.34}$$

and subsequently modeling the state transition by the point mass model

$${}^{\mathcal{F}}T\,(s, v, a) = \begin{pmatrix} 1 & t_\mathrm{s} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} s \\ v \end{pmatrix} + \begin{pmatrix} {t_\mathrm{s}}^2/2 \\ t_\mathrm{s} \end{pmatrix} a. \tag{6.35}$$

The next state along the route is obtained as

$$(s', v') = {}^{\mathcal{F}}T\,(s, v, a)\,, \tag{6.36}$$

which can be transformed back to the Cartesian frame C by using the posterior lateral offset d' along the route

$$\boldsymbol{s}' = \left[{}^{C}_{\mathcal{F}}M\,(s', d'), v'\right]^\mathsf{T} = [x', y', v']^\mathsf{T}. \tag{6.37}$$

The posterior lateral offset d' can be modeled either coarsely as linearly decreasing from d to zero with time, or precisely with pure pursuit control. The Equations (6.34) to (6.36) are together referred as ${}^{\mathcal{F}}T_{\vartheta}$.

Transition Model of the Ego Vehicle

The deterministic transition presented above can be extended to a probabilistic one by first using the deterministic transition to Frenet coordinates

$$(s', d', v', a) = {}^{\mathcal{F}}T_{\vartheta}(x, y, v, a) \tag{6.38}$$

and sampling both the longitudinal position in Frenet coordinates and the speed from a Gaussian distribution, i.e. $\tilde{s}' \sim \mathcal{N}(s', \sigma_s)$ and $\tilde{v}' \sim \mathcal{N}(v', \sigma_v)$. The variance term σ_s is obtained by projecting the position uncertainty stored in the environment model onto the reference path by using Equation (3.4). Sampled longitudinal position value $\tilde{s}'$ is transformed to the Cartesian frame C by using d' and the next state of the ego vehicle is obtained

$$\boldsymbol{s}'_{\vartheta_0} = \left[{}^{C}_{\mathcal{F}}M(\tilde{s}', d'), v' \right]^{\mathsf{T}} = [\tilde{x}', \tilde{y}', \tilde{v}']^{\mathsf{T}}. \tag{6.39}$$

Transition Model of Other Vehicles

The action obtained from the policy serves as a direct input to the transition model of the ego vehicle. However, the actions of the ego vehicle indirectly affect the actions of other vehicles. Other vehicles generate an action from their own policy and perform their state transition with this action. Because transition models are called in MCTS with every particle, a complex policy that considers interactions among all vehicles is computationally not affordable. Therefore, interactions of other vehicles are limited to their reaction to the ego vehicle. A simple yet efficient way for such a behavior modeling is to utilize the IDM, introduced in Section 2.1.3.

The reaction of vehicle k can be modeled with

$$a_{\vartheta_k} = a_{\text{IDM}} + \mathcal{N}(0, \sigma_{\text{IDM}}). \tag{6.40}$$

Subsequently, the new state in the Frenet frame

$$(s', d', v', a_{\vartheta_k}) = {}^{\mathcal{F}}T_{\vartheta}(x, y, v, a_{\vartheta_k}) \tag{6.41}$$

can be transformed back to the Cartesian frame and augmented with the previously sampled route r

$$\boldsymbol{s}'_{\vartheta_k} = \left[{}^{C}_{\mathcal{F}}M\,(s', d'), v', r \right]^{\mathsf{T}} = [x', y', v', r]^{\mathsf{T}}. \tag{6.42}$$

6.4.3 Observation Model

The IPFT requires an observation model to sample observations from states $\boldsymbol{o} \sim Z(\cdot \mid \boldsymbol{s})$, and to evaluate the likelihood of observations given states $Z(\boldsymbol{o} \mid \boldsymbol{s})$. Even though the general definition of an observation model depends on the previous action, *cf*. Equation (6.17), it can be modeled independently of it, as the current state already incorporates the effect of that previous action.

Sampling Observations

Observations comprise the state of all vehicles in the scene model including the state of the ego vehicle. These observations can be sampled either from a black-box generative model or from an explicit model.

The observation of the ego vehicle can be modeled by sampling it from

$$\boldsymbol{o}_{\vartheta_0} \sim Z(\cdot \mid \boldsymbol{s}_{\vartheta_0}) = \mathcal{N}(\bar{\boldsymbol{o}}_{\vartheta_0}, \Sigma_{\vartheta_0}) \tag{6.43}$$

where $\bar{\boldsymbol{o}}_{\vartheta_0} \in \mathbb{R}^3$ is the mean value and $\Sigma_{\vartheta_0} \in \mathbb{R}^{3\times3}$ is the diagonal covariance matrix with elements σ_{x,ϑ_0}, σ_{y,ϑ_0}, and σ_{v,ϑ_0} of the trivariate Gaussian distribution. Its parameters are obtained from fused perception information.

Observation of an "other" vehicle k is modeled alike

$$\boldsymbol{o}_{\vartheta_k} \sim Z(\cdot \mid \boldsymbol{s}_{\vartheta_k}) = \mathcal{N}(\bar{\boldsymbol{o}}_{\vartheta_k}, \Sigma_{\vartheta_k}), \tag{6.44}$$

but it contains an additional term $\sigma_{\psi,\vartheta_k}$ for the yaw angle $\hat{\psi}$. Therefore, $\bar{\boldsymbol{o}}_{\vartheta_k} \in \mathbb{R}^4$ and $\Sigma_{\vartheta_k} \in \mathbb{R}^{4\times4}$.

The estimated yaw angle $\hat{\psi}$ of ϑ_k is calculated for sampled Cartesian positions of the observation

$$\hat{\psi}_{\vartheta_k} = \psi_{\text{interp}}(\boldsymbol{o}_{x,\vartheta_k}, \boldsymbol{o}_{y,\vartheta_k}). \tag{6.45}$$

The function ψ_{interp} is defined along the centerline of a chosen route and returns *interpolated* yaw angles along the centerline, see Figure 6.3. It calculates an interpolation factor by transforming a given Cartesian position to its Hesse-Normal form and comparing the longitudinal position of the nearest point on the line segment to the length of the line segment. This factor is used for linear interpolation with the normal vectors of subsequent line segments. The baseline of this approach is known as *Phong Shading* in computer vision [Pho75] which has later been applied to motion planning as *pseudodistances* [Zie17, p. 30]. This work extends it to yaw angles.

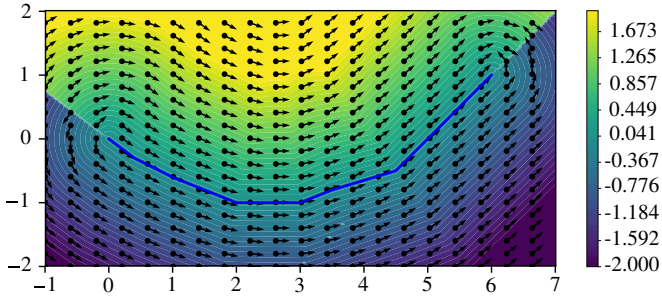


Figure 6.3: Interpolated distances and gradients calculated for the blue polyline.

Observation Likelihood

The likelihood of an observation, required by the weighted particle filter, can be defined as the product of observation likelihood of all vehicles in the scene

$$Z(\boldsymbol{o} \mid \boldsymbol{s}) = \prod_{k=0}^{k_S} P(\boldsymbol{o}_{\vartheta_k} \mid \boldsymbol{s}_{\vartheta_k}). \tag{6.46}$$

This implies independence assumption between the noise in the observation of any vehicle which is common practice [TBF05, p. 152].

Modeling vehicles driving along a certain route eases calculation of observation likelihood of vehicle positions. By transforming Cartesian positions to the Frenet frame, route referenced positions are evaluated. The longitudinal component $\boldsymbol{o}_s$ in Frenet coordinates serves as a simple model for observation likelihood of position, and the lateral component $\boldsymbol{o}_d$ as a convenient feature to inspect deviations from the given route.

The unknown route of other vehicles leads to different observation likelihood formulations for the ego vehicle and other vehicles. The observation likelihood of the ego vehicle is defined as the product of observation likelihoods of longitudinal position and speed, which are modeled with Gaussian distribution

$$P(\boldsymbol{o}_{\vartheta_0} \mid \boldsymbol{s}_{\vartheta_0}) = P(\boldsymbol{o}_{s,\vartheta_0} \mid \boldsymbol{s}_{s,\vartheta_0}) P(\boldsymbol{o}_{v,\vartheta_0} \mid \boldsymbol{s}_{v,\vartheta_0}) \tag{6.47}$$

$$= \mathcal{N}(\boldsymbol{o}_{s,\vartheta_0} \mid \boldsymbol{s}_{s,\vartheta_0}, \sigma^{\mathrm{PF}}_{s,\vartheta_0}) \mathcal{N}(\boldsymbol{o}_{v,\vartheta_0} \mid \boldsymbol{s}_{v,\vartheta_0}, \sigma^{\mathrm{PF}}_{v,\vartheta_0}). \tag{6.48}$$

Observation likelihood of other vehicles is defined with an additional likelihood term of $\boldsymbol{s}_{\vartheta_k}$ being on route r given the observation

$$P(\boldsymbol{o}_{\vartheta_k} \mid \boldsymbol{s}_{\vartheta_k}) = P(\boldsymbol{o}_{s,\vartheta_k} \mid \boldsymbol{s}_{s,\vartheta_k}) P(\boldsymbol{o}_{v,\vartheta_k} \mid \boldsymbol{s}_{v,\vartheta_k}) P(\boldsymbol{o}_{r,\vartheta_k} \mid \boldsymbol{s}_{r,\vartheta_k}). \tag{6.49}$$

Whereas the first two are calculated similarly to the ego vehicle, the last one is calculated from three features: 1) lateral offset to the centerline of the route, $\boldsymbol{o}_d$ 2) yaw angle difference between the observed yaw angle and the estimated yaw angle for that position, $\Delta\psi_{\vartheta_k} = \boldsymbol{o}_{\psi,\vartheta_k} - \hat{\psi}_{\vartheta_k}$ 3) speed on that route, $\boldsymbol{o}_v$

$$P(\boldsymbol{o}_{r,\vartheta_k} \mid \boldsymbol{s}_{r,\vartheta_k}) = P(\boldsymbol{o}_{d,\vartheta_k} \mid \boldsymbol{s}_{r,\vartheta_k}) P(\Delta\psi_{\vartheta_k} \mid \boldsymbol{s}_{r,\vartheta_k}) P(\boldsymbol{o}_{v,\vartheta_k} \mid \boldsymbol{s}_{r,\vartheta_k}) \tag{6.50}$$

$$\propto \mathcal{N}(\boldsymbol{o}_{d,\vartheta_k} \mid 0, \sigma^{\mathrm{PF}}_{d}) \mathcal{N}(\Delta\psi_{\vartheta_k} \mid 0, \sigma^{\mathrm{PF}}_{\Delta\psi}) \mathcal{N}(\boldsymbol{o}_{v,\vartheta_k} \mid v_r, \sigma^{\mathrm{PF}}_{v_r}). \tag{6.51}$$

The standard deviations used for likelihood calculation ($\sigma^{\mathrm{PF}}_{(\cdot)}$) are chosen slightly larger than the typical values observed in the scene model. Therefore, they are larger than the values used for sampling observations. Increasing these standard deviations mitigate potential particle deprivation problems.

It should be underlined that the likelihood equations above are not normalized. However, as particle weights are normalized in the subsequent steps during particle filtering, this would be superfluous.

6.4.4 Reward Model

The reward model has goal-driven and information-gathering reward terms

$$\rho(\boldsymbol{b}, \boldsymbol{a}, \boldsymbol{b}') = R(\boldsymbol{s}, \boldsymbol{a}) + w_{\mathcal{I}} \mathcal{I}_{\gamma}(\boldsymbol{b}, \boldsymbol{b}'). \tag{6.52}$$

All rewards are modeled as negative.

Goal-Driven Rewards

The goal-driven rewards correspond to the driving goals presented in Chapter 4. They account for deviations from the desired speed, acceleration in longitudinal as well as lateral direction, and collisions

$$R(\boldsymbol{s}, \boldsymbol{a}) = R_v(\boldsymbol{s}) + R_{a,\mathrm{lon}}(\boldsymbol{a}) + R_{a,\mathrm{lat}}(\boldsymbol{a}) + R_{\mathrm{coll}}(\boldsymbol{s}, \boldsymbol{a}).$$

The first term is defined similar to the driving objective in Equation (4.16) and the next two are defined similar to the first two terms of Equation (4.17). The last term R_{coll} takes collisions of the ego vehicle into account

$$R_{\mathrm{coll}}(\boldsymbol{s}, \boldsymbol{a}) = \begin{cases} 0 & \text{no collision} \\ w_{\mathrm{coll}} & \text{ego vehicle collides.} \end{cases} \tag{6.53}$$

The weighting factor for collision w_{coll} is chosen by magnitudes larger than other terms. The collision checks during state transitions are done by using the criterion from [Eri04, p. 223].

Information-Gathering Rewards

Every variable in the state representation of other vehicles might have an effect on the reward obtained upon selection of an action. Gathering information on them is therefore beneficial for the solver. However, the information-based reward calculation becomes computationally expensive with increasing dimension. An algorithm should therefore inspect only the most decisive variables.

Maneuvers intentions comprise the most important source of uncertainty, as discussed in Section 2.3. POMDP modeling in this work considers only route

intentions, as mentioned before. Therefore, the information reward is defined as the scaled information gain in the route intention probability of the vehicles in the current scene

$$\mathcal{I}_\gamma(\boldsymbol{b}, \boldsymbol{b}') = \sum_{k=1}^{k_S} \left(H(\boldsymbol{b}_{r,\vartheta_k}) - H(\boldsymbol{b}'_{r,\vartheta_k}) \right). \tag{6.54}$$

As route intentions are discrete variables, no kernel density estimation is required. This is an advantage over choosing variables for information-based rewards, such as positions, which can take continuous values.

6.4.5 Rollout Policy and Belief Initialization

The POMDP solvers that depend on MCTS execute rollouts, whenever a new node is generated, see Section 6.3.1. In motion planning, constant velocity motion serves as a lightweight yet accurate rollout policy π_{rollout}.

State particles are initialized similar to observation sampling introduced in Section 6.4.3. The only difference is, instead of sampling observation from posterior states, states are sampled from observations.

The state of the ego vehicle contains observable parameters and can therefore be sampled directly from observation, whereas the state of other vehicles contains the hidden variable route intention. This variable is sampled from a discrete probability distribution

$$P(r_{\vartheta_k} = r_{\vartheta_k,i} \mid \boldsymbol{o}_d, \Delta\psi) = \frac{P(\boldsymbol{o}_d \mid \boldsymbol{s}_{r,i}) P(\Delta\psi \mid \boldsymbol{s}_{r,i})}{\sum_{j=1}^{|\mathcal{R}|} P(\boldsymbol{o}_d \mid \boldsymbol{s}_{r,j}) P(\Delta\psi \mid \boldsymbol{s}_{r,j})}, \tag{6.55}$$

where $P(\boldsymbol{o}_d \mid \boldsymbol{s}_{r,i})$ and $P(\Delta\psi \mid \boldsymbol{s}_{r,i})$ are similar to Equation (6.51).

7 Evaluation

The planning under uncertainty problem for autonomous vehicles is modeled and solved with two distinct methods. The contributions for both methods are evaluated either by driving experiments or simulations, depending on how well they demonstrate their efficacy.

This chapter evaluates the proposed contributions of the individual methods, in the order presented in the preceding sections. It starts with the numerical optimization based planning and presents results for uncertainty-free case first. Then it proceeds with analyzing the effect of uncertainties. Subsequently, it delves into the planning with the POMDP framework. It evaluates the framework in intersection crossing scenarios and shows that such scenarios benefit from active information gathering. Afterwards, it demonstrates the effect of employing model-based bandit algorithms in the chosen class of POMDP solvers. The chapter finally compares both planning approaches and discusses their strengths and weaknesses.

7.1 Planning with Numerical Optimization

The decision-theoretic MPC features comfort and safety in the presence of incomplete environment information. To demonstrate the advantage of the proposed methods, they are inspected independently.

The planning algorithm is implemented with two different optimization methods: penalty methods and interior-point methods. The planner that utilizes a penalty method is implemented by using the nonlinear least-squares optimization library Ceres [AM+15]. The library internally involves a forward mode automatic differentiation module called Jet and a sparse Cholesky factorization for solving the linearized problem. A planning problem of 200 parameters is solved under 50 ms by a 2.6GHz Intel Xeon E5-2640v3 processor launched in 2014.

Constrained optimization algorithms based on penalty methods do not guarantee feasibility on their solution, as argued in Section 5.3. This creates a big problem especially in cases where the feasible solution is confined in a relatively narrow space. Changing the penalty function with a logarithmic barrier is not straight forward and requires an elaborate step factor selection algorithm. For this purpose, the decision-theoretic MPCC algorithm is implemented by using interior-point solver library IPOPT and the derivatives are calculated with the CPPAD library. As introduced in Chapter 5, IPOPT is renowned as the most robust and powerful nonlinear solver available with a non-commercial license, and CPPAD is the automatic differentiation library that shows the best performance. Using the linear solver MUMPS [Ame+19], the decision-theoretic MPCC with all the constraints defined in Section 5.6.3 is solved under 200 ms by a 3.80GHz Intel Core i7-10700K processor launched in 2020.

Driving experiments are performed either on the autonomous vehicle Bertha [Taş+18], or on CoCar [Koh+13]. Simulation experiments are performed on the simulation environment P3IV [Taş21] by using the drone dataset InteractionDataset [Zha+19] for scenario definitions. Both experiments are run on ROS [Hel+16] and utilize Lanelet2 map format [Pog+18; PJ20].

7.1.1 Planning in Fully Observable Environments

The most fundamental requirement for a planner is the ability to plan comfortable and safe motion, without considering any uncertainties. The scenarios and logs of the cooperative autonomous driving contest the Grand Cooperative Driving Challenge (GCDC) 2016 [Eng+16] served as a perfect setting to demonstrate the capabilities of the motion planner quantitatively.

The challenge composed of two scenarios, cooperative intersection crossing and cooperative lane change, and defined rules and goals to make the competition safe and evaluable. The participant agents were cars and trucks only and the scoring was merely determined by the longitudinal performance. In this sense, by defining custom goals that included error-tolerant zones, the GCDC had a synthetic nature. Nevertheless, this international event provided a setting, where experienced teams in autonomous driving demonstrated their algorithms.

The use of automatic differentiation allowed changing the cost function and constraint definitions on-the-fly. On the system-side, ROS allowed changing

parameter values in runtime. These instruments enabled flexibility in adapting the motion planner for the needs of the competition.

The motion planner was tuned to achieve the highest score, and planned only for the longitudinal motion. In this way, the number of elements in the state vector, as well as the number of applied constraints were reduced, and the computational resources were canalized to reach higher replanning frequencies with harder convergence conditions. The customized planner used the penalty function implementation and had a runtime of 15 ms on average.

In the first scenario, namely the cooperative intersection, the goal is that all vehicles leave the intersection in the shortest time without violating the defined minimum safety distance and the maximum speed while yielding to the organizer vehicle ID 3. After the start signal, all vehicles must enter the intersection zone exactly after 20 s and traverse it ideally with 30 km/h.

Figure 7.1 shows the speed profiles and the travelled distance of an arbitrary heat in the intersection scenario. The speed profile of Bertha has a small overshoot. Firstly, this is due to the slightly penalized tolerance band that is covered by the range cost terms defined in Equation (4.13), and secondly, to the unmodeled dynamics of the vehicle. Despite the speed profile of the station ID 3 having small jitters, Bertha drives quite smoothly. All vehicles hold the safety distance for all times they are inside the intersection zone. The video record of this heat is presented in frames of 0.25 s intervals in Figure 7.2.

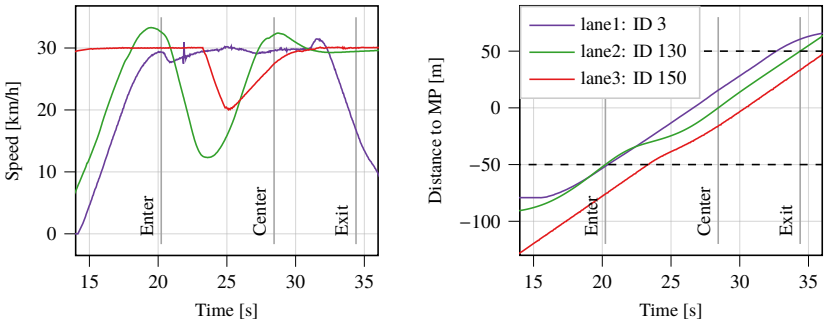


Figure 7.1: Logs from a cooperative intersection heat. The vertical lines indicate the times at which Bertha (ID 130) entered the intersection zone, was at its center, and left it.

Figure 7.2: Cooperative intersection crossing.[1] Frames are incremented every 0.25 s. Later frames, in which the vehicle ID 150 is visible, are omitted.

[1] Video available online at `https://youtu.be/41lCu6IvBB4`

7.1.2 Changes in Driving Goals and Constraints

In the intersection crossing scenario, the driving goals and intentions of the vehicles do not change. Therefore, the constraints applied on the planner are stationary. While driving, constraints and objectives change continuously depending on the environment and the maneuver intentions of other agents. An instantaneous constraint activation can result in poor performance for planning in receding horizon. A linear constraint activation scheme that activates the constraints gradually is proposed in Section 4.1.3 to address this issue.

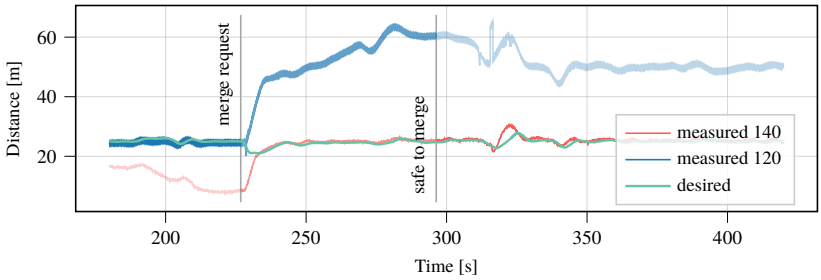
The cooperative lane change scenario in GCDC 2016 provides a very good benchmark setting to evaluate the efficacy of such a constraint activation scheme. The scenario resembles the case where the left lane of a two lane highway is blocked by a (virtual) construction zone. Two platoons driving on right and left lanes have to merge before the construction zone. The maneuver intentions are transmitted over V2V-communication and therefore do not entail any uncertainty. The scoring is determined by how smooth and fast two platoons merge without violating the safety distance.

The vehicles in this scenario start either on the left or the right lane. Two platoons first come side by side, then they pair, and perform lane changes. For a lane change, the merging vehicle signals “merge request” and thereupon, the vehicle on the right side opens a gap and signals “safe to merge”, once the required gap is created. Afterwards, the lane change takes place.

Figure 7.3 depicts the relative distance of Bertha based on V2V-communication logs in an arbitrary lane change scenario. Before the “merge request” at 226.7 s, Bertha follows the vehicle with the station ID 120, while driving slightly behind ID 130, the vehicle in the left lane. After the “merge request”, it gradually switches its objective to follow the vehicle with the ID 140, resulting in a gentle opening of the desired gap defined by the competition rules. The relative distances depicted in the figure display a smooth transition.

The results presented in the figures represent only a few heats of the challenge. In the GCDC 2016, Bertha scored the best in motion planning performance and the second-best in cumulative evaluation, which includes aspects like human-machine interface. This great result in planning performance lies, among others, in the flexibility gained with the use of automatic differentiation and flexible parameter tuning with ROS.

(a) A frame taken after the “safe to merge” signal. ID 140 is changing its lane.



(b) Relative distance of Bertha against other vehicles. All lines have the same width.

Figure 7.3: Logs from an arbitrary cooperative lane change heat. Bertha is the 4th car driving on the right lane. ID 120 is its leader and ID 140 is the car that is changing its lane.

7.1.3 Noncompliant Traffic Participants

The evaluation presented up to this point used transmitted positions and intentions of vehicles. In driving, maneuver intentions of participants are not observable and in some cases, exhibit noncompliant behaviors. Section 4.4 proposed an approach that yields proactive motion plans for such cases. The applicability of this approach can be demonstrated both in simulation and with driving experiments, whereas the latter requires consideration of aspects like time delays and has a higher credibility. Therefore, this work presents results obtained from the latter.

The driving experiments are conducted at a T-intersection, where the ego vehicle has the right-of-way and an oncoming vehicle must give way to the ego

vehicle. Figure 7.4 shows one of the driving experiments. The ego vehicle is the gray vehicle driving autonomously along a priority road. The oncoming black vehicle approaches the intersection at a speed that suggests it will not give way to the ego vehicle, but applies braking at the very last moment. The black vehicle is intentionally driven manually to replicate different driving styles and demonstrate the robustness of the algorithm. To exclude the effect of the limited sensor range and therewith occlusions, the black vehicle transmits its position and speed via V2V-communication.

The frames obtained from a video of a test run is shown in Figure 7.4. The planned motion of the ego vehicle is shown in Figure 7.5. The ego vehicle approaches the intersection at its desired speed. It recognizes that the oncoming vehicle will not be able to brake comfortably to give way to it, and therefore, it does not release its stop constraint defined in Equation (4.53). Frames 3 to 11 in Figure 7.5 depict the full stop motion. The active stop line constraint causes a slight reduction of speed as the vehicle approaches the intersection. Once it is clear that the oncoming vehicle will give way to the ego vehicle with a comfortable braking deceleration, the constraint is released. Subsequently, the ego vehicle continues to drive along the priority road at its desired speed.

Although results from only a single run are presented in this work, the reproducibility and robustness are shown in repeated runs performed at the RobustSENSE project final event, and are demonstrated to the experts of the field.

7.1.4 Imperfect Perception

The proactive safety conditions introduced in Chapter 4 allow safe intersection crossing under limited visibility. Although such problems are easier to handle when compared with the presence of noncompliant traffic participants, the evaluations demonstrate interesting properties.

Figure 7.6a shows a scenario, where the ego vehicle depicted in blue drives in urban areas. The vehicle has a limited sensor range and the buildings around hinder its visible field. It approaches an intersection, where it has to yield to any vehicle from the intersecting route. Because of the limited sensor range and occlusions caused by the buildings, it is unaware of the intersection ahead. The deviations of speed from the desired speed, as well as the acceleration and

Figure 7.4: Safe intersection crossing in the presence of noncompliant participants. The autonomous vehicle (gray) has the right way. The speed of the oncoming black vehicle suggests that it will not yield to gray vehicle. Frames are incremented every 0.25 s.

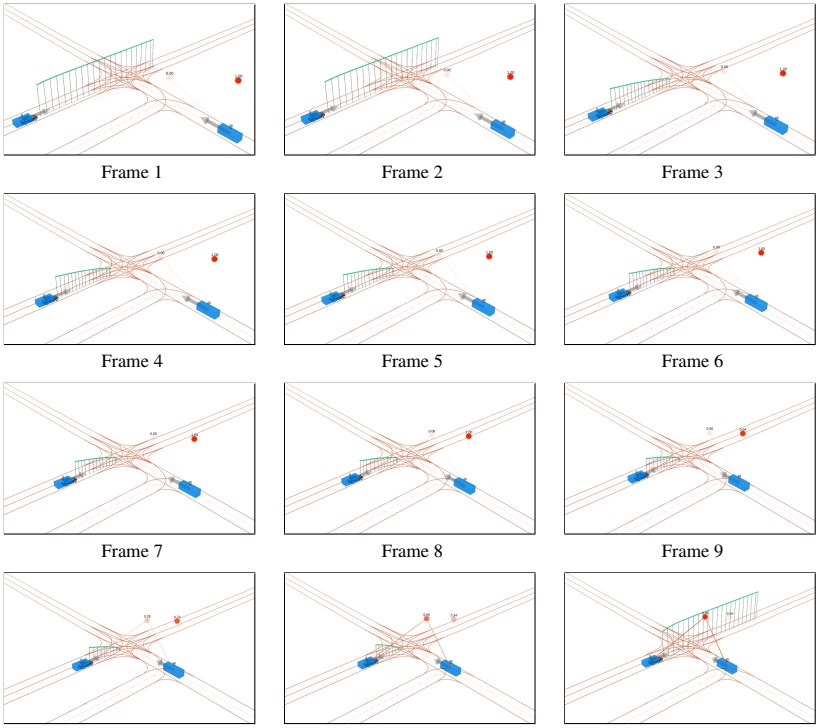
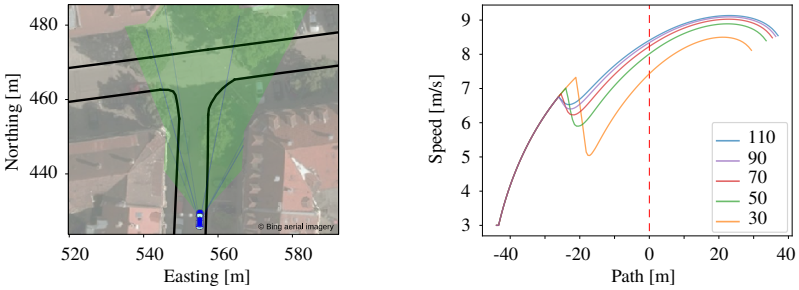


Figure 7.5: Planned motion of the intersection crossing scenario.[2] Frames are incremented every 0.25 s, but their numbers are not synchronized with the frame number of the camera images in the previous figure.

[2] Video available online at `https://youtu.be/lOpE5uldJiI`

jerk values are penalized with quadratic loss terms (see Equation (4.12)). Until recognizing the intersection zone, its speed is constrained to keep its standstill distance shorter than its visible distance along the driving corridor.



(a) Scenario.

(b) Speed for various sensor ranges.

Figure 7.6: Effect of limited visibility simulated on an uncontrolled intersection.

Figure 7.6b displays motion profiles for various sensor ranges on a path-speed diagram. The center of the intersection is set as zero in path coordinates. All profiles in the diagram have the same length as they are of the same duration. The profiles indicate that long sensor ranges enable the vehicle to recognize the intersection early and allow a comfortable reduction in speed. For shorter sensor ranges, the vehicle detects the intersecting route later and closer to the intersection, causing a harsh deceleration.

Once the vehicle has enough visible distance along the intersecting route, it starts accelerating. However, due to the limited visible field, it cannot reach its desired speed for all inspected sensor ranges. After reaching a peak speed at roughly 20 meter, the vehicle starts decelerating again, because of buildings and road curvature. This indicates that sensor ranges longer than 50 meter are not necessarily advantageous in urban scenarios.

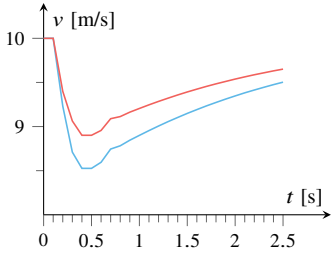
It must be underlined that the replanning time t_c plays an important role in reaching the desired speed: long replanning times impede reaching desired travel speeds when considering visible field.

7.1.5 Postponing Decisions While Ensuring Safety

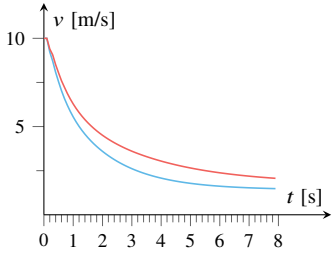
The decision-theoretic MPCC is evaluated in simulation, first for the phantom object scenario, described in Section 5.6.3, and then for the intersection crossing scenarios provided by the drone dataset.

In the phantom object scenario, the ego vehicle detects 15.0 m ahead a vehicle with 50 % existence probability, driving at 2.0 m/s. This scenario is evaluated for two distinct settings. In the first setting, the phantom detection disappears after 0.3 s, where its existence remains unclear in the second setting.

Figure 7.7 compares the effect of using decision-theoretic MPCC with a conventional MPCC framework for both settings. The speed profile displays the smooth yet safe reaction obtained by using decision-theoretic MPCC.



(a) Phantom detection identified. The reference contour and the initialization are reset after phantom detection is identified.



(b) Phantom detection persists. The decision-theoretic planner reduces its speed to the speed of the phantom object later.

Figure 7.7: Speed profile comparison of a conventional MPCC (—) with a decision-theoretic MPCC (—) in a phantom object detection scenario. Whereas the conventional MPCC reacts more directly to the phantom detection, the decision-theoretic one finds a balance between criticality and comfort.

Planning homotopy-class-free motion and postponing decisions is especially useful in intersection crossing scenarios. Figure 7.8 presents a planning instance at a roundabout scenario, where the ego vehicle considers maneuver probabilities of other vehicles, that are obtained from a particle filter based prediction module. The figure consists of three subplots depicting 1) the predicted and planned positions of the vehicles, 2) vehicle positions and uncertainties at the time of planning, and 3) the profile of the planned motion. The results show quite smooth motion profiles.

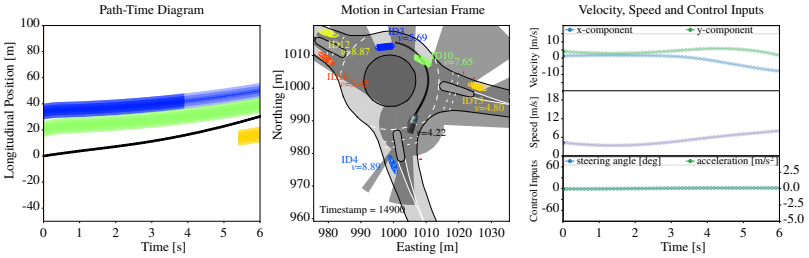


Figure 7.8: Smooth planning with decision-theoretic MPCC. The ego vehicle is depicted in black. Vehicles in the environment model $\mathcal{V}_{\mathcal{E}}$, i.e. inside the visible field, in gray, are represented with their position uncertainty and a semi-transparent rectangle centered at their mean position. The rectangles are opaque for vehicles in the scene model $\mathcal{V}_{\mathcal{S}}$.

7.2 Planning with the POMDP Framework

The evaluation of planning with the POMDP framework is divided into two parts. The first part, consisting of two sections, utilizes the IPFT solver and aims planning in intersection scenarios in real-time. It evaluates results for the state-based reward definition, and then for the belief-based reward definition with the purpose of active information gathering. To do this, an open-sourced implementation in the programming language C++ of the solver algorithm is tested with the P3IV simulation framework. The second part utilizes the POMCP algorithm implemented in Python and inspects the convergence of bandit algorithms in different settings.

7.2.1 Intersection Crossing in Real-World Scenarios

For autonomous driving, a planning horizon t_{h} of at least 5 s is required. In order to achieve real-time capability for this planning horizon, the POMDP model is configured for a time discretization t_{s} of 0.5 s. The computation time of the solver is set to 1 s, but the optimum action over solver time is monitored for evaluation. P3IV allows for asynchronous simulation, enabling simulation update rates higher than computation times. Therefore, the time discretization of the simulation environment is set to 0.1 s. The solver creates a new search tree at every planning instance and does not reuse the previously created one.

For evaluating the solution, two distinct plots are implemented. The first one is divided into three subplots depicting the current scenario and belief, see Figure 7.9. The left-hand side depicts the true position and speed of the vehicles. It additionally shows the corridor options of an oncoming vehicle. The subplot in the center depicts belief states and the centerlines of the driving corridors. Additionally, it displays some random particles from the search tree at various planning times, which correspond to different tree levels, on the map. These particles are either simulated observations of the other vehicle and are denoted with $+$, or motion states of the ego vehicle. The motion states are denoted either with $\times$ indicating they end in a collision, or with $\bullet$. The subplot on the right-hand side shows the route probabilities of the vehicles over time.

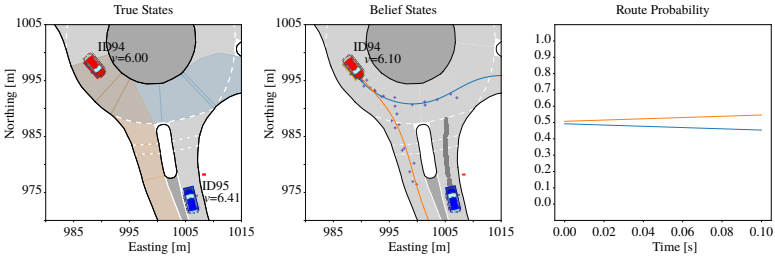


Figure 7.9: Baseline roundabout scenario obtained from InteractionDataset.

The second plot inspects the search tree and it is divided into three subplots as well, see Figure 7.10. The left-hand side shows the Q-values of different actions over solver time. The subplot in the center integrates chosen accelerations in the search tree and plots the resulting speed profiles over time. The line opacity of the profiles denotes the frequency of individual actions. Integrated actions that lead to collision are depicted in red, whereas collision-free actions are denoted in green. The dashed red line depicts the desired speed and the continuous blue line depicts the optimal action sequence. The right-hand side shows the Q-value estimates, which incorporate the cost associated with the collision probability, on a time-speed diagram. The values are obtained by averaging all actions in the search tree that yield the corresponding speed value. It must be underlined that this figure is bound to approximation errors, as the order of actions have an influence on the corresponding Q-values. Nevertheless, it provides an interpretable representation. Similar to the plot in the center, the continuous blue line depicts the optimal action sequence.

The evaluation is performed in a scenario, where there is a vehicle k driving inside the roundabout, to which the ego vehicle has to yield, if the vehicle does not take the exit. The scenario is shown in Figure 7.9. The motion of the vehicle k is read from the dataset. Therefore, its actions are independent of the actions of the ego vehicle. The parameters of the scenario are provided in Appendix A.4. The scenario is intentionally selected such that the route classification yields wrong results until the intersection. Even though the planner can successfully identify the route intention in most of the cases, this scenario serves as a stress test.

The Q-value profile and hence the optimal action sequence of the planner strongly depend on the parameters of the model. Figure 7.10 shows the results for $v_{\text{des, ego}} = 4.0\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$, immediately after the oncoming vehicle is detected. The planner chooses to yield to the oncoming vehicle. However, for $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 5.0\,\text{m/s}$, the selected action sequence is considerably different, as shown in Figure 7.11. The planner intrinsically prefers a maneuver of another homotopy class: the ego vehicle enters the roundabout before the other vehicle. This is a remarkable result, highlighting that planning with the POMDP formulation internally selects the optimal homotopy class, unlike conventional planners.

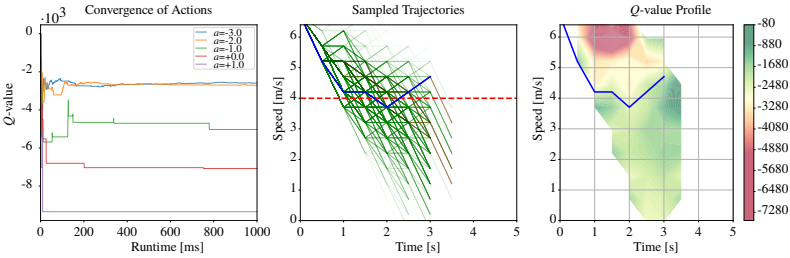


Figure 7.10: Search tree for $v_{\text{des, ego}} = 4.0\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$.

Figures 7.12 and 7.13 show the evolution of the scenario after 1.3 s. Since the motion of the vehicle k is independent of the actions of the ego vehicle, its position and route probabilities are exactly the same for both parameter configurations. In contrary, the position and the planned motion of the ego vehicle are significantly different. For parameter values where the desired speed of the ego vehicle is lower than the other vehicle, the ego vehicle continues

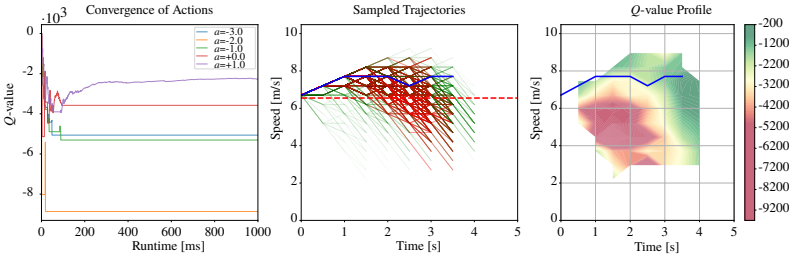
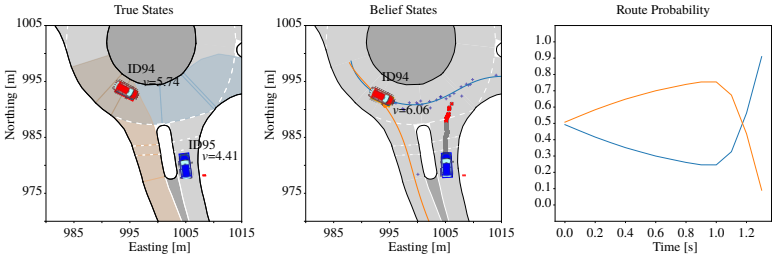
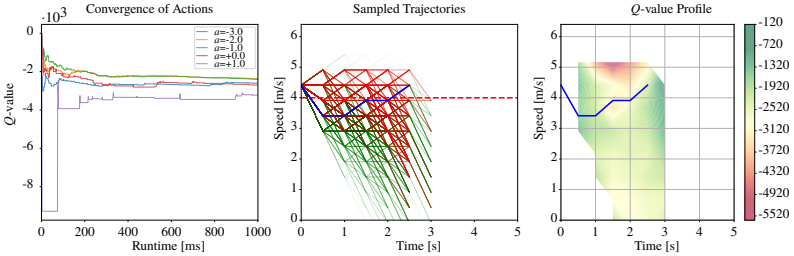


Figure 7.11: Search tree for $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 5.0\,\text{m/s}$.

to decelerate and starts accelerating once the other vehicle has passed (see Figure 7.12b). For parameter values of the other homotopy class, it continues to accelerate to avoid any collision (see Figure 7.13). Results after 1.1 s and 2.1 s are presented in Figures A.2 to A.4 in Appendix.

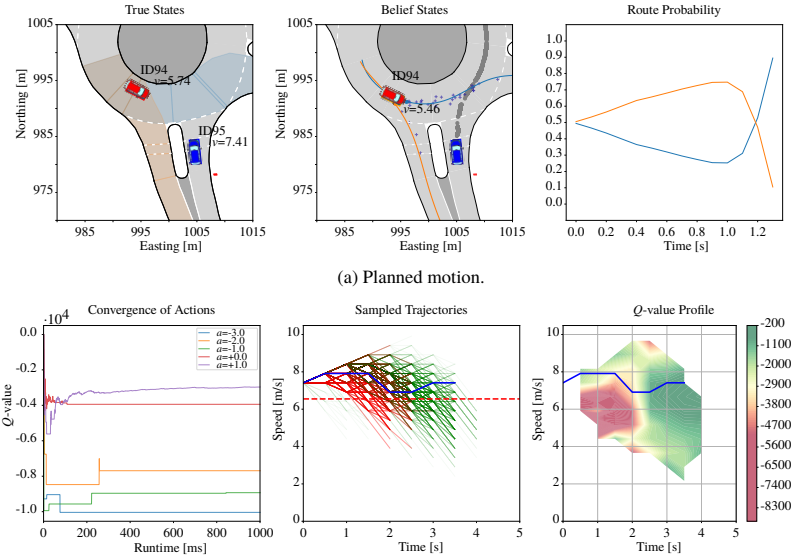


(a) Planned motion.



(b) Search tree.

Figure 7.12: Evolution of the scenario after 1.3 s, for $v_{\text{des, ego}} = 4.0\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$.



(a) Planned motion.

(b) Search tree.

Figure 7.13: Evolution of the scenario after 1.3 s, for $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 5.0\,\text{m/s}$.

The results indicate that the optimal action sequence depends strongly on the models and its parameters. Different desired speed configurations result in motion profiles of different homotopy classes. Their correct estimation is essential in obtaining comfortable motion. In the evaluations above, v_{des} of the vehicles are not conflicting: if one is set high, the other one is set low. If conflicting values are chosen, the vehicle executes a harsh braking, as expected. Results for such parameter values are provided in Figure A.5 in Appendix.

Besides desired speed, uncertainties play an important role on the chosen actions. Increasing uncertainties lead to broader areas of low Q-values, resulting in a defensive driving style.

Solver parameters have a strong influence on the solution. The hyperparameter analysis in the IPFT paper shows that the optimal particle set size for the benchmark problem is $m = 20$, whereas the difference after $m = 4$ is rather insignificant. Motion planning for autonomous driving is much more complicated

than the benchmark problem, and therefore choosing $m = 20$ for the selected computation time $t_c = 1.0$ enables a planning horizon $t_h = 3.0$ on average. This value is insufficient for planning. Therefore, the evaluations above are done for $m = 10$, which yields $t_h = 4.0$ on average.

The UCB exploration constant c is another decisive parameter value. Its value is half of the biggest weight in reward terms, as practiced commonly. Increasing it leads to excessive exploration: actions in the areas that lead to collision are exhaustively explored. On the contrary, choosing a small value results in a greedy policy approximation. Once it has found a collision free action sequence, it barely explores other actions, resulting in a narrow but deep search tree.

Another important solver parameter is the observation progressive widening (PW) factor k_o. Whereas actions are selected from a discrete set, observations are continuous, and therefore, PW might be beneficial for building deep search trees. Experiments show that observation PW is strongly correlated with exploration constant c. Enabling it causes the search tree to try actions for a broader range of observations, resulting in shallow search trees. Although the effect can partly be compensated with low c values, it usually yields inferior results [Bec21]. For this reason, observation branching is disabled.

All tree search plots show that the immediate optimal action usually does not change after a solver time of $t_c = 200$ ms in most cases, as the search tree does not encounter anything unexpected as it deepens.

7.2.2 Intersection Crossing with Active Information Gathering

The preceding intersection crossing evaluations are done for state-based rewards only. Section 6.4.4 presented belief-based information gathering rewards on the route intention of another vehicle. By incorporating such rewards, the IPFT algorithm can execute actions that reveal route intentions. It does this by optimizing for reactions through the behavior model of other vehicle defined in the POMDP formulation. Therefore, in order to evaluate if the reaction of the other vehicle matches the expectations, it requires a clear feedback on the actions it executes.

The POMDP model between the action of the ego vehicle and the reaction of the other vehicle k on the same route has a loosely-coupled feedback, which is particularly difficult for the solver to operate on. The execution of the action chosen by the policy of the ego vehicle is a noise-bound process. The reaction of the vehicle k is modeled by using IDM, where modeling errors are covered by an additive Gaussian noise σ_{IDM}, see Section 6.4.2. Subsequently, the ego vehicle's observation of this new state is subject to noise as well. These accumulating uncertainties result in a loosely-coupled reaction model. In contrast to standardized benchmark problems, where the agent has a direct feedback, this loosely-coupled problem formulation is much harder to tackle.

The effect of active information gathering is evaluated at a T-intersection scenario depicted in Figure 7.14. The oncoming vehicle, depicted in red, has two route options: driving straight and yielding to the ego vehicle, or turning right. It is programmed so that it drives at a constant speed and takes the exit on the right. The task of the ego vehicle is to execute actions that maximize its knowledge about the route intention of the vehicle. The scenario and solver parameters are presented in Appendix A.4.

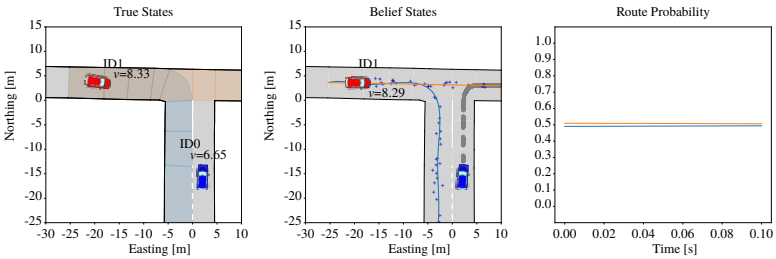


Figure 7.14: The T-intersection scenario used in active information gathering evaluations.

The efficacy of the approach is first evaluated with a modified behavior model and then with the original POMDP formulation presented in Section 6.4.2. In order to make the route intention identification independent of geometric features, and to demonstrate the applicability of the approach for maneuver intention identification, the probability calculation defined in Equation (6.51) is reduced to its third component first, i.e. speed only.

Action Reinforced Behavior Model

Establishing a more direct feedback is beneficial for performing information gathering. Instead of reducing modeling uncertainties in a naïve way, which can cause particle deprivation and overfitting, the transition model of the other vehicle can be revised. If they are travelling on the same route, the reaction of the other vehicle can be reinforced by the actions of the ego vehicle such that

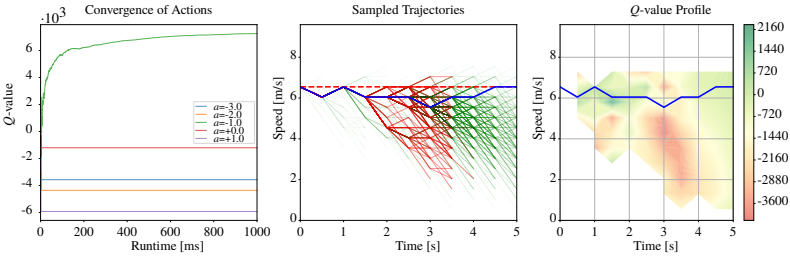
$$a_{\mathrm{IDM}} = \begin{cases} 1.5a_{\mathrm{IDM}} & (a_{\mathrm{IDM}} < 0 \wedge a_{\mathrm{ego}} < 0) \ \vee \ (a_{\mathrm{IDM}} > 0 \wedge a_{\mathrm{ego}} \geq 0), \\ 0.5a_{\mathrm{IDM}} & (a_{\mathrm{IDM}} < 0 \wedge a_{\mathrm{ego}} \geq 0) \ \vee \ (a_{\mathrm{IDM}} > 0 \wedge a_{\mathrm{ego}} < 0). \end{cases} \tag{7.1}$$

The resulting a_{IDM} is passed to Equation (6.40). Despite uncertainty, the reaction of the other vehicle can be easily revealed. It must be highlighted that this model is designed for initial evaluations, as accelerations are unobservable in reality.

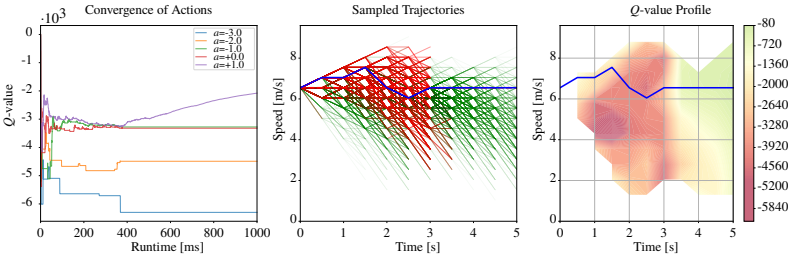
Figure 7.15 compares the effect of active information gathering on the search tree of the reinforced formulation presented above. All solver parameters are left unchanged. In active information gathering, the solver executes a targeted and consecutive deceleration and acceleration. Strikingly, this action is found quite rewarding, such that it tries other actions only once. The solver builds a narrow-structured tree during the initial stages. In comparison, the setting without information gathering builds a broader tree and the ego vehicle prefers to accelerate in order to enter the intersection before the oncoming vehicle.

The Q-value contours for both cases are substantially different due to information rewards. However, both cases have the same optimal policy by the virtue of potential-based reward shaping. It must be underlined that the Q-value visualization in the active information gathering setting has more approximation errors, as different action sequences can create entirely different information rewards.

Figure 7.16 depicts the speed and route probabilities over time. Performing active information gathering is especially beneficial in the very first moments after detection. With increasing time, the POMDP can inherently infer the route intention of the other vehicle from speed likelihoods. The speed profiles do not indicate a significant difference for this simulation run.



(a) With information rewards.



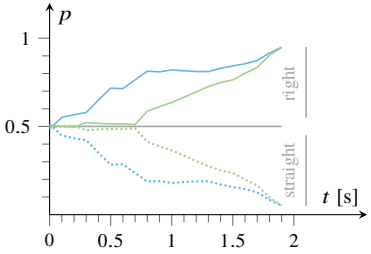
(b) Without information rewards.

Figure 7.15: POMDP search tree with and without active information gathering. The Q-values of the search tree with active information gathering include belief-based information rewards. Therefore, its Q-value profile is substantially different.

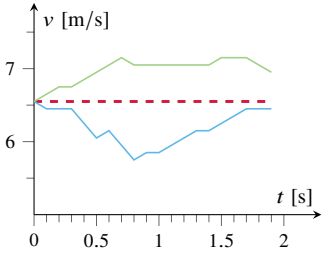
Non-Reinforced Behavior Model

The results for action reinforced behavior model suggest that planning with the original POMDP formulation might benefit from active information gathering. Since the framework approximates the optimal policy for the POMDP with sampling, the results are analyzed for 100 runs for a comprehensive evaluation.

Figure 7.17 compares the effect of using and not using active information gathering in the T-intersection scenario. Although the difference between them is not as obvious as in the reinforced case, the benefits are still visible. The route probabilities increase slightly later, as the observations are less-directly linked with actions of the ego vehicle. The search tree analysis, unlike the reinforced case, is indistinguishable from the state-based reward case. Therefore, the search tree plots are not presented. The speed plot reveals a remarkable result:



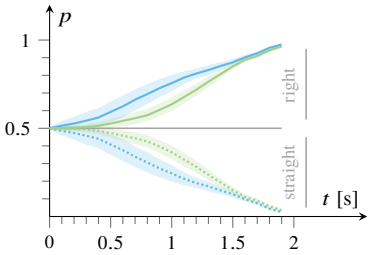
(a) Route probabilities.



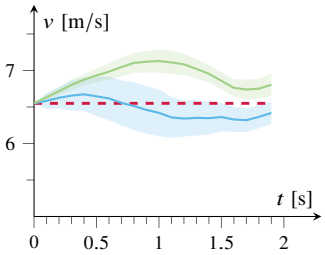
(b) Speed profile.

Figure 7.16: Evolution of the predicted route probability and the speed profile over time for the reinforced motion model. The blue line ( ) represents the results for the case with active information gathering, and the green line ( ) is for the case without information rewards. The dotted line ( ) in the route probability shows the probability of driving straight. The red dashed line ( ) in the speed profile shows the desired travel speed.

the state-based reward formulation accelerates to speeds higher than the desired speed. Because of the asymmetric reward function on speed, speeds lower than the desired speed are penalized with a tolerant loss, whereas higher speeds are penalized with a quadratic one. This suggests that the action sequence in state-based formulation has a lower utility than for active information gathering.



(a) Route probabilities.



(b) Speed profile.

Figure 7.17: Evolution of the predicted route probability and the speed profile over time for the original POMDP model obtained from 100 simulation runs.

Planning with the POMDP framework strongly depends on the chosen models and their parameters. The evaluations presented above are performed for $\sigma_{\text{IDM}} = 0.1$. The benefits of active information gathering disappear when this value is increased to $\sigma_{\text{IDM}} = 1.5$. Nevertheless, the evaluation shows that, under a reasonable uncertainty, information gathering improves the utility in problems where the optimal policy is substantially affected by the present uncertainties.

A remarkable result from the active information gathering experiments is that the solver is able to reach a planning horizon $t_{\text{h}} = 5.0$ constantly. Compared with the previous experiments, the increase in the horizon is due to disabling geometry-based features in route prediction. This indicates that matching Cartesian coordinates to the Frenet frame with the interpolated distance and yaw angle scheme presented in Section 6.4.3 accounts for a computational overhead of 20 %.

The results of using geometric features for route prediction with active information gathering is presented in Figure A.6 in Appendix. The figures show that geometric features dominate the route probability results as the vehicle comes closer to the intersection.

Active information gathering relies on estimating probability distributions with particles, where larger particle sets enable better estimates. Experiments have shown that larger particle sets result in steeper increase in route probabilities. However, the planning horizon becomes shorter in return. Furthermore, with increasing particle set sizes, the vehicle executes repeated deceleration and acceleration sequences. This is an intuitive result, as achieving a higher confidence becomes harder with growing particle set sizes. Reducing the information gathering weight alleviates such repeated actions, yet the route probability increases quite late.

The results for particle set sizes $m = 4$ without geometric features are presented in Figure A.7 in Appendix. The results are comparable with $m = 10$. Furthermore, with $m = 4$, a planning horizon longer than $t_{\text{h}} = 5.0$ is obtained. However, in the presence of multiple other vehicles, this size is insufficient. Qualitative experiments show that a particle set size $m \simeq 10$ is a suitable choice.

7.2.3 Exploiting Lipschitz Continuity of Rewards

Section 6.3.2 derived two multi-armed bandits (MABs) for Lipschitz continuous rewards by proposing to utilize them in sampling-based POMDPs to improve their converge. However, it is yet unclear if the motion planning problem formulation in Section 6.4 has a Lipschitz continuous reward profile.

Lipschitz continuous transition model and reward model yield a Lipschitz continuous reward [AML18]. The reward definition for motion planning includes a binary collision term, rendering the reward function discontinuous. In contrast, state transitions and observations are subject to uncertainties, which in turn can alleviate such discontinuities.

This work empirically evaluates the continuity of reward functions. In the following, the continuity of the Q-value function is analyzed first, then the effect of utilizing a Lipschitz MAB is benchmarked. To demonstrate the generality of the results, the POMCP algorithm defined in Section 6.3.3 is utilized as the solver.

(a) Straight driving. (b) Traversing curves. (c) Imminent collision. (d) Intersection crossing.

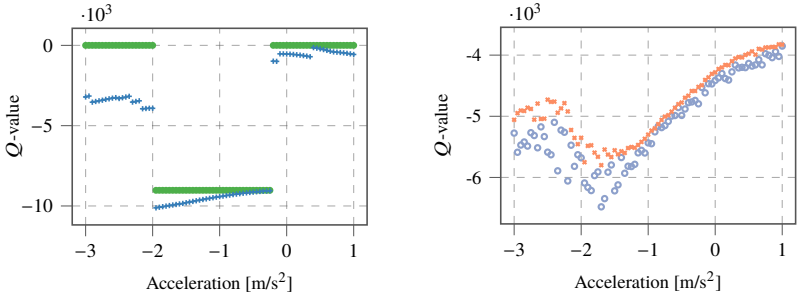
Figure 7.18: Traffic scenes used for development and in the evaluations.

The evaluations are performed on the traffic scenes depicted in Figure 7.18. The first two scenes do not behold any discontinuities and are used for initial testing, see Figures 7.18a and 7.18b. The next scene requires interaction with the oncoming participants, see Figure 7.18c. The last one further requires identification whether the other participant is on the collision path, see Figure 7.18d. The latter two scenes are used to define three scenarios: a collision scenario $\mathtt{S_{Coll}}$ that is subject to imminent collision, and two intersection scenarios $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$, which pose low and high probabilities of collision, respectively. The parameter of the scenarios and the values affecting the results are provided in Appendix A.4.

Continuity of Q-Value Function

The continuity evaluation is done by inspecting the Q-value in the root node of the belief tree for equidistant actions and running simulations for every action value. For an accurate result, the action space is finely discretized with $\Delta a = 0.05\,\mathrm{m/s^2}$, and for every action value, a simulation run with 10^6 particles is executed. The succeeding action values after the first action value are selected from 5 equidistant actions with the UCB bandit.

To highlight the discontinuity in the reward function, $\mathtt{S_{I\text{-}Hi}}$ is analyzed first. Figure 7.19a illustrates the Q-value function without considering the noise in transition and observation models. The rewards of the collision term are depicted with ● markers. The overlying + markers correspond to the Q-values of all the rewards. It can clearly be seen that $a \in (-2, -0.2]$ results in collision. Figure 7.19b displays the reward function for the same scenario, but this time while considering the aforementioned noise. The results are presented for both *coarse* and *fine* approximations, denoted with ○ and × respectively. The coarse uses 10^4 particles, whereas the fine one is obtained by sampling 10^7 particles and doubled resolution of the observation discretization. Additionally, the bandit of the fine approximation selects from 17 actions. Even though such large number of samples and high resolutions are not realistic, this fine approximation represents the edge case. Compared with coarse approximation, the reduction in the variance is evident. But in return, the discontinuity in the Q-values starts to appear, eventually requiring a larger Lipschitz constant $\mathscr{L}$.

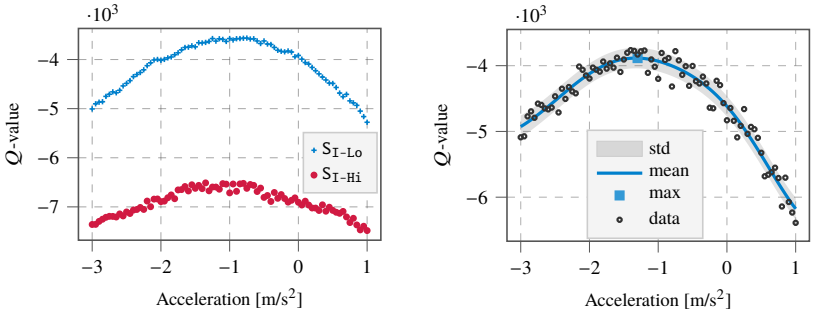


(a) Collision term only & All terms without uncertainty.

(b) 5 actions, coarse, 10^4 particles & 17 actions, fine, 10^7 particles.

Figure 7.19: Q-value profiles of $\mathtt{S_{Coll}}$.

The Q-value profiles of the intersection scenarios are presented in Figure 7.20. The first subfigure presents the results for $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$ for the same number of particles. They have the same shape, whereas $\mathtt{S_{I\text{-}Hi}}$ has lower Q-values and higher variance compared to $\mathtt{S_{I\text{-}Lo}}$. This is an expected result due to the higher collision probability of $\mathtt{S_{I\text{-}Hi}}$. The higher variance in case of an imminent collision results from the optimistic nature of the UCB bandit. The second subfigure presents Q-value $\mathtt{S_{I\text{-}Lo}}$ and Gaussian process regression on the individual Q-values for a lower quantity of samples. The variance is higher and Q-values are less than for 10^4 particles.



(a) Intersection scenarios both sampled with 10^4 particles.

(b) Gaussian process regression on $\mathtt{S_{I\text{-}Lo}}$ sampled with 10^2 particles.

Figure 7.20: Q-value profiles of $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$.

Efficiency of Lipschitz Multi-Armed Bandits

The uncertainties in the transition and observation models alleviate the discontinuities in the Q-value function and allow for Lipschitz continuity assumption. In the following, the MABs presented in Section 6.3.2 are benchmarked.

A common metric to benchmark convergence is to inspect the *mean absolute error* (MAE) between the best value obtained and the true value. For sampling-based POMDPs, the MAE between the current highest Q-value (Q_n^*) and the optimal Q-value (Q^*) over the number of sampling episodes n can be taken as the performance measure. Q^* can be calculated by sampling 10^7 particles while employing the UCB bandit. Since the resulting Q-value profile has stochastic nature, the noise can be eliminated by performing Gaussian process

regression, as depicted in Figure 7.20b. The maximum of the underlying function corresponds to Q^*.

To account for random events, the MAE is calculated for m simulation runs

$$\text{MAE}_n = \frac{1}{m} \sum_{i=1}^{m} |Q^*_{i,n} - Q^*|$$

where $m = 100$ and $n = 2 \cdot 10^4$ episodes.

Lipschitz bandits require the Lipschitz constant $\mathscr{L}$ to be set beforehand. An over- or underestimation of its value results in suboptimal performance, nevertheless its optimal value is scenario-specific and is not known in advance. The optimal Lipschitz constants for the scenarios are given in Appendix A.4. For benchmarks $\mathscr{L} = 2000$ is selected empirically.

The convergence results for different numbers of actions in $\mathtt{S_{Coll}}$ is given in Figure 7.21. For low numbers of available actions, the difference in convergence is insignificant. With increasing numbers of actions, POSLB and POSLB-V show increasingly superior performance. In general, the bandit variants that consider empirical variances perform comparably. The convergence of POSLB is subject to jitter and has a greater variance compared to other bandits. This is an expected result of the Lipschitz continuity assumption. Nevertheless, for increasing numbers of actions, Lipschitz bandits demonstrate better convergence than their UCB counterparts.

Figure 7.22 presents the results for 9 and 33 actions in $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$ scenarios. Even though the convergence behavior of the bandits is quite similar to those of $\mathtt{S_{Coll}}$ in general, the POSLB bandit displays striking results in Figure 7.22d. It has a very large variance and the difference between its MAE with UCB is smaller than other cases. This is caused by misleading rollouts that point to a different region of the action space to be optimal. The Lipschitz assumption causes the bandit to select actions in that region. This effect diminishes with increasing tree depths. Since POSLB-V incorporates estimated variance in action selection and selects an action with higher variance more often, it compensates the drawbacks of misleading rollouts. A further remarkable result is observed, when the convergence results of $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$ are compared against each other. At first, the Lipschitz assumption seems to be more effective for $\mathtt{S_{I\text{-}Hi}}$. However, this is due to the selected value of the Lipschitz constant.

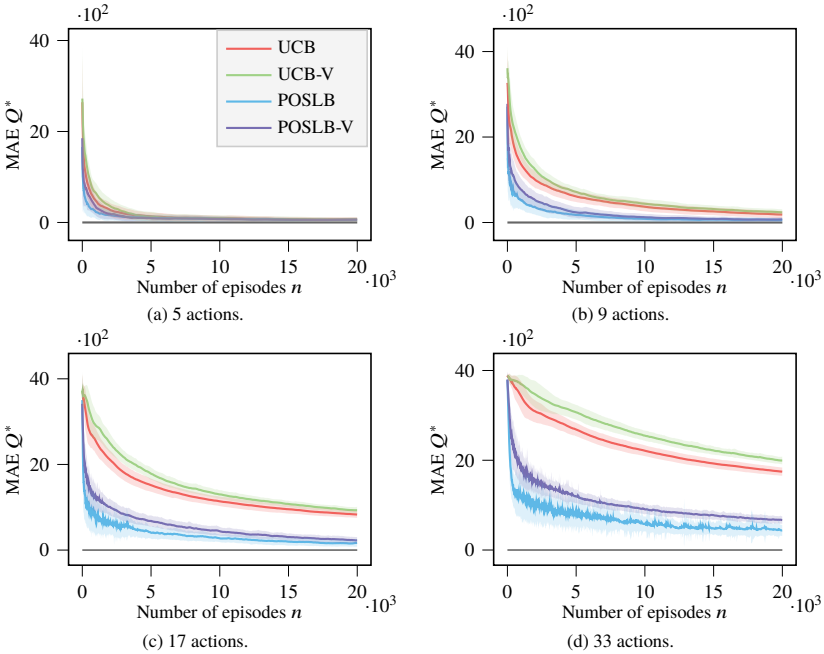
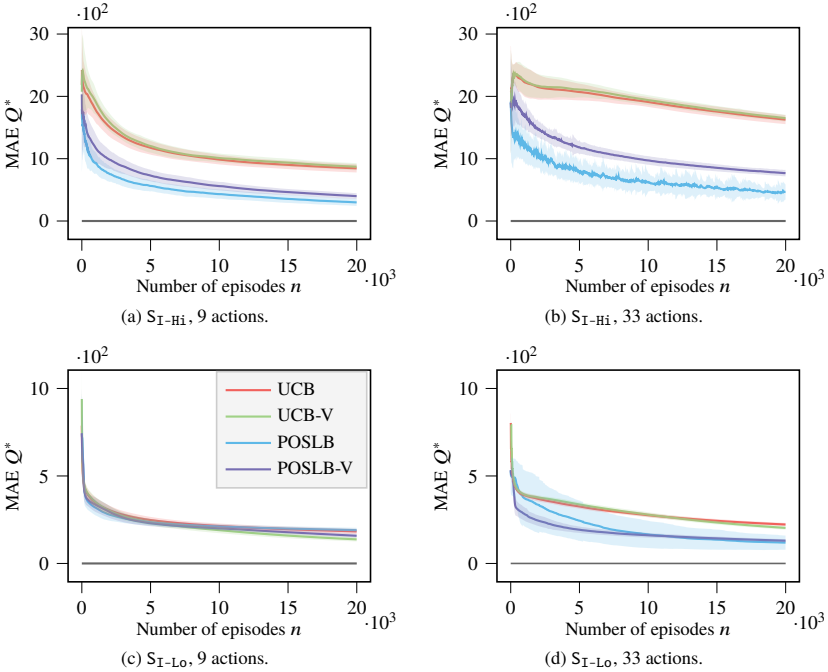


Figure 7.21: Mean absolute error in $\mathtt{S_{Coll}}$ for different numbers of actions.

$\mathscr{L} = 2000$ underestimates the Lipschitz continuity of $\mathtt{S_{I\text{-}Hi}}$ more than it does for $\mathtt{S_{I\text{-}Lo}}$, see Table A.6 in Appendix.

The results for 5 actions are omitted, as the difference in the convergence properties of the bandits are insignificant. The results for 17 actions are between 9 and 33 action cases. They are provided in Figure A.8 in Appendix.

In a sampling-based POMDP, the quality of the solution strongly depends on the depth of the search tree. The Lipschitz bandits must enable the building of deeper trees to provide better convergence. Figure 7.23 compares the tree depths for UCB and POSLB bandits on $\mathtt{S_{Coll}}$ for 9 actions. The bar widths represent the number of nodes created at each depth level, whereas the color represents the number of particle visits. UCB has a greater branching factor, resulting in shallower trees compared with POSLB, as expected. The results for other scenarios and action combinations are alike.



(a) $S_{\texttt{I-Hi}}$, 9 actions. (b) $S_{\texttt{I-Hi}}$, 33 actions.

(c) $S_{\texttt{I-Lo}}$, 9 actions. (d) $S_{\texttt{I-Lo}}$, 33 actions.

Figure 7.22: Mean absolute error in $S_{\texttt{I-Hi}}$ and $S_{\texttt{I-Lo}}$ for different number of actions.

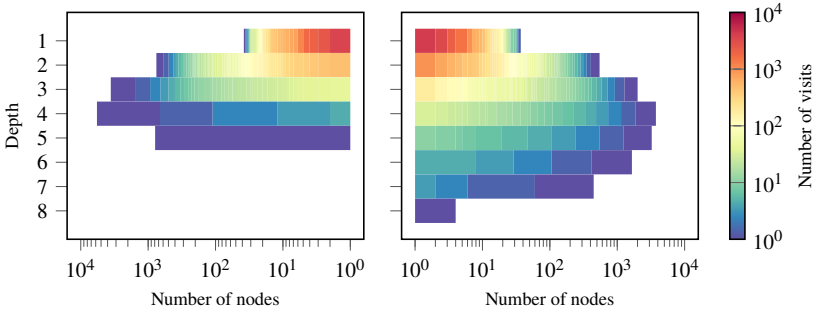


Figure 7.23: Tree depth of UCB (left) and POSLB (right) bandits in $S_{\texttt{I-Hi}}$ for the same number of episodes.

The evaluations are done for a constant number of episodes. The Lipschitz bandit algorithms are more complex when compared with UCB, resulting in an increased computational complexity. The experiments show that, even in the computationally most demanding case of 33 actions, the increase remains below 10% in the worst case and is 5% on average. These values are obtained as an average of all scenarios.

7.3 A Comparison of the Proposed Planning Frameworks and Discussions

Planning by utilizing numerical optimization and sampling-based POMDP frameworks operate fundamentally different from each other. Therefore, a direct, quantitative comparison is not reasonable. Nevertheless, analysis and remarks that reveal design choices as well as highlight their advantages and disadvantages are presented next. These serve as a basis for comparing both planning frameworks.

Planning with Numerical Optimization

The main difficulty while developing planning algorithms that utilize numerical optimization lies in problem modeling and initialization. Problems that are solved with Newton's method must be twice continuously differentiable, and initialized in the vicinity of the local minimum that is sought for. This is a stringent requirement, limiting its flexible use when compared with algorithms that employ nonparametric approaches.

Finding an appropriate initialization is not a trivial problem in motion planning. In every planning instance new objects may appear, invalidating the previous solution obtained from a receding horizon planning. There are two alternative ways to cope with this problem by modeling. The first option is to initialize the problem with full-braking motion. However, such a *cold* start makes planning in receding horizon computationally more expensive. Additionally, in case of the MPCC formulation, as the vehicles' contouring errors are defined with respect to an initial progress profile, it is inapplicable. The second option is to use a simple driving model, such as IDM. IDM satisfies the goal of achieving

naturalistic driving objectives and provides a feasible initialization to the solver. The experiments conducted show that for basic planning formulations, the planner does not find any optimization potential after several iterations, and terminates. Even though an initialization with IDM seems favorable, it considers reactive capabilities only in the longitudinal direction. Therefore, utilizing a solver that can operate with an infeasible initialization is highly advantageous.

In autonomous driving, the drivable area that is free of static obstacles is frequently represented with polygons. Such polygon boundaries serve as a linear constraint to optimization-based planning algorithms and for this reason, they do not pose considerable complexity to the planner algorithm. However, there may be other agents in the environment that are moving inside this area. Conventional MPC approaches treat these objects as obstacles with a known motion profile, and shrink the driving corridor according to their predicted motion. Although this results in good convergence and fast solver times, it prevents probabilistic processing of the environment by the motion planner, hindering its interaction capabilities.

These stringent requirements on the model allow for fast runtimes in return. Newton's method can find the solution in fewer iterations in such problems. However, this comes at the cost of computing the Hessian in every iteration, which can be expensive to calculate. Therefore, instead of calculating the exact value of the Hessian, it can be approximated with Quasi-Newton Methods. A qualitative comparison of both approaches on motion planning problems show a slightly better result in favor of the exact Hessian methods.

Nonlinear optimization algorithms utilize linear solvers internally. The nonlinear solver utilized in the decision-theoretic MPCC internally utilizes the MUMPS solver, as mentioned in Section 7.1. Several benchmarks suggest that other linear solvers such as MA27, MA57 [HSL13] show better performance for the problem size of the planning problem. However, these solvers did not perform considerably better for motion planning applications.

Further runtime evaluations for optimization-based planning have shown that typically only 10 %–15 % of the solution time is spent on calculating derivatives. Another analysis reveals that the difference in runtimes of using Runge-Kutta method and Euler's method for colocation are insignificant. In another benchmark, comparing the runtimes of kinematic vehicle models and linearized kinematic models has shown that the linearized model does not produce any

runtime improvement while using a nonlinear solver. Contrary to the expectations, such a linearization introduces the risk of worsening the convergence for cases where the optimal solution is away from the initialization, i.e. point of linearization. The use of a low power CPU, specifically the 1.80Ghz Intel Core i7-8550U from 2017, results in solver times being roughly doubled. Even though the current runtimes may be regarded as high, with hardware-focused library compilation and code optimization, faster runtimes can be achieved.

Planning with the POMDP Framework

Planning with the POMDP formulation has several advantages over MPC-based approaches. First, sampling-based solvers are anytime capable. Although a good belief initialization can improve the quality of the solution, it is not essential in guaranteeing convergence, unlike optimization-based approaches.

Sampling-based POMDPs are rich in their representational power and allow flexible modeling. They incorporate prediction in planning and can accurately predict the reaction of other agents against each other during planning. Even though the uncertainties are modeled such that the resulting belief space is Gaussian, they can be utilized in arbitrary representations. Sampling-based POMDPs can additionally leverage heuristics from deep neural networks, making them fast and accurate. They can be mixed and extended with other approaches and have therefore an enormous improvement potential.

Compared with optimization-based approaches, they are more generalizable and maintain their robustness in a variety of driving scenarios, as they do not require continuous differentiability and allow higher flexibility in modeling. Due to the nature of sampling, even in the case of implementation errors, they can operate with higher reliability and are less error-prone.

The advantages of utilizing the POMDP framework comes at the cost of high computational complexity. Even their sampling-based solvers have high runtimes. There are two alternative ways to improve their speed. The first option is to parallelize the MCTS. A recent work follows this option and demonstrates promising results in continuous domains [KHZ20]. The other option is to utilize MABs that exploit the continuity of the reward function, as done in this work.

For exploiting the continuity of the reward function, the POSLB bandit is utilized in this work. There are other MABs that assume Lipschitz continuous returns among the bandit arms [KSU08; Bub+08; MM10; BSY11; Sli19, p. 39]. However, these bandits choose a new arm from continuous action space in every round and never sample an action twice. Therefore, these bandits cannot be utilized for building a belief tree.

Learning rate and the backup operator play an important role in the speed of POMDP solvers. Although many solvers utilize a learning rate $\alpha = 1$, a rate depending on the current number of the sampling episode is chosen in this work, as indicated in Section 6.3.3. This choice results in faster convergence, as proposed by [EM03]. Another modification on POMCP is the backup operator. Although robust backup has theoretical guarantees, the maximum backup is more intuitive and shows better convergence properties. These results are also confirmed by another recent publication that employs POMDPs for planning [Bey+21].

The evaluations and remarks on the strengths and weaknesses of both proposed approaches are briefly summarized in Table 7.1.

Method	Speed	Flexibility	Extendability	Robustness	Error-Safe
Decision-theoretic MPCC	+++	+	-	++	-
Sampling-based POMDP	+	+++	+++	++	++

Table 7.1: Comparison of the two planning methods proposed.

8 Conclusion & Future Directions

Accumulating uncertainties due to imperfect perception and partial observability frequently result in defensive motion plans in autonomous driving. Planning safe, but not overcautious motion is a topic of active research, which has been addressed in many works. This work develops two novel motion planners that can compensate the deficiencies of perception and prediction, while remaining safe even in the worst-case evolution of a driving scenario. Furthermore, the second planner executes dedicated actions that maximize the vehicle's knowledge about its environment.

This work first analyzed the source and the implications of uncertainties in autonomous driving. In order to tackle partial observability, it delved into their mathematical modeling and the notion of information. Subsequently, it discussed the properties of an optimal motion for driving while interacting with other agents and reviewed approaches that attempt to ensure safety. While many works addressed different aspects of criticality assessment and safe planning, this work studied safety in receding horizon planning. Based on inevitable collision states, it developed a proactive, chance-constrained safe motion planning approach that creates a direct link between decision making and motion planning.

The introduced motion planning method, i.e. the decision-theoretic MPCC, is capable of meeting all the aforementioned requirements except for active information gathering. For planning motion that maximizes the agent's knowledge about the environment, the POMDP framework was inspected next. The POMDP framework optimizes over a large sequence of actions and observations at the cost of high computational complexity. Therefore, this work focused on solvers that approximate the solution by sampling. After reviewing state-of-the-art solver algorithms, a new POMDP solver was presented and used to solve motion planning problems in the POMDP formulation.

Planning with the POMDP formulation, due to the resulting computational complexity, was limited to planning with a few actions. For this reason, previous works presented them as behavior planning methods. Insights from model predictive control motivated studying the reward structure of the model used for planning with the POMDP formulation, which in turn revealed its Lipschitz continuity. Multi-armed bandits that exploit this continuity in sampling were proposed to improve their runtimes.

The model predictive control based motion planner was implemented by utilizing open-source non-commercial numerical optimization libraries. The first version of the planner relied on constrained nonlinear optimization with penalty methods. It showed excellent results in international autonomous driving contests and ordinary urban driving scenarios. Convenient driving corridor representations allowed the penalty function constrained solver to yield reliable results even for lateral maneuvering in urban driving. For more complicated motion planning tasks, the problem was solved with an interior-point nonlinear optimization solver that is robust against poor initialization. The simulation results from arbitrary real driving scenarios showed that uncertainty-aware, risk sensitive proactive motion plans are especially advantageous where the current environment information entails considerable amount of uncertainty. The simulations were restricted to scenarios where all traffic participants are vehicles, but the approach can be generalized for cyclists and pedestrians – if the dynamic boundaries are defined accordingly.

Framing the motion planning problem as a decision process and solving it with sampling-based POMDPs requires utilizing reward shaping to promote information gathering actions. Reward shaping for active information gathering in MCTS has been a unique contribution of this work. Motion planning with active information gathering for autonomous driving is done in real-time when a simple motion model is used. Utilizing Lipschitz continuity during sampling in MCTS enables massive performance improvements especially for a high number of actions, which is the default setting in motion planning.

Apart from the developments in motion planning, the close relationship of prediction and planning led to the development of a sampling-based, and a reachable set-based motion prediction algorithm. The reachable set-based prediction library is utilized in the safe planning approach within the arbitration framework of Bertha. Both planning approaches can also be employed within this framework or any other behavior tree.

Future Work

The contributions of this work create several directions for future work in different categories. The first category is regarding planning with the MPC formulation. As a fallback plan, i.e. Z-plan, full-braking maneuver is selected. An alternative is to plan other types of maneuvers such as lane change or swerving. However, these must be thoroughly inspected and carefully developed, as this may end with an increased hazard due to imperfect environment perception. Another possible direction is the utilized vehicle model. The nonlinear optimization algorithm allows integration of more complex vehicle models, making the planner especially suitable for fallback maneuver computation. Last but not least, Dempster-Shafer theory can be utilized for weighting the individual maneuver options against each other.

The second category of future work encompasses planning with the POMDP framework. The POMDP modeling in this work does not consider a limited visible field and the existence probabilities of other participants. However, the state representation can easily be extended with such additional features. The experiments evaluated active information gathering only to reveal maneuver intentions of other participants. Another application is the visible field maximization for regions of interest. Even though this is an inferior problem that can be solved using state-based rewards, it requires lateral offsetting from the center of the driving corridor. A potential extension would utilize an appropriate vehicle model to cover such actions.

In planning with the POMDP framework, actions and observations are sampled from Gaussian distribution. Real applications enable defining an upper and lower bound, resulting in a truncated Gaussian distribution. Utilization of specialized sampling procedures for such distributions can improve the efficiency further.

Exploiting the Lipschitz continuity of rewards enables significant performance improvements. A theoretical analysis on the validity of Lipschitz continuity and the regret of utilizing Lipschitz bandits will make this approach more sound. Recent works suggest that problems with information rewards also have a Lipschitz continuous reward function [Feh+18]. A potential further work includes inspecting the convergence of utilizing Lipschitz bandits for POMDPs with belief-based rewards. Finally, the use of generative models for sampling in conjunction weighted particle filtering is also a vital issue for future research.

A major problem in planning is the identification of driving goals and behavior. The parameter values quantifying them change continuously while driving. The quality of motion from both MPC and POMDP planning frameworks strongly depend on their accurate values. Thus, learning them significantly contributes to the quality.

Realizing autonomous driving is an important milestone for increasing driving safety and improving traffic flow efficiency. This work has made a significant contribution to model-based motion planning for autonomous driving. Future achievements in motion planning will strongly benefit from the developments in machine learning, as goal definitions pose a multi-objective problem of information gathering and utility maximization that can only be learned from driving data. Integrating them into model-based approaches will make autonomous driving robust and reliable.

References

[AB+20] E. Andreotti, P. Boyraz, et al. “Mathematical Definitions of Scene and Scenario for Analysis of Automated Driving Systems in Mixed-Traffic Simulations”. In: *IEEE Transactions on Intelligent Vehicles* 6.2 (2020), pp. 366–375. DOI: 10.1109/TIV.2020.3031981 (cit. on p. 7).

[ABG19] J. P. Alsterda, M. Brown, and J. C. Gerdes. “Contingency model predictive control for automated vehicles”. In: *Proceedings of the American Control Conference (ACC)*. Philadelphia, PA, USA, July 2019, pp. 717–722. DOI: 10.23919/ACC.2019.8815260 (cit. on pp. 20, 51).

[ACF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. “Finite-Time Analysis of the Multiarmed Bandit Problem”. In: *Machine Learning* 47.2 (2002), pp. 235–256. DOI: 10.1023/A:1013689704352 (cit. on p. 89).

[AD17] F. Altché and A. De La Fortelle. “Partitioning of the Free Space-Time for On-Road Navigation of Autonomous Ground Vehicles”. In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. Melbourne, Australia, Dec. 2017, pp. 2126–2133. DOI: 10.1109/CDC.2017.8263961 (cit. on p. 12).

[AFS12] M. Ali, P. Falcone, and J. Sjöberg. “Threat assessment design under driver parameter uncertainty”. In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. Maui, HI, USA, Dec. 2012, pp. 6315–6320. DOI: 10.1109/CDC.2012.6425989 (cit. on p. 51).

[AM+15] S. Agarwal, K. Mierle, et al. *Ceres Solver*. Version 1.10.0. Nov. 9, 2015. URL: http://ceres-solver.org (cit. on p. 111).

[AM16] M. Althoff and S. Magdici. “Set-Based Prediction of Traffic Participants on Arbitrary Road Networks”. In: *IEEE Transactions on Intelligent Vehicles* 1.2 (2016), pp. 187–202. DOI: 10.1109/TIV.2016.2622920 (cit. on p. 50).

[Ame+19] P. Amestoy, A. Buttari, J.-Y. L’Excellent, and T. Mary. “Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures”. In: *ACM Transactions on Mathematical Software* 45.1 (2019), pp. 1–26. DOI: 10.1145/3242094 (cit. on p. 112).

[AML18] K. Asadi, D. Misra, and M. Littman. “Lipschitz Continuity in Model-based Reinforcement Learning”. In: *Proceedings of the IEEE International Conference on Machine Learning (ICML)*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm, Sweden: PMLR, July 2018, pp. 264–273 (cit. on p. 133).

[AMS07] J.-Y. Audibert, R. Munos, and C. Szepesvári. “Tuning Bandit Algorithms in Stochastic Environments”. In: *Algorithmic Learning Theory*. Ed. by M. Hutter, R. A. Servedio, and E. Takimoto. Berlin, Germany: Springer, 2007, pp. 150–165. DOI: 10.1007/978-3-540-75225-7_15 (cit. on p. 90).

[And+19a] H. Andersen, J. Alonso-Mora, Y. H. Eng, D. Rus, and M. H. Ang. “Trajectory Optimization and Situational Analysis Framework for Autonomous Overtaking With Visibility Maximization”. In: *IEEE Transactions on Intelligent Vehicles* 5.1 (2019), pp. 7–20. DOI: 10.1109/TIV.2019.2955361 (cit. on p. 20).

[And+19b] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. “CasADi – A software framework for nonlinear optimization and optimal control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4 (cit. on p. 72).

[Aou+13] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How. “Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns”. In: *Autonomous Robots* 35.1 (2013), pp. 51–76. DOI: 10.1007/s10514-013-9334-3 (cit. on p. 19).

[Ara+10] M. Araya-López, O. Buffet, V. Thomas, and F. Charpillet. “A POMDP Extension with Belief-Dependent Rewards”. In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Red Hook, NY, USA: Curran Associates, Inc., 2010, pp. 64–72 (cit. on p. 98).

[Ara13] M. Araya-López. “Near-Optimal Algorithms for Sequential Information-Gathering Decision Problems”. PhD thesis. Nancy, France: Université de Lorraine, 2013. URL: https://tel.archives-ouvertes.fr/tel-00943513 (cit. on pp. 182, 183).

[Art+20] A. Artunedo, J. Villagra, J. Godoy, and M. D. Del Castillo. “Motion planning approach considering localization uncertainty”. In: *IEEE Transactions on Vehicular Technology* 69.6 (2020), pp. 5983–5994. DOI: 10.1109/TVT.2020.2985546 (cit. on p. 19).

[Art+99] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. “Coherent Measures of Risk”. In: *Mathematical Finance* 9.3 (1999), pp. 203–228. DOI: 10.1111/1467-9965.00068 (cit. on p. 48).

[Aru+02] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188. DOI: 10.1109/78.978374 (cit. on pp. 29, 85).

[ASB07a] M. Althoff, O. Stursberg, and M. Buss. “Online verification of cognitive car decisions”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Istanbul, Turkey, June 2007, pp. 728–733. DOI: 10.1109/IVS.2007.4290203 (cit. on p. 50).

[ASB07b] M. Althoff, O. Stursberg, and M. Buss. “Safety assessment of autonomous cars using verification techniques”. In: *Proceedings of the American Control Conference (ACC)*. New York, NY, USA, July 2007, pp. 4154–4159. DOI: 10.1109/ACC.2007.4282809 (cit. on p. 50).

[Åst65] K. J. Åström. “Optimal control of Markov processes with incomplete state information I”. In: *Journal of Mathematical Analysis and Applications* 10 (1965), pp. 174–205. DOI: 10.1016/0022-247X(65)90154-X (cit. on p. 82).

[Ban+18] H. Banzhaf, M. Dolgov, J. Stellet, and J. M. Zöllner. “From footprints to belief prints: Motion planning under uncertainty for maneuvering automated vehicles in dense scenarios”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, Nov. 2018, pp. 1680–1687. DOI: 10.1109/ITSC.2018.8569897 (cit. on p. 19).

[Bar19] J. T. Barron. “A General and Adaptive Robust Loss Function”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, June 2019, pp. 4331–4339. DOI: 10.1109/CVPR.2019.00446 (cit. on pp. 40, 184).

[Bay+18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. “Automatic differentiation in machine learning: a survey”. In: *Journal of machine learning research* 18 (2018). DOI: 10.5555/3122009.3242010 (cit. on p. 71).

[BDJ01] K. A. Brookhuis, D. De Waard, and W. H. Janssen. “Behavioural impacts of advanced driver assistance systems–an overview”. In: *European Journal of Transport and Infrastructure Research* 1.3 (2001), pp. 245–253. DOI: 10.18757/ejtir.2001.1.3.3667 (cit. on p. 1).

[BDK20] K. Brown, K. R. Driggs-Campbell, and M. J. Kochenderfer. “Modeling and Prediction of Human Driver Behavior: A Survey”. In: (2020). arXiv: 2006.08832 [eess.SY]. URL: https://arxiv.org/abs/2006.08832 (visited on 08/23/2021) (cit. on p. 13).

[Bec21] M. Beck. “Motion Planning for Automated Vehicles in Uncertain Environments”. Master’s thesis. Karlsruhe, Germany: Institute of Measurement and Control Systems, Karlsruhe Institute of Technology, 2021 (cit. on p. 127).

[Bel21] B. Bell. *CppAD: a package for C++ algorithmic differentiation*. Version 20210000.6. Apr. 27, 2021. URL: https://coin-or.github.io/CppAD/doc/cppad.htm (visited on 01/08/2022) (cit. on p. 72).

[Ben+14] K. Bengler, K. Dietmayer, B. Färber, M. Maurer, C. Stiller, and H. Winner. "Three decades of driver assistance systems: Review and future perspectives". In: *IEEE Intelligent Transportation Systems Magazine* 6.4 (2014), pp. 6–22. DOI: 10.1109/MITS.2014.2336271 (cit. on p. 1).

[Ben+15] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller. "The combinatorial aspect of motion planning: Maneuver variants in structured environments". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Seoul, Korea, June 2015, pp. 1386–1392. DOI: 10.1109/IVS.2015.7225909 (cit. on pp. 12, 17, 41).

[Ber+11] A. Berthelot, A. Tamke, T. Dang, and G. Breuel. "Handling uncertainties in criticality assessment". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany, June 2011, pp. 571–576. DOI: 10.1109/IVS.2011.5940483 (cit. on p. 45).

[Ber+12] A. Berthelot, A. Tamke, T. Dang, and G. Breuel. "A novel approach for the probabilistic computation of time-to-collision". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Madrid, Spain, June 2012, pp. 1173–1178. DOI: 10.1109/IVS.2012.6232221 (cit. on p. 45).

[Bey+21] H. Bey, M. Sackmann, A. Lange, and J. Thielecke. "Handling Prediction Model Errors in Planning for Automated Driving Using POMDPs". In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Indianapolis, IN, USA, Sept. 2021, pp. 439–446. DOI: 10.1109/ITSC48978.2021.9564408 (cit. on pp. 21, 142).

[BFD08] C. R. Baker, D. I. Ferguson, and J. M. Dolan. "Robust mission execution for autonomous urban driving". In: *Robotics Institute* (2008), p. 178. DOI: 10.3233/978-1-58603-887-8-155 (cit. on p. 53).

[BGD14] S. Brechtel, T. Gindele, and R. Dillmann. “Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China, Oct. 2014, pp. 392–399. DOI: 10.1109/ITSC.2014.6957722 (cit. on p. 21).

[Bha+21] R. Bhattacharyya, S. Jung, L. A. Kruse, R. Senanayake, and M. J. Kochenderfer. “A Hybrid Rule-Based and Data-Driven Approach to Driver Modeling Through Particle Filtering”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021). Early Access, pp. 1–14. DOI: 10.1109/TITS.2021.3119415 (cit. on p. 103).

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006. ISBN: 978-0387-31073-2 (cit. on pp. 26–28).

[BKO19] M. Bansal, A. Krizhevsky, and A. S. Ogale. “ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst”. In: *Proceedings of the Robotics: Science and Systems Conference, RSS XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*. Ed. by A. Bicchi, H. Kress-Gazit, and S. Hutchinson. 2019. DOI: 10.15607/RSS.2019.XV.031 (cit. on p. 22).

[BLW06] L. Blackmore, H. Li, and B. Williams. “A probabilistic approach to optimal robust path planning with obstacles”. In: *Proceedings of the American Control Conference (ACC)*. Minneapolis, MN, USA, June 2006, pp. 2831–2837. DOI: 10.1109/ACC.2006.1656653 (cit. on p. 47).

[Bou+18] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. “Scalable Decision Making with Sensor Occlusions for Autonomous Driving”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, May 2018, pp. 2076–2081. DOI: 10.1109/ICRA.2018.8460914 (cit. on p. 21).

[Bri+19] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora. “Model predictive contouring control for collision avoidance in unstructured dynamic environments”. In: *IEEE Robotics and Automation Letters*

4.4 (2019), pp. 4459–4466. DOI: 10.1109/LRA.2019.2929976 (cit. on p. 46).

[Bro+12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (2012), pp. 1–43. DOI: 10.1109/TCIAIG.2012.2186810 (cit. on p. 88).

[Brü+21] T. Brüdigam, M. Olbrich, D. Wollherr, and M. Leibold. "Stochastic model predictive control with a safety guarantee for automated driving". In: *IEEE Transactions on Intelligent Vehicles* (2021). Early Access, pp. 1–1. DOI: 10.1109/TIV.2021.3074645 (cit. on p. 20).

[BSY11] S. Bubeck, G. Stoltz, and J. Y. Yu. "Lipschitz Bandits without the Lipschitz Constant". In: *Algorithmic Learning Theory - Proceedings of the 22nd International Conference, ALT 2011, Espoo, Finland, October 5-7, 2011*. Ed. by J. Kivinen, C. Szepesvári, E. Ukkonen, and T. Zeugmann. Berlin, Germany: Springer, 2011, pp. 144–158. DOI: 10.1007/978-3-642-24412-4_14 (cit. on p. 142).

[Bub+08] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. "Online Optimization in X-Armed Bandits". In: *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS), Vancouver, British Columbia, Canada, December 8-11, 2008*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Red Hook, NY, USA: Curran Associates, Inc., 2008, pp. 201–208 (cit. on p. 142).

[Bur14] J. Burkardt. "The truncated normal distribution". In: *Department of Scientific Computing Website, Florida State University* (2014), pp. 1–35 (cit. on p. 25).

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004. DOI: https://doi.org/10.1017/CBO9780511804441 (cit. on pp. 31, 67).

[BZS14] P. Bender, J. Ziegler, and C. Stiller. “Lanelets: Efficient map representation for autonomous driving”. In: *Proceedings of the Intelligent Vehicles Symposium (IV)*. Dearborn, MI, USA, June 2014, pp. 420–425. DOI: 10.1109/IVS.2014.6856487 (cit. on p. 9).

[Cam+14] G. R. de Campos, A. H. Runarsson, F. Granum, P. Falcone, and K. Alenljung. “Collision avoidance at intersections: A probabilistic threat-assessment and decision-making system for safety interventions”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China, Oct. 2014, pp. 649–654. DOI: 10.1109/ITSC.2014.6957763 (cit. on pp. 48, 60).

[Car+14] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli. “Stochastic predictive control of autonomous vehicles in uncertain environments”. In: *12th International Symposium on Advanced Vehicle Control*. Tokyo, Japan, Sept. 2014, pp. 712–719 (cit. on pp. 20, 48).

[CC06] G. C. Calafiore and M. C. Campi. “The scenario approach to robust control design”. In: *IEEE Transactions on Automatic Control* 51.5 (2006), pp. 742–753. DOI: 10.1109/TAC.2006.875041 (cit. on p. 48).

[CC63] A. Charnes and W. W. Cooper. “Deterministic equivalents for optimizing and satisficing under chance constraints”. In: *Operations research* 11.1 (1963), pp. 18–39. DOI: 10.1287/opre.11.1.18 (cit. on p. 47).

[CD02] D. Crisan and A. Doucet. “A Survey of Convergence Results on Particle. Filtering Methods for Practitioners”. In: *IEEE Transactions on Signal Processing* 50.3 (2002), pp. 736–746. DOI: 10.1109/78.984773 (cit. on p. 29).

[CE06] G. C. Calafiore and L. El Ghaoui. “On Distributionally Robust Chance-Constrained Linear Programs”. In: *Journal of Optimization Theory and Applications* 130.1 (2006), pp. 1–22. DOI: 10.1007/s10957-006-9084-x (cit. on p. 47).

[CG14] Y. Chow and M. Ghavamzadeh. “Algorithms for CVaR Optimization in MDPs”. In: *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems (NIPS), December 8-13 2014, Montreal, Quebec, Canada*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Red Hook, NY, USA: Curran Associates, Inc., 2014, pp. 3509–3517 (cit. on p. 48).

[Cha+19] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. “MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction”. In: (Oct. 2019). Ed. by L. P. Kaelbling, D. Kragic, and K. Sugiura, pp. 86–99. URL: http://proceedings.mlr.press/v100/chai20a.html (cit. on p. 13).

[Cho+05] H. M. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, and R. C. Arkin. *Principles of robot motion: theory, algorithms, and implementation*. Cambridge, MA, USA: MIT Press, 2005. ISBN: 0-262-03327-5 (cit. on p. 38).

[Cor+09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009. ISBN: 978-0-262-03384-8 (cit. on p. 17).

[Cou+11] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard. “Continuous Upper Confidence Trees”. In: *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*. Ed. by C. A. C. Coello. Berlin, Germany: Springer, 2011, pp. 433–445. DOI: 10.1007/978-3-642-25566-3_32 (cit. on p. 95).

[CPI14] A. Constantin, J. Park, and K. Iagnemma. “A margin-based approach to threat assessment for autonomous highway navigation”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, MI, USA, June 2014, pp. 234–239. DOI: 10.1109/IVS.2014.6856584 (cit. on p. 52).

[CSU21] S. Casas, A. Sadat, and R. Urtasun. “MP3: A Unified Model to Map, Perceive, Predict and Plan”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Virtual, June 2021, pp. 14403–14412 (cit. on p. 22).

[De +14] A. De La Fortelle, X. Qian, S. Diemer, J. Grégoire, F. Moutarde, S. Bonnabel, A. Marjovi, A. Martinoli, I. Llatser, A. Festag, et al. "Network of automated vehicles: the AutoNet 2030 vision". In: *World Congress on Intelligent Transport Systems (ITS)*. Detroit, MI, USA, Sept. 2014. URL: https://hal-mines-paristech.archives-ouvertes.fr/hal-01063484 (cit. on p. 1).

[DE14] F. Damerow and J. Eggert. "Predictive risk maps". In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China, Oct. 2014, pp. 703–710. DOI: 10.1109/ITSC.2014.6957772 (cit. on p. 52).

[DGA00] A. Doucet, S. Godsill, and C. Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and Computing* 10.3 (2000), pp. 197–208. DOI: 10.1023/A:1008935410038 (cit. on p. 85).

[DKH15] M. Dolgov, G. Kurz, and U. D. Hanebeck. "Chance-constrained model predictive control based on box approximations". In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. Osaka, Japan, Dec. 2015, pp. 7189–7194. DOI: 10.1109/CDC.2015.7403353 (cit. on p. 47).

[Dre+15] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh. "Efficient Guiding Strategies for Testing of Temporal Properties of Hybrid Systems". In: *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*. Ed. by K. Havelund, G. J. Holzmann, and R. Joshi. Vol. 9058. Cham, Switzerland: Springer, 2015, pp. 127–142. DOI: 10.1007/978-3-319-17524-9_10 (cit. on p. 51).

[Dre78] Z. Drezner. "Computation of the bivariate normal integral". In: *Mathematics of Computation* 32.141 (1978), pp. 277–279. DOI: 10.2307/2006276 (cit. on p. 48).

[DSW15] B. Duraisamy, T. Schwarz, and C. Wohler. "On track-to-track data association for automotive sensor fusion". In: *Proceedings of the International Conference on Information Fusion (FUSION)*. Washington, DC, USA, July 2015, pp. 1213–1222 (cit. on p. 73).

[Eck+16] A. Eck, L.-K. Soh, S. Devlin, and D. Kudenko. “Potential-Based Reward Shaping for Finite Horizon Online POMDP Planning”. In: *Autonomous Agents and Multi-Agent Systems* 30 (2016), pp. 403–445. DOI: 10.1007/s10458-015-9292-6 (cit. on p. 97).

[EDK15] J. Eggert, F. Damerow, and S. Klingelschmitt. “The foresighted driver model”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Seoul, South Korea, June 2015, pp. 322–329. DOI: 10.1109/IVS.2015.7225706 (cit. on p. 13).

[EM03] E. Even-Dar and Y. Mansour. “Learning Rates for Q-Learning”. In: *Journal of Machine Learning Research* 5 (2003), pp. 1–25 (cit. on pp. 94, 142).

[Eng+16] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff. “The Grand Cooperative Driving Challenge 2016: boosting the introduction of cooperative automated vehicles”. In: *IEEE Transactions on Wireless Communications* 23.4 (2016), pp. 146–152. DOI: 10.1109/MWC.2016.7553038 (cit. on p. 112).

[EP08] A. Eidehall and L. Petersson. “Statistical threat assessment for general road scenes using Monte Carlo sampling”. In: *IEEE Transactions on Intelligent Transportation Systems* 9.1 (2008), pp. 137–147. DOI: 10.1109/TITS.2007.909241 (cit. on p. 52).

[Eri04] C. Ericson. *Real-Time Collision Detection*. San Francisco, CA, USA: Morgan Kaufmann, 2004. ISBN: 978-1-55860-732-3 (cit. on p. 109).

[Fan+20] L. Fang, Q. Jiang, J. Shi, and B. Zhou. “TPNet: Trajectory Proposal Network for Motion Prediction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, June 2020, pp. 6797–6806. DOI: 10.1109/CVPR42600.2020.00683 (cit. on p. 13).

[FAR21] *Fatality Analysis Reporting System (FARS) Encyclopedia*. National Highway Traffic Safety Administration. 2021. URL: http://www-fars.nhtsa.dot.gov/Main/index.aspx (visited on 07/18/2021) (cit. on p. 1).

[Feh+18] M. Fehr, O. Buffet, V. Thomas, and J. Dibangoye. "ρ-POMDPs have Lipschitz-Continuous ϵ-Optimal Value Functions". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Red Hook, NY, USA: Curran Associates, Inc., 2018, pp. 6933–6943 (cit. on p. 145).

[Fin09] T. Finch. "Incremental calculation of weighted mean and variance". In: *University of Cambridge* 4.11-5 (2009), pp. 41–42 (cit. on p. 94).

[Fli+95] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. "Flatness and defect of non-linear systems: introductory theory and examples". In: *International Journal of Control* 61.6 (1995), pp. 1327–1361. DOI: 10.1080/00207179508921959 (cit. on p. 38).

[FT20] J. Fischer* and Ö. Ş. Taş*. "Information Particle Filter Tree: An Online Algorithm for POMDPs with Belief-Based Rewards on Continuous Domains". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. PMLR. 2020, pp. 3177–3187 (cit. on p. 98).

[GC11] A. Garivier and O. Cappé. "The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond". In: *COLT 2011 - The 24th Annual Conference on Learning Theory, June 9-11, 2011, Budapest, Hungary*. Ed. by S. M. Kakade and U. von Luxburg. JMLR, 2011, pp. 359–376 (cit. on p. 90).

[Ger91] A. Geraci. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE, 1991, pp. 1–217. DOI: 10.1109/IEEESTD.1991.106963 (cit. on p. 44).

[GGG20] J. Y. Goh, T. Goel, and C. J. Gerdes. "Toward Automated Vehicle Control Beyond the Stability Limits: Drifting Along a General Path". In: *Journal of Dynamic Systems, Measurement, and Control* 142.2 (2020), p. 021004. DOI: 10.1115/1.4045320 (cit. on p. 36).

[GI83] D. Goldfarb and A. Idnani. “A numerically stable dual method for solving strictly convex quadratic programs”. In: *Mathematical programming* 27.1 (1983), pp. 1–33. DOI: 10.1007/BF02591962 (cit. on p. 68).

[Gif+18] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli. “The control toolbox — An open-source C++ library for robotics, optimal and model predictive control”. In: *Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. Brisbane, Australia, May 2018, pp. 123–129. DOI: 10.1109/SIMPAR.2018.8376281 (cit. on pp. 70, 72).

[Gin+08] T. Gindele, D. Jagszent, B. Pitzer, and R. Dillmann. “Design of the planner of Team AnnieWAY’s autonomous vehicle used in the DARPA Urban Challenge 2007”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Eindhoven, The Netherlands, June 2008, pp. 1131–1136. DOI: 10.1109/IVS.2008.4621268 (cit. on p. 60).

[GNS09] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. Philadelphia, PA, USA: SIAM, 2009. DOI: 10.1137/1.9780898717730 (cit. on p. 65).

[Gon+15] D. González, J. Pérez, V. Milanés, and F. Nashashibi. “A review of motion planning techniques for automated vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2015), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841 (cit. on pp. 18, 19).

[Gra18] A. Gramacki. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Cham, Switzerland: Springer, 2018. DOI: 10.1007/978-3-319-71688-6 (cit. on p. 30).

[GS62] D. Gale and L. S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15. DOI: 10.1080/00029890.1962.11989827 (cit. on p. 103).

[Har28] R. V. Hartley. “Transmission of information 1”. In: *Bell System Technical Journal* 7.3 (1928), pp. 535–563. DOI: 10.1002/j.1538-7305.1928.tb01236.x (cit. on p. 31).

[Hau97] M. Hauskrecht. “Incremental Methods for Computing Bounds in Partially Observable Markov Decision Processes”. In: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*. Ed. by B. Kuipers and B. L. Webber. Cambridge, MA, USA: The MIT Press, 1997, pp. 734–739 (cit. on p. 87).

[HBC14] A. Houénou, P. Bonnifait, and V. Cherfaoui. “Risk assessment for collision avoidance systems”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China, Oct. 2014, pp. 386–391. DOI: 10.1109/ITSC.2014.6957721 (cit. on p. 48).

[HD15] U. D. Hanebeck and M. Dolgov. “Adaptive lower bounds for gaussian measures of polytopes”. In: *Proceedings of the International Conference on Information Fusion (FUSION)*. Washington, DC, USA, July 2015, pp. 1489–1496 (cit. on p. 47).

[Hel+16] A. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera, and N. O. Salscheider. “Robot Operating System: A Modular Software Framework for Automated Driving”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil, Nov. 2016, pp. 1564–1570. DOI: 10.1109/ITSC.2016.7795766 (cit. on p. 112).

[Hil07] J. Hillenbrand. “Fahrerassistenz zur Kollisionsvermeidung”. PhD thesis. Karlsruhe, Germany: Karlsruhe Institute of Technology (KIT), 2007. DOI: 10.5445/IR/1000007728 (cit. on p. 45).

[HKZ19] L. Hewing, J. Kabzan, and M. N. Zeilinger. “Cautious Model Predictive Control using Gaussian Process Regression”. In: *IEEE Transactions on Control Systems Technology* 28.6 (2019), pp. 2736–2743. DOI: 10.1109/TCST.2019.2949757 (cit. on pp. 20, 75).

[HSD17] S. Hoermann, D. Stumper, and K. Dietmayer. “Probabilistic Long-Term Prediction for Autonomous Vehicles”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA, June 2017, pp. 237–243. DOI: 10.1109/IVS.2017.7995726 (cit. on p. 184).

[HSK06] J. Hillenbrand, A. Spieker, and K. Kroschel. “Efficient decision making for a multi-level collision mitigation system”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Tokyo, Japan, June 2006, pp. 460–465. DOI: 10.1109/IVS.2006.1689671 (cit. on p. 45).

[HSL13] HSL. *A collection of Fortran codes for large scale scientific computation*. 2013. URL: http://www.hsl.rl.ac.uk (visited on 05/21/2019) (cit. on p. 140).

[Hub+18] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller. “Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction”. In: *IEEE Transactions on Intelligent Vehicles* 3.1 (2018), pp. 5–17. DOI: 10.1109/TIV.2017.2788208 (cit. on pp. 21, 102).

[Hub+19] C. Hubmann, N. Quetschlich, J. Schulz, J. Bernhard, D. Althoff, and C. Stiller. “A POMDP Maneuver Planner For Occlusions in Urban Scenarios”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, June 2019, pp. 2172–2179. DOI: 10.1109/IVS.2019.8814179 (cit. on p. 21).

[IPM13] A. H. Ibrahim, C. Pegard, and E.-M. Mouaddib. “Collision prediction between vehicles in an unstructured environment”. In: *Proceedings of the International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. Sousse, Tunisia, Dec. 2013, pp. 509–514. DOI: 10.1109/STA.2013.6783179 (cit. on p. 46).

[ISO09] *Risk management–principles and guidelines*. ISO 31000:2009(E). Geneva, Switzerland: International Organization for Standardization, Nov. 2009 (cit. on p. 44).

[Iva+20] B. Ivanovic, A. Elhafsi, G. Rosman, A. Gaidon, and M. Pavone. “MATS: An Interpretable Trajectory Forecasting Representation for Planning and Control”. In: *Proceedings of the Conference on Robot Learning (CoRL)*. Ed. by J. Kober, F. Ramos, and C. J. Tomlin. Virtual Event / Cambridge, MA, USA: PMLR, Nov. 2020, pp. 2243–2256 (cit. on p. 22).

[Joã09] João Rui Leal. *CppADCodeGen*. Version 2.4.3. 2020-10-09. URL: https://github.com/joaoleal/CppADCodeGen (visited on 11/04/2021) (cit. on p. 72).

[Joh14] S. G. Johnson. *The NLopt nonlinear-optimization package*. 2014. URL: https://nlopt.readthedocs.io/en/latest/ (visited on 10/29/2021) (cit. on p. 68).

[Kam+08] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schröder, M. Thuy, M. Goebl, F. v. Hundelshausen, et al. "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge". In: *Journal of Field Robotics* 25.9 (2008), pp. 615–639. DOI: 10.1002/rob.20252 (cit. on p. 53).

[Kam+21] D. Kamran, T. Engelgeh, M. Busch, J. Fischer, and C. Stiller. "Minimizing Safety Interference for Safe and Comfortable Automated Driving with Distributional Reinforcement Learning". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic, Sept. 2021, pp. 1236–1243. DOI: 10.1109/IROS51168.2021.9636847 (cit. on p. 22).

[KHZ20] K. Kurzer, C. Hörtnagl, and J. M. Zöllner. "Parallelization of Monte Carlo Tree Search in Continuous Domains". In: (2020). arXiv: 2003.13741 [cs.LG]. URL: https://arxiv.org/abs/2003.13741 (visited on 10/10/2021) (cit. on p. 141).

[Kim+12] J. Kim, G. Bhatia, R. Rajkumar, and M. Jochim. "SAFER: System-level Architecture for Failure Evasion in Real-time Applications". In: *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*. San Juan, PR, USA, Dec. 2012, pp. 227–236. DOI: 10.1109/RTSS.2012.74 (cit. on p. 53).

[Koc15] M. J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. Cambridge, MA, USA: MIT Press, 2015. ISBN: 978-0-262-02925-4 (cit. on p. 80).

[Koh+13] R. Kohlhaas, T. Schamm, D. Lenk, and J. M. Zöllner. "Towards driving autonomously: Autonomous cruise control in urban environments". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Gold Coast City, Australia, June 2013, pp. 109–114. DOI: 10.1109/IVWorkshops.2013.6615235 (cit. on p. 112).

[Kos+19] M. Koschi, C. Pek, S. Maierhofer, and M. Althoff. “Computationally efficient safety falsification of adaptive cruise control systems”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Auckland, New Zealand, Oct. 2019, pp. 2879–2886. DOI: 10.1109/ITSC.2019.8917287 (cit. on p. 51).

[Kra88] D. Kraft. *A software package for sequential quadratic programming*. Cologne, Germany: Wissenschaftliches Berichtswesen der Deutschen Forschungs-und Versuchsanstalt für Luft-und Raumfahrt, 1988 (cit. on p. 68).

[KS06] L. Kocsis and C. Szepesvári. “Bandit Based Monte-Carlo Planning”. In: *Machine Learning: ECML 2006, Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*. Ed. by S. M. Fürnkranz J. Scheffer T. Berlin, Germany: Springer, 2006, pp. 282–293. DOI: 10.1007/11871842_29 (cit. on p. 88).

[KSU08] R. Kleinberg, A. Slivkins, and E. Upfal. “Multi-armed bandits in metric spaces”. In: *STOC ’08: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2008, pp. 681–690. DOI: 10.1145/1374376.1374475 (cit. on p. 142).

[KTH07] A. Kesting, M. Treiber, and D. Helbing. “General lane-changing model MOBIL for car-following models”. In: *Transportation Research Record: Journal of the Transportation Research Board* 1999.1 (2007), pp. 86–94. DOI: 10.3141/1999-10 (cit. on p. 62).

[Kuh53] H. W. Kuhn. “11. Extensive Games and the Problem of Information”. In: *Contributions to the Theory of Games (AM-28), Volume II*. Ed. by H. W. Kuhn and A. W. Tucker. Princeton, NJ, USA: Princeton University Press, 1953, pp. 193–216. DOI: 10.1515/9781400881970-012 (cit. on p. 7).

[Kur+21] K. Kurzer, P. Schörner, A. Albers, H. Thomsen, K. Daaboul, and J. M. Zöllner. “Generalizing Decision Making for Automated Driving with an Invariant Environment Representation using Deep Reinforcement Learning”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, June 2021,

pp. 994–1000. DOI: 10.1109/IV48863.2021.9575669 (cit. on p. 45).

[KW19] M. J. Kochenderfer and T. A. Wheeler. *Algorithms for Optimization*. Cambridge, MA, USA: MIT Press, 2019. ISBN: 978-0-262-03942-0 (cit. on pp. 65, 66).

[KZ86] 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. DOI: 10.1177/027836498600500304 (cit. on p. 16).

[LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006. ISBN: 978-0-521-86205-9 (cit. on pp. 16, 38).

[LDM15] A. Liniger, A. Domahidi, and M. Morari. "Optimization-Based Autonomous Racing of 1:43 Scale RC Cars". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647. DOI: 10.1002/oca.2123 (cit. on p. 75).

[Lee+17] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. "Desire: Distant future prediction in dynamic scenes with interacting agents". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, July 2017, pp. 336–345. DOI: 10.1109/CVPR.2017.233 (cit. on p. 13).

[Lee+20] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer. "Adaptive stress testing: Finding likely failure events with reinforcement learning". In: *Journal of Artificial Intelligence Research* 69 (2020), pp. 1165–1201. DOI: 10.1613/jair.1.12190 (cit. on pp. 49, 51).

[Lie+12] M. Liebner, M. Baumann, F. Klanner, and C. Stiller. "Driver intent inference at urban intersections using the intelligent driver model". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Madrid, Spain, June 2012, pp. 1162–1167. DOI: 10.1109/IVS.2012.6232131 (cit. on p. 184).

[Lin18] A. Liniger. *Path planning and control for autonomous racing*. Zurich, Switzerland: ETH Zurich, 2018. DOI: 10.3929/ethz-b-000302942 (cit. on p. 20).

[LK21] V. Lefkopoulos and M. Kamgarpour. “Trajectory Planning Under Environmental Uncertainty with Finite-Sample Safety Guarantees”. In: *Automatica* 131 (2021), p. 109754. DOI: 10.1016/j.automatica.2021.109754 (cit. on p. 48).

[LKH10] B. Luders, M. Kothari, and J. How. “Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty”. In: *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. Toronto, Ontario, Canada, Aug. 2010, p. 8160. DOI: 10.2514/6.2010-8160 (cit. on pp. 19, 48).

[LKK15] D. Lenz, T. Kessler, and A. Knoll. “Stochastic model predictive controller with chance constraints for comfortable and safe driving behavior of autonomous vehicles”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Seoul, South Korea, June 2015, pp. 292–297. DOI: 10.1109/IVS.2015.7225701 (cit. on pp. 20, 48).

[LKK16] D. Lenz, T. Kessler, and A. Knoll. “Tactical cooperative planning for autonomous highway driving using Monte-Carlo Tree Search”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden, June 2016, pp. 447–453. DOI: 10.1109/IVS.2016.7535424 (cit. on p. 21).

[LMG10] D. Lam, C. Manzie, and M. Good. “Model predictive contouring control”. In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. Atlanta, Georgia, USA, Dec. 2010, pp. 6137–6142. DOI: 10.1109/CDC.2010.5717042 (cit. on p. 75).

[Luo+19] Y. Luo, H. Bai, D. Hsu, and W. S. Lee. “Importance sampling for online planning under uncertainty”. In: *The International Journal of Robotics Research* 38.2-3 (2019), pp. 162–181. DOI: 10.1177/0278364918780322 (cit. on p. 22).

[LVL14] S. Lefèvre, D. Vasquez, and C. Laugier. “A survey on motion prediction and risk assessment for intelligent vehicles”. In: *Robomech Journal* 1.1 (2014), p. 1. DOI: 10.1186/s40648-014-0001-z (cit. on p. 13).

[Lyo14] A. Lyon. “Why are Normal Distributions Normal?” In: *The British Journal for the Philosophy of Science* 65.3 (2014), pp. 621–649. DOI: 10.1093/bjps/axs046 (cit. on p. 23).

[MA16] S. Magdici and M. Althoff. “Fail-safe motion planning of autonomous vehicles”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil, Nov. 2016, pp. 452–458. DOI: 10.1109/ITSC.2016.7795594 (cit. on p. 50).

[Mac03] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge, MA, USA: Cambridge University Press, 2003. ISBN: 978-0-521-64298-9 (cit. on pp. 27, 29).

[Mag18] S. Magureanu. “Efficient Online Learning under Bandit Feedback”. PhD thesis. Stockholm, Sweden: KTH Royal Institute of Technology, 2018 (cit. on p. 91).

[Mar19] C. C. Margossian. “A review of automatic differentiation and its efficient implementation”. In: *WIREs Data Mining and Knowledge Discovery* 9.4 (2019), e1305. DOI: 10.1002/widm.1305 (cit. on p. 71).

[MB01] M. M. Minderhoud and P. H. Bovy. “Extended time-to-collision measures for road traffic safety assessment”. In: *Accident Analysis & Prevention* 33.1 (2001), pp. 89–97. DOI: 10.1016/S0001-4575(00)00019-1 (cit. on p. 45).

[Mih+02] L. Mihaylova, T. Lefebvre, H. Bruyninckx, K. Gadeyne, and J. D. Schutter. “A Comparison of Decision Making Criteria and Optimization Methods for Active Robotic Sensing”. In: *Revised Papers from the 5th International Conference on Numerical Methods and Applications, NMA 2002, Borovets, Bulgaria, August 20-24, 2002*. Ed. by I. Dimov, I. Lirkov, S. Margenov, and Z. Zlatev. Berlin, Germany: Springer, 2002, pp. 316–324. DOI: 10.1007/3-540-36487-0_35 (cit. on p. 31).

[Mir+16] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. “Learning to Navigate in Complex Environments”. In: (2016). arXiv: 1611.03673 [cs.AI]. URL: http://arxiv.org/abs/1611.03673 (visited on 12/10/2021) (cit. on p. 22).

[MM10] O.-A. Maillard and R. Munos. “Online Learning in Adversarial Lipschitz Environments”. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona,*

Spain, September 20-24, 2010, Proceedings, Part II. Ed. by J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag. Berlin, Germany: Springer, 2010, pp. 305–320. DOI: 10.1007/978-3-642-15883-4_20 (cit. on p. 142).

[MM15] R. Matthaei and M. Maurer. "Autonomous driving–a top-down-approach". In: *at-Automatisierungstechnik* 63.3 (2015), pp. 155–167. DOI: 10.1515/auto-2014-1136 (cit. on p. 53).

[MP17] A. Majumdar and M. Pavone. "How should a robot assess risk? towards an axiomatic theory of risk in robotics". In: *Robotics Research, The 18th International Symposium, ISRR 2017, Puerto Varas, Chile, December 11-14, 2017*. Ed. by N. M. Amato, G. Hager, S. L. Thomas, and M. Torres-Torriti. Vol. 10. Cham, Switzerland: Springer, 2017, pp. 75–84. DOI: 10.1007/978-3-030-28619-4_10 (cit. on p. 48).

[Nau+21] M. Naumann, F. Wirth, F. Oboril, K.-U. Scholl, M. Soledad Elli, I. Alvarez, J. Weast, and C. Stiller. "On Responsibility Sensitive Safety in Car-following Situations - A Parameter Analysis on German Highways". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, June 2021, pp. 83–90. DOI: 10.1109/IV48863.2021.9575420 (cit. on p. 49).

[Nau21] M. Naumann. *Probabilistic Motion Planning for Automated Vehicles*. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie. Karlsruhe, Germany: KIT Scientific Publishing, 2021. DOI: 10.5445/KSP/1000126389 (cit. on p. 21).

[Neu+16] M. Neunert, M. Giftthaler, M. Frigerio, C. Semini, and J. Buchli. "Fast derivatives of rigid body dynamics for control, optimization and estimation". In: *Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. San Francisco, CA, USA, Dec. 2016, pp. 91–97. DOI: 10.1109/SIMPAR.2016.7862380 (cit. on pp. 70, 72).

[NHR99] A. Y. Ng, D. Harada, and S. J. Russell. "Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping". In: *Proceedings of the Sixteenth International Conference on Machine Learning, (ICML 1999), Bled, Slovenia, June*

27 - 30, 1999. Ed. by D. S. Bratko I. San Francisco, CA, USA: Morgan Kaufmann, 1999, pp. 278–287 (cit. on p. 97).

[NS07] A. Nemirovski and A. Shapiro. "Convex Approximations of Chance Constrained Programs". In: *SIAM Journal on Optimization* 17.4 (2007), pp. 969–996. DOI: 10.1137/050622328 (cit. on p. 47).

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. New York, NY, USA: Springer Science & Business Media, 2006. DOI: 10.1007/978-0-387-40065-5 (cit. on pp. 65–69).

[Orz+21] P. F. Orzechowski, C. Burger, M. Lauer, and C. Stiller. "Decision-making for automated vehicles using behavior-based arbitration schemes". In: *at - Automatisierungstechnik* 69.2 (2021), pp. 171–181. DOI: 10.1515/auto-2020-0099 (cit. on p. 51).

[PA18] C. Pek and M. Althoff. "Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization". In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, Sept. 2018, pp. 1447–1454. DOI: 10.1109/ITSC.2018.8569425 (cit. on p. 51).

[PA20] C. Pek and M. Althoff. "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization". In: *IEEE Transactions on Robotics* 37.3 (2020), pp. 798–814. DOI: 10.1109/TRO.2020.3036624 (cit. on p. 20).

[Pad+16] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. "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. DOI: 10.1109/TIV.2016.2578706 (cit. on pp. 18, 19).

[Pet+13] D. Petrich, T. Dang, D. Kasper, G. Breuel, and C. Stiller. "Map-based long term motion prediction for vehicles in traffic environments". In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. The Hague, The Netherlands, Oct. 2013, pp. 2166–2172. DOI: 10.1109/ITSC.2013.6728549 (cit. on p. 9).

[PF05] S. Petti and T. Fraichard. “Safe motion planning in dynamic environments”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Edmonton, Alberta, Canada, Aug. 2005, pp. 2210–2215. DOI: 10.1109/IROS.2005.1545549 (cit. on pp. 34, 50).

[PGT03] J. Pineau, G. J. Gordon, and S. Thrun. “Point-based value iteration: An anytime algorithm for POMDPs”. In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by G. Gottlob and T. Walsh. San Francisco, CA, USA: Morgan Kaufmann, 2003, pp. 1025–1032 (cit. on p. 87).

[PGT06] J. Pineau, G. Gordon, and S. Thrun. “Anytime point-based approximations for large POMDPs”. In: *Journal of Artificial Intelligence Research* 27.1 (2006), pp. 335–380. DOI: 10.1613/jair.2078 (cit. on p. 82).

[Pha+20] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff. “Covernet: Multimodal behavior prediction using trajectory sets”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, June 2020, pp. 14062–14071. DOI: 10.1109/CVPR42600.2020.01408 (cit. on p. 13).

[Pho75] B. T. Phong. “Illumination for computer generated pictures”. In: *Communications of the ACM* 18.6 (1975), pp. 311–317. DOI: 10.1145/360825.360839 (cit. on p. 107).

[PJ20] F. Poggenhans and J. Janosovits. “Pathfinding and Routing for Automated Driving in the Lanelet2 Map Framework”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, Sept. 2020, pp. 1–7. DOI: 10.1109/ITSC45102.2020.9294376 (cit. on pp. 9, 112).

[PK19] F. Pusse and M. Klusch. “Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Paris, France, June 2019, pp. 1013–1020. DOI: 10.1109/IVS.2019.8814125 (cit. on p. 22).

[PKI15] J. Park, S. Karumanchi, and K. Iagnemma. “Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1101–1115. DOI: 10.1109/TRO.2015.2459373 (cit. on p. 12).

[PLG21] V. Z. Patterson, F. E. Lewis, and J. C. Gerdes. “Optimal Decision Making for Automated Vehicles Using Homotopy Generation and Nonlinear Model Predictive Control”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, July 2021, pp. 1045–1050. DOI: 10.1109/IV48863.2021.9575302 (cit. on pp. 12, 41).

[Pog+18] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. “Lanelet2: A high-definition map framework for the future of automated driving”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, Nov. 2018, pp. 1672–1679. DOI: 10.1109/ITSC.2018.8569929 (cit. on pp. 9, 112).

[Pre+92] W. H. Press, S. A. Teukolsky, B. P. Flannery, and W. T. Vetterling. *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Cambridge, MA, USA: Cambridge University Press, 1992. DOI: 10.1021/ja965936f (cit. on p. 24).

[PT87] C. H. Papadimitriou and J. N. Tsitsiklis. “The Complexity of Markov Decision Processes”. In: *Mathematics of Operations Research* 12.3 (1987), pp. 441–450. DOI: 10.1287/moor.12.3.441 (cit. on p. 83).

[Qia+16] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle. “Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil, Nov. 2016, pp. 205–210. DOI: 10.1109/ITSC.2016.7795555 (cit. on p. 20).

[Raj11] R. Rajamani. *Vehicle Dynamics and Control*. Boston, MA, USA: Springer, 2011. DOI: 10.1007/978-1-4614-1433-9 (cit. on p. 38).

[RC04] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. New York, NY, USA: Springer, 2004. DOI: 10.1007/978-1-4757-4145-2 (cit. on p. 27).

[Ric+19] S. Richter, S. Wirges, H. Königshof, and C. Stiller. "Fusion of range measurements and semantic estimates in an evidential framework / Fusion von Distanzmessungen und semantischen Größen im Rahmen der Evidenztheorie". In: *tm - Technisches Messen* 86.s1 (2019), pp. 102–106. DOI: 10.1515/teme-2019-0052 (cit. on p. 73).

[RN16] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Bengaluru, India: Pearson Education Limited, 2016. ISBN: 978-93-325-4351-5 (cit. on pp. 16, 83).

[RU+00] R. T. Rockafellar, S. Uryasev, et al. "Optimization of conditional value-at-risk". In: *Journal of Risk* 2 (2000), pp. 21–42. DOI: 10.21314/JOR.2000.038 (cit. on p. 48).

[Sad+16] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan. "Information gathering actions over human internal state". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, Oct. 2016, pp. 66–73. DOI: 10.1109/IROS.2016.7759036 (cit. on pp. 20, 21).

[SAE21] *SAE J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. ISO/SAE PAS 22736:2021. Geneva, Switzerland: International Organization for Standardization, Apr. 2021 (cit. on pp. 1, 2).

[Sal+20] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. "Trajectron++: Dynamically-Feasible Trajectory Forecasting with Heterogeneous Data". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVIII*. Ed. by A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm. Cham, Switzerland: Springer, 2020, pp. 683–700. DOI: 10.1007/978-3-030-58523-5_40 (cit. on p. 22).

[Sch+14] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. "Motion planning with sequential convex optimization and convex collision checking". In: *The International Journal of Robotics Research* 33.9 (2014),

pp. 1251–1270. DOI: 10.1177/0278364914528132 (cit. on p. 46).

[Sch+17] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus. "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model". In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2017), pp. 2994–3008. DOI: 10.1109/TITS.2017.2771351 (cit. on pp. 20, 75).

[Sch+21] W. Schwarting, A. Pierson, S. Karaman, and D. Rus. "Stochastic Dynamic Games in Belief Space". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 2157–2172. DOI: 10.1109/TRO.2021.3075376 (cit. on p. 21).

[Sch14] C. Schmidt. *Fahrstrategien zur Unfallvermeidung im Straßenverkehr für Einzel-und Mehrobjektszenarien*. Vol. 30. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie. Karlsruhe, Germany: KIT Scientific Publishing, 2014. DOI: 10.5445/KSP/1000039757 (cit. on p. 54).

[Sch21] J. C. Schulz. "Interaction-Aware Probabilistic Behavior Prediction of Traffic Participants in Urban Environments". PhD thesis. Munich, Germany: Technische Universität München, 2021. URL: http://mediatum.ub.tum.de/?id=1543207 (cit. on p. 103).

[Sha48] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x (cit. on p. 31).

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London, UK: Chapman & Hall, 1986. ISBN: 978-0-412-24620-3 (cit. on p. 30).

[SK18] Z. N. Sunberg and M. J. Kochenderfer. "Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces". In: *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*. Ed. by M. de Weerdt, S. Koenig, G. Röger, and M. T. J. Spaan. Palo Alto, CA, USA: AAAI Press, 2018, pp. 259–263. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17734 (cit. on pp. 21, 96, 97).

[SK20] Z. Sunberg and M. J. Kochenderfer. “Improving Automated Driving through Planning with Human Internal States”. In: (2020). arXiv: 2005.14549 [cs.AI]. URL: https://arxiv.org/abs/2005.14549 (visited on 10/19/2021) (cit. on p. 21).

[SKA18] S. Sontges, M. Koschi, and M. Althoff. “Worst-case analysis of the time-to-react using reachable sets”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China, June 2018, pp. 1891–1897. DOI: 10.1109/IVS.2018.8500709 (cit. on p. 51).

[Sli19] A. Slivkins. “Introduction to multi-armed bandits”. In: *Foundations and Trends® in Machine Learning* 12.1-2 (2019), pp. 1–286. DOI: 10.1561/2200000068 (cit. on pp. 89, 142).

[SN99] A. T. Schwarm and M. Nikolaou. “Chance-constrained model predictive control”. In: *AIChE Journal* 45.8 (1999), pp. 1743–1752. DOI: 10.1002/aic.690450811 (cit. on p. 47).

[Som+13] A. Somani, N. Ye, D. Hsu, and W. S. Lee. “DESPOT: Online POMDP Planning with Regularization”. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Red Hook, NY, USA: Curran Associates, Inc., 2013, pp. 1772–1780 (cit. on p. 22).

[SS04] T. Smith and R. G. Simmons. “Heuristic Search Value Iteration for POMDPs”. In: *UAI ’04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, Banff, Canada, July 7-11, 2004*. Ed. by D. M. Chickering and J. Y. Halpern. Arlington, VA, USA: AUAI Press, 2004, pp. 520–527 (cit. on p. 87).

[SS73] R. D. Smallwood and E. J. Sondik. “The optimal control of partially observable Markov processes over a finite horizon”. In: *Operations Research* 21.5 (1973), pp. 1071–1088. DOI: 10.1287/opre.21.5.1071 (cit. on p. 87).

[SSF15] U. Schwesinger, R. Siegwart, and P. Furgale. “Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*

(ICRA). Seattle, WA, USA, May 2015, pp. 63–68. DOI: 10.1109/ICRA.2015.7138981 (cit. on p. 47).

[SSS17] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "On a Formal Model of Safe and Scalable Self-driving Cars". In: (2017). arXiv: 1708.06374 [cs.RO]. URL: http://arxiv.org/abs/1708.06374 (visited on 11/02/2021) (cit. on p. 49).

[Ste+15] J. E. Stellet, J. Schumacher, W. Branz, and J. M. Zöllner. "Uncertainty propagation in criticality measures for driver assistance". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Seoul, South Korea, June 2015, pp. 1187–1194. DOI: 10.1109/IVS.2015.7225844 (cit. on p. 45).

[Ste+16] J. E. Stellet, P. Vogt, J. Schumacher, W. Branz, and J. M. Zöllner. "Analytical derivation of performance bounds of autonomous emergency brake systems". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden, June 2016, pp. 220–226. DOI: 10.1109/IVS.2016.7535389 (cit. on p. 45).

[SV10] D. Silver and J. Veness. "Monte-Carlo Planning in Large POMDPs". In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. Ed. by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Red Hook, NY, USA: Curran Associates Inc., 2010, pp. 2164–2172 (cit. on pp. 87, 93).

[SWA16] M. Schreier, V. Willert, and J. Adamy. "An integrated approach to maneuver-based trajectory prediction and criticality assessment in arbitrary road environments". In: *IEEE Transactions on Intelligent Transportation Systems* 17.10 (2016), pp. 2751–2766. DOI: 10.1109/TITS.2016.2522507 (cit. on p. 45).

[Taş+16] Ö. Ş. Taş, F. Kuhnt, J. M. Zöllner, and C. Stiller. "Functional System Architectures towards Fully Automated Driving". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden, June 2016, pp. 304–309. DOI: 10.1109/IVS.2016.7535402 (cit. on pp. 1, 2, 44).

[Taş+17] Ö. Ş. Taş, S. Hörmann, B. Schäufele, and F. Kuhnt. “Automated Vehicle System Architecture with Performance Assessment”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Yokohama, Japan, Oct. 2017, pp. 1–8. DOI: 10.1109/ITSC.2017.8317862 (cit. on pp. 2, 3, 53).

[Taş+18] Ö. Ş. Taş*, N. O. Salscheider*, F. Poggenhans*, S. Wirges*, C. Bandera*, M. R. Zofka*, T. Strauss*, J. M. Zöllner*, and C. Stiller*. “Making Bertha Cooperate–Team AnnieWAY’s Entry to the 2016 Grand Cooperative Driving Challenge”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2018), pp. 1262–1276. DOI: 10.1109/TITS.2017.2749974 (cit. on pp. 38, 44, 112).

[Taş14] Ö. Ş. Taş. “Integrating Combinatorial Reasoning and Continuous Methods for Optimal Motion Planning of Autonomous Vehicles”. English. MA thesis. Karlsruhe, Germany: Karlsruhe Institute of Technology (KIT), Sept. 2014. DOI: 10.5445/IR/1000045135 (cit. on pp. 9, 12, 41, 68).

[Taş21] Ö. Ş. Taş. *P3IV: Probabilistic Prediction and Planning for Intelligent Vehicles Simulator*. 2021. URL: https://github.com/fzi-forschungszentrum-informatik/P3IV (visited on 11/08/2021) (cit. on p. 112).

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005. ISBN: 978-0-262-20162-9 (cit. on pp. 86, 107).

[THH00] M. Treiber, A. Hennecke, and D. Helbing. “Congested traffic states in empirical observations and microscopic simulations”. In: *Physical Review E* 62.2 (2000), pp. 1805–1824. DOI: 10.1103/PhysRevE.62.1805 (cit. on p. 13).

[THL21] Ö. Ş. Taş, F. Hauser, and M. Lauer. “Efficient Sampling in POMDPs with Lipschitz Bandits for Motion Planning in Continuous Spaces”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, June 2021, pp. 1081–1088. DOI: 10.1109/IV48863.2021.9575303 (cit. on p. 92).

[THS18] Ö. Ş. Taş, F. Hauser, and C. Stiller. “Decision-Time Postponing Motion Planning for Combinatorial Uncertain Maneuvering”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, Nov. 2018, pp. 2419–2425. DOI: 10.1109/itsc.2018.8569580 (cit. on pp. 42, 50).

[TS18] Ö. Ş. Taş and C. Stiller. “Limited Visibility and Uncertainty Aware Motion Planning for Automated Driving”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Changshu, China, June 2018, pp. 1171–1178. DOI: 10.1109/ivs.2018.8500369 (cit. on pp. 20, 34, 50, 51, 53).

[TS20] Ö. Ş. Taş and C. Stiller. “Tackling Existence Probabilities of Objects with Motion Planning for Automated Urban Driving”. In: (2020). arXiv: 2002.01254 [cs.RO]. URL: https://arxiv.org/abs/2002.01254 (visited on 08/09/2021) (cit. on pp. 18, 74).

[Ulb+15] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. “Defining and substantiating the terms scene, situation, and scenario for automated driving”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Gran Canaria, Spain, Sept. 2015, pp. 982–988. DOI: 10.1109/ITSC.2015.164 (cit. on pp. 7–9).

[UM13] S. Ulbrich and M. Maurer. “Probabilistic Online POMDP Decision Making for Lane Changes in Fully Automated Driving”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. The Hague, The Netherlands, Oct. 2013, pp. 2063–2067. DOI: 10.1109/ITSC.2013.6728533 (cit. on p. 21).

[VAK10] A. K. Verma, S. Ajit, and D. R. Karanki. *Reliability and Safety Engineering*. London, UK: Springer, 2010. DOI: 10.1007/978-1-4471-6269-8 (cit. on p. 44).

[Van04] D. H. Van Hessem. “Stochastic inequality constrained closed-loop model predictive control – with application to chemical process operation”. PhD thesis. Delft, The Netherlands: Delft University Press, 2004. ISBN: 978-9040724893 (cit. on p. 48).

[VT11] M. P. Vitus and C. J. Tomlin. “On feedback design and risk allocation in chance constrained control”. In: *Proceedings of the IEEE Conference on Decision and Control and European Control Conference (CDC/ECC)*. Orlando, FL, USA, Dec. 2011, pp. 734–739. DOI: 10.1109/CDC.2011.6160721 (cit. on pp. 20, 48).

[VT13] M. P. Vitus and C. J. Tomlin. “A probabilistic approach to planning and control in autonomous urban driving”. In: *Proceedings of the IEEE Conference on Decision and Control (CDC)*. Florence, Italy, Dec. 2013, pp. 2459–2464. DOI: 10.1109/CDC.2013.6760249 (cit. on pp. 20, 47).

[War+14] J. Ward, G. Agamennoni, S. Worrall, and E. Nebot. “Vehicle collision probability calculation for general traffic scenarios under uncertainty”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, MI, USA, June 2014, pp. 986–992. DOI: 10.1109/IVS.2014.6856430 (cit. on p. 45).

[WB06] A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106.1 (2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y (cit. on p. 69).

[WBS21] L. Wang, C. Burger, and C. Stiller. “Reasoning about Potential Hidden Traffic Participants by Tracking Occluded Areas”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Indianapolis, IN, USA, Sept. 2021, pp. 157–163. DOI: 10.1109/ITSC48978.2021.9564584 (cit. on p. 52).

[Wei09] F. Weißel. *Stochastische modell-prädiktive Regelung nichtlinearer Systeme*. Vol. 2. Karlsruhe Series on Intelligent Sensor-Actuator-Systems. Karlsruhe, Germany: KIT Scientific Publishing, 2009. DOI: 10.5445/KSP/1000010573 (cit. on p. 78).

[WJW20] A. Wang, A. Jasour, and B. C. Williams. “Non-gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6041–6048. DOI: 10.1109/LRA.2020.3010755 (cit. on p. 20).

[Wra+21] K. H. Wray, B. Lange, A. Jamgochian, S. J. Witwicki, A. Kobashi, S. Hagaribommanahalli, and D. Ilstrup. “POMDPs for Safe Visibility Reasoning in Autonomous Vehicles”. In: *Proceedings of the IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. Tokoname, Japan, Mar. 2021, pp. 191–195. DOI: 10.1109/ISR50024.2021.9419519 (cit. on p. 21).

[Wri05] M. Wright. “The interior-point revolution in optimization: history, recent developments, and lasting consequences”. In: *Bulletin of the American Mathematical Society* 42.1 (2005), pp. 39–56. DOI: 10.1090/S0273-0979-04-01040-7 (cit. on p. 69).

[Yeu12] R. W. Yeung. *A First Course in Information Theory*. Boston, MA, USA: Springer, 2012. DOI: 10.1007/978-1-4419-8608-5 (cit. on p. 182).

[Yur+20] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. “A survey of autonomous driving: Common practices and emerging technologies”. In: *IEEE Access* 8 (2020), pp. 58443–58469. DOI: 10.1109/ACCESS.2020.2983149 (cit. on p. 2).

[YVJ19] M.-Y. Yu, R. Vasudevan, and M. Johnson-Roberson. “Occlusion-aware risk assessment for autonomous driving in urban environments”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2235–2241. DOI: 10.1109/LRA.2019.2900453 (cit. on p. 52).

[ZAG06] Y. Zhang, E. K. Antonsson, and K. Grote. “A new threat assessment measure for collision avoidance systems”. In: *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Toronto, Ontario, Canada, Sept. 2006, pp. 968–975. DOI: 10.1109/ITSC.2006.1706870 (cit. on p. 45).

[Zen+19] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. “End-to-end interpretable neural motion planner”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, June 2019, pp. 8660–8669. DOI: 10.1109/CVPR.2019.00886 (cit. on p. 22).

[Zha+19] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kümmerle, H. Königshof, C. Stiller, A. de La Fortelle, et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: (2019). arXiv: 1910.03088 [cs.RO]. URL: http://arxiv.org/abs/1910.03088 (visited on 12/08/2021) (cit. on p. 112).

[Zie+14a] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, et al. "Making Bertha drive — An autonomous journey on a historic route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20. DOI: 10.1109/MITS.2014.2306552 (cit. on pp. 46, 68).

[Zie+14b] J. Ziegler, P. Bender, T. Dang, and C. Stiller. "Trajectory planning for Bertha — a local, continuous method". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, MI, USA, June 2014, pp. 450–457. DOI: 10.1109/IVS.2014.6856581 (cit. on p. 38).

[Zie17] J. Ziegler. *Optimale Trajektorienplanung für Automobile*. Vol. 35. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie. Karlsruhe, Germany: KIT Scientific Publishing, 2017. DOI: 10.5445/KSP/1000056530 (cit. on pp. 66, 68, 107).

[ZS10] J. Ziegler and C. Stiller. "Fast collision checking for intelligent vehicle motion planning". In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. La Jolla, CA, USA, June 2010, pp. 518–522. DOI: 10.1109/IVS.2010.5547976 (cit. on p. 46).

A Appendix

A.1 Appendix on Modeling Uncertain Information

A.1.1 Truncated Gaussian Distribution

Density calculation of a truncated univariate Gaussian distribution (TUVG) is done by limiting the density of its *parent* Gaussian distribution into a bounded interval. The density of a TUVG $\psi(\breve{\mu}, \breve{\sigma}, a, b)$ with $\breve{\mu}, \breve{\sigma}$ the mean and the variance of its parent and $a \leq x \leq b$ its probable interval, can be calculated as

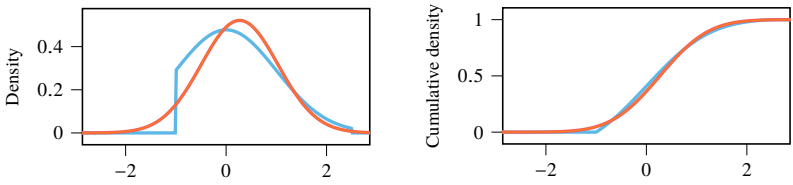
$$\psi(x \mid \breve{\mu}, \breve{\sigma}, a, b) = \begin{cases} 0 & x < a, \\ \eta\left(\phi(\xi(x \mid \breve{\mu}, \breve{\sigma}))\right) & a \leq x \leq b, \\ 0 & b < x. \end{cases} \tag{A.1}$$

$\xi(x \mid \mu, \sigma) = \frac{x-\mu}{\sigma}$ maps the parent Gaussian distribution to the standard Gaussian distribution $\mathcal{N}(0, 1)$. Since the density is bounded into the interval $[a, b]$, the density obtained from ϕ is multiplied with $\eta = z^{-1}$, where z is a normalizing constant $z = \Phi\left(\xi\left(b \mid \breve{\mu}, \breve{\sigma}\right)\right) - \Phi\left(\xi\left(a \mid \breve{\mu}, \breve{\sigma}\right)\right)$. Likewise, its CDF is

$$\Psi(x \mid \breve{\mu}, \breve{\sigma}, a, b) = \begin{cases} 0 & x < a, \\ \eta(\Phi\left(\xi\left(x \mid \breve{\mu}, \breve{\sigma}\right)\right) - \Phi\left(\xi\left(b \mid \breve{\mu}, \breve{\sigma}\right)\right)) & a \leq x \leq b, \\ 1 & b < x. \end{cases} \tag{A.2}$$

The density calculations above transform the density of a TUVG to a scaled and shifted Gaussian distribution $\mathcal{N}(\mu, \sigma)$. By slicing the resulting internal distribution into intervals, it yields accurate results. The internal distribution

serves as a rough approximation of the original TUVG, benefiting from the features of a UVG distribution (see Figure A.1).



(a) $\psi(x \mid \breve{\mu}, \breve{\sigma}, a, b)$ (b) $\Psi(x \mid \breve{\mu}, \breve{\sigma}, a, b)$

Figure A.1: Truncated univariate Gaussian distribution with $\breve{\mu} = 0.00$, $\breve{\sigma} = 1.00$ bounded into the interval $-1.0 \leq x \leq 2.5$. It is internally mapped into $\mathcal{N}(0.27, 0.77)$.

A.1.2 Further Information Measures

In information theory, information quantification that solely reflects the shape of the distribution is often not sufficient. A measure reflecting difference in both shape and shift between a target distribution P and a proposal distribution Q is frequently required. Such a measure must always return a non-negative value and take zero only if $P = Q$ [Yeu12, p. 19]. The *relative entropy*, or the *Kullback-Leibler divergence* (KLD), satisfies these requirements

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}. \tag{A.3}$$

This definition requires using the convention $P(x) \log \left(\frac{P(x)}{0}\right) = \infty$ for continuity. There are two things to underline on the use of KLD: First, $D_{\mathrm{KL}}(P \parallel Q)$ is not symmetric in P and Q, and second, it does not satisfy the triangle inequality. Therefore, although KLD is a statistical distance measure, it is not a metric.

Information theory contains many further alternative measures for information quantification. ℓ_p-norms, such that $\|P(x)\|_p$ with $p \in [1, \infty)$ can be seen as approximations of KLD where $Q \coloneqq \mathcal{U}$ [Ara13, p. 45]. In this way, the non-negativity condition of KLD is satisfied for any distribution. Similarly, if the probability distributions are inspected on a *probability simplex*, where each point represents a probability distribution, the distribution with the maximal

uncertainty corresponds to the center of that simplex. The distance of a point to this center is called *distance to the simplex center* (DSC)

$$D_{\mathrm{DSC}}(P) = D_{\gamma}(P, \mathscr{U}) \tag{A.4}$$

and can be used as an information measure [Ara13, p. 47]. The distance D_{γ} to the uniform distribution $\mathscr{U}$ can be calculated with the Bhattacharyya distance, which measures the amount of overlap between two distributions, or an ℓ_p-norm. In case of an ℓ_p-norm, as the probabilities at any point in a probability simplex sum up to one, $p > 1$ must be chosen.

A.2 Appendix on Mathematical Models

A.2.1 Intelligent Driver Model

The Intelligent Driver Model (IDM) yields the acceleration of a vehicle as

$$a_{\mathrm{IDM}} = a_{\mathrm{des}} + a_{\mathrm{int}}, \tag{A.5}$$

where the first term yields a comfortable acceleration to desired speed

$$a_{\mathrm{des}} = a^{+}\left(1 - \left(\frac{v}{v_{\mathrm{des}}}\right)^{4}\right),$$

and the second term accounts for interaction in presence of other vehicles

$$a_{\mathrm{int}} = -a^{+}\left(\frac{s_{\mathrm{int}}}{s}\right)^{2},$$

with

$$s_{\mathrm{int}} = s_{\mathrm{min}} + v\, t_{\mathrm{hw}} + \left(\frac{v\, v_{\mathrm{rel}}}{2\sqrt{a^{+}a^{-}_{\mathrm{cft}}}}\right).$$

The parameters are: s distance between vehicles in the Frenet frame, s_{min} stand-still distance, t_{hw} time headway, v current speed of the vehicle, v_{des} desired speed of the vehicle, v_{rel} relative speed of the vehicle to its leader, a^{-}_{cft} comfortable deceleration, and a^{+} maximum acceleration. The values of these

parameters vary for different driver profiles and can be estimated as proposed in [HSD17]. Their commonly used values are presented in [Lie+12]. The interaction term takes only negative values, meaning that the vehicle can interact only by braking.

A.2.2 Runge-Kutta Integration

$$\boldsymbol{k}_1 = \mathbf{f}(\boldsymbol{x}^{(i)}, \boldsymbol{a}^{(i)}) \tag{A.6a}$$

$$\boldsymbol{k}_2 = \mathbf{f}(\boldsymbol{x}^{(i)} + \frac{t_\mathrm{s}}{2}\boldsymbol{k}_1, \boldsymbol{a}^{(i)}) \tag{A.6b}$$

$$\boldsymbol{k}_3 = \mathbf{f}(\boldsymbol{x}^{(i)} + \frac{t_\mathrm{s}}{2}\boldsymbol{k}_2, \boldsymbol{a}^{(i)}) \tag{A.6c}$$

$$\boldsymbol{k}_4 = \mathbf{f}(\boldsymbol{x}^{(i)} + t_\mathrm{s}\,\boldsymbol{k}_1, \boldsymbol{a}^{(i)}) \tag{A.6d}$$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \frac{t_\mathrm{s}}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{A.6e}$$

A.2.3 A General and Adaptive Loss Function

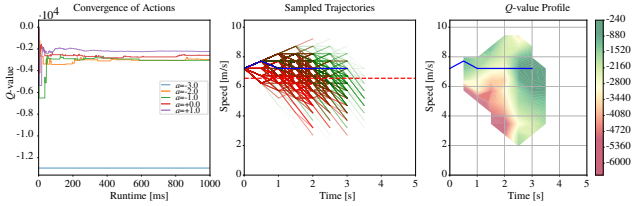
$$\rho(x, \alpha, c) = \begin{cases} \frac{1}{2}(x/c)^2 & \text{if } \alpha = 2 \\ \log\left(\frac{1}{2}(x/c)^2 + 1\right) & \text{if } \alpha = 0 \\ 1 - \exp\left(-\frac{1}{2}(x/c)^2\right) & \text{if } \alpha = -\infty \\ \frac{|\alpha - 2|}{\alpha}\left(\left(\frac{(x/c)^2}{|\alpha - 2|} + 1\right)^{\alpha/2} - 1\right) & \text{otherwise} \end{cases} \tag{A.7}$$

$$\frac{\partial \rho}{\partial x}(x, \alpha, c) = \begin{cases} \frac{x}{c^2} & \text{if } \alpha = 2 \\ \frac{2x}{x^2 + 2c^2} & \text{if } \alpha = 0 \\ \frac{x}{c^2}\exp\left(-\frac{1}{2}(x/c)^2\right) & \text{if } \alpha = -\infty \\ \frac{x}{c^2}\left(\frac{(x/c)^2}{|\alpha - 2|} + 1\right)^{(\alpha/2 - 1)} & \text{otherwise} \end{cases} \tag{A.8}$$

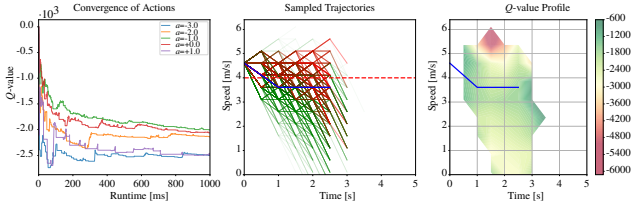
For additional information, please refer to [Bar19].

A.3 Additional Results

A.3.1 Active Information Gathering

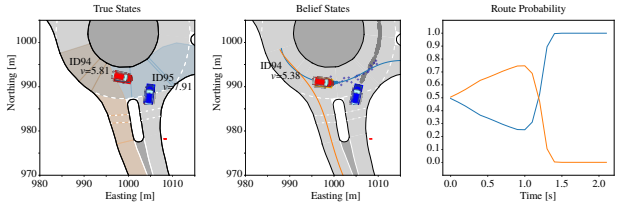


(a) $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 5.0\,\text{m/s}$.

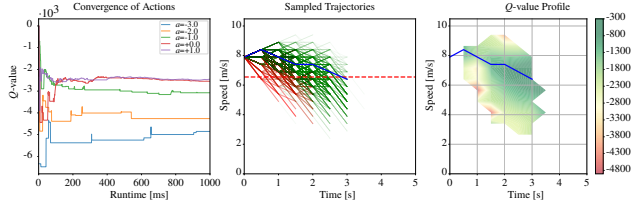


(b) $v_{\text{des, ego}} = 4.0\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$.

Figure A.2: Before the ego vehicle realizes it has misidentified the route intention; after 1.1 s.

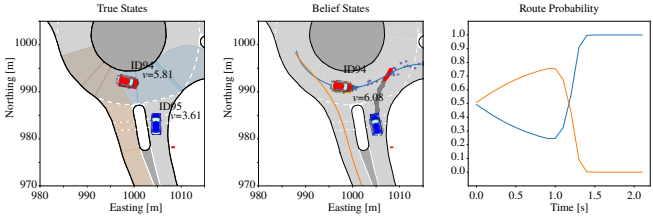


(a) Planned motion.

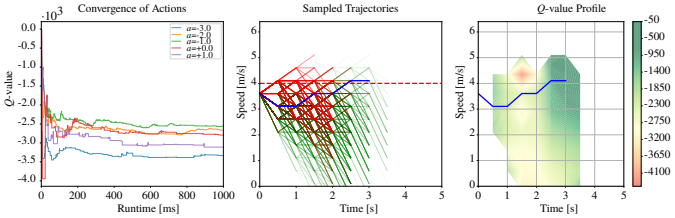


(b) Search tree.

Figure A.3: Results after 2.1 s for $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 5.0\,\text{m/s}$.

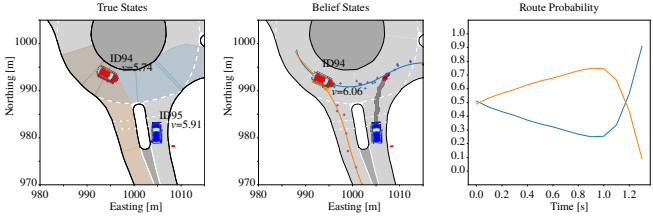


(a) Planned motion.

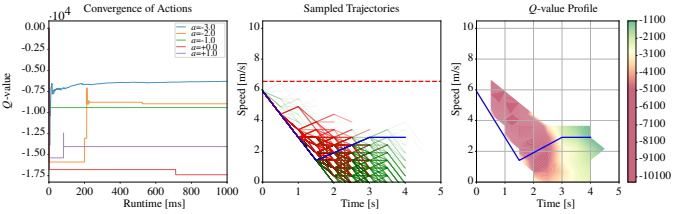


(b) Search tree.

Figure A.4: Results after 2.1 s for $v_{\text{des, ego}} = 4.0\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$.

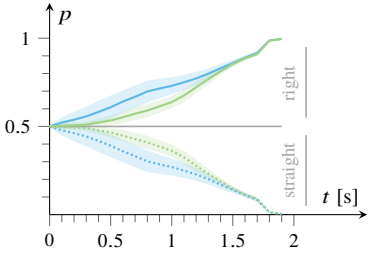


(a) Planned motion.

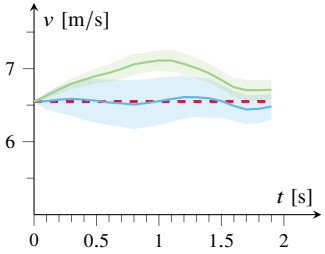


(b) Search tree.

Figure A.5: Harsh braking after identifying the correct route intention for conflicting v_{des} values, i.e. $v_{\text{des, ego}} = 6.55\,\text{m/s}$ and $v_{\text{des, }k} = 10.0\,\text{m/s}$; after 1.3 s.

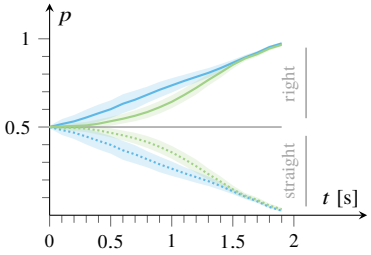


(a) Route probabilities.

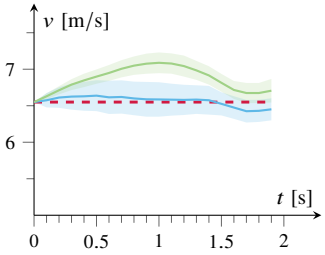


(b) Speed profile.

Figure A.6: Evolution of the predicted route probability and the speed profile over time while using geometric features for route probability identification. Particle set size is $m = 10$ and the behavior model of the vehicle is not reinforced. The blue line ( ) represents the results for the case with active information gathering, and the green line ( ) is for the case without information rewards. The dotted line ( ) in the route probability shows the probability of driving straight. The red dashed line ( ) in the speed profile shows the desired travel speed.



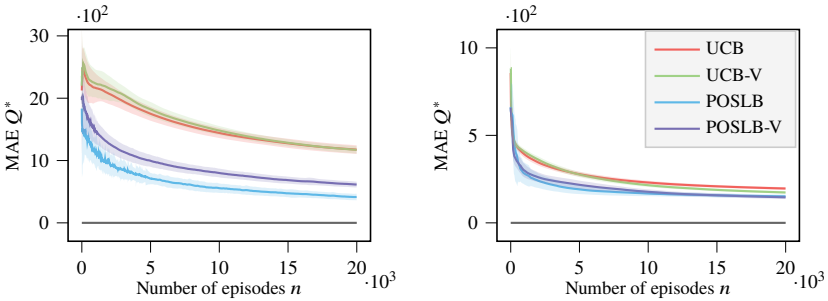
(a) Route probabilities.



(b) Speed profile.

Figure A.7: Evolution of the predicted route probability and the speed profile over time for a particle set size $m = 4$. Geometric features are not used for route probability identification and the behavior model of the vehicle is not reinforced.

A.3.2 Lipschitz Continuity of Rewards



(a) $S_{\texttt{I-Hi}}$. (b) $S_{\texttt{I-Lo}}$.

Figure A.8: Mean absolute error for 17 actions in intersection scenarios.

A.4 Parameter Values Used in the Evaluations

Scenario Parameters

The scenario used in Section 7.2.1 corresponds to the `DR_DEU_Roundabout_OF` scenario of InteractionDataset, track file number 0, timestamp 214000. The ego vehicle is the ID 95.

The parameters of the T-intersection scenario used in Section 7.2.2 for active information gathering evaluations are presented in Table A.1.

Vehicle	Distance to Merge Point	Speed
ego	15.97	6.55
other	22.37	8.33

Table A.1: Parameters of the T-intersection scenario used in active information gathering evaluations.

The parameters that define the criticality of the scenarios in the Lipschitz continuity evaluations are presented in Table A.2.

Scenario	Vehicle	Time-to-Intersection (s)			
$S_{\texttt{Coll}}$	ego	2.11			
	vehicle2	2.71			
$S_{\texttt{I-Lo}}$	ego	5.33	5.14	6.81	
	vehicle1	3.99	4.20	4.78	
	vehicle2	6.35	6.14	7.89	7.73
$S_{\texttt{I-Hi}}$	ego	2.66	2.28	5.63	
	vehicle1	2.78	3.23	4.52	
	vehicle2	3.42	3.05	6.21	5.92

Table A.2: Times until the merge point (MP) for vehicles in the Lipschitz continuity evaluations.

Solver Parameters

Parameters of the IPFT-based motion planner are presented in Tables A.3 and A.4.

Parameter	Value	Unit
m	10	-
γ	0.95	-
c	4000	-
d_{max}	10	-

Table A.3: Parameters of the IPFT used in Sections 7.2.1 and 7.2.2.

Parameter	Value	Unit
t_s	0.5	s
s_{min}	2	m
t_h	1.0	s
a^+	0.73	m/s^2
a^-_{cft}	−1.67	m/s^2
a^-	−7.0	m/s^2
σ_{s,ϑ_0}	0.1	m
σ_{v,ϑ_0}	0.2	m/s
σ_{IDM}	1.5	m/s^2
σ_{x,ϑ_0}	1.0	m
σ_{y,ϑ_0}	1.0	m
σ_{v,ϑ_0}	0.5	m/s
σ_{x,ϑ_0}	1.0	m
σ_{y,ϑ_0}	1.0	m
σ_{v,ϑ_0}	0.5	m/s
σ_{x,ϑ_k}	4.0	m
σ_{y,ϑ_k}	4.0	m
σ_{v,ϑ_k}	2.0	m/s
$\sigma_{\psi,\vartheta_k}$	5.0	°
$\sigma^{PF}_{s,\vartheta_k}$	1.0	m
$\sigma^{PF}_{s,\vartheta_0}$	4.0	m
σ^{PF}_{d}	0.75	m
$\sigma^{PF}_{\Delta\psi}$	10.0	°
σ^{PF}_{v}	3.0	m/s
$\sigma^{PF}_{v_r}$	3.0	m/s
w_{coll}	−10 000	-
w_v	−100	-
$w_{a,lat}$	−50	-

Table A.4: Scenario parameters used in Sections 7.2.1 and 7.2.2.

The solver parameters of the POMCP in the Lipschitz continuity evaluations are presented in Table A.5.

Parameter	Value	Unit
t_s	1	s
s_{goal}	15	m
s_{veh}	2	m
s_{min}	2	m
t_h	1.5	s
a^+	0.73	m/s^2
a^-_{cft}	−1.67	m/s^2
σ_a	3	m/s^2
$\sigma_{\boldsymbol{o},s}$	0.2	m
$\sigma_{\boldsymbol{o},v}$	1.0	m/s
d_{thresh}	1	m
w_{coll}	−10 000	-
w_v	−100	-
$w_{j,lon}$	−100	-
$w_{a,lat}$	−100	-
ω	0.77	-
γ	0.95	-
c	10 000	-
$\mathscr{L}$	2000	s^2/m
d_{max}	20	-

Table A.5: Parameters used in the Lipschitz continuity evaluations. The effect of σ_s and σ_v are covered with σ_a and the effect of σ_{IDM} is covered with $\sigma_{\boldsymbol{o},s}$ and $\sigma_{\boldsymbol{o},v}$.

$\lvert\mathcal{A}\rvert$	Straight	Curve	S$_{\texttt{Coll}}$	S$_{\texttt{I-Lo}}$	S$_{\texttt{I-Hi}}$
5	1247	1847	1003	1241	573
9	1271	2157	1981	1345	728
17	1336	2280	2742	1453	787
33	1370	2246	1260	1432	1033

Table A.6: Estimated Lipschitz constant $\mathscr{L}$ [s^2/m].

$\lvert\mathcal{A}\rvert$	Straight	Curve	S$_{\texttt{Coll}}$	S$_{\texttt{I-Lo}}$	S$_{\texttt{I-Hi}}$
5	−0.30	−2788.93	−4033.37	−3557.44	−6963.64
9	−12.52	−2695.64	−3950.14	−3283.98	−6639.67
17	−6.63	−2676.03	−3923.30	−3230.04	−6286.98
33	−2.03	−2726.60	−3908.22	−3249.59	−5927.48

Table A.7: Optimal Q-value.