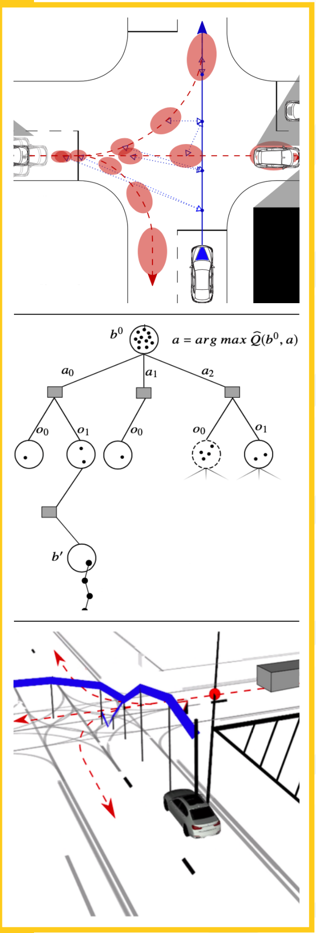


CONSTANTIN HUBMANN

Belief State Planning for Autonomous Driving

Planning with Interaction, Uncertain Prediction and Uncertain Perception

KIT Scientific Publishing

Constantin Hubmann

Belief State Planning for Autonomous Driving

Planning with Interaction, Uncertain Prediction
and Uncertain Perception

Schriftenreihe
Institut für Mess- und Regelungstechnik,
Karlsruher Institut für Technologie (KIT)

Band 048

Eine Übersicht aller bisher in dieser Schriftenreihe erschienenen Bände finden Sie am Ende des Buchs.

Belief State Planning for Autonomous Driving

Planning with Interaction, Uncertain Prediction and Uncertain Perception

by
Constantin Hubmann

KIT Scientific Publishing

Karlsruher Institut für Technologie
Institut für Mess- und Regelungstechnik

Belief State Planning for Autonomous Driving: Planning with Interaction, Uncertain Prediction and Uncertain Perception

Zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften von der KIT-Fakultät für Maschinenbau des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Constantin Hubmann, M. Sc.

Tag der mündlichen Prüfung: 8. November 2019
Hauptreferent: Prof. Dr.-Ing. Christoph Stiller
Korreferent: Prof. Dr. Mykel Kochenderfer

Impressum

KIT Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2021 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1613-4214
ISBN 978-3-7315-1039-0
DOI 10.5445/KSP/1000122855

Foreword

This thesis presents the results of my time as a PhD student, working in cooperation between the BMW Group and the Karlsruhe Institute of Technology (KIT). I would like to express my gratitude to the people who joined me on this path over the last years and contributed to my graduate career:

My sincere thanks go to Prof. Dr.-Ing. Stiller for the supervision of this thesis. I am especially thankful for his guidance and motivational support which made me follow my first ideas to the end. Further thanks go to all the PhD students of Prof. Dr.-Ing. Stiller which made every visit to the lab, summer seminars and conferences a fruitful experience.

A special thanks go to Prof. Dr. Kochenderfer of Stanford University for acting as co-examiner of this thesis. I feel honored having an expert in the field of decision making under uncertainty reviewing this thesis.

I also want to thank Prof. Dr.-Ing. Werner Huber at the BMW Group who initiated this thesis and gave me the chance to work on my research ideas. Furthermore, I want to greatly thank my supervisor at the BMW Group, Dr.-Ing. Daniel Althoff, for the many challenging discussions about my approaches, co-authoring of publications and general advice. I also want to thank my manager, PD Dr.-Ing. Moritz Werling, for many discussions and especially for creating a great, academic environment inside the BMW Group that enabled this thesis.

Furthermore, I want to thank the students I supervised and greatly enjoyed working with, Marvin Becker and Nils Quetschlich, for their dedication and effort.

I also want to greatly thank the other PhD students in my group, namely Jens Schulz, Christian Pek, Sascha Steyer, Kai Stiens and Branka Mircevska. This thesis would have not been possible without the coffee breaks, lunches and after work beers which led to endless motivational, technical and fun discussions. The time would have not been the same without you guys.

I also want to express my deep gratitude to my family, who always supported me during my education. Finally, my deepest thanks go to Annelie, for supporting me in every intense time and for sharing this journey.

Munich, August 2019 *Constantin Hubmann*

Abstract

This thesis presents a behavior planning algorithm for automated driving in urban environments with an uncertain and dynamic nature. The uncertainty in the environment arises by the fact that the intentions as well as the future trajectories of the surrounding drivers cannot be measured directly but can only be estimated in a probabilistic fashion. Even the perception of objects is uncertain due to sensor noise or possible occlusions. When driving in such environments, the autonomous car must predict the behavior of the other drivers and plan safe, comfortable and legal trajectories. Planning such trajectories requires robust decision making when several high-level options are available for the autonomous car.

Current planning algorithms for automated driving split the problem into different subproblems, ranging from discrete, high-level decision making to prediction and continuous trajectory planning. This separation of one problem into several subproblems, combined with rule-based decision making, leads to sub-optimal behavior.

This thesis presents a global, *closed-loop* formulation for the motion planning problem which intertwines action selection and corresponding prediction of the other agents in one optimization problem. The global formulation allows the planning algorithm to make the decision for certain high-level options implicitly. Furthermore, the *closed-loop* manner of the algorithm optimizes the solution for various, future scenarios concerning the future behavior of the other agents. Formulating prediction and planning as an intertwined problem allows for modeling interaction, i.e. the future reaction of the other drivers to the behavior of the autonomous car.

The problem is modeled as a partially observable Markov decision process (POMDP) with a discrete action and a continuous state and observation space. The solution to the POMDP is a policy over belief states, which contains different reactive plans for possible future scenarios. Surrounding drivers are modeled with interactive, probabilistic agent models to account for their prediction uncertainty. The field of view of the autonomous car is simulated ahead over the whole planning horizon during the optimization of the policy. Simulating the possible, corresponding, future observations allows the algorithm to select actions that actively reduce the uncertainty of the world state. Depending on the scenario, the behavior of the autonomous car is optimized in (combined lateral and) longitudinal direction. The algorithm is formulated in a generic way and solved *online*, which allows for applying the algorithm on various road layouts and scenarios.

While such a generic problem formulation is intractable to solve exactly, this thesis demonstrates how a sufficiently good approximation to the optimal policy can be found *online*. The problem is solved by combining state of the art Monte Carlo tree search algorithms with near-optimal, domain specific roll-outs.

The algorithm is evaluated in scenarios such as the crossing of intersections under unknown intentions of other crossing vehicles, interactive lane changes in narrow gaps and decision making at intersections with large occluded areas. It is shown that the behavior of the *closed-loop* planner is less conservative than comparable *open-loop* planners. More precisely, it is even demonstrated that the policy enables the autonomous car to drive in a similar way as an omniscient planner with full knowledge of the scene. It is also demonstrated how the autonomous car executes actions to actively gather more information about the surrounding and to reduce the uncertainty of its belief state.

Kurzfassung

Diese Arbeit stellt einen neuen Ansatz für die Verhaltensgenerierung automatisierter Fahrzeuge in dynamischen, urbanen Umgebungen vor. Der Fokus der Arbeit liegt im Besonderen auf der Berücksichtigung von Unsicherheiten die in einem urbanen Umfeld vorkommen. Diese Unsicherheiten existieren, da die Intention der anderen Fahrer, ihr individuelles Fahrverhalten sowie mögliche Interaktionen mit dem autonomen Fahrzeug nicht deterministisch sondern nur probabilistisch vorhergesagt werden können. Zudem ist die Wahrnehmung der anderen Verkehrsteilnehmer durch die Sensorik zumindest Messrauschen unterworfen, kann aber auch aufgrund von Verdeckungen unvollständig sein.

Bisherige Ansätze zur Verhaltensgenerierung für das automatisierte Fahren lösen das Problem durch eine Aufteilung in verschiedene Teilprobleme: die Entscheidungsfindung für eine bestimmte Fahroption auf höchster Ebene, die Prädiktion der anderen Fahrer sowie die Planung einer kontinuierlichen Trajektorie. Diese Aufteilung des Problems, sowie die Verwendung regelbasierter Ansätze zur Entscheidungsfindung, führt in vielen Fällen zu suboptimalem Fahrverhalten.

Diese Arbeit präsentiert einen global optimalen *Closed-Loop* Ansatz, der das Auswählen einer Aktion des autonomen Fahrzeuges sowie die Prädiktion der anderen Verkehrsteilnehmer in einer gekoppelten Problemformulierung beschreibt. Die globale Formulierung erlaubt hierbei, dass Entscheidungen für Fahroptionen auf höchster Ebene implizit als Teil des Planungsproblems getroffen werden können. Der *Closed-Loop* Ansatz optimiert ausserdem das Verhalten des autonomen Fahrzeuges für mehrere, mögliche zukünftige Szenarien bezüglich des Verhaltens der anderen Verkehrsteilnehmer. Die kombinierte Formulierung der Planung für das autonome Fahrzeug sowie der Prädiktion für die anderen Verkehrsteilnehmer erlaubt die Modellierung von Interaktion. Dies bedeutet, dass die Reaktion der anderen Fahrzeuge auf das Verhalten des autonomen Fahrzeugs bei der Verhaltensplanung bereits berücksichtigt wird.

Das Problem ist als teilweise beobachtbarer Markov Entscheidungsprozess (POMDP) auf einem kontinuierlichen Zustands- und Beobachtungsraum mit diskreten Aktionen modelliert. Diese Formulierung wird durch eine Policy gelöst, welche reaktive Aktionen für zukünftige Ereignisse enthält. Das unbekannte, zukünftige Verhalten der Fahrer in der Umgebung des autonomen Fahrzeuges wird mit Hilfe von probabilistischen, interaktiven Fahrermodellen beschrieben. Das Sichtfeld des autonomen Fahrzeuges wird während der Optimierung der Policy über den kompletten Planungshorizont simuliert. Ebenso werden mögliche zukünftige Messungen des autonomen Fahrzeuges simuliert, was dem Algorithmus erlaubt Aktionen zu wählen, welche die Unsicherheit des Weltzustandes aktiv minimieren. Der Algorithmus optimiert das Verhalten je nach Modellierung nur in longitudinaler oder zugleich auch in lateraler Richtung Eine generische Problemformulierung sowie dessen Lösen zur Laufzeit erlauben einen Einsatz des Algorithmus in vielfältigen Szenarien.

Diese generische Problemformulierung exakt zu lösen ist nach gegenwärtigem Stand der Forschung nicht möglich. Dennoch zeigt diese Arbeit wie eine ausreichend gute Approximation der optimalen Lösung sogar während der Laufzeit (*online*) gefunden werden kann. Dies ist möglich indem hochmoderne, stochastische Verfahren (Monte Carlo Baum Suche) mit spezifischen Heuristiken des jeweiligen Problems kombiniert werden.

Der Algorithmus wird vielfach in der Simulation evaluiert. Dies geschieht in Szenarien mit kreuzendem Verkehr mit verschiedenen, möglichen Intentionen, interaktiven Spurwechseln in sehr kleine Lücken sowie Szenarien mit grossen Sensorverdeckungen an Kreuzungen. Es wird gezeigt, dass der *Closed-Loop* Planungsansatz ein weniger konservatives Verhalten ermöglicht als vergleichbare *Open-Loop* Planer. Ausserdem wird gezeigt, dass die Policy nahezu ein Fahrverhalten ermöglicht, welches ansonsten nur mit einem allwissenden Planer erreicht werden kann. Zudem wird gezeigt, dass der Algorithmus in der Lage ist, aktiv Aktionen zu wählen, die die Unsicherheit des aktuellen Zustandes reduzieren.

Contents

Foreword . . . . **i**

Abstract . . . . **iii**

Kurzfassung . . . . **v**

Notation and Symbols . . . . **xi**

1 Introduction **1**
1.1 Motivation: Motion Planning Under Uncertainty 4
1.2 Related Work: Motion Planning 6
1.2.1 Properties of Planning Algorithms 7
1.2.2 Consideration of Constraints 8
1.2.3 Graph Search for Trajectory Planning 8
1.2.4 Probabilistic Search for Trajectory Planning 10
1.2.5 Variational Trajectory Planning 11
1.2.6 Trajectory Planning for Autonomous Vehicles 12
1.3 Related Work: Motion Planning Architectures 14
1.3.1 Non-Interactive Planning with Given Prediction . . . 16
1.3.2 (Interactive) Planning with Given Maneuvers 19
1.3.3 Optimizing Interactive Maneuvers 21
1.4 Motion Planning with Policies 23
1.4.1 Open-Loop Planning 23
1.4.2 Closed-Loop Planning 23
1.4.3 Definition of Policy Optimization 24
1.5 Closed-Loop Behavior Planning Under Uncertainty 25
1.6 Contributions and Outline 27

2 Background **31**
2.1 Planning with Deterministic Models 31

2.2 Planning with Probabilistic Models 33
2.3 Planning with State Uncertainty 34
2.3.1 Complexity of Solving POMDPs 35
2.3.2 Solving POMDPs 36
2.3.3 The Simplified QMDP Formulation 40
2.3.4 Policy Optimization: Online vs Offline 40
2.4 Solving POMDPs in this Thesis 41
2.4.1 Monte Carlo Tree Search 42
2.4.2 MCTS for POMDPs 43
2.4.3 UCT Action Selection 45
2.4.4 Belief State Tracking and Observation Clustering . . 46
2.4.5 Calculating Optimized Roll-Outs 47
2.4.6 Creating Consistent Plans 48
2.4.7 Batch Sampling of Episodes 49
2.5 Reducing the Dimensionality of the Action Space 50

3 Planning for Combinatorial Decision Making 51
3.1 Related Work 52
3.2 Problem Formulation 53
3.3 Approach 54
3.3.1 Transition Model 55
3.3.2 Cost Function 56
3.3.3 Domain Specific Heuristics 61
3.3.4 Goal State Formulation 61
3.3.5 Implementation 62
3.4 Results 63
3.4.1 Performance 63
3.4.2 Qualitative Simulation Scenario 63
3.5 Summary 66

4 Planning with Uncertain Intentions of Crossing Traffic . 67
4.1 Related Work 69
4.2 Problem Formulation 71
4.3 Approach 73
4.3.1 State Space 73
4.3.2 Action and Transition Model 74
4.3.3 Reward Model 76
4.3.4 Observation Model 76

4.3.5 Implementation 78

4.4 Results 79

4.4.1 Convergence 79

4.4.2 Policy Behavior Planning 82

4.5 Summary 88

5 Coupled 2D Planning for Interactive Merging 91

5.1 Related Work 93

5.1.1 Gap Assessment Algorithms 93

5.1.2 Planning-Based Algorithms 94

5.2 Approach 95

5.2.1 State Space 96

5.2.2 Action and Transition Model 97

5.2.3 Motion Model of Surrounding Agents 98

5.2.4 Observation Model 99

5.2.5 Reward Model 100

5.2.6 Learned Yielding Model 101

5.2.7 Implementation 103

5.3 Results 104

5.3.1 Analysis of Belief State Policy 104

5.3.2 Online Simulation 106

5.4 Summary 108

6 Planning under Sensor Occlusions 109

6.1 Related Work 111

6.2 Approach 113

6.2.1 State Space 113

6.2.2 Observation Model 114

6.2.3 Representation of Phantom Vehicles 116

6.2.4 Action and Transition Model 117

6.2.5 Reward Model 119

6.2.6 Implementation 120

6.3 Results 120

6.3.1 Static Occlusion 120

6.3.2 Dynamic Occlusion 124

6.3.3 2D Motion Primitives 127

6.4 Summary 128

7 Conclusion **129**
7.1 Future Research Directions 131

Bibliography **133**

Notation and Symbols

Abbreviations

ABT	Adaptive Belief Tree
ACC	Adaptive Cruise Control
BFS	Breadth-First Search
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
DARPA	Defense Advanced Research Projects Agency
DESPOT	Determinized Sparse Partially Observable Tree
DFS	Depth-First Search
DNN	Deep Neural Net
DQL	Deep Q-Learning
DRL	Deep Reinforcement Learning
FoV	Field of View
HSVI	Heuristic Search Value Iteration
ICS	Inevitable Collision States
IDM	Intelligent Driver Model
IMM	Interacting Multiple Model
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MIP	Mixed Integer Programming
MIQP	Mixed Integer Quadratic Programming
MOMDP	Mixed Observability Markov Decision Process
MPC	Model Predictive Control
MPDM	Multi Policy Decision Making
PBVI	Point-Based Value Iteration
PCLRHC	Partially Closed-Loop Receding Horizon Control
PGM	Probabilistic Graphical Model
POMCP	Partially Observable Monte Carlo Planning
POMDP	Partially Observable Markov Decision Process
PRM	Probabilistic Road Maps

QMDP	Fully Observable Value Approximation
QP	Quadratic Programming
RL	Reinforcement Learning
RRT	Rapidly Exploring Random Trees
SARSOP	Successive Approximations of the Reachable Space under Optimal Policies
SQP	Sequential Quadratic Programming
TAPIR	Toolkit for Approximating and Adapting POMDP Solutions in Real Time
UCT	Upper Confidence Bound for Trees

Symbols

General

$\Vert \cdot \Vert, \Vert \cdot \Vert_2$	Euclidean norm of a vector
$\lvert \cdot \rvert$	cardinality of a set/absolute value of a scalar
$\mathrm{E}[\cdot]$	expected value
$\mathcal{N}$	Normal distribution
$P(\cdot)$	probability
$P(\cdot \mid \cdot)$	conditional probability
$f(\cdot)$	deterministic function
$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
$(\cdot)^*$	optimal value
$\widehat{(\cdot)}$	estimated value
$(\cdot)(t)$	value $(\cdot)$ at continuous time t
$(\cdot)^{(t)}$	value $(\cdot)$ at discrete time t

Planning

$\mathcal{C}_{\text{config}}$	configuration space of the robot
$\mathcal{C}_{\text{free}}$	free space of the robot
N_k	agent k in the environment
p_k	path of agent k

$\mathcal{P}_k$	set of path hypotheses of agent k
r_k	a certain route in the topological map
$\mathcal{R}$	set of all routes in the topological map
m_k	a high-level maneuver of agent k
$\mathcal{M}_k$	set of available maneuvers of agent k
ξ_k	trajectory of agent k
$\tilde{\xi}_k^{t_0:T}$	set of predicted trajectories of agent k for $[t_0, T]$
$v_{\mathrm{des}}(s)$	reference velocity on the path at position s
$\kappa(s)$	road curvature at position s
J	cost function
a_k	action of agent k
x	state of the environment
$h(x)$	heuristic estimate of future costs starting at x
s_k	longitudinal position of agent N_k on its path
v_k	longitudinal velocity of agent N_k on its path
l_k	lane of agent N_k
d_k	lateral position of agent N_k
m_k	interaction friendliness of N_k
g_l	existence state of phantom car in a occlusion
Ψ_l	field of view on the path of phantom car N_l
t	time
t_{hor}	planning horizon

POMDP

b, $b(x)$	belief state, probability of being in state x
$\pi(\cdot, a)$	policy, mapping an action a on a (belief) state $(\cdot)$
$R(x, a)$	reward for choosing action a in state x
$T(x, a, x')$	probability of traversing to x' after choosing a in x
$Z(x', a, o)$	probability of observing o when in x' after choosing a
$V(\cdot)$	value of a certain (belief) state $(\cdot)$
$Q(\cdot, a)$	expected value when executing a in a (belief) state $(\cdot)$
γ	discount factor of the rewards
c	UCT factor balancing exploration and exploitation
α_a	vector representing the reward $R(\cdot, a)$ for every state x

ABT

$\mathcal{T}$	belief tree
u	A sampled episode in the belief tree
$U(b,a)$	The set of all episodes which select a in b
$U(b)$	The set of all episodes passing b
n	The depth in the belief tree
$o_{\max}$	Maximum observation distance of one cluster

1 Introduction

The transportation industry faces the biggest change in its history: the automation of vehicles. Fully autonomous systems exist to date only in closed, structured environments, such as factories and manufacturing cells. Nowadays, academia and industry work closely together on the transfer of such autonomous systems to public environments [36]. This is the case for small, unmanned delivery robots, autonomous vehicles and even aerial vehicles such as drones.

Especially, automated vehicles are in the focus as they are currently of high interest to the industry. The capabilities of advanced driver assistance systems are enhanced over the years to continually improve safety and comfort (see [88] for a definition of the different levels of automation). These systems extend the degree of automation but still rely on a human driver to bear responsibility. Nonetheless, completely automated systems are now on the verge of becoming reality in small, geo-fenced areas [64] and are considered to have game changing capabilities for the transportation industry.

The largest desired effect is hereby a expected, potential decline of accidents and fatalities. This is the case as 94% of all accidents in the United States are attributed to driver errors, with recognition (41%) and decision error (33%) being the most critical reasons [97]. Such a drastically reduced amount of accidents, combined with a reduced fuel consumption and optimized traffic management may reduce the costs of operating a car. This will provide access to individual transportation possibilities for people without driving license or for elderly, intoxicated or disabled persons who are physically unable to drive [78]. Especially, ride-hailing services may start to provide on-demand transportation in inner cities at a reduced rate compared to common taxi services. This is due to reduced operating costs because of spared drivers [76].

It is of major importance how such autonomous systems behave in urban traffic. The generated behavior must not only comply with traffic rules but also be comfortable to gain the trust of the potential passengers [49]. Furthermore, such cars must guarantee a certain degree of safety to be accepted by society and regulating authorities. At the same time, autonomous vehicles cannot drive too conservatively without creating frustration among surrounding human drivers [26] or even getting completely stuck as they do not dare to move in scenarios with a high degree of uncertainty [103]. To design such a system, different problems have to be tackled.

A **perception system** records raw sensor data of the environment. This sensor data is processed to detect static and dynamic objects. In a second step, the dynamic objects are tracked over time. The sensor data of the perception system may also be used to localize the robot in the environment.

A **prediction system** uses the information about the tracked objects as input and provides their predicted future behavior.

The **motion planner** is responsible for guiding the autonomous car in a safe, comfortable and legal way through the traffic. It is a crucial part of such an autonomous system as it must cope with the accumulated uncertainties of the previous layers while it must present a safe and comfortable plan at the same time. This means, the planner must be able to act in semi-structured, dynamic and uncertain environments. The environment is semi-structured

Figure 1.1: Left: ‘Electricity may be the driver’, advertisement of the *Central Power and Light Company* in 1956 (graphic from [108]) . Right: Nonetheless, it took until 2015, that vehicles, being rigorously designed for autonomous driving, were hitting urban roads for testing passenger rides (graphic from [1], ©Waymo).

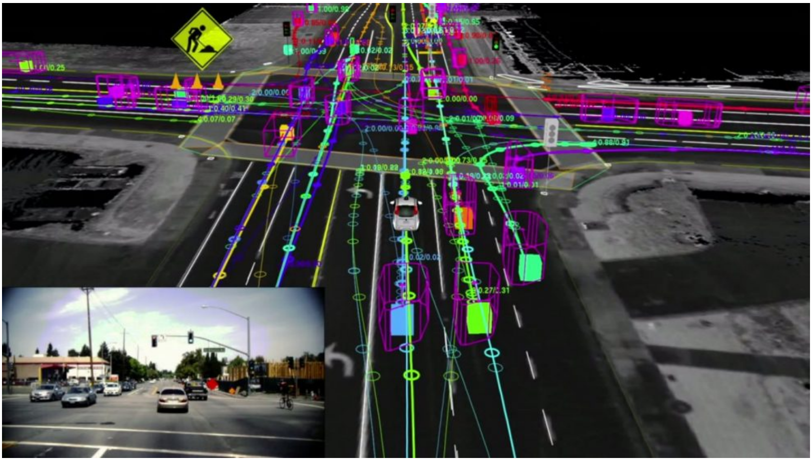


Figure 1.2: A typical intersection as seen by an autonomous vehicle. Future trajectories of the other vehicles are not known, but possible hypotheses can be made in a probabilistic fashion (graphic from [106], ©Waymo).

because of the fact that other dynamic agents move on predefined entities such as lanes, pavements, etc. Nonetheless, topological maps may be outdated and the motion of the other agents is not necessarily limited to these entities. The uncertain nature of the environment arises because of the limitations (range, non-observable states) and noise in the perception system. Therefore, the location of the autonomous car as well as the future trajectories of the surrounding traffic can only be estimated in a probabilistic fashion. Especially, the uncertain future behavior of the surrounding traffic poses a challenge for generating sensible behavior for the autonomous car. This is the case as there are numerous possible future scenarios which must be considered by the autonomous vehicle. For example, it cannot be determined if another car will drive straight or turn right at an intersection. figure 1.2 shows an example of the variety of possible future trajectories of a real-world scenario.

The goal of this thesis is to develop a new behavior planning algorithm for autonomous vehicles in urban environments.

The underlying idea of this work is, that generating an optimal behavior for the autonomous car and predicting the uncertain future behavior of the other agents is a coupled problem that must be modeled in a coupled manner to generate optimal behavior. While this results in a very hard problem formulation, the goal of this thesis is to present an *online* algorithm for this problem.

The further introduction is structured as follows. At first, section 1.1 presents the different uncertainties that arise in urban environments. In the following, section 1.2 formally defines the problem of motion planning followed by an overview of the state of the art in the field. Subsequently, section 1.3 gives an overview of different planning architectures which can be used to generate a behavior for the autonomous car. Section 1.4 gives an introduction about the advantages of using policies in motion planning instead of trajectories. In section 1.5, the main idea of this thesis is described. The last section of this chapter, section 1.6, describes the main contributions as well as the outline of the whole thesis.

1.1 Motivation: Motion Planning Under Uncertainty

A motion planning algorithm for urban traffic scenarios must cope with the uncertainty of various possible future scenarios. This uncertain prediction of the other drivers is modeled in this work as described in the following (see figure 1.3 for an illustration).

At first, the intended path to follow of the other vehicle is not known but can only be estimated in a probabilistic fashion. This is referred to as *unknown intention* of the other agents. Secondly, assuming that the path of the other vehicle is known, the motion on the path is dependent on the style of the respective driver which is, again, unknown. This *uncertainty* is described by a *probabilistic driver model*. Additionally, the potential influence of the future motion of the autonomous car on the behavior of the other agent must be modeled. This interplay is probabilistic (e.g. other vehicles yield to the autonomous car or not) and is denoted as *interaction* throughout this thesis. Additionally, the uncertain measurements of the configuration of the other vehicles are referred to as *sensor uncertainty*, while the uncertainty, describing if other vehicles can be perceived at all, is denominated as *occlusion uncertainty*.

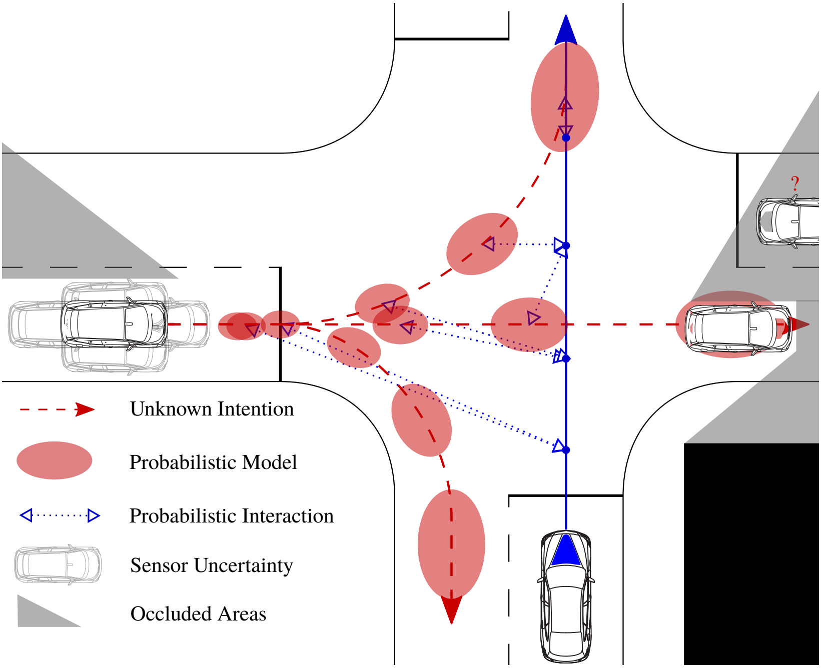


Figure 1.3: The planned path of the autonomous vehicle is depicted in blue, while the possible motion hypotheses of the other car is drawn in red. Planning the motion for the autonomous car requires us to take various uncertainties into account. This is at first the unknown behavior of the other agents (due to their unknown intention, probabilistic driver models and uncertain interaction with the autonomous car). Additionally, sensor noise as well as the existence probability of possibly occluded objects must be respected [124].

Motion planning algorithms must consider prediction uncertainties to plan safe and comfortable trajectories. The motion prediction of the other drivers can be represented in various ways.

The future behavior of the other agents can be presented by **trajectories** which are either learned from data or modeled from human experience. This approach is not capable of respecting every possible, future behavior, even if sets of possible future trajectories are considered. Therefore it does not guarantee safety.

Probability density functions can be used to describe the probability of possible future configurations. While this allows for a precise modeling, describing these probability distributions can be difficult. This is the case, as strongly non-Gaussian distributions are difficult to model. Gaussian distributions on the other hand are often unable to describe the real distribution and also introduce so-called long tails. To overcome the problem of long tails, chance constraints are often used to cut off the unlimited Gaussian distribution at a certain point.

Reachability Analysis calculates an over approximated set of possible future vehicle configurations [2]. The approach allows us to guarantee safety of an evaluated trajectory, given assumptions on the worst case behavior of the other agents. Nonetheless, it may result in conservative trajectories as the reachable set grows drastically over simulated time when future observations are not considered.

1.2 Related Work: Motion Planning

The general problem of motion planning is to generate a possible path p_0 or trajectory ξ_0 from a given start state x_{start} to a goal state x_{goal} with $x \in \mathcal{X}$. The robot can traverse from one state to the other by using a certain control action $a \in \mathcal{A}$. The trajectory must consider the dynamic and kinematic constraints of the robot N_0 while respecting constraints of the environment such as static and dynamic obstacles $N_{1:K}$. A potential trajectory of the autonomous vehicle, ξ_0, is evaluated by a cost function J which may be denoted as a weighted sum of different measures such as total acceleration/jerk and collisions.

The goal of an optimal planning algorithm is to find the optimal trajectory ξ_0^*, defined as

$$\xi_0^* := \arg\min_{\xi_0} \int_0^{t_{\text{goal}}} J(x(t), a(t))\underset{.}{t}, \tag{1.1}$$

for given system dynamics $\dot{x}(t) = f(x(t), a(t))$ and inequality and equality constraints:

$$h_i(x(t), a(t)) \leq 0, \text{for } i \in [1, \ldots, m], m \in \mathbb{N}_0 \tag{1.2}$$

$$g_j(x, a) = 0, \text{for } j \in [1, \ldots, n], n \in \mathbb{N}_0. \tag{1.3}$$

In the context of autonomous driving, constraints such as speed limits, traffic rules (e.g. traffic lights), lane boundaries, static objects and dynamic objects must be considered. Respecting dynamic objects is nontrivial as their future trajectory is not known (probabilistic prediction) and may also depend on the executed trajectory of the autonomous vehicle (interaction). This is the case as the different agents cannot be considered as independent which makes it a coupled problem.

The problem of optimally considering the uncertainty of future states can be addressed by planning in the space of policies instead of in the space of trajectories. An introduction to the planning of policies instead of trajectories is given in section 1.4.

In the following, general characteristics of motion planning algorithms are introduced first. This is followed by on overview of the state of the art in motion planning.

1.2.1 Properties of Planning Algorithms

Motion planning algorithms can be described by several different characteristics. The most common ones are shortly reviewed in the following.

Optimality - The algorithm guarantees to find the *global optimal* solution if it exists. This is opposed to *local* algorithms, which find one local optimum, depending on their initial solution.

Completeness - The algorithm guarantees to find a solution if one exists. This can be relaxed to *Resolution Completeness* for grid-based planners, where the planner is guaranteed to find the solution if the underlying grid cells are sufficiently small. It can also be relaxed to *Probabilistic Completeness* which assures that the probability to find a solution converges to one over runtime. This is for example the case for many sampling based techniques.

Anytime - *Anytime* algorithms find an initial solution first and improve it over time as long as further runtime is given. This is often the case for sampling based algorithms such as Monte Carlo algorithms.

Online - An algorithm which is able to find a solution *online*, i.e. during runtime. This allows the robot to not have an a priori plan for every possible scenario, but to solve the current situation when it occurs. It is desired to have a planner which computes the solution *online*, s.t. it can account for changes in the environment [94] (see section 2.3.4 for a detailed description of the advantages).

1.2.2 Consideration of Constraints

Furthermore, motion planning algorithms can be distinguished in terms of what kind of constraints are incorporated in the planning problem.

Dynamic constraints - The dynamics of the autonomous robot are considered in the formulation by expressing them in the constraint equalities.

Kinematic constraints - Constraints concerning the degrees of freedom in the movement of an autonomous robot. Of special interest in the context of autonomous mobile robots are holonomic constraints. A holonomic robot has only holonomic constraints, i.e. equality constraints based on coordinates and time but no time derivatives. This allows the robot to move in a certain direction independently of the current velocity. A non-holonomic robot has inequality constraints which include time derivatives of the coordinates. A standard car is for example a non-holonomic system as its capability to move in lateral direction is dependent of its longitudinal velocity [35].

Kinodynamic constraints - A motion planning algorithm which considers *dynamic* as well as *kinematic* constraints is referred to as kinodynamic planner.

Topologic and Traffic Rule Constraints - Further constraints may arise in the area of autonomous vehicles when the topological map and traffic rules must be considered.

1.2.3 Graph Search for Trajectory Planning

Search based motion planning algorithms (also called geometric motion planning algaorithms) aim to find the optimal trajectory by searching on a constructed graph. This allows for global, non-convex optimization due to the combinatorial nature of the algorithms, but also requires some sort of dis-

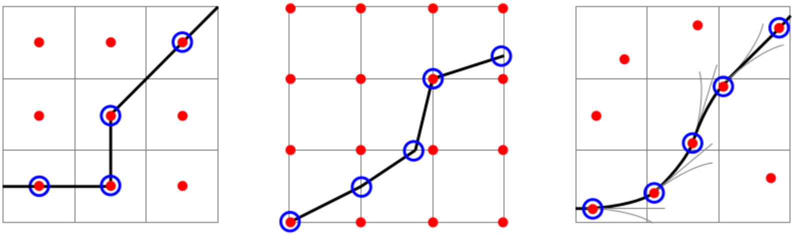


Figure 1.4: Comparison of different motion planning algorithms for grid based search (graphic from [73]). A^* (left) connects the center of cells which does not allow for considering kinematic constraints of the vehicle. While *Field* D^* [30] allows to plan smoother paths/trajectories by interpolating on the edges of the corners of the grid, kinematic constraints can still not be considered. The hybrid A^* (right) uses motion primitives to account for the kinematic constraints of the robot and assigns them to the related grid cells [73]. While this reduces the size of the search tree, the algorithm is not guaranteed to find the optimal solution anymore.

cretization (state or action discrete). The respective algorithms can be distinguished by the search algorithm itself and the way the graph is constructed.

Grid based motion planners generate the graph by discretization of the configuration space C_{config} of the robot with a grid first. By assuming, that a transition between neighboring cells is possible, the grid may be searched with graph search algorithms such as Breadth-First Search (BFS), Depth-First Search (DFS), the well-known Dijkstra algorithm [25] and its heuristic based extension A^* [86]. As grids are not able to account for the kinematic constraints of the robot, new search algorithms were introduced to find smoother trajectories (*Field* D^* [30]) or to even account for kinematic constraints by sacrificing optimality [73]. The different ideas are sketched in figure 1.4.

As the number of grid cells grows dramatically when the kinematics of the robot must be considered, another idea is to either create a state-lattice which is constructed with feasible motion primitives in a way, such that smooth transitions at every state are guaranteed (see figure 1.5) [82].

Instead of creating this state-lattice beforehand, it may also be constructed *online* during search by expanding only the required nodes [83]. The growth of the graph can be reduced by truncating branches whose estimated remaining costs are too high. The costs can be estimated by heuristics which represent a lower bound on the future costs. Typical heuristics for autonomous driving are

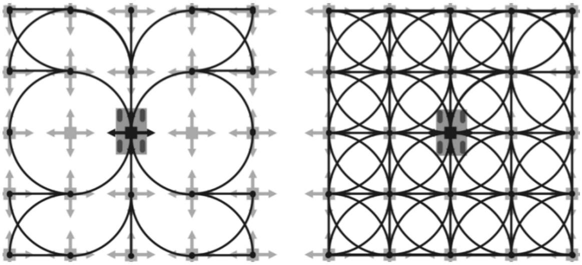


Figure 1.5: An example of a constructed state-lattice. The motion primitives are chosen in a way that continuous transitions from one motion primitive to another are always possible (graphic from [82]).

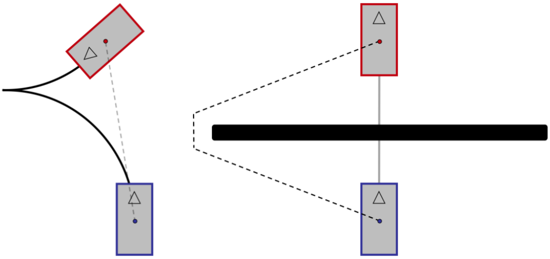


Figure 1.6: Two different heuristics for planning a trajectory to the goal configuration (blue). Either the non-holonomic characteristic of the car is considered (left) or potential obstacles are considered (right) (graphic from [68]).

presented in figure 1.6. Further possibilities to construct a search-graph are for example Voronoi diagrams, visibility graphs and cell decomposition [61].

1.2.4 Probabilistic Search for Trajectory Planning

Another possibility to construct a search graph is to use various sampling techniques to cover the configuration space C_{config}. These algorithms may provide only probabilistic completeness [61].

The idea of Probabilistic Road Maps (PRM) is to sample possible states from the configuration space C_{config} and test if they have a potential collision with obstacles. If the state is collision free, it is connected with a local planner to a neighboring state, given that a certain distance metric is fulfilled. This is done until the resulting graph efficiently covers the search space or until the maximum runtime is reached. The resulting graph may then be searched in a second step with one of the graph search algorithms presented in section 1.2.3. Potentially existing non-holonomic constraints of the robot may hereby be considered by a subsequent local planning stage.

Another popular algorithm is the Rapidly Exploring Random Trees (RRT) algorithm [62]. Instead of creating the graph at arbitrary sampled positions in the configuration space C_{config}, a tree is grown by steering it in the most undiscovered areas of the C_{config} by sampling. By sampling uniformly in the state space coordinates, the probability of sampling a state in a certain area is proportional to the size of its Voronoi region. In other words, sparsely explored areas in the state space are more likely to be sampled during the construction of the graph.

1.2.5 Variational Trajectory Planning

The goal of variational approaches is to formulate the problem with a convex cost functional. If such a convex problem formulation is possible, it allows to solve the problem on a continuous state space by use of various gradient descent methods. In general, the advantage of variational approaches is, that these formulations can be solved very fast and the solution is continuous. Nonetheless, because of the convex approximation of the problem, only a local minimum is found. This local minimum is closest to an initial solution (e.g. a reference path, reference trajectory). Therefore, to find the global optimal solution, it must be ensured that the initial solution is close enough to the optimal solution [61].

Despite the local nature of the algorithms, they often can be extended to find the global optimum by being parameterized for different minima. This is for example the case for the Mixed Integer Programming (MIP) extension of convex optimization algorithms such as Quadratic Programming (QP).

A well-known variational motion planning algorithm is the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) algorithm [119]. It can be used for optimizing paths and trajectories locally. CHOMP uses functional gradient techniques to optimize smoothness and collision avoidance simultaneously.

Another popular variational approach for path and trajectory planning is presented in [89]. The approach uses a sequential convex optimization formulation for the planning of collision-free trajectories. The sequential nature of the algorithm penalizes collisions with a hinge loss in an inner loop and uses an outer loop to increase the penalty coefficients if necessary.

1.2.6 Trajectory Planning for Autonomous Vehicles

This section reviews domain specific trajectory planning algorithms for autonomous driving. In this context, trajectory planning algorithms typically generate a trajectory which is local or global optimal on a certain time horizon (typically $3\,\mathrm{s} - 10\,\mathrm{s}$) or a spatial length. A local optimal trajectory refers hereby to a trajectory describing a local minimum in the cost function. The trajectory planner has no information about the long-term navigation goal but is parameterized continuously by a higher layer. The main focus of the trajectory planner is to optimize a reference trajectory or reach a goal state while optimizing comfort and respecting constraints [78].

In the following, popular trajectory planning algorithms are presented. The interested reader is referred to the surveys [36, 78] for a broader overview about motion planning algorithms for autonomous driving.

Global trajectory planners:

In [29], different geometric splines are used for creating simple longitudinal and lateral motion patterns. The resulting trajectories are evaluated and, depending of a rule-based decision maker, combined with other possible longitudinal profiles. As soon as a sufficiently good trajectory is found, it is tracked by a motion controller.

A resolution complete, optimal trajectory planning method is presented in [112]. Kinematically feasible trajectories are sampled on the Frenet frame by use of quintic polynomials. In a second step, the trajectory candidates of the created manifold are evaluated given a certain cost function. This leads to a jerk-optimal solution, given the discretization of the sampled polynomials.

A spatio-temporal state lattice is used by [116] to plan trajectories in on-road driving scenarios with dynamic obstacles. The resulting trajectories are based on quintic polynomials and are second-order continuous.

Local trajectory planners:

The variational algorithm, Sequential Quadratic Programming (SQP) is used in [115] to generate continuous trajectories. While the method is local, constraints for static and dynamic objects are introduced in a way such that the resulting solution is often globally optimal.

Another variational approach is presented in [38]. A local trajectory is generated by use of a linear time-varying Model Predictive Control (MPC), which is formulated as a QP. The algorithm needs an initial solution which is improved by a gradient descent algorithm. This guarantees to find a kinematically feasible, locally optimal solution very fast.

Another QP formulation is used by the authors of [74] which use the QP problem formulation at first to generate a longitudinal speed profile. In a second step, the speed profile is also optimized in lateral direction. This separation between longitudinal and lateral optimization allows on the one hand for two simple problem formulations, but constrains the solution space on the other hand.

The presented algorithms in this section allow to plan trajectories from a start state x_{start} to a goal state x_{goal}. While the formulation of these problems often allows to even consider the kinodynamic constraints of the robot, they lack the possibility of explicitly considering interaction and the uncertainties of real-world environments. This is due to their simplified problem formulations which are used to make the problem either tractable at all or to represent the trajectory in a continuous fashion.

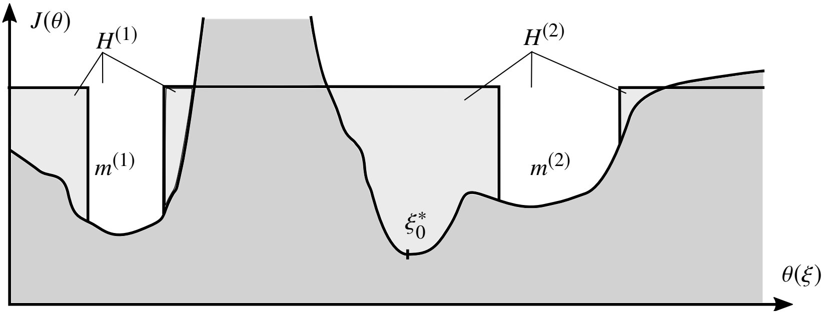


Figure 1.7: Schematic of a possible cost function evaluating the parameters of different continuous trajectories. It can be seen that two possible homotopies ($H^{(1)}$ and $H^{(2)}$) exist (e.g. passing a object on the left or right side). The a priori defined maneuvers ($m^{(1)}$ and $m^{(2)}$) are able to extract two local minima. Nonetheless, the global optimal trajectory ξ_0^* may not be found.

1.3 Related Work: Motion Planning Architectures

An autonomous vehicle must not only optimize its trajectory but also make high-level decisions, given the uncertain nature of the prediction. These decisions are for example to either stop in front of or to traverse a zebra crossing, pass before or behind a crossing pedestrian or to decide for a certain gap during lane changes.

Formulating a continuous trajectory optimization problem which incorporates decision making, interaction and uncertain prediction is possible but intractable to solve. Therefore, various planning architectures exist which split the problem into different subproblems. Every subproblem considers only a single aspect of the motion planning problem, which makes it tractable to solve. A trajectory planning algorithm (as the ones presented in section 1.2.6) may therefore be just a part of a larger planning architecture.

This section gives an overview of different, popular planning architectures for autonomous vehicles. As there is no unified solution to the problem of autonomous driving, many approaches exist. Designs focus either on autonomous vehicles [78] in general, on cooperative vehicles [102] or specific frameworks are used for tasks such as the the Defense Advanced Research Projects Agency (DARPA) challenge [34, 73]. A general architecture or clear definition of different frameworks is both missing and difficult to establish due to the early stage of this technology.

Nonetheless, the following five modules are present in motion planning designs:

Navigation layer - A navigation system is responsible of guiding the autonomous car through the route network. It defines for example on which lane to drive.

Behavior layer - The behavioral layer makes the high-level decisions such as in which gap to merge, passing before or after a crossing vehicle, etc.

The behavior can either be optimized itself (as done in this thesis) or realized as a selector, choosing an a priori defined maneuver $m \in \mathcal{M}$. A maneuver $m \in \mathcal{M}$ denotes an abstract, high-level behavior, which can be described in an human understandable way (such as turning right, or crossing before another vehicle). The goal of selecting a maneuver a priori to the planner, is to constrain the problem to a small, convex solution space. A maneuver may be equivalent to the global optimal behavior but may not necessarily. An optimized behavior on the other hand represents the global optimal behavior on the planning horizon which may not be found by using predefined maneuvers. While a precise definition of a maneuver does not exist, it is often compared to the mathematical concept of *homotopies*. Two continuous trajectories are in the same *homotopy* class if a continuous, collision-free projection exists that transforms one trajectory to the other one. For example, two different trajectories, one overtaking an obstacle on the right side and one on the left side lie in different *homotopies*, assuming a 2-dimensional configuration space. General concepts for finding of *homotopies* are presented for path-planning in [12] and for trajectory planning in [11]. Nonetheless, extracting the constraints of *homotopies* is a complex problem in dynamic, urban environments due to to the high uncertainty in the prediction of others. In [101], an approach is presented for the retrieval of different driving corridors in autonomous driving scenarios. It assumes a set-based prediction (realized with constant velocity assumptions) to tackle the problem of uncertain prediction. The relation between maneuvers and homotopies is visualized in figure 1.7.

Prediction layer - The prediction module is responsible for providing the estimated, future configurations of the surrounding traffic.

Trajectory layer - The trajectory planning layer is able to provide a smooth, continuous trajectory. It is mostly dependent on input from the behavior planner (parameterization) and the prediction module.

Controller - Finally, the generated trajectory is tracked with a controller.

The order of this list does neither imply, that the layers must be executed in that order nor that a sequential execution of the modules is the only possibility. Especially, the behavior, prediction and trajectory layers may be interleaved.

Different approaches in the area of motion planning for autonomous driving may be distinguished based on how and in what hierarchical level the following aspects are considered:

- the various uncertainties (see figure 1.3)
- interaction
- the type of constraints (objects, traffic rules, kinematics, dynamics).

The following notation is used throughout the thesis: The autonomous car is defined as N_0 and the surrounding vehicles are defined as N_k, with $k \in [1, K]$. Correspondingly, the state of vehicle N_k at time t is defined as x_k^t. The set of predicted, future trajectories of the other vehicles $N_{1:K}$ in the time interval $[t_0, t_0 + T]$ is denoted as $\widetilde{\xi}_{1:K}^{t_0:T}$. The trajectory of an agent N_k is defined over time as $\xi_k(t)$.

The next sections present often used architectures which deal with possible combinations of the behavior, trajectory and prediction layer in different ways. Possible algorithms which are used within the architectures are reviewed. Following the focus of this thesis, the architectures are differed by how interaction and prediction is considered throughout the motion planning process.

1.3.1 Non-Interactive Planning with Given Prediction

A common approach is to separate prediction and planning. In this case, all the trajectories of the other agents, $\widetilde{\xi}_{1:K}^{t_0:T}$, are predicted first.

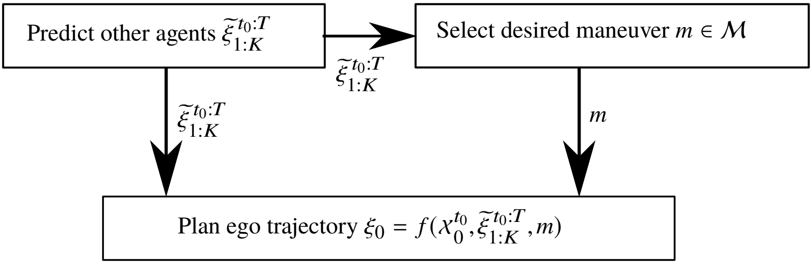

Predict other agents $\tilde{\xi}_{1:K}^{t_0:T}$
Select desired maneuver $m \in \mathcal{M}$
$\tilde{\xi}_{1:K}^{t_0:T}$
$\tilde{\xi}_{1:K}^{t_0:T}$
m
Plan ego trajectory $\xi_0 = f(x_0^{t_0}, \tilde{\xi}_{1:K}^{t_0:T}, m)$

Figure 1.8: The classic separation of prediction and planning is a hierarchical design. In a first step, the other agents are predicted until the planning horizon T and a maneuver $m \in \mathcal{M}$ is chosen. In a second step, the trajectory ξ_0 of the ego vehicle is planned, given the existing prediction of the other vehicles and a desired maneuver (such as Adaptive Cruise Control (ACC), lane change).

Given the predicted trajectories, $\tilde{\xi}_{1:K}^{t_0:T}$, a maneuver m is selected for the autonomous car (by the behavior layer) and a correspondent trajectory is planned by the trajectory planner (see figure 1.8).

This approach is based on the assumption, that the probabilistic future behavior of the other agents is independent of the future behavior of the autonomous agent, i.e.:

$$P(\tilde{\xi}_{1:K}^{t_0:T} | \xi_0) = P(\tilde{\xi}_{1:K}^{t_0:T}). \tag{1.4}$$

While this is a heavily used assumption in most prediction algorithms [65], it is only valid if the behavior of the other agents $N_{1:K}$ is independent of the behavior of N_0. This assumption is valid for scenarios such as a leading vehicle on a highway, but in urban scenarios various counterexamples exist (as demonstrated in figure 1.9). Interactive behavior is hereby simply retrieved by the reactive, replanning behavior of the algorithm [103].

As the interactive behavior of the other vehicles is not considered during planning, this approach can lead to too conservative planning, especially when considering many predicted trajectories [103].

This is because the behavior layer must choose a maneuver m in a way such that it can be executed safely, without respecting that other agents may react to the maneuver. The maneuver selector of such architectures is often realized as a *rule-based* system, which chooses a certain maneuver m and parameterizes the trajectory planner accordingly (e.g. to cross an intersection or to stop before).

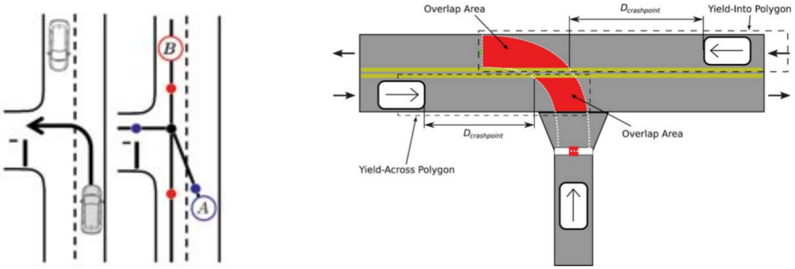


Figure 1.9: Two possible rule based maneuver selection algorithms from participants of the DARPA urban challenge (namely vehicle Annieway (left) and vehicle BOSS (right)). The decision of entering the intersection or stopping before is made in both cases with rules concerning the positions and velocities of all vehicles. Nonetheless, the behavior of the other agents is highly dependent of the autonomous car's behavior (graphics from [47] (left) and [107] (right)).

The trajectory planner is realized as planning algorithm (see section 1.2.6 for an overview), optimizing on a certain temporal/spatial horizon. Two possible maneuver selectors are shown in figure 1.9. Both planning algorithms make the decision to cross or not with rules depending on the position and velocity of the other car. Possible trajectory planners for this scenario are presented in [112, 115].

Possible algorithms for predicting the other cars may rely on physics-based, model-based or interaction-aware models as surveyed in [65]. As part of this thesis, prediction algorithms have also been designed which combine all of these three methods. In [120], a interactive forward simulation of the scene is performed, which generates a likelihood for each discrete maneuver given the current scene. A classifier uses then the prior as well as current measurements to provide probabilities for each maneuver. Another approach is to track various possible interactive motion models with a particle filter [126] or a multiple model unscented Kalman filter to infer the latent variables of the driver model (e.g. the route intention) [127].

Such *rule-based* systems allow us to solve many of the simpler problems in real-world environments [73, 117] by use of a fast replanning behavior. Nonetheless, handcrafting the decision rules for different cases is a costly process which may require individual solutions for each scenario.

However, the potential prediction of the other vehicles has to be constrained to the most likely cases to impede conservative behavior. This is the case as the many, possible predicted trajectories of the other agents narrow the free space in which the robot is able to plan. This is also enforced as most trajectory planning algorithms are not able to consider the interaction explicitly itself.

If interaction is not considered, i.e. the reaction of other agents to the trajectory of the autonomous car is not modeled, the robot may only execute very conservative trajectories. In the worst-case, it may even lead to standstill behavior because the robot cannot find a trajectory which does not collide with one of the various possible predictions of the other agents. This is the so called *freezing robot problem* [103].

1.3.2 (Interactive) Planning with Given Maneuvers

Another possibility is to design a framework in such a way that the behavior layer retrieves the different maneuver possibilities first. Following a general concept of robotics motion planning, the idea is to avoid a potentially exhaustive search of the whole configuration space. Therefore, the idea is to extract the different path or trajectory homotopies first to constrain the following optimization problem to this subspace [11]. The maneuvers can be described with driving corridors represented by one reachable set per homotopy/maneuver [101], with different constraints retrieved by a forward search [90], or directly with a designed policy for every given maneuver [32, 23].

In a second step, a trajectory is planned for every potential maneuver $m \in \mathcal{M}$ and the best one is selected for execution. This idea is demonstrated in figure 1.10. This method has especially its advantages when used with variational methods. As these methods are only able to find a local minimum, the repeated calculation for every possible maneuver, may allow to find the global optimum.

In [9], a deterministic local planner is used on every set of maneuver constraints to find the global optimum. Uncertainty and interaction are hereby not considered. As the authors mention, it is difficult to extract all constraints and calculating a trajectory for every maneuver may become intractable for combinatorial problems with a high amount of possibilities.

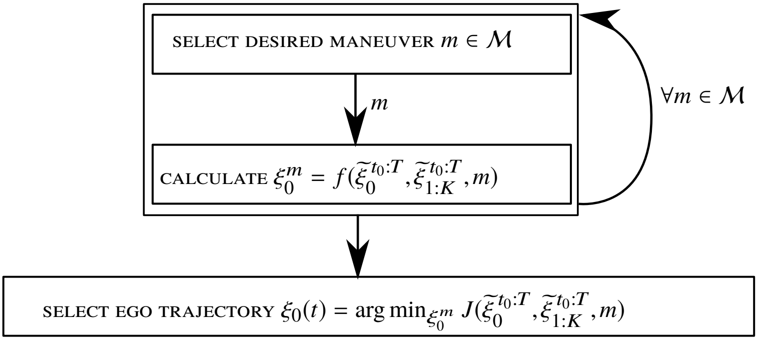

SELECT DESIRED MANEUVER $m \in \mathcal{M}$
m
$\forall m \in \mathcal{M}$
CALCULATE $\xi_0^m = f(\widetilde{\xi}_0^{t_0:T}, \widetilde{\xi}_{1:K}^{t_0:T}, m)$
SELECT EGO TRAJECTORY $\xi_0(t) = \arg\min_{\xi_0^m} J(\widetilde{\xi}_0^{t_0:T}, \widetilde{\xi}_{1:K}^{t_0:T}, m)$

Figure 1.10: At first a certain set of possible maneuvers $\mathcal{M}$ is retrieved (e.g. cross the intersection before/after the other car). Now a trajectory is planned for each of the maneuvers in $\mathcal{M}$ and evaluated with a cost function J. The trajectory of the maneuver with the minimum cost is then selected.

Different approaches consider planning and prediction as a combined problem, such that the prediction of the other agents is also part of the planning problem. In [118], the interaction in terms of the expected reaction of other vehicles to the planned ego trajectory is considered in a deterministic way.

The authors of [84] use a Mixed Integer Quadratic Programming (MIQP) approach. They assume static obstacles as well as dynamic agents with a single predicted trajectory. Various maneuver possibilities for passing the objects in different sequences are extracted and used for the formulation of the MIQP formulation.

A combined planning and state estimation approach for all different maneuvers is presented in [90]. The authors introduce so called *collective maneuvers*, which describe the possible maneuver combinations in a certain scene. Every vehicle tracks the maneuver probability of every other car online. By use of a cooperative cost function every agent plans interactive maneuvers in a collective maneuver set which is formulated as MIP.

In [37], the different maneuver homologies (i.e. relaxed homotopies) are extracted by introducing *pseudo homologies* first. Hereby the homology assumption is further relaxed, so that trajectories with different end states may lie in the same homology, as long as both end states fulfill some conditions about how they relate to each other.

The authors of [32, 23] formulate the planning problem as a Markov Decision Process (MDP) with possible, hand-tuned policies for each maneuver. This allows for stochastic forward simulations of the whole scene given the different policies to determine the expected cumulative reward of every policy. The potentially non-linear and probabilistic transition function of the Markov Decision Process (MDP) allows considering interaction as well as uncertainty in the planning problem.

In general, the approach of simply optimizing for one (section 1.3.1) or several maneuvers (section 1.3.2) has drawbacks. At first, all maneuvers must be enumerated beforehand, which is a challenging task that does not guarantee finding the optimal behavior. Additionally, the solution space is absolutely constrained to these a priori maneuvers. Solutions which e.g. optimize for two possible maneuvers are not found.

1.3.3 Optimizing Interactive Maneuvers

The approaches in the previous two sections depend on the a priori selection of a certain maneuver m or even a set of maneuvers for the autonomous vehicle. Instead of defining certain maneuvers beforehand and planning corresponding local optimal trajectories, graph-based search techniques allow to find a global optimal solution [101] directly. This global optimal solution may contain a high-level maneuver implicitly, such that there is no need anymore for an a priori maneuver selection.

Such a search may either assume the prediction of the other agents as independent or also model the interactive behavior of the other agents by representing them directly in the state space. A probabilistic search, such as Monte Carlo Tree Search (MCTS) allows even for representing the possibly interactive behavior of the other drivers in a probabilistic instead of a deterministic fashion. The general framework is shown in figure 1.11.

As it is intractable to search the whole configuration space, it is essential for these methods to constrain the search. This may be done with heuristics, probabilistic sampling or intelligently designed search graphs.

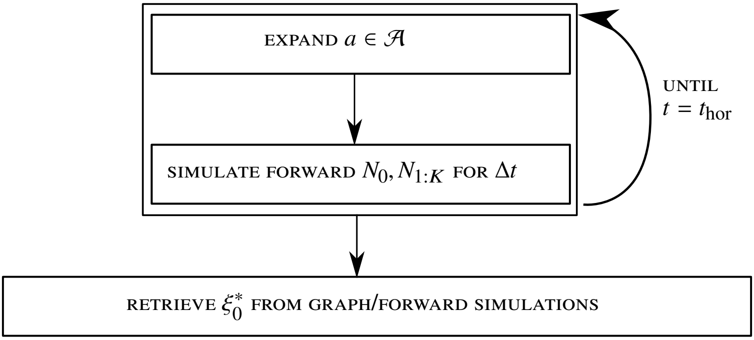

EXPAND $a \in \mathcal{A}$
SIMULATE FORWARD $N_0, N_{1:K}$ FOR Δt
UNTIL $t = t_{\text{hor}}$
RETRIEVE ξ_0^* FROM GRAPH/FORWARD SIMULATIONS

Figure 1.11: A search based optimization technique provides a trajectory solution directly and optimizes the maneuver implicitly. It searches the whole configuration space C_{config} by expanding possible motion primitives and simulating the other vehicles simultaneously.

In [44], a visibility graph in the reachable set is created around the edges of dynamic objects. This allows to create a minimum sized graph which can be searched for the optimal solution. The graph resembles generated maneuver hypotheses due to its minimum size and structure around existing objects. The behavior of the other drivers is predicted in advance, such that it allows not for interactive behavior.

The authors of [67] present a MCTS based planner for highway driving. The approach is strongly focused on modeling the potential interaction during planning but does not account for uncertainties.

In [59], a decentralized cooperative MCTS approach for lane changes is presented. Every agent optimizes a cooperative cost function, while neglecting uncertainties. The authors use a hierarchical MCTS variation with macro actions and progressive widening to allow for continuous actions on a constrained search space.

While these methods have the advantage of finding a global optimum, they are considered difficult to use in reality. This is the case as the exhaustive search may limit the *online* capability of the algorithm and reduces its completeness to resolution completeness. This leads to a small, discretized action set $\mathcal{A}$ to constrain the possible branching factor. Such a discretized action set reduces the completeness of the planner to resolution completeness as the reachable state space is limited. This is due to the limited number of actions and the time resolution.

1.4 Motion Planning with Policies

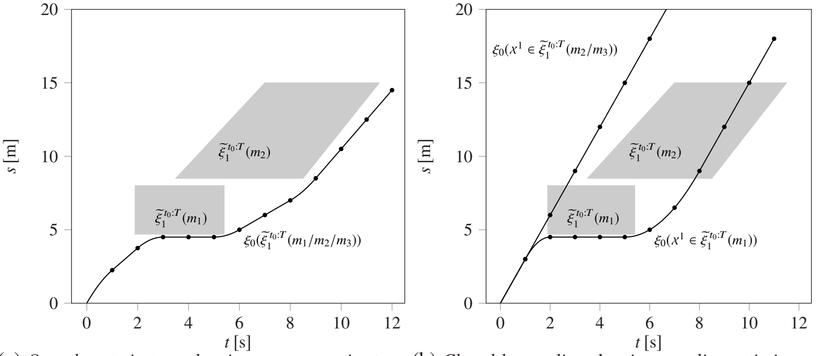
Most existing motion planning algorithms plan a trajectory from a given start state x_{start} to a goal state x_{goal}. Executing a global optimal trajectory leads to global optimal behavior, given that the system dynamics are deterministic. This work assumes perfect localization and control during the execution of a trajectory. Nonetheless, the future trajectories of the surrounding traffic are not known and can only be described in a probabilistic fashion. Two distinct approaches can be differed for this problem, namely *open-loop* and *closed-loop* planning.

1.4.1 Open-Loop Planning

Open-loop motion planning algorithms do not consider future measurements which arrive during the execution of the planned motion. In this case, two possibilities exist. Firstly, every possible future motion of the other vehicle can be respected by the planner. In figure 1.12a, a scenario is shown where the planner has to consider three possible predictions (turning left/right and passing the intersection) of the other vehicle for the scenario presented in figure 1.3. Respecting every possible prediction leads to safe but potentially conservative behavior. Another possibility is to consider only the most likely prediction(s). While this allows for less conservative plans, safety cannot be guaranteed as not every possible prediction is considered in the planning stage.

1.4.2 Closed-Loop Planning

Closed-loop motion planning on the other hand allows to consider the possible future observations in the planning stage. Instead of a trajectory, these algorithms plan a policy which contains reactive plans for different possible future scenarios. Such a policy is shown in figure 1.12b for the scenario presented in figure 1.3. The policy contains two plans about how to react to the observation



(a) *Open-loop* trajectory planning: a conservative trajectory evades all possible, predicted trajectories.

(b) *Closed-loop* policy planning: a policy optimizes the expected reward and provides different plans, depending of the next observed state.

Figure 1.12: Transfer of the scenario in figure 1.3 into a spatio-temporal cost map. The longitudinal position on the planned path of the autonomous vehicle is plotted on the y-axis, the planning time on the x-axis. The predicted maneuvers of the car on the left of the intersection are depicted in grey, the planned trajectory of the autonomous vehicle is depicted in black. The other car has three potential maneuvers (m_1 = cross straight, m_2 = turn left, m_3 = turn right). Planning of *closed loop* policies allow for less conservative driving. The policy contains future plans for all possible observations. If it will be observed at $t = 1$ that the other car turns left, the autonomous car merges before the other car. If maneuver m_1 is observed, it will execute the plan to pass behind the crossing car. If the other car is observed to turn right, the autonomous car has the same behavior as for maneuver m_2.

which arrives at $t = 1$. This allows for less conservative initial actions as the policy allows to react to the future observation. *Closed-loop* planning is rarely used for *online* planning. This is the case as it must consider an infinite amount of possible measurements which makes the approach often intractable to use. In [28], the authors present Partially Closed-Loop Receding Horizon Control (PCLRHC) to overcome this problem. The idea is to only consider the most likely future observation. While this makes the approach tractable, it may lead to unsafe behavior as relevant future observations may be ignored.

1.4.3 Definition of Policy Optimization

The goal of a policy is to map an action on a state to maximize the reward over all possible, future scenarios (see figure 1.12b). Therefore, a potential policy is evaluated by its expected reward to account for the uncertainty of the future observations.

To account for these requirements, the motion planning problem is described as follows. A probabilistic transition model is defined as $T(x', x, a) = P(x'|x, a)$, describing the probability of ending in a new state x', after executing a certain action a in state x. A reward function $R(x, a)$ is defined for executing an action $a \in \mathcal{A}$ in a state $x \in \mathcal{X}$.

Instead of minimizing a cost functional, the optimization aims for finding the policy which maximizes the expected, discounted future reward:

$$\pi^* := \underset{\pi \in \Pi}{\arg\max}\ \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^t\right]. \tag{1.5}$$

The policy defines a mapping from states on actions, $\pi : \mathcal{X} \rightarrow \mathcal{A}$, such that the $a = \pi(x)$. The future reward may be discounted by a discount factor γ, to favor immediate rewards over long-term rewards. This formulation allows to model combinatorial decision problems with a non-linear, probabilistic transition function T and a deterministic reward function $R(x, a)$.

1.5 Closed-Loop Behavior Planning Under Uncertainty

This thesis describes a new behavior planner for autonomous driving in urban environments. The underlying idea is to formulate the behavior planner itself as an optimization problem on a receding horizon.

The presented planner is not dependent on any a priori maneuver generation but finds the global optimal solution on the receding horizon *online*. This means that decisions, such as passing before or after a crossing pedestrian are made implicitly by the optimization problem. The solution may represent a typical high-level maneuver implicitly but not necessarily. This is enabled by formulating the problem as a search-based planning problem, which respects static/dynamic objects as well as other constraints such as traffic rules.

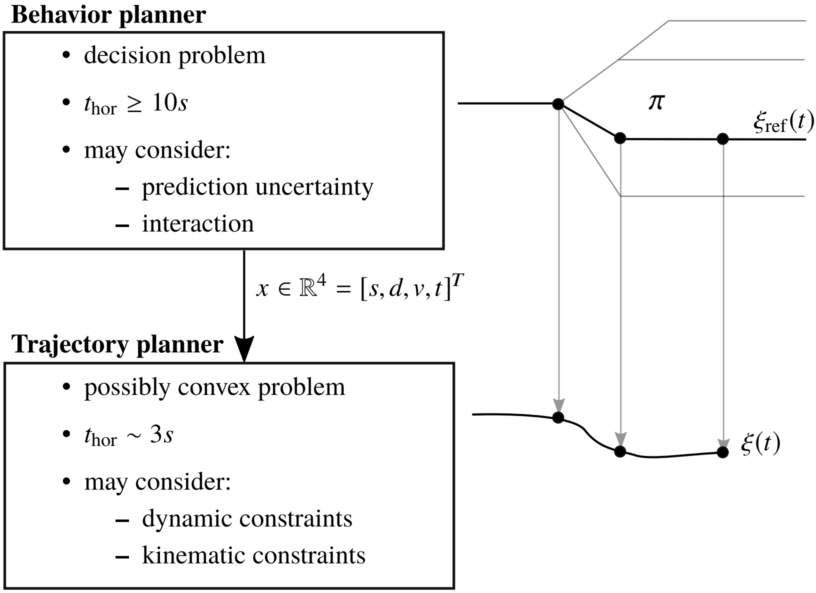


Figure 1.13: The behavior planner optimizes the policy on a receding horizon in a global manner and parameterizes the trajectory planner for finding the continuous local optimal trajectory in the global minimum. The provided reference trajectory is the most probable trajectory in the policy, s.t. $\xi_{\text{ref}} = \arg\max_{\xi \in \pi} P(\xi)$.

The planner respects all relevant uncertainties of the other agents (see figure 1.3) as well as their potential interaction with the autonomous car.

Considering the various different uncertainties with an *open-loop* planner would lead to very conservative results due to the large manifold of possible, predicted trajectories. To overcome this problem, this thesis presents a *closed-loop* planner which generates a policy over an uncertain belief space. The policy contains reactive plans for possible future observations, i.e. measurements of the uncertain behavior of the other agents. The policy is optimized for the most probable future scenarios and also incorporates at what point in the future certain beliefs (e.g. over the yielding behavior of other agents) become more certain.

The problem is modeled as a Partially Observable Markov Decision Process (POMDP). Solving this problem formulation exactly is known to be an intractable problem (see section 2.3.1). Therefore, the problem is reduced by using only a discrete set of actions. The policy is retrieved from simulating thousands of possible scenarios *online* on the planning horizon. This allows to provide an optimized behavior *online*, that contains respective decisions and considers uncertainty and interaction.

While such a policy considers the aforementioned uncertainties, interaction and has the capability of decision making, the discrete actions lead to non-optimal trajectories concerning comfort and smoothness. Therefore, the separation between behavior planning and trajectory planning is also used within this work. The behavior planner generates an optimal plan first, but under different optimization criteria than the trajectory planner. In a second step, a local trajectory planning algorithm optimizes the most probable trajectory in the generated policy of the POMDP on a shorter horizon (figure 1.13). This is e.g. done with optimization criteria like minimizing jerk [112] or minimizing the total turn rate of the steering angle [38]. Simpler trajectory planning approaches minimize acceleration by smoothing the speed profile without considering the kinodynamic constraints [113].

1.6 Contributions and Outline

The main contributions of this thesis are the following:

Firstly, it presents a **global** optimization formulation for autonomous driving scenarios. This allows to generate a *global optimal behavior* instead of behaviors which display certain high-level, hand-selected maneuvers only.

Secondly, this thesis shows how *various uncertainties* can be explicitly modeled by a **POMDP formulation**. The uncertainties are namely the unknown intentions of other drivers, their uncertain prediction, possible interaction, noisy sensor measurements as well as the uncertainty introduced by occlusions. The problem is formulated on a continuous state/belief space and uses a discrete action space to optimize the behavior of the autonomous car.

By designing a *closed-loop policy* planner which respects not only the current belief state but also the *most likely future scenarios*, less conservative behavior can be realized. This formulation even allows the agent to actively reduce the uncertainty by executing *information gathering* actions. Additionally, modeling *intertwined prediction and planning* allows to consider the reaction of other agents to the trajectory of the autonomous car.

Thirdly, it is shown how this POMDP formulation can be solved **online**. This is possible by extending *state of the art solvers* with *domain specific heuristics* which allows to focus on promising branches in an otherwise intractable graph search.

Finally, an extensive **evaluation** demonstrates the capabilities of the planner for scenarios such as the crossing of intersections, lane changes in dense traffic and the handling of occlusions. It is shown how the presented *closed-loop* planner outperforms common *open-loop* planning approaches. It is demonstrated how the planner allows for non-conservative behavior in uncertain environments that can only be achieved by common *open-loop* planners if they have full knowledge of the future scene, i.e. are omniscient.

The remaining thesis is structured as follows: Chapter 2 will introduce the general POMDP formulation and explain in detail how POMDPs are solved in this thesis. This is combined with general background and characteristics of POMDPs including various techniques to solve them.

In Chapter 3, a deterministic planning algorithm based on the popular A^* formulation is introduced. It demonstrates how an optimal behavior planning algorithm can be realized without using a predefined maneuver set. The algorithm considers the predicted trajectory of dynamic objects and other dynamic constraints (such as switching traffic lights). While it does not consider any uncertainties, it demonstrates planning based decision making.

In Chapter 4 the POMDP planning approach is presented for intersection scenarios. The algorithm optimizes a longitudinal policy under the uncertainty of the unknown intention, driver models and interaction of the other agents.

Chapter 5 extends the action set, s.t. the policy is optimized in a 2D space. This combined longitudinal and lateral optimization is combined with a belief state formulation which includes the friendliness of the other drivers. It is demonstrated how this formulation allows the agent to merge in too small gaps by actively considering the potential interaction with the surrounding traffic.

Finally, Chapter 6 introduces how reasoning over potentially existing agents in occluded areas can be formulated as a POMDP. The approach allows the autonomous car to actively gather more information about occluded areas by explicitly considering the field of view of the car in the forward simulation.

Chapter 7 summarizes the results of the thesis and gives an outlook how the approach may be extended and improved in the future.

2 Background

This chapter introduces algorithms and their mathematical notations used in this thesis. The theoretical foundations of sequential decision making under different degrees of uncertainty are introduced. This is followed by an overview of different solvers and a detailed discussion on how POMDPs are solved in this thesis. Finally, the reduction of a 2-dimensional planning problem is discussed, the so called *path-velocity* decomposition.

Sequential decision making models a problem where a series of decisions have to be made. The problems may be distinguished by the nature of the transition model and by the observability of the state. The transition model can either be deterministic or probabilistic and the state of the environment may either be fully observable or only partially observable. Partial observability of the state leads to a description of the environment with a belief state as the real state cannot be measured.

2.1 Planning with Deterministic Models

Given a set of discrete, fully observable states, a set of discrete actions and a deterministic state transition function, s.t. $x' = f(x, a)$, the sequential decision making problem can be solved by the graph search algorithms presented in section 1.2.3. This is the case as the deterministic nature of the state transition function and the fully observable state allows to present the planning problem as a graph.

One of the most well-known algorithms for graph search is the Dijkstra algorithm [25]. It keeps track of two sets, a set of open nodes $\mathcal{X}_{\text{open}}$ (not yet expanded) and a set of closed nodes $\mathcal{X}_{\text{closed}}$ (already expanded). The idea behind the Dijkstra algorithm is to select the node with the minimum cost-to-come in $\mathcal{X}_{\text{open}}$ as the next node to be expanded. This allows to steer the search in a promising direction and guarantees to find the shortest path [86]. Despite the directed search, the algorithm has still a worst-case runtime of $\mathcal{O}(|E|+|V|\log|V|)$. The algorithms in this thesis use a extension of the Dijkstra algorithm, the so called A^* algorithm [86]. It uses a heuristic to truncate non-promising branches early. Instead of expanding the node with the lowest costs, i.e. $x = \arg\min_{\mathcal{X}_\text{open}} x.c$, the A^* algorithm expands the node which describes the state $x = \arg\min_{\mathcal{X}_\text{open}} x.c + h(x)$. The heuristic function $h(x)$ estimates the

Algorithm 1 A* graph search on a receding horizon

```
$x_\text{start}, t_\text{hor}, \mathcal{A}$          //$x_\text{start}$: start state, $t_\text{hor}$: planning horizon
function $A^*(x_\text{start}, t_\text{Hor}, \mathcal{A})$
$\mathcal{X}_\text{open} = [], \mathcal{X}_\text{closed} = []$
$\mathcal{X}_\text{open} \leftarrow x_\text{start}$
while $\mathcal{X}_\text{open} \neq \emptyset$ do
    $x \leftarrow \arg\min_{\mathcal{X}_\text{open}} x.g$          //$x.g$: heuristic + cost-to-come
    if $x.t \geq t_\text{hor}$ then
        return $x$
    end if
    for all $a \in A$ do
        $x' \leftarrow x + a$
        if $x' \notin \mathcal{X}_\text{closed}$ then
            $x'.c \leftarrow x.c + J(x, a, x')$
            $x'.g \leftarrow x'.c + h(x')$
            $x'.p \leftarrow x$                    //$x'.p$: parent state of $x'$
            if $x' \in \mathcal{X}_\text{open}$ and $x'.g < x'_{old}.g$ then
                Remove $x'$ in $\mathcal{X}_\text{open}$
            end if
            add $x'$ to $\mathcal{X}_\text{open}$
        end if
    end for
    add $x$ to $\mathcal{X}_\text{closed}$
end while
return trace back trajectory from $x$
```

remaining total costs from state x to the goal state. As long as the heuristic function h is a lower bound to the real costs as well as consistent, the A^* algorithm is complete. Consistency is fulfilled if $h(x) \leq J(x, a, x') + h(x') \forall x, x'$. The A^* algorithm is presented in Alg. 1 for planning on a limited time horizon.

2.2 Planning with Probabilistic Models

For the case of a probabilistic state transition function $T(x' \mid x, a) := P(x' \mid x, a)$, the sequential decision making problem can be modeled as a Markov Decision Process (MDP). An MDP is defined by the tuple $\langle \mathcal{X}, \mathcal{A}, T, R, \gamma \rangle$. The reward function, $R(x, a)$, defines a reward for choosing an action $a \in \mathcal{A}$ in state x. The reward is discounted over time t by a discount factor γ to favor immediate rewards over long term rewards. The state $x \in \mathcal{X}$ of an MDP is assumed to be fully observable. The goal of an MDP is to find an optimal policy, π^*, which maximizes the expected cumulative discounted reward, i.e. the value $V(x)$ when starting in state x and following the optimal policy thereafter:

$$\pi^*(x) := \arg\max_{\pi} V^{\pi}(x). \tag{2.1}$$

To solve an MDP, a technique called *dynamic programming* is used. It is described by Bellman's principle of optimality:

> *Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [8].*

In other words, an optimal solution to a problem is composed by optimal solutions of the subproblems. This is known as the *Bellman* equation [8] which defines the optimal value function, V^* of an MDP as follows:

$$V^*(x) = \max_{a \in \mathcal{A}} \left[R(x, a) + \gamma \sum_{x' \in \mathcal{X}} T(x, a, x') V^*(x') \right]. \tag{2.2}$$

Widely known algorithms such as policy iteration and value iteration are based on the idea of the *Bellman* equation [51]. These algorithms iteratively perform backups over the (whole) state space $\mathcal{X}$ and update the value function accordingly until convergence is reached. Although they guarantee to find the optimal policy, they are are mostly used for *offline* MDP approaches. Their general computational complexity and the curse of dimensionality makes them comparably slow and therefore only suited for low dimensional problems. On the contrary, approximate techniques such as *sparse sampling* or MCTS are better suited for *online* MDP approaches [51].

Certain assumptions even allow to simplify solving an MDP. In the case of a linear system with a quadratic cost function, the general MDP formulation becomes a *linear quadratic regulator* and can be solved analytically.

For more details about MDPs, the reader is referred to [51] and [86].

2.3 Planning with State Uncertainty

MDPs rely on the assumption that a state is fully observable. POMDPs on the other hand do not rely on this assumption and are formulated over a probabilistic belief state instead of a fully observable state. Since the current state is not known, the belief state is described by $b(x) \forall x \in \mathcal{X}$, i.e. the probability of being in a certain state x.

A POMDP is defined by the tuple $\langle \mathcal{X}, \mathcal{A}, T, \mathcal{O}, Z, R, b^0, \gamma \rangle$. The state is defined as $x \in \mathcal{X}$ and a possible action of the agent is defined as $a \in \mathcal{A}$. $T(x, a, x') = P(x'|x, a)$ is the transition probability of ending in state x' when executing action a in state x. $R(x, a)$ is the reward for selecting action a in state x. The initial belief of the problem is b^0. Additionally, the discount factor $\gamma \in [0, 1)$ is used to favor immediate rewards over long-term rewards.

The differences to an MDP are the possible observations $o \in \mathcal{O}$ and the observation function Z which allows to describe future beliefs $b(x)$, given possible future observations o, a prior belief b, the state transition probabilities T and the observation function Z.

The observation function $Z(x', a, o) = P(o|x', a)$ provides the probability to observe a certain observation o after taking action a and ending in the new state x'. The policy of a POMDP maps a belief state b an action, $\pi : b \mapsto a$. The solution of a POMDP is the optimal policy, π^*, which maximizes the expected discounted cumulative reward

$$\pi^* := \arg\max_{\pi} \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(b^t, \pi(b^t)) | b^0, \pi\right]. \tag{2.3}$$

The definition of the optimal value function can also be transferred to POMDPs, such that the corresponding *Bellman* equation for belief states, $V^*(b)$ is defined as follows:

$$V^*(b) := \max_{a \in \mathcal{A}} \left[R(b, a) + \gamma \sum_{b' \in \mathcal{B}} T(b, a, b') V^*(b') \right]. \tag{2.4}$$

The reward model over a belief state, $R(b, a)$, is defined as:

$$R(b, a) = \sum_{x \in \mathcal{X}} b(x) R(x, a). \tag{2.5}$$

2.3.1 Complexity of Solving POMDPs

POMDPs are often considered to be computationally intractable to solve exactly. This is the case for two reasons:

The first reason is due to the *curse of dimensionality*: Even a limited number of discrete states $|\mathcal{X}|$ leads to a $(|\mathcal{X}| - 1)$-dimensional continuous belief space $\mathcal{B}$ [81]. Therefore, naive discretization of the belief space results in an exponential number of belief states over the number of states.

The second reason is the *curse of history*: The number of possible action-observation sequences, starting at a belief b^0, is $(|\mathcal{A}||\mathcal{O}|)^n$ and therefore grows exponentially with the length of the history n. Nonetheless, to calculate exact solutions to POMDPs, all possible histories must be considered.

This makes finding an optimal policy for a finite-horizon POMDP PSPACE complete [79]. Solving a POMDP for an infinite horizon is even undecidable [70].

Nonetheless, many algorithms have been developed which calculate approximate solutions for POMDPs as surveyed in [85]. These approximate techniques allow to successfully approximate optimal solutions *online* on large state spaces ($|\mathcal{X}| > 100.000$).

These positive results lead to further investigations about what underlying characteristics of POMDPs make them easier to solve. The authors of [41] introduce a so called covering number of a POMDP as the number of balls of a certain diameter which are needed to cover the reachable belief space. It is shown that a solution can be calculated in time polynomial in the covering number of a reachable belief state and the authors argue that the number of states may be a poor measure of the complexity of a POMDP.

2.3.2 Solving POMDPs

In the following, it is explained how the belief state of the world can be estimated over time and how POMDPs can be solved. An overview of the most common *online* POMDPs solvers is given.

Belief Update

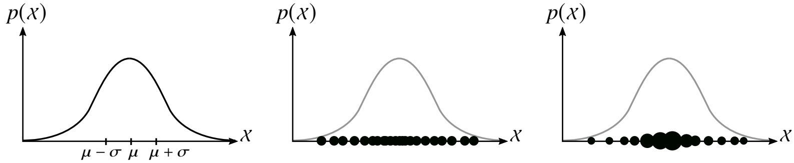
The policy contains the optimal action $a^t = \pi(b^t)$ at time t, given the current belief b^t. The current belief b^t can be estimated with recursive Bayesian estimation by using the last belief state, b^{t-1}, the last action a^{t-1} as well as the actual observation o^t:

$$b^{t+1}(x^{t+1}) = P(o^t, a^t, b^t). \tag{2.6}$$

The Bayesian filter can also be used for state estimation only, as done in [126], to track the model parameters of the surrounding agents. In general, formulations with discrete states and formulations with linear-Gaussian transition models and Gaussian observation models can be solved exactly [51]. For discrete state problems a Bayesian discrete state filter is used. Problems on a continuous state space with a linear-Gaussian transition and observation model can be solved exactly with the well-known *Kalman* filter [46]. If the transition model is non-linear, exact solutions do not exist anymore. For the case of a non-linear, continuous transition function various modifications of the *Kalman* filter exist.

Algorithm 2 Unweighted particle filter

```
function UPDATE BELIEF($(b, a, o)$)
$b \leftarrow \emptyset$
for $i \leftarrow 1$ to $|b|$ do
    $x \sim b$
    repeat
        $x' \sim T(x, a, x')$
        $o' \sim Z(o, a, x')$
    until $o' = o$
    add $x'$ to $b$
end for
return $b'$
```



(a) Analytic description of $\mathcal{N}(\mu, \sigma)$.

(b) Description with an unweighted set of particles.

(c) Description with a weighted set of particles.

Figure 2.1: Different methods to present the probability density function of a normal distribution. The representation with particles allows for describing arbitrary probability density functions.

The *extended* and *unscented Kalman* filters allow to find non-optimal solutions in these cases [45] by using linearized transition functions or approximated Gaussian distributions.

The above mentioned approaches do only work, if the probability density function of the belief state can be described analytically, e.g. with a (set of) Normal distributions. Another possibility is to describe the probability density function, by a set of (weighted) state instance, i.e. particles. The underlying idea is shown in figure 2.1 for the case of a normal distribution. Additionally, typical filters based on the idea of particles are able to track the probability even for highly non-linear or even non-continuous transition models. This work uses an unweighted particle filter to track the current belief state, the pseudo code of the algorithm is presented in Alg. 2 and is inspired by [51].

Point-Based Solvers

As introduced in section 2.3.1, one of the main difficulties in solving POMDPs is the continuous belief state. Nonetheless, the authors of [100] showed, that the value function of a POMDP is always piece-wise linear and convex in the belief (see the left figure in figure 2.2 for an example). This characteristic allows a simple representation of the optimal value function over the continuous belief by a set of α-vectors. An alpha vector α_a contains the reward for every possible state, assuming action a, s.t. $\alpha_a = R(\cdot, a)$. The alpha vector describes a $|\mathcal{X}|$-dimensional hyperplane in the belief space $\mathcal{B}$ and allows to describe the value function as

$$V(b) = \max_{\alpha} \sum_{x \in \mathcal{X}} \alpha(x) b(x). \tag{2.7}$$

The idea of POMDP value iteration is to describe every possible plan until a certain depth by α vectors. Nonetheless, solving a POMDP exactly with such an approach is infeasible as the number of needed alpha vectors $|\alpha_\pi|$ for optimization depth of n grows exponentially, s.t. $|\alpha_\pi| = |\mathcal{A}|^{(|O|^n-1)/(|O|-1)}$ [51].

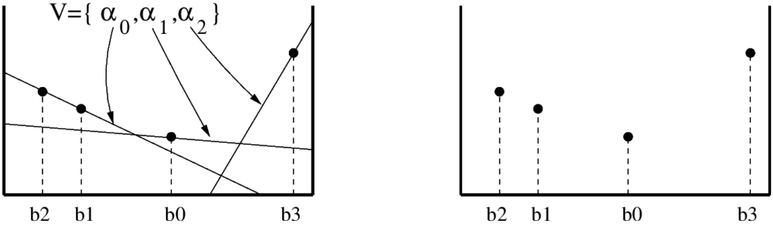


Figure 2.2: Left: Point-Based Value Iteration (PBVI) keeps a set of α-vectors to approximate the value function over the whole belief. Right: Grid based approaches approximate the value function only for a set of belief-value pairs [14] (graphic from [81]).

The idea of *point-based* algorithms is to overcome that problem by backing up the value function only for a discrete set of chosen belief-points. The first algorithm, alleviating this idea, was the PBVI algorithm, proposed by [81]. The algorithm selects an initial set of belief points and estimates their value by use of α-vectors. In a next step, further belief points are added if it introduces a strong improvement on the value of the belief. Two other known algorithms are Heuristic Search Value Iteration (HSVI) [98] and Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) [57]. Both algorithms use graph search to limit the optimization to the reachable belief while pruning the tree with upper and lower bounds. A complete survey of *point-based* solvers can be found in [93].

Sampling-Based Solvers

While the point-based solvers in the previous section simulate full belief trajectories, sampling-based solver use simulated histories to estimate the value of certain belief states.

The belief state itself is hereby represented by particles (similar to the belief state of a particle filter). Potential histories are simulated based on a particle. Simply representing the belief state by a set of particles may sound like a simple approach compared to representing the value function over a continuous belief with α-vectors. Nonetheless, the characteristic of MCTS based search, i.e. focusing on the most promising branch, allows to produce fast approximations to the optimal value function for relevant beliefs.

The first Monte-Carlo algorithm for POMDPs was introduced by [96]. The idea is to represent the current belief by a set of particles and construct a belief tree by sampling possible episodes in the reachable belief. That way, the optimized policy is only calculated for the current belief. Representing the belief state by a set of particles allows to update the belief by an unweighted particle filter given the current observation. This allows to keep the relevant part of the belief tree alive instead of reconstructing the tree from scratch. The authors of the Adaptive Belief Tree (ABT) algorithm adapt the algorithm to be able to keep the belief tree even in cases where the model changes by detecting the relevant episodes. A Determinized Sparse Partially Observable Tree (DESPOT) is constucted in [99]. The idea is to constrain the growing of the belief tree by observing if new branches do change the current optimal policy or not. The authors of [21] drastically speed up the idea of DESPOT by parallelizing action selection as well as roll-outs on a CPU and GPU simultaneously.

2.3.3 The Simplified QMDP Formulation

While the POMDP formulation is very generic, simplified formulations exist. A popular one is the so-called Fully Observable Value Approximation (QMDP). The idea behind a QMDP is to calculate a solution under the assumption that the state uncertainty disappears after the fist planning step, i.e. the next state is fully observable [69].

The QMDP approximation is defined as follows:

$$Q(b,a) = \sum_{x \in X} b(x) Q_{\mathrm{MDP}}(x,a). \tag{2.8}$$

This leads to selection of the action which maximizes the long term reward, weighted over all states. The underlying assumption, that all uncertainty vanishes in the next step, makes it a overly confident, unsafe planner. Nonetheless, it can be shown that a QMDP solution is an upper bound to the value function of a POMDP [51]. This makes it very interesting for use as heuristic.

2.3.4 Policy Optimization: Online vs Offline

In the context of solving a POMDP, *offline* is referred to finding a sufficiently optimized policy over the whole belief space prior to execution. An *online* solver on the other hand, calculates only the optimal action for the current belief assuming a certain optimization depth n. After execution of the optimal action, the belief is updated and the optimal policy is optimized again for the new belief.

As it is computationally challenging to solve a POMDP *online*, one might argue that solving it *offline* could be a possibility.

Nonetheless, this is not the case due to various reasons. At first, finding a (belief) state representation for every possible scenario with a completely varying number of agents, lanes, traffic regulations, etc. is intractable for planning-based approaches due to the sheer manifold of situations. Even if such a representation could be found, the dimensionality of the state space may explode as every detail (geometries, lane markings) of the scene must be incorporated in the state space. This makes the problem to solve by magnitudes harder. In the case of *online* approaches on the other hand, only the dynamic agents must be part of a state space whereas scene information such as lane geometries can simply be considered as actual parameters of the motion models of the agents which does not change the dimensionality of the state space. Moreover, *online* solver calculate the optimal policy for the current belief only which allows them to only consider the reachable belief space on a limited horizon. Additionally, even if a generic problem formulation can be found and solved, describing and storing the resulting policy may become difficult due its sheer size.

Nonetheless, the idea of *offline* approaches is pursued in different, promising ways. Learning based approaches try to tackle the problem of a generic input space by using either various, preprocessed top-down views as input for a deep imitation learning architecture (as done by Waymo's ChauffeurNet [6]) or by learning directly from front camera images [13]. Nonetheless, these approaches are currently in an early stage and are limited by the amount of possible training data (including corner-cases) and have difficulties to give guarantees on the learned behavior in the Deep Neural Net (DNN). Approaches which combine both worlds exist and are very promising. The these cases, the idea is to have an *online* graph/tree search which relies heavily on heuristics, which are learned *offline*. Such a approach is for example used for the artificial intelligence used to play and win against humans in the board game Go [95]. Hence the problem is solved *online* in this thesis.

2.4 Solving POMDPs in this Thesis

In this thesis the Toolkit for Approximating and Adapting POMDP Solutions in Real Time (TAPIR) [50] is used to solve a POMDP *online*. It is an implementation of the ABT algorithm [58], one of the fastest POMDP solvers today. The algorithm is anytime and capable of solving large POMDPs even on continuous belief states *online*.

It approximates the optimal policy by *Monte Carlo* sampling of potential episodes in the reachable belief space. ABT uses MCTS, but modified for POMDPs. This section explains at first the standard MCTS algorithm. It is followed by the transfer of MCTS to POMDPs and further details on how ABT and the implementation in this thesis works.

2.4.1 Monte Carlo Tree Search

MCTS was first introduced in [22] where it was used for sequential decision making in games. The transition model in games is probabilistic as the next state of the game depends on the controllable action of the agent but also on the unknown action of the other player. The general idea of MCTS is to combine a deterministic tree search with random sampling. This allows to solve MDP formulations, which cannot be solved by traditional graph-search due to the non-deterministic nature of their transition model.

A major breakthrough occurred with the introduction of the Upper Confidence Bound for Trees (UCT) algorithm [52]. UCT balances the search in a way which allows to sample promising branches more frequently to achieve a very precise estimate of the value of promising branches. Less promising actions on the other hand are sampled less often. This procedure is very powerful and allows to solve otherwise infeasible, probabilistic graph search. The idea behind the UCT is similar to the exploration-exploitation dilemma in Reinforcement Learning (RL), but happens completely *offline*. The nodes of the tree correspond to states. The algorithm performs many simulations of so-called possible histories to estimate the state-action value function $Q(x, a)$. One simulation is composed by four steps as shown in figure 2.3:

1. *Selection*: Traverse the tree until an expandable node is reached (non-terminal state and unexpanded children)
2. *Expansion*: Expand the node
3. *Simulation*: Do a roll-out with a default policy to estimate future rewards
4. *Backpropagation*: The received rewards are propagated back to update the statistics of prior nodes

The combination of a smart selection method (e.g. the UCT algorithm) and the fast estimation of future rewards by a sufficiently good default policy allows to reduce the search space drastically, which gives the algorithm its *online* capabilities. For a more detailed description of MCTS, the reader is referred to [19].

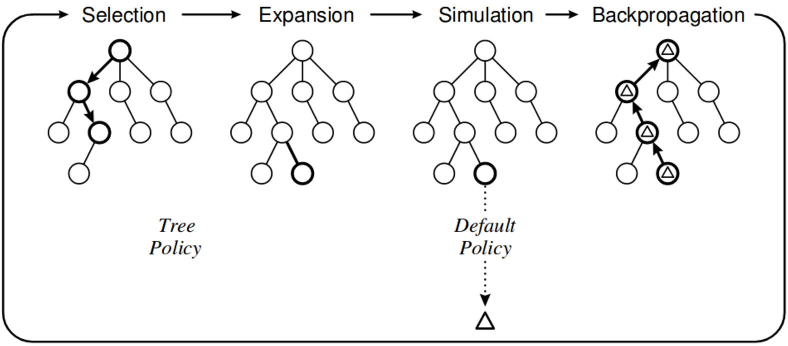


Figure 2.3: The 4 steps of MCTS: Selection, Expansion, Simulation, Backpropagation (graphic from [19]).

2.4.2 MCTS for POMDPs

The idea of using MCTS for POMDPs was introduced in [96] and has been advanced by ABT [58]. In the following, the ABT algorithm is explained in detail as it serves as an algorithmic foundation in this thesis.

ABT describes the root node b of the belief tree $\mathcal{T}$ by a set of particles. To construct $\mathcal{T}$, ABT selects randomly one of the particles of the root node and samples a so called episode u (see figure 2.4). An episode is one possible scenario with a maximum length of the planning horizon t_{hor}. It is described by a sequence of quadruples (x, a, o, R). The tree is traversed by the UCT algorithm during the *selection* step (see section 2.4.3 for further details). If an expandable node is reached, a not yet expanded action is sampled. This is followed by sampling a new state x' and by sampling a corresponding observation o' using the transition function T and the observation function Z. The newly discovered belief state b', is therefore described by only one state particle until further episodes reach that belief. From that belief, an optimal roll-out strategy is determined *online* in the *simulation* step (see section 2.4.5 for further details on the roll-out) to get a first estimate of $V(b')$. The goal of the ABT algorithm is to approximate the optimal policy π^*(Equation (2.3)).

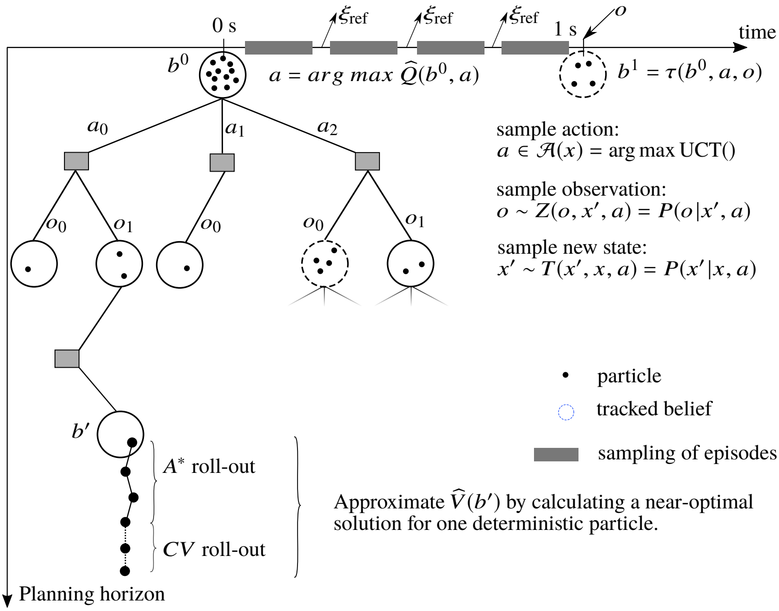


Figure 2.4: Construction of the belief tree by *online* sampling of possible episodes (graphic from [124], ©2018 IEEE).

Estimating $\pi^*(b)$ is done by estimating the Q-function $Q(b, a)$ first. It describes the expected reward, given a certain action, s.t.

$$Q(b, a) = R(b, a) + \gamma \sum_{o \in O} \tau(b, a, o) V^*(\tau(b, a, o)). \tag{2.9}$$

This allows to retrieve the optimal policy $\pi^*(b)$ as

$$\pi^*(b) := \underset{a \in \mathcal{A}}{\arg\max}\ Q(b, a) \tag{2.10}$$

with $\tau(b, a, o)$ being the belief update function, s.t. $b' = \tau(b, a, o)$, given the old belief b, the received observation o and the previously executed action a.

In case of an MDP, the state-value Q-function, $Q(x, a)$, is directly estimated via the cumulative rewards following action a. The difference in the case of

POMDPs is that the Q-function is estimated via the recorded episodes u. After every episode, the rewards of the episode are *backpropagated* and $Q(b,a)$ of passed beliefs is updated. The Q-function is estimated as

$$\widehat{Q}(b,a) = \frac{1}{|U_{(b,a)}|} \sum_{u \in U_{(b,a)}} V(u,n) \tag{2.11}$$

with $U_{(b,a)} \subseteq U$ being the subset of the set of all sampled episodes U, that contain the sequence (b,a). The depth of the tree is defined as n, the value of an episode u starting from depth n is defined as $V(u,n)$. The value of a history $V(u,n)$ is defined as

$$V(u,n) = \sum_{i=n}^{|u|} \gamma^{i-n} R(u_i.x, u_i.a). \tag{2.12}$$

2.4.3 UCT Action Selection

Action selection is required during the *selection* step and in the *expansion* step. Both can be formulated together as

$$a := \begin{cases} \sim \mathcal{A}' & ,\text{if} \mathcal{A}' \neq \emptyset \\ \underset{a \in \mathcal{A}}{\arg\max}\ \widehat{Q}(b,a) + c\sqrt{\frac{log(|U_b|)}{|U_{(b,a)}|}} & ,\text{otherwise} \end{cases} \tag{2.13}$$

with $\mathcal{A}'$ being the uniform distribution over all actions which have not yet been selected in the corresponding belief state, $|U_b|$ the number of episodes containing belief b and $|U_{(b,a)}|$ the number of episodes executing action a in b. A scalar, proportional exploration coefficient is defined as c and allows to trade off between exploration and exploitation. This type of action selection is called UCT [52].

The benefit of the UCT algorithm is twofold. First, it balances exploration and exploitation. This avoids searching the belief space exhaustively and at the same time allows to estimate promising solutions precisely very fast. Secondly, the search focus on promising actions is needed to converge to the optimal Q-function, given enough runtime. The Q-function $\widehat{Q}(b,a)$ is estimated via the mean of all episodes starting at b. It does only converge to the optimal Q-function $Q^*(b,a)$ if the actions of the optimal policy are selected more often via the UCT action-selection (see equation (2.13)). Despite that the algorithm may find non-optimal solutions given not enough runtime, it is observed that simply more conservative policies are found in this case. This is the case as the branch of the currently best action is most likely expanded and even an unlikely collision is found very fast.

2.4.4 Belief State Tracking and Observation Clustering

The current belief state is represented via a set of particles containing the possible state instances (see e.g. Figure 2.4). The belief state is tracked with an unweighted particle filter using simple rejection sampling for two reasons (see Alg. 2). First, the probability distribution of the belief state may become very different to Normal distributions during the interactive forward simulation. Therefore more exact filtering methods cannot be used. Secondly, tracking the belief state directly in the belief tree allows to conserve the relevant part of the tree and not to reconstruct the complete tree anew. This is possible by using the subtree which is constructed by all the episodes, that contain the sequence of the previously selected action and the new belief.

Observation Clustering

The belief update as well as the sampling of episodes in the belief tree requires the binary comparison of two continuous observations. This is done by comparing two observations, o_1 and o_2, with their maximum Euclidean distance $o_{\max}$ as follows:

$$o_1 = o_2, \text{iff } ||o_1 - o_2||_2 \leq o_{\max}. \tag{2.14}$$

During the simulation of the belief tree, the observations which are following a certain action a must be clustered into a discrete number of possible observations. This is the case as the structure of a tree can only be generated when a discrete number of observations exist. Respecting the continuous nature of the observation space also in the belief tree would lead to an infinite number of possible observations.

The clustering method is presented in figure 2.5. Every possible observation cluster is defined by a certain observation $o_{1:4}$. If a new observation arrives,

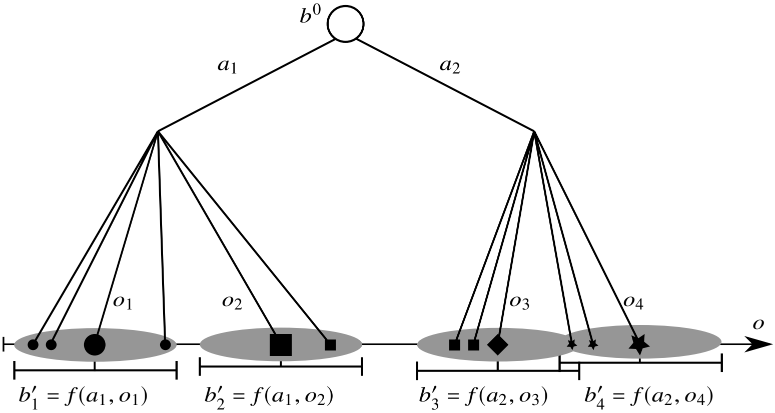


Figure 2.5: The clustering of continuous observations into a discrete set of possible observation clusters. Every observation cluster is defined by one certain observation $o_{1:4}$ which defines the center of an observation.

it is tried to match it on one of the existing observation clusters. If this is not possible, the new observation defines a new observation cluster itself. This may lead to observation clusters which do overlap as shown in figure 2.5 for the clusters generated by observations o_3 and o_4. A new observation which matches both clusters is assigned to the cluster, which has been generated first. This clustering is suboptimal due to two reasons. At first the center of the clusters is selected without having seen all the data. Secondly, possible overlapping clusters can introduce a higher uncertainty in a belief on a certain depth of the tree than necessary. This may result in a suboptimal policy.

2.4.5 Calculating Optimized Roll-Outs

As soon as a new belief state is explored (labeled b' in figure 2.4), the initial value of the belief state is set to a heuristic estimate to allow the UCT algorithm to converge faster. This can be done by a default roll-out strategy or even a random walk until the planning horizon. Nonetheless, a good approximation of the optimal value function of the newly explored belief $\widehat{V}^*(b)$ allows the algorithm to converge by magnitudes faster [19]. In this theses, the heuristic value is calculated by solving a deterministic, simplified problem *online* as soon as a new belief state is encountered. As the new belief state is at that point described by one sample only, the belief state is deterministic until more episodes pass this belief. By additionally removing the Gaussian noise on the motion models of the other agents, the planning problem becomes deterministic.

The optimization problem is solved by either a Dijkstra or A^* graph search throughout the thesis. Solving a graph search on a longer horizon is computationally too expensive for use as a heuristic. This is the case as the heuristic will be called once per episode, i.e. several 1000 times during one optimization run. Therefore, an approximation is used to the complete search by aborting the graph search after a certain number of steps and using constant velocity actions afterwards. Throughout this thesis, optimization is used for the first three steps, followed by a constant velocity roll-out until the optimization horizon.

2.4.6 Creating Consistent Plans

It is desired that the behavior layer creates consistent plans. This means that the reference trajectory does not jump from one behavior to another without major changes in the predicted behavior of the other agents. Nonetheless, this undesired effect can occur because of the fact that the algorithm replans frequently and that the reference trajectory is not tracked accurately. This is the case as the trajectory planner optimizes the trajectory with another cost function, focusing on comfort instead of behaviors. Next to this desired deviation of the trajectory planner, the controller itself is not able to track the trajectory exactly. Therefore, the driven trajectory deviates from the planned ξ_{ref} (see figure 2.6 for a visualization). In the case of replanning the trajectory, which happens frequently in a receding horizon control, the planner may find another, cheaper maneuver due to the different start state. To prohibit such a jump in behaviors, the current desired state on the reference trajectory instead of the measured state is used for replanning. The idea is presented in figure 2.6 with an example based on trajectories but works in the same way for closed loop planning with policies. This method is also used for replanning of trajectories without integrating a drift due to control errors [110].

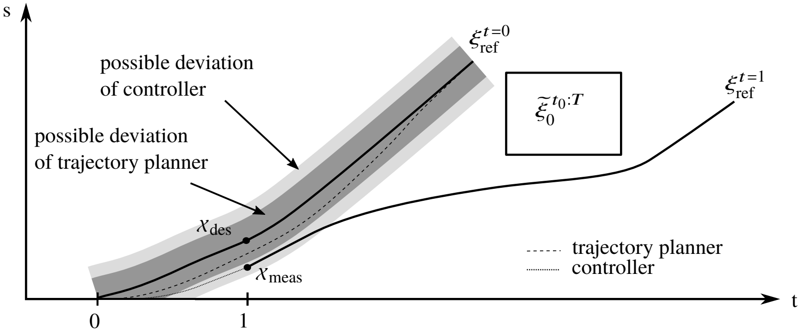


Figure 2.6: The trajectory planner as well as the control algorithm deviate from the reference trajectory ξ_{ref}. Frequent replanning behavior may therefore result in a jump from one maneuver ($\xi_{\text{ref}}^{t=0}$, i.e. passing before the dynamic agent) to another ($\xi_{\text{ref}}^{t=1}$, i.e. passing behind the dynamic agent). To avoid this undesired behavior, replanning at $t = 1$ is performed from the currently desired state $x_{\text{des}} = \xi_{\text{ref}}^{t=0}(1)$ instead of the currently measured state x_{meas}.

2.4.7 Batch Sampling of Episodes

The ABT algorithm is anytime. Therefore, the planner has to trade off between the used time interval for the optimization of the policy and the introduced delay until sampling of episodes is done and a policy is available. The more time is used for sampling of episodes, the better the policy is approximated. On the other hand, the earlier the policy is send to the trajectory layer, the less delay is introduced between the sensor measurements and the corresponding policy. Hence, it is chosen in this work to approximate the policy by sampling possible episodes for 200 ms (see figure 2.4). After 200 ms, the reference trajectory ξ_{ref} is extracted from the approximated, optimal policy $\widehat{\pi}^*$. It is send to the trajectory planning layer for execution. The reference trajectory is extracted from the optimal policy by choosing the most likely trajectory which is included in the policy:

$$\xi_{\text{ref}} = \arg\max_{\xi \in \pi} P(\xi). \tag{2.15}$$

While the solution is optimized for 200 ms, a step size of $\Delta t = 1s$ is used to construct the tree to allow for a planning horizon of 8-10 s. Therefore, the particle filter can only match an observation, which is received after 1 s. The spare time until a new observation arrives is used to sample additional episodes, to make the tree more robust. This is done in several blocks of 200 ms and the updated ξ_{ref} is send to the planning layer after every block (see figure 2.4 for more details).

2.5 Reducing the Dimensionality of the Action Space

The autonomous agent is operating in a dynamic, 2-dimensional workspace (assuming a planar environment). To allow the autonomous agent to reach various configurations in the workspace, the action space $\mathcal{A}$ must contain actions for its longitudinal accelerations as well as different steering angles and also the combinations thereof. Nonetheless, the size of the action space grows exponentially with its dimension, due to combinations of actions in different dimensions. Even if the 2-dimensional action set $\mathcal{A}$ is reduced by non-holonomic constraints, it may be intractable to solve if the number of discrete actions in each dimension increase.

The idea of the *path-velocity decomposition* [48] is to decompose the problem into two simpler subproblems:

1. plan a path around static obstacles while considering kinematic constraints
2. plan the longitudinal speed profile considering dynamic objects and dynamic constraints.

The *path-velocity* decomposition is applicable, as long as the path is independent of the longitudinal velocity. As this is the case for many scenarios of automated driving (see [113] for a comparison of scenarios), the *path-velocity* decomposition is widely used for motion planning algorithms for autonomous driving. In this thesis, *path-velocity* decomposition is used throughout all chapters, except for the algorithms concerning lane changes (chapter 5) and in parts for the algorithms handling occlusions (chapter 6). These scenarios present a strongly coupled problem which require combined longitudinal and lateral optimization.

3 Planning for Combinatorial Decision Making

This chapter describes an optimal behavior planner for urban environments. Most current motion planning frameworks are based on rule-based decision making for one of a discrete set of potential maneuvers (see section 3.1). Such an approach relies on an a priori selection of a discrete set of maneuvers. Nonetheless, the a priori definition of such a set as well as the design of a logical-reasoning based arbiter for the maneuver decision may become infeasible in complex, urban environments. This is the case as their countless, different road topologies including intersecting lanes (as opposed to highways), a varying number of other vehicles/pedestrians and multiple traffic rules lead to a large amount of parameters to be considered. This is either intractable with a rule-based system or leads to suboptimal behavior.

The main contribution of this section is the presentation of a globally optimal planner that optimizes in the space of behaviors and trajectories. Its non-convex formulation allows for planning of global optimal trajectories on a *receding* time horizon. That implies that the planning algorithm itself allows for implicit decision making. It considers various different events and constraints at the same time in the optimization formulation. These are traffic rules (e.g. traffic lights, speed limits) as well as dynamic objects which are formulated as constraints. The planner is an *open loop* planner, i.e. an estimated future trajectory of the other agents is considered but future, possible observations are not considered. The result is a reference trajectory representing an optimal behavior which contains all decisions implicitly. A key idea of the algorithm is to do a *path-velocity decomposition* (see section 2.5) first and optimize the velocity only in longitudinal direction on the preplanned path. While this does not allow for combined, longitudinal and lateral optimization, the optimal solution may still be found in most scenarios (see [113] for a comparison of the different cases). The longitudinal formulation enables fast solving of an A^* formulation by use of a heuristic based on the idea of Inevitable Collision

States (ICS) [31]. The provided trajectories are dynamically feasible, safe and legal. Also, comfort is optimized over the complete planning horizon. The presented deterministic planner can be used as a standalone, *open loop*, behavior planning module. Additionally, it is also used as heuristic itself in the presented algorithms in the following chapters.

The chapter is based on and was previously published in [121].

3.1 Related Work

Solving the global optimization problem under the high demand of combined longitudinal and lateral comfort optimization and environment constraints is considered to be computationally intractable [116]. Nonetheless, motion planning algorithms which solve single, capsulated subproblems, exist. In [71, 63] the approaching of and decision making at traffic lights is handled from an energy-optimal perspective while [72] minimizes jerk while making crossing decisions at intersections. Anticipatory, energy efficient approaching on slower vehicles is presented in [55]. A supervised learning model for car following behavior is presented in [114]. An algorithm which also does a path-velocity decomposition [48] and plans in the velocity-time frame is presented in [43]. It creates a graph first by planning trajectories between the edges of crossing vehicles and searches the generated graph for the minimum-cost solution afterwards. This allows for global optimization but on a very sparse graph.

As these algorithms only solve certain subproblems, a local trajectory planner is normally embedded in a framework, where a higher layer does the decision making for a certain behavior and parameterizes the trajectory planner accordingly. These decision making systems are often rule-based and for example formulated as a decision tree [3] or as a state machine as teams of the DARPA Urban Challenge [107, 117] did. Another approach is to represent the situation with high-level, semantic states and search the generated graph in a second step [53]. Nonetheless, the retrieved solution may be dynamically infeasible and must therefore be validated [54]. The approach also requires a set of rules which describe how the current situation may be processed into the high-level, abstract state space. Instead of rule-based systems, also the maneuver with the minimum acceleration or minimum total cost may be chosen, as done in [111, 56]. While the approach of only considering a limited amount of predefined maneuvers is feasible on highways, this may become intractable in urban environments due to the high amount of varying topological situations and corresponding maneuver possibilities. Especially, more complex maneuvers such as *early braking during following behavior because of a traffic light switching to red in a larger distance* is difficult to be designed by a rule-based system.

None of these algorithms is able to find the optimal maneuver in urban environments under the combined consideration of comfort-optimization, respecting traffic rules and a varying number of dynamic objects.

3.2 Problem Formulation

The algorithm expects a global path r_0 of the autonomous vehicle N_0. This path can be retrieved by a path planning algorithm such as the variational, local planner presented in [119]. The path must be at least represented by sampled global positions $q_i \in \mathbb{R}^2$ and $i \in [0, \dots, I], I \in \mathbb{N}$, with an assumed, sufficiently dense, spatial sampling distance such that an approximate transformation to the Frenet frame is possible. The transformation to the Frenet frame describes the arc length $s \in \mathbb{R}$ along the path r_0 from a path origin q_0 to the current point q_i as: $q_i \rightarrow s$ [112].

The absolute velocity is bounded by $\dot{s} \in [0, v_{\max}]$ with $v_{\max}(s)$ being a function of the path's curvature κ at distance s, i.e. $v_{\max}(s) = f(\kappa(s))$ and the vehicle's acceleration a being the system's bounded input $a \in [a_{\min}, a_{\max}]$. The motion of the autonomous car on its path can be described by a set of linear, differential equations:

$$\begin{bmatrix} \dot{s} \\ \ddot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s \\ \dot{s} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} a. \tag{3.1}$$

Along the given path r_0 exists a finite set E of events e_i that are described by 4-tuples: $e_i = \left(t^{es}_{\text{start}}, t^{es}_{\text{end}}, s^{e_i}_{0,\text{start}}, s^{e_i}_{0,\text{end}} \right)$. The event e occupies the lane for a time interval $[t^{e_i}_{\text{start}}, t^{e_i}_{\text{end}}]$ at positions $\left[s^{e_i}_{0,\text{start}}, s^{e_i}_{0,\text{end}} \right]$. These events must not intersect with the position s of the autonomous vehicle at any time. In addition, a finite set L of traffic laws $l_i(s)$ is imposed along the road and limits for example the absolute velocity. The goal of the driving strategy is to generate an optimal behavior in longitudinal direction, represented as trajectory. This may be formulated as the following combinatorial optimization problem:

$$a(t) = \arg\min_{a(t)} \sum_{t=0}^{t_{\text{hor}}} J(s, \dot{s}, \ddot{s}, \kappa(s), E, L). \tag{3.2}$$

As this problem does, in general, not only have one global minimum but different local ones (i.e. different maneuvers) as well as various constraints (e.g. speed limits), it is a constrained non-convex problem.

The task of the behavior planner is hereby to find a global optimum, i.e. a long term solution ($t_{\text{hor}} \geq 10s$) for the combinatorial optimization problem. The reference solution is then provided to the trajectory planner presented in [112] and executed in a jerk optimal way. This combination of a global and a local planner is described in section 1.5 illustrated in figure 1.13.

3.3 Approach

This algorithm plans global optimal behavior that consider traffic laws, long term comfort and human driving conventions.

The corresponding optimization problem is formulated as a global, discrete planning problem [61] and solved with an A^* graph search [86]. The state space $\mathcal{X} \subseteq \mathbb{R}^3$ describes only the configuration of the autonomous robot. This is sufficient as the problem is solved *online* and because the other agents are predicted independently and because possible future measurements are not taken into account. Therefore, the state is described by $x = [s, v, t]^{\mathrm{T}} \in \mathcal{X}$, with s being the longitudinal position along the path, v being the longitudinal velocity and with t being the the corresponding time. The search graph is expanded *online* by use of a set of discrete set of actions $\mathcal{A}$, used for a sample time of Δt. As no states with a negative velocity, $v_i < 0$ are considered, the result is a directed, acyclic graph.

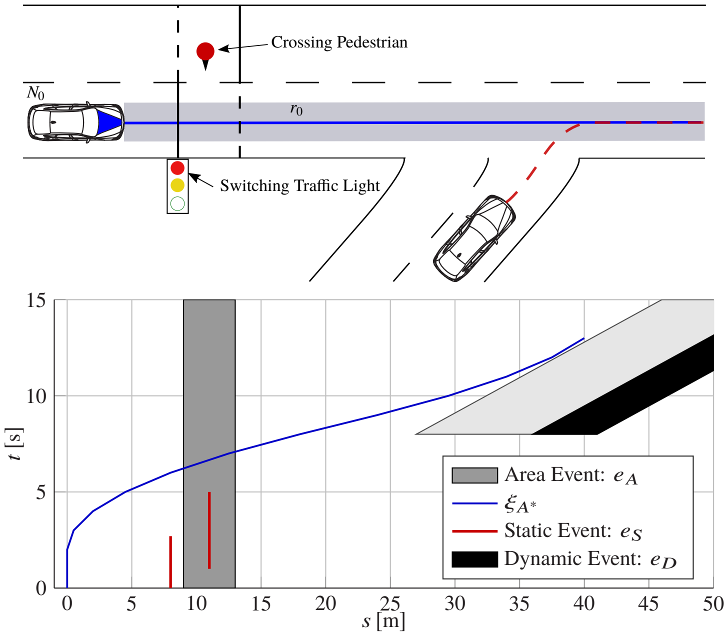


Figure 3.1: Idea: The algorithm maps a typical urban traffic scene (top figure) into a spatio-temporal cost map (bottom figure) along the planned path of the autonomous car. The objects of the traffic scene are represented as constraints in the free space of the cost map. This allows to find a global optimal behavior while respecting various events simultaneously (graphic from [121], ©2016 IEEE).

3.3.1 Transition Model

The discretized state transition from a state x to the next state x' is formulated as

$$x' = \begin{bmatrix} s' \\ v' \\ t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \end{bmatrix} \begin{bmatrix} s \\ v \\ t \\ 1 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \\ \Delta t \\ 0 \end{bmatrix} a \tag{3.3}$$

with a being the action of the autonomous car, i.e. the acceleration, selected in state x and executed for Δt.

3.3.2 Cost Function

The step cost $J(x, a, x', E, L)$ is the cost for taking action a in state x to traverse to state x'. The total cost of a trajectory is the sum of all intermediate costs [86, p. 68], s.t.

$$J(\xi) = \sum_{x = x_{\text{start}}}^{x_{\text{goal}}} J(x, a, x', E, L). \tag{3.4}$$

The goal is to find the path from the start state to a goal state with minimal costs. A sum of different costs terms is used to describe the different optimization objectives:

$$J(x, a, x', E, L) = J_{\text{V}}(x', L) + J_A(a) + J_E(x, x', E) \tag{3.5}$$

with $J_{\text{V}}(x')$ being the cost for any deviation to the desired speed, $J_A(a)$ being the cost for taking action a and $J_E(x, x', E)$ being the cost for a collision while traversing from x to x'.

Action Set

The discrete set of actions $\mathcal{A}$ represents different accelerations during a discrete planning step Δt. Punishing accelerations quadratically reduces the duration and intensity of acceleration which increases the driver's comfort. Therefore, the acceleration costs are defined as:

$$J_A(a) = a^2. \tag{3.6}$$

Reference Velocity

A so called reference velocity is defined to guide the autonomous vehicle to drive with appropriate velocity. Therefore, a desired speed $v_{\text{des}}(s)$ is defined along the planned path, r_0, of the autonomous car. The desired speed at position s is the desired speed under the assumption that no events exist. It is a combination of the current legal speed limit $v_{\text{law}}(s)$ and $v_{\text{curve}}(s)$, i.e. the limit introduced by the road's curvature. The maximum curve speed v_{curve} is defined as in [56] via a maximum allowed lateral acceleration, $a_{lat,max}$, in the curve which is defined by its curvature $\kappa(s)$ along the path of the autonomous vehicle:

$$v_{\text{curve}}(s) = \sqrt{\frac{a_{\text{lat,curve}}}{\kappa(s)}}. \tag{3.7}$$

The desired speed is retrieved by applying a smoothed curve approach filtering to the minimum of the different velocities:

$$v_{\text{des}}(s) = \min(v_{\text{law}}(s), v_{\text{curve}}(s)). \tag{3.8}$$

This is shown in figure 3.2.

The velocity-dependent costs, J_{V}, are defined by the deviation to the desired velocity v_{des}. Too high velocities are punished quadratically, too low velocities are punished linearly to allow lower velocities during decelerating upon events (as red traffic lights). $J_{\text{V}}(x_{i+1})$ is then defined as follows:

$$J_{\text{V}}(s') = \begin{cases} (v' - v_{\text{des}}(s'))^2, & v' > v_{\text{des}}(s') \\ 0, & v' = v_{\text{des}}(s') \\ \frac{1}{2}(v_{\text{des}}(s') - v'), & v' < v_{\text{des}}(s') \end{cases}. \tag{3.9}$$

This is done for curve approaches but not for curve departures. This is the case, as the approach phase helps the algorithm to converge faster at these points. When leaving curves, the delta to the desired velocity automatically draws the velocity to the desired velocity.

Representation of the Environment

Along the path, there may be merging/crossing cars, traffic lights and crossing pedestrians. Independently of their different causes of existence and type, a behavior planner must incorporate all events in the decision making process. The idea of this work is to present a very generic event definition, capable of presenting arbitrary scenarios along the road. The set of existing events is defined as $E = \{e_1, e_2, \ldots, e_I\}$. A set of events may consist of different types of events such that $E = E_S \cup E_D \cup E_A$ with E_S and E_A representing events with

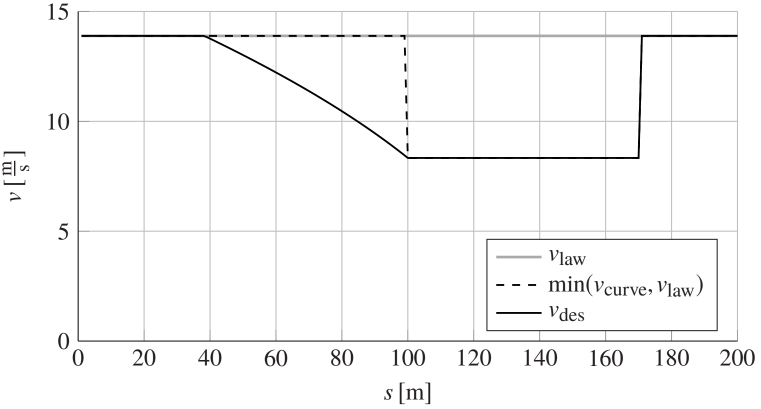


Figure 3.2: Along the path r_0, the desired velocity is limited by speed limits and the curvature. The desired velocity is retrieved by smoothing the jumps to lower velocities with an approach deceleration (graphic from [121], ©2016 IEEE).

a constant position/area on the path r_0 and E_D events with a time-dependent position.

A static event e_S prohibits the autonomous vehicle to traverse a certain position s on its path r_0 at a certain position during a time interval $\left[t_{\mathrm{start}}^{es}, t_{\mathrm{end}}^{es}\right]$ and can therefore be defined with the 3-tuple

$$e_S = \left(t_{\mathrm{start}}^{es}, t_{\mathrm{end}}^{es}, s^{es}\right). \tag{3.10}$$

Dynamic events on the other hand have a time dependent position such that $s^{es} = f(t)$. In addition, a specific dynamic event e_D has a defined length of the corresponding object, l^{e_D}, and therefore a plane in the position-time dimension of the state space $\mathcal{X}$ (see Fig. 3.1). It also allows for the definition of a following distance d^{e_D}, which defines a spatio-temporal cost map M^{e_D}, to realize a smooth following behavior of the autonomous car. The cost map M^{e_D} is realized as an increasing linear function, defined by $s^{e_D}(t), d^{e_D}, l^{e_D}$ in the interval $[t_{\mathrm{start}}^{e_{D,i}}, t_{\mathrm{end}}^{e_{D,i}}]$.

Therefore, the dynamic event is defined as the 5-tuple

$$e_D = \left(t_{\text{start}}^{e_D}, t_{\text{end}}^{e_D}, s^{e_D}(t), d^{e_D}, l^{e_D}\right). \tag{3.11}$$

Area events are defined as a conditional spatial cost map. They get activated when the autonomous vehicle is in the area and $v = 0$ holds. Therefore, they are only defined by a spatial tuple, such that

$$e_A = \left(s_{\text{start}}^{e_A}, s_{\text{end}}^{e_A}\right). \tag{3.12}$$

The total event-based costs are defined as

$$\begin{aligned} J_E(\chi, \chi', E) &= J_{E_S}(\chi, \chi') + J_{E_D}(\chi') + J_{E_A}(\chi') \\ &= \sum_{E_S} J_{e_S}(\chi, \chi') + \sum_{E_D} J_{e_D}(\chi') + \sum_{E_A} J_{e_A}(\chi') \\ &\text{with} \\ J_{e_S} &= \begin{cases} \infty, & \text{if} \overrightarrow{\chi\chi'} \cap e_S \neq \emptyset \\ 0, & \text{otherwise} \end{cases}, \\ &\text{and} \\ J_{e_D} &= \begin{cases} \infty, & \text{if} \chi' \in e_D \\ M^{e_D}(\chi'), & \text{if} \chi' \in M^{e_D} \\ 0, & \text{otherwise} \end{cases}, \\ &\text{and} \\ J_{e_A} &= \begin{cases} c_{\text{area}}, & \text{if} s \in \left[s_{\text{start}}^{e_A}, s_{\text{end}}^{e_A}\right] \cap v = 0 \\ 0, & \text{otherwise} \end{cases}. \end{aligned} \tag{3.13}$$

The motion primitive connecting χ and χ' is defined as $\overrightarrow{\chi\chi'}$. Allowing the ego to stop on an area can be balanced with the area costs, c_{area}.

The various, different event types define interfaces which allow to frame relevant things along the road as events. This enables easy extension of the behavior planner as extending simply means to frame a new event as one of the different event types and add it to the cost function. The behavior planner will immediately respect the new event and consider it in the combined decision making process. Hence, adding an event is completely independent from the other events. This limits the complexity of the algorithm from a design perspective as every event may be considered and added independently of the others.

Modeling of Traffic Lights as Dynamic Events

The decision making and speed adaptation at traffic lights motivated different other publications [72,56,63]. Therefore, the representation of a traffic light as static event is demonstrated. The interval $\left[t_{\text{start}}^{es}, t_{\text{end}}^{es}\right]$ defines the forbidden (red phase) time interval, while s^{es} is the position of the traffic light. If no further CAR2X information is available, the current green and red phase is assumed to last forever. During a yellow phase, the legal length of the yellow phase is used to predict the traffic signal switch, such that t_{start}^{es} is the predicted start of the red phase and t_{end}^{es} is set to infinity. That way, the algorithm implicitly handles the decision to pass or not to pass a (recently switched) traffic light. While the algorithm's event is therefore independent to potentially available CAR2X information that information can be easily added if available by a different event handling.

Modeling of Leading/Merging Vehicles as Dynamic Events

A dynamic event e_D, which is in front of the autonomous vehicle is parameterized by the 5 tuple $\left(t_{\text{start}}^{e_D}, t_{\text{end}}^{e_D}, s^{e_D}(t), d^{e_D}, l^{e_D}\right)$. As the focus of this work is on the planning algorithm but not on predicting the longitudinal behavior of other vehicles, the velocity of dynamic events is assumed to be constant (i.e. a constant velocity prediction). Nonetheless, a better prediction function can be easily included by replacing the linear function $s^{e_D}(t)$. In addition, lane changes (onto the path r_0) are predicted with a simple rule-based classifier based on a threshold concerning the other vehicles lateral position and lateral velocity. The time interval $\left[t_{\text{start}}^{e_D}, t_{\text{end}}^{e_D}\right]$ of the corresponding event is set accordingly. Such an intention estimation enables foresighted decision making in terms of early reaction to the planned trajectory of the other vehicles.

Modeling of Crosswalks and Intersections as Area Events

Area events are used to allow the autonomous vehicle to cross certain areas, but to prohibit the autonomous vehicle to enter certain areas when they cannot be left again. This may be the case when the path of the autonomous vehicle lies on an oncoming/crossing lane (e.g. in the case of overtaking/intersection crossing) or on a zebra crossing.

3.3.3 Domain Specific Heuristics

The A* algorithm uses a heuristic to speed up the graph search by truncating non-promising branches early (see Alg. 1 for details). Such a heuristic must be admissible (underestimate the real costs) and consistent (the heuristic must be monotonically decreasing along a path to the goal). The idea of this work is to use the concept of Inevitable Collision States (ICS) [31] as a heuristic. An ICS is a state from which at least one collision is inevitable in the future given the available system input. When a new state x is generated, it is tested for being an ICS. If this is the case, the remaining estimated costs are at least the collision costs. By setting the heuristic value of the state, $h(x)$ to the collision costs, an admissible and consistent heuristic is found which furthermore allows to react to upcoming events which are currently ahead of the currently expanded graph depth or even the planning horizon itself. In the case of a movement in a one dimensional direction, the test for an ICS can be done analytically and is therefore fast enough to be used as heuristic. Formally written, a newly expanded state x may be labeled as an ICS if and only if

$$\forall a \in \mathcal{A}, \exists e \in E_S \cup E_D : \{s + v \cdot t + \frac{1}{2}at^2 | t \in [0 \;\; \infty[\} \cap e \neq \emptyset. \qquad (3.14)$$

The concept is demonstrated for static events in Fig. 3.3.

3.3.4 Goal State Formulation

A state x is defined as a goal state x_G if $t_G = t_{\text{hor}}$ as done for MPC approaches [60]. This ensures a constant behavior length in the time domain. Setting the goal state condition to a more complex equation, advanced problems may be

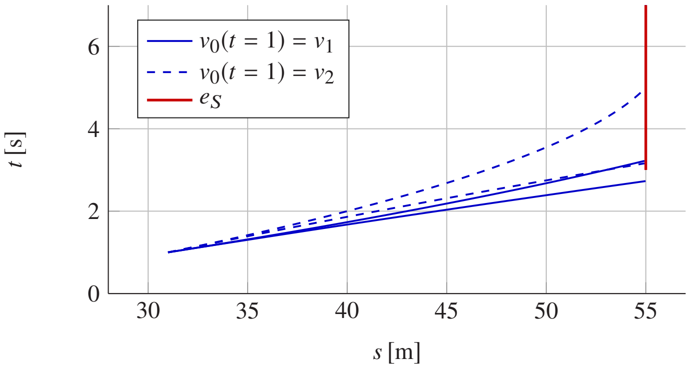


Figure 3.3: Analytic calculation of the Inevitable Collision States [31]. The plot shows the potential trajectories with maximum acceleration/deceleration for the two different states x_1 and x_2 with $v_1 > v_2$. If a collision is inevitable, the heuristic of this state may be set to ∞. It can be seen, that a collision cannot be avoided for x_2, while the static event can be avoided for state x_1. Therefore, $h(x_1) = 0$ while $h(x_2) = \infty$ (graphic from [121], ©2016 IEEE).

tackled. The behavior planner may be used for example for gap approaches for lane changes as demonstrated in successive work of this algorithm [113].

3.3.5 Implementation

As the A^* planner considers only one simple prediction for the surrounding traffic, it must run with a higher frequency to reactively account for sudden changes in the environment. Therefore, the behavior planner is set to a re-planning frequency of 10 Hz. As the behavior planner shall only provide a reference solution for the trajectory planning layer, the step size is chosen in a coarse way of $\Delta t = 1$ s to allow for a long planning horizon of $t_{\text{hor}} = 13$ s. The set of actions $\mathcal{A}$ is $\mathcal{A} = \{-2, -1, 0, 1\}$. It is important that the behavior planner generates consistent behavior. Therefore, instead of planning from the actual, measured state $x_{\text{meas}}(t_0)$ the currently desired state, retrieved from the previous planning step, $x_{\text{des}}(t_0)$ is used as the start state x_{start} (see section 2.4.6). Furthermore, to fulfill Bellman's Principle of Optimality in a discrete planning problem, the actions must be sampled at the same absolute points in time. This is impossible when sample steps of $\Delta t = 1$ s are used with a planning frequency

$f = 10\,\text{Hz}$. Therefore, the first planning step is not executed with a length of Δt but with the temporal difference to the last solution's second state.

To prevent the generated graph from expanding too many nodes, only the cheapest state of two very close states is expanded. This is a similar approach as done for the Hybrid A^* planner [73]. Closeness between state A and B is defined by

$$(t_A - t_B)^2 + (s_A - s_B)^2 + (v_A - v_B)^2 < 1. \tag{3.15}$$

3.4 Results

The behavior planner is implemented in the software framework for automated driving which is used at the BMW Group for research and development. It is tested in a complex simulation scenario which allows to show the capabilities of the planner.

3.4.1 Performance

The algorithm's performance is evaluated on a simulated round course containing four different intersections with traffic lights, various road curvatures and randomly generated traffic. The system runs on an Intel Core i7-4900MQ CPU at 2.8 Ghz. The runtime of the algorithm depends strongly on the length of the planning horizon t_{hor} and the micro traffic situation, i.e. the number and configuration of the different events. Therefore, the worst-case runtime is approximated empirically by running many simulations in an urban scenario, using different planning horizons. Figure 3.4 shows the runtime for a worst case scenario during driving on a evaluation circuit with four traffic lights and intersections. The average runtime is lower than the worst-case runtime by a factor of 10. While this gives an idea of the runtime of the planner, more efficient implementations exist.

3.4.2 Qualitative Simulation Scenario

To evaluate the different capabilities of the algorithm, a complex situation is set up and evaluated in a simulation run. The situation with its recorded

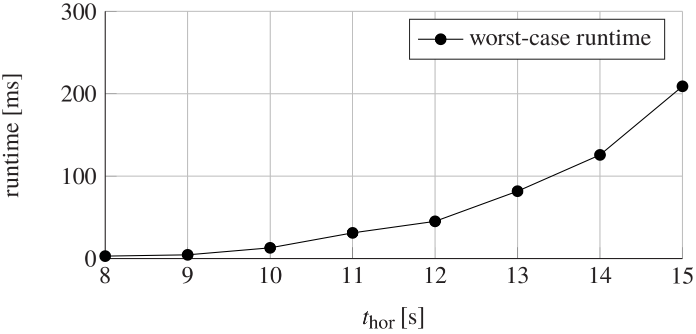
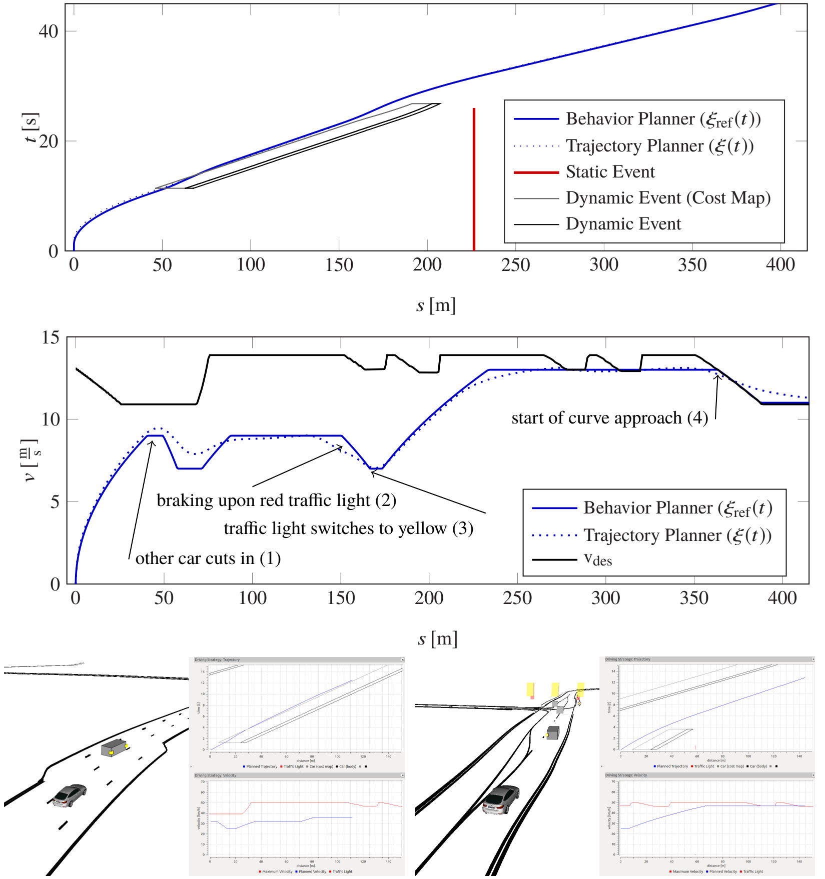


Figure 3.4: Empirical worst-case performance of the algorithm for different planning horizons t_{hor} during the simulation of the round course scenario (graphic from [121], ©2016 IEEE).

data is shown in figure 3.5 and described in the following. The autonomous car drives along a road, when another agent cuts in before it (Sit. 1). The other car is predicted to enter the autonomous vehicle's lane in two seconds. Adding this cut-in action as a dynamic event allows the autonomous vehicle to react anticipatory and cooperatively by starting to decelerate already before the other car enters the lane. Subsequently, the autonomous car follows the other car with ACC behavior, realized by the spatio-temporal cost map. At (2), the autonomous vehicle starts to brake upon a red traffic light, which is in front of the vehicle running ahead. At (3), the traffic light switches from red to yellow and the forbidden passing time (ongoing yellow period) of the traffic light is predicted to last for another second. In addition, it is detected that the preceding vehicle changes its lane to the left and it is predicted to actually leave the lane in two seconds. It can be seen in figure 3.5b, how the driving strategy optimizes its behavior over these different events. While the vehicle ahead is braking during its lane change, the autonomous vehicle already starts to accelerate to v_{des} as it incorporates the prediction of the vehicle in front (when it will leave the lane) and that the currently red-yellow traffic light will have switched at arrival time. It can also be seen, that the maximum velocity is constrained by the slight road curvature, such that v_{des} is lower than v_{law}. In this situation, the algorithm optimizes its behavior under consideration of other vehicles, a currently switching traffic light and the road's curvature.



(a) Snapshot of situation 1: A cut-in is detected by the autonomous car. The time of the entering the autonomous lane is predicted and allows for a foresighted reaction.

(b) Snapshot of situation 3: The traffic light ahead switches from red to yellow. Therefore, the current red-yellow phase is predicted to last for one second (duration of red-yellow phase). As the vehicle ahead is predicted to leave the autonomous lane in 4 seconds, the autonomous car can start to accelerate.

Figure 3.5: Overview of the driven trajectories of the simulation scenario, described in section 3.4.2 (graphic from [121], ©2016 IEEE).

3.5 Summary

This chapter demonstrates how a global, *open loop* planner can be used for implicit decision making for autonomous driving. Furthermore, it is shown how the provided reference trajectory of the global planner can be used for the parameterization of a local trajectory planner. The introduced global planner is able to consider various events and provides an optimal solution accordingly. Generic interface formulations for so called *static*, *dynamic* and *area events* allows for fast and simple extension of the algorithm to consider more incidents on the road. Although deterministic interaction could be considered in the framework, interaction as well as uncertainties are not modeled in this approach. The presented algorithm is only able to handle deterministic prediction(s) of the surrounding traffic but may reach its limits for scenarios with many agents, very uncertain prediction or required interaction. Nonetheless, this algorithm can be succesfully used for motion planning in the presented urban scenarios. This is, as fast replanning behavior may also allow to solve complexer scenarios including interaction and uncertain prediction. Additionally, this algorithm will serve as a heuristic for the probabilistic algorithms, presented in the next chapters of this thesis.

4 Planning with Uncertain Intentions of Crossing Traffic

The *open-loop* behavior planner in chapter 3 transfers traffic rules and the predicted behavior of other agents in the environment directly into a spatio-temporal cost map. That planner considers only the most likely prediction of every agent and does neglect the information of possible future observations. The uncertainty in the prediction is addressed by frequent replanning. This is possible if the various possible future scenarios do not differ heavily (e.g. ACC scenarios).

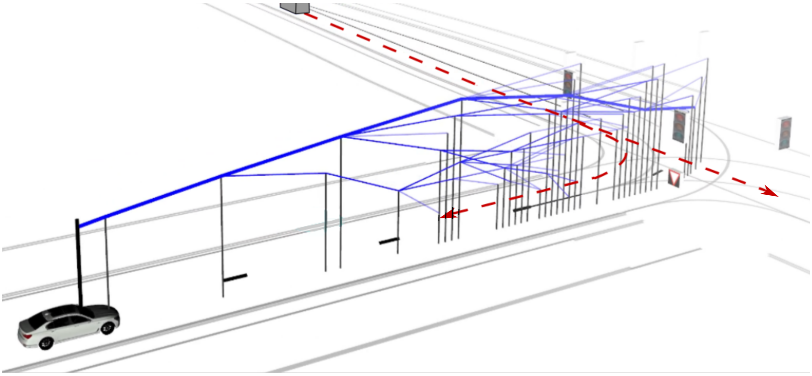


Figure 4.1: Visualization of the *closed-loop*, *online* algorithm: The planning algorithm approximates the optimal policy on a receding horizon for the most probable future scenarios *online*. The optimal policy π^* is shown in blue, plotted with its velocity over the longitudinal distance. The policy branches for cases where different, possible, future observations lead to different optimal actions for the autonomous car. It even finds behaviors for the different homotopy classes automatically. The other vehicles follows one of two possible paths (drive straight, turn right) and are modeled with interactive, probabilistic driver models.

In urban environments on the other hand, the manifold of possible paths of the other vehicles (i.e. intention uncertainty) is larger due to a road topology with splitting/joining/crossing road elements (e.g. at intersections). Additionally, sensor and model uncertainty create even more possible predicted trajectories in even different homotopies (pass before or after). Moreover, the uncertain interactive nature of the agents must be modeled to account for the interrelationships between the actions of the agents. A summary of all these uncertainties is displayed in figure 1.3.

Extending the idea of the A^* planner in chapter 3 by simply adding every possible maneuver in the spatio-temporal cost-map would lead to very conservative behavior or even standstill [103]. This is the case as the autonomous vehicle plans a trajectory which avoids every possible future trajectory of the other agents. In the worst case, the only safe trajectory is standstill.

To overcome this drawback, this chapter presents a problem formulation as global, *closed-loop* planner on a receding horizon. A POMDP is used to formulate the problem due to its generic nature (see section 2.3 for the formulation). The solution to a POMDP is an optimal policy instead of an optimal trajectory which optimizes the expected reward, starting from an initial belief state. It contains reactive plans for possible, future observations. A path-velocity decomposition is used to design a longitudinal planning problem. The capabilities of the planner are demonstrated for the crossing of intersections with a various number of other agents and road geometries.

The key contributions of this chapter are as follows:

- consideration of uncertain driver models and uncertain intentions for other agents
- consideration of interactive behavior of the other vehicles
- explicitly take possible, future observations into account
- *online* optimization of a *closed-loop* formulation on a continuous state and belief space
- combination of MCTS with a deterministic A^* roll-out heuristic for fast convergence to the optimal policy

This chapter is based on and was previously published in [122, 124, 129].

4.1 Related Work

Low-level motion planning approaches separate the problems of planning and prediction and consider uncertainties most times only in a limited way. These simplifications are done to allow to formulate the problem in continuous time with continuous actions. Hence, it is formulated on a state space of higher dimension including e.g. derivatives of acceleration and velocity of steering angle [38]. This allows to plan a very smooth, continuous trajectory which is e.g. jerk optimal [112].

In the following, algorithms which do respect uncertainties and/or interaction are reviewed. The focus is strongly on algorithms in the context of intersection crossing of autonomous vehicles.

Uncertainty

One possibility to integrate the controller uncertainties directly in the controller itself is demonstrated in [66]. By using a stochastic MPC with chance constraints, the execution uncertainty of the robot may be directly considered in the controller.

The well-known sampling-based RRT* algorithm is extended in [7] to include localization and controller uncertainties via a Gaussian belief space.

The authors of [24] build so called risk maps with the existence probabilities of other vehicles given their potential future maneuvers. Then a trajectory is planned on the combined risk maps, which therefore reacts to the most probable case. In a second step, the generated trajectory is extended with branching back-up trajectories for the case that improbable scenarios are happening.

Interaction

Interaction can be modeled as a multi-agent planning approach [90]. By preselecting a discrete set of possible, collective maneuvers, each problem can be formulated as QP problem and solved via a MIP. The maneuver of the other vehicles is unknown but tracked with an Interacting Multiple Model (IMM) filter.

In [118], a reactive model is integrated in the prediction model of the other drivers. The variational problem formulation is extended with a dynamic programming approach over the possibly interacting trajectories. This allows to consider interaction while using a variational formulation.

Belief State Planning with a POMDP

A popular formulation of a belief state planning algorithm is a POMDP (section 2.3). This is because its generic formulation allows to describe combinatorial optimization problems with state and model uncertainty, while no limitations on the transition function exist. Nonetheless, this generic formulation makes it also difficult to solve. Therefore, it is often solved *offline*. In [92], the merging at a T-Junction is formulated as a Mixed Observability Markov Decision Process (MOMDP) on a discrete action and observation space. A simple behavior model is used for the other agent, based on their intended route at the intersection and the level of aggressiveness. The authors demonstrate promising results for one real-world scenario.

While these results are promising, *offline* approaches do not generalize well for complex scenarios with a variable number of agents and lanes (see section 2.3.4). In [5], a similar scenario to the one presented in [92] is shown and solved *online* with the MCTS based solver Partially Observable Monte Carlo Planning (POMCP). While the state space and action space is discretized, promising results are shown for various planning problems in environments with dynamic agents.

The authors of [4] use a *online* POMDP for the navigation in environments, densely populated with pedestrians. They represent the unknown intentions of the pedestrians as latent variables in their belief state. As they are operating at low speeds, a safe state can be reached very quickly.

The complex POMDP model is simplified in [32] by having a discrete set of policies instead of actions for the other vehicles as well as for the autonomous vehicle. A Bayesian model and the Viterbi algorithm are used to calculate the belief state over possible policies of the other agents *online*.The policy for the autonomous vehicle is then selected by the expected reward of each policy, given the most probable policies of the other vehicles.

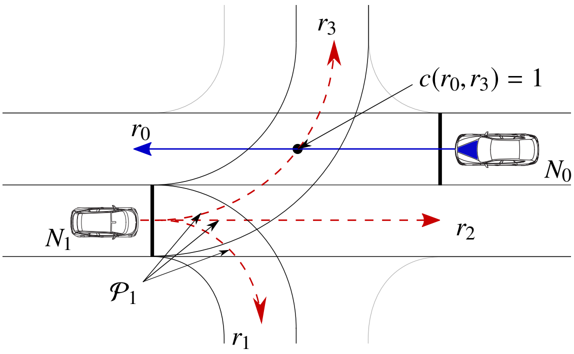


Figure 4.2: A typical urban intersection with the autonomous car N_0 driving on path r_0 and one oncoming vehicle N_1 which may intersect with r_0 (graphic from [124], ©2018 IEEE).

The authors of [15] demonstrate an *online* POMDP planner for intersection scenarios. They track the belief state with an IMM and describe the behavior of the other vehicles with two possible models: constant velocity or constant acceleration. The other vehicles are modeled to switch their model with a certain probability. The problem is modeled on a continuous state space with continuous actions. The performance of the POMDP is nonetheless constrained by the simple motion models of the other vehicles.

4.2 Problem Formulation

This work focuses on the *online* decision making for the ego vehicle, i.e. the generation of a sequence of desired accelerations $a_0 = (a_0^{t_0}, a_0^{t_1}, a_0^{t_2}, \dots)$, e.g. for traversing an unsignalized intersection with an arbitrary layout and a variable number of other traffic participants with unknown intentions and probabilistic motion models.

The path of the ego vehicle p_0 is assumed to be collision-free regarding static-obstacles and is either generated by a path planner a priori or simply retrieved from the road geometry of a given map. In a second step, the longitudinal velocity is planned along p_0. This practice is referred to as *path-velocity* *decomposition* in the literature [48] and reduces the trajectory planning problem to a one dimensional workspace.

The environment is populated by a set of agents $\mathcal{N} = \{N_0, \dots, N_K\}$, with $K \in \mathbb{N}_0$ and the ego vehicle N_0. Every other agent N_k, with $k \in \{1, \dots, K\}$, has a set of future path hypotheses. The path of the ego vehicle, r_0, and all other path hypotheses are retrieved from the topological map $\mathcal{R}$, defined as $\mathcal{R} = \{r_0, r_1, \dots, r_2\}$, with $I \in \mathbb{N}_0$, $r_i = \{\overrightarrow{q_{i,0}q_{i,1}}, \dots, \overrightarrow{q_{i,J-1}q_{i,J}}\}$ for $i \in \{0, \underset{.}{,} I\}$, $j \in \{0, \dots, J\}$ and $J \in \mathbb{N}_0$, and $q_{i,j} \in \mathbb{R}^2$ being the position of waypoint j of route i. Every agent N_k is assumed to drive on a certain route on which its motion is described by $v_k(t) \in [0, v_{\max}]$ for time $t \in [0, \infty)$. For every agent N_k a set of possible path hypotheses is defined as $\mathcal{P}_k \subseteq \mathcal{R}$.

As the various route elements may intersect with each other, an intersection function $c(r_i, r_j)$ is defined as

$$c(r_i, r_j) = \begin{cases} 1, & \text{if } r_i \cap r_j \neq \emptyset \\ 0, & \text{otherwise} \end{cases} \quad \forall i, j \in \{0, \dots, I\}, i \neq j. \tag{4.1}$$

The different paths are retrieved from the road network and may therefore be referred to as routes. An example of this route definition can be seen in figure 4.2.

Given the uncertainty about the movement of the other cars, the autonomous vehicle has to continuously choose an optimal acceleration a^* to maximize the expected, cumulative discounted future reward:

$$a^* := \arg\max_a \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R^t | x^0\right]. \tag{4.2}$$

The reward may take into account collisions, the total acceleration (providing comfort) and the deviation to a traffic-law and curvature based reference velocity

4.3 Approach

This chapter describes the algorithm to generate an optimized policy for the autonomous vehicle *online*. The main focus is hereby that it takes the uncertain future behavior of the other vehicles into account. The approach describes the road layout in a generic way and can therefore be used for arbitrary intersections. An external prediction algorithm is not needed as the other agents are simulated stepwise ahead as part of a forward simulation. Various models are used, one for each of the different possible maneuvers. The models contain interactive behavior which allows the planning of complex and interactive maneuvers for the autonomous car. To reduce the dimension of the state space and to simplify the representation, possible path hypotheses acite generated for the other vehicles. The path hypotheses are extracted from the topological map. This allows a low-dimensional, compact agent representation with the path of the other agent as hidden variables (figure 4.3). The configuration of the other agents can then simply be described in longitudinal direction on their path. The problem description from Section 4.2 is formulated as a POMDP to allow for the representation of state uncertainty (belief states) and model uncertainty.

The POMDP problem formulation is solved *online* with the library TAPIR, which is an implementation of the sampling-based ABT algorithm (see [50]). Because the utilized ABT algorithm samples multiple episodes to approximate the solution, the model properties of the POMDP (e.g. probability distributions) do not need to be specified explicitly but as a generative model. As previously presented in the longitudinal planning approach in the authors' previous work [121], the algorithm provided here solves the motion planning problem in a coarse way on the behavioral layer (see [102] for the definition). The solution is an optimized policy. Parts of the policy are then provided to the trajectory planning layer for smooth execution.

4.3.1 State Space

The motion models of the different agents are not independent, as interactive behavior is represented in the forward simulation. Therefore, all agents are represented in the state space. A certain state $x \in \mathcal{X}$ is defined in continuous space as

$$x = [x_0, x_1, x_2, \ldots, x_K]^{\mathrm{T}}. \tag{4.3}$$

The state of the autonomous car is represented by x_0 and the surrounding agents are represented by $x_k, k \in \{1, \ldots, K\}$. The configuration of all agents is described by their longitudinal position s_k on their path p_k by use of the Frenet-Serret formulas.

The state of the autonomous car is thus defined as

$$x_0 = [s_0, v_0]^{\mathrm{T}} \tag{4.4}$$

and the state of the other vehicles is

$$x_k = [s_k, v_k, p_k]^{\mathrm{T}}. \tag{4.5}$$

The path p_k defines the latent variable which cannot be measured directly but only inferred over time. Nonetheless, a discrete set $\mathcal{P}_k$ of potential path candidates is retrieved from the topological map for every vehicle N_k. The notation of the state space is illustrated in Figure 4.3.

4.3.2 Action and Transition Model

A simple physical transition model is used instead of more advanced kinematic models as the planning problem is solved on the behavior layer. The transition model of the other vehicles is a discrete time physical model with a step size of Δt:

$$\begin{bmatrix} s'_k \\ v'_k \\ p'_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_k \\ v_k \\ p_k \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \\ \Delta t \\ 0 \end{bmatrix} a_k, k \in \{0, \ldots, K\}. \tag{4.6}$$

As mentioned in Section 4.2, the path of an agent is assumed to be constant, such that $p'_k = p_k$. It represents the hidden state which is not dynamic. The acceleration a_k is retrieved from a car-following model (Intelligent Driver Model (IDM) [104]) which also respects reference velocity, v_{ref}, which is

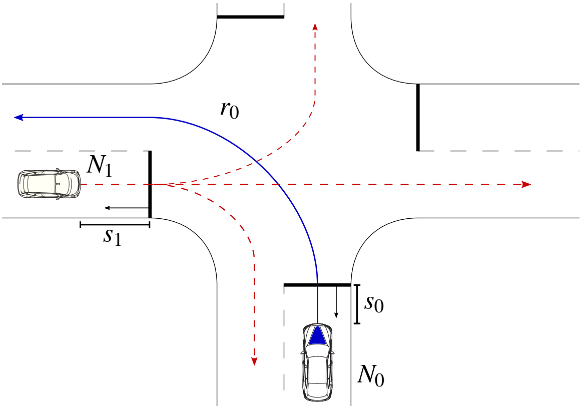


Figure 4.3: Visualization of the definition of the state space (graphic from [124], ©2018 IEEE).

retrieved by constraining the lateral acceleration (as shown in section 3.3.2). The IDM is extended by adding an interaction-based acceleration a_{int}:

$$a_{int,k} = \begin{cases} 0\,\frac{\text{m}}{\text{s}^2}, & \text{if} c(r_k, r_0) = 0, \\ -1.5\,\frac{\text{m}}{\text{s}^2}, & \text{if} c(r_k, r_0) = 1 \wedge (t_{c,k} - t_{c,0}) \in [1,5] \end{cases}. \tag{4.7}$$

Assuming a constant velocity v_k, $t_{c,k}$ is the time needed by agent k to reach the conflict point where both paths intersect. The interaction based acceleration is an empirically chosen heuristic value, but could also be learned from training data or be a probabilistic function.

The acceleration is also constrained by a maximum acceleration $a_{\max}$. The acceleration of the other vehicles is additionally perturbed by use of Gaussian noise to represent the model uncertainty. Nonetheless, precise motion models (either learned or tuned) are favorable to keep the variance low. This is the case as a high variance leads to a high degree of uncertainty of the future position/velocity of the other vehicles, which may lead to a more conservative policy. The resulting total acceleration for every other agent is therefore:

$$a_k = \min(a_{\text{ref},k} + a_{\text{int},k}, a_{\max}) + \mathcal{N}(0, \sigma^2). \tag{4.8}$$

The transition model of the ego vehicle is defined in the same way as for the other vehicles in equation (4.6), except that its path must not be incorporated in its state. This is the case as the autonomous vehicle has only one path, calculated a priori.

4.3.3 Reward Model

The reward $R(\mathcal{X}, a)$ model is defined as follows:

$$R(\mathcal{X}, a) = R_{\text{coll}}(\mathcal{X}) + R_{\text{vel}}(\mathcal{X}) + R_{\text{acc}}(a). \tag{4.9}$$

The term R_{coll} punishes a collision with a high negative reward. The second term, $R_{\text{v}}(\mathcal{X})$, punishes the deviation to a reference velocity (defined as a desired velocity on a road without vehicles, see section 3.3). It is defined as, $R_{\text{vel}} = -K_{\text{v}+}(v_{\text{ref}} - v_0)^2, \text{if} v_0 > v_{\text{ref}}$ and $R_v = -K_{v-}(v_{\text{ref}} - v_0), \text{if} v_0 < v_{\text{ref}}$. By quadratically punishing too high velocities, the ego vehicle is more unlikely to clearly overshoot the desired velocity. Punishing lower velocities in a linear way motivates the planner to drive with the desired velocity but allows for slower solutions (e.g. because of a temporarily occupied lane). The third term, R_{acc} punishes accelerations to avoid unnecessary reactive behavior.

4.3.4 Observation Model

An observation $o \in \mathcal{O}$ is also defined on the whole state vector, with

$$o = [o_0, o_1, \ldots, o_K]^{\text{T}}. \tag{4.10}$$

The observation of the autonomous vehicle is defined as o_0 and the observations of the other vehicles are defined as o_k with $k \in \{1, \ldots, K\}$.

The state of the autonomous vehicle is considered as fully observable and therefore the state and the corresponding observation is equal:

$$o_0 = [s_0, v_0]^{\text{T}}. \tag{4.11}$$

The possible route of the other vehicles is not directly observable. Therefore, neither the route, nor the longitudinal position on the (unknown) route can be measured. Instead, the longitudinal position is transformed to global coordinates which are used as observation, s.t.:

$$o_k = [v_k, x_k, y_k]^{\mathrm{T}}. \tag{4.12}$$

The ABT algorithm solves a POMDP formulation by generating the belief tree via sampling of possible episodes. Hereby, the observation model $Z(o, x', a) = P(o|x', a)$ does not need to be given explicitly but possible observations must be sampled when episodes are generated.

The path p_k of another vehicle N_k is generally unobservable. After generating a new state with the transition model $x'_k = \left[s'_k, v'_k, p'_k\right]^{\mathrm{T}}$, this state is used to generate the corresponding observation. By using the unambiguous transformation ${}^{W}T_L$, a possible future observation following the new state can be created: $[s', v', p'] \xmapsto{{}^{W}T_L} \left[v'_{obs}, x'_{obs}, y'_{obs}\right]$ (see Figure 4.2). Comparing the real, measured observations with the previously generated observations will allow to infer the hidden state over time.

Additionally, observation noise is embedded in the model. Instead of simply adding Gaussian noise to the deterministic observation which is retrieved from the new state x', the uncertainty of the perception concerning the lane of the other vehicle is included. The perception of the other vehicle is normally tracked and probabilistically mapped on a certain route. This uncertainty is modeled with the probabilities from a Bayes classifier. It simulates the uncertainty concerning the tracked route of another vehicle in future time steps. The Bayes classifier uses a 2-dimensional feature vector f_k (velocity and position based, see Figure 4.4) for vehicle N_k, that can be generated from the observation space:

$$f_k = \begin{bmatrix} f_{k,1} \\ f_{k,2} \end{bmatrix} = \begin{bmatrix} |v'_k - v_{\mathrm{ref},r_i}(s'_k)| \\ \left\| [x'_k \;\; y'_k]^{\mathrm{T}} - [x_{k,\mathrm{pred},r_i} \;\; y_{k,\mathrm{pred},r_i}]^{\mathrm{T}} \right\|_2 \end{bmatrix}. \tag{4.13}$$

The probability of vehicle N_k being on a certain route r, $P(p_k = r)$, with$r \in \mathcal{P}_k$ can be defined via Bayes rule as

$$P(p_k = r_i | f_{k,1}, f_{k,2}) = \frac{P(r_i) P(f_{k,1}, f_{k,2} | r_i)}{P(f_{k,1}, f_{k,2})}. \tag{4.14}$$

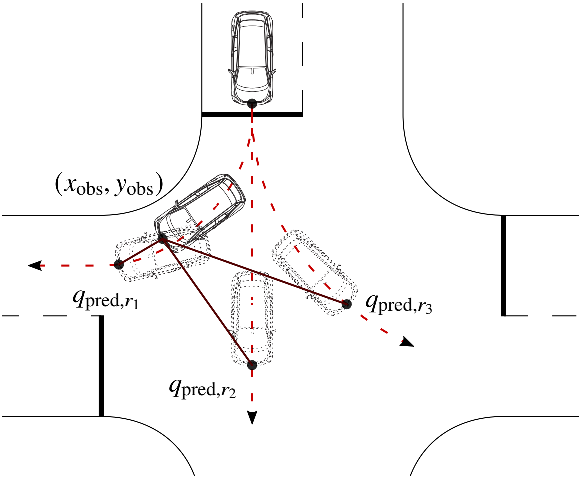


Figure 4.4: Demonstration of the distance feature $f_{k,2}$ for the generation of the observation classifier. It is defined as the Euclidean distance between the simulated observation's position $(x_{\text{obs}}, y_{\text{obs}})$ and the assumed Euclidean position $q_{\text{pred},\text{r}_\text{i}}$ given a certain path hypothesis $p_k = r_i \in \mathcal{P}$ (graphic from [124], ©2018 IEEE).

With the assumption that every route has the same a priori probability, s.t. $(P(p_k = r_1) = P(p_k = r_2) = P(p_k = r_3) \ldots)$, the law of total probability and the assumption of independent features, equation (4.14) may be rewritten to:

$$P(p_k = r_i | f_{k,1}, f_{k,2}) = \frac{P(f_{k,1}|r_i)P(f_{k,2}|r_i)}{\sum_{l=1}^{I} P(f_{k,1}|r_l)P(f_{k,2}|r_l)}. \tag{4.15}$$

$P(f_{1/2}|r_i)$ can be learned from sample data, or simply designed as done in this work. It is normally distributed with $P(f_1|r_i) = \mathcal{N}(0, 4.0)$ and $P(f_2|r_i) = \mathcal{N}(0, 6.0)$ to simulate the uncertainty of the lane object matching. The observation o_k is generated for every particle based on the probability of route r_i that is sampled from equation (4.15).

4.3.5 Implementation

The POMDP formulation in this chapter is solved as described in section 2.4. As roll-out plan, a graph search is combined with a constant velocity roll-out

Table 4.1: Simulation parameters

c	20000	t_{hor}	8	R_{coll}	-10000
γ	1	K_{v+}	-100	K_{v-}	-100

as described in section 2.4.5. The important parameters of the POMDP are given in table 4.1.

4.4 Results

This section shows the results of the POMDP behavior planner. The evaluation is twofold: At first the convergence as well as the policies for various uncertainties are shown. This is done with a simple example to show the capabilities of the planner. As the planner approximates the optimal policy *online*, the intent of the first section is to show with what probability the optimal action is found. The second part of the evaluation demonstrates the capabilities of the planner in full simulation scenarios for the crossing at a complex intersection. A proprietary simulator at BMW Group is used for the simulations [39]. The system (containing the simulation environment and the algorithm) runs on an Intel Core i7-4910MQ CPU with 2.9 GHz for the simulation scenarios.

4.4.1 Convergence

The ABT algorithm approximates the optimal policy *online* by sampling of episodes. During the runtime of the anytime algorithm (see section 2.4), the optimal policy is constructed from the sampled episodes. In the following, it is evaluated how well the optimal policy can be approximated for the scenario shown in figure 4.5. The convergence rate is shown in figure 4.6 as a function over the optimization time of the anytime algorithm. The convergence results are shown for various heuristics to motivate their usage. The approximation quality is evaluated with a loss function, defined as the absolute difference between the optimal action of the ground truth and the approximated Q-value:

$$L(\widehat{Q}, Q^*) = |\widehat{Q}(b, a^*) - Q^*(b, a^*)|. \tag{4.16}$$

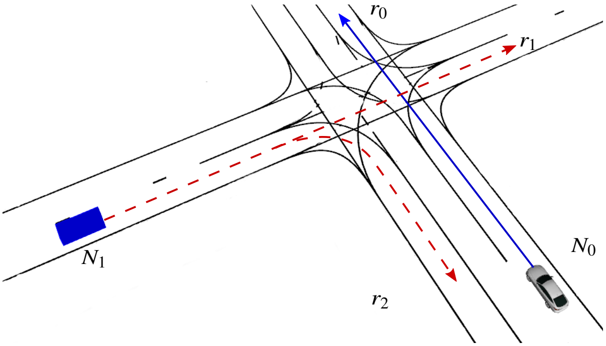


Figure 4.5: The simple example scenario for the evaluation of convergence to the optimal policy. The scenario is chosen such that the path probabilities of the initial belief are $P(p_1 = r_1) = 0.05$ and $P(p_1 = r_2) = 0.95$. The initial velocities are $v_0 = 8.6\ \frac{\text{m}}{\text{s}}$ and $v_1 = 8\ \frac{\text{m}}{\text{s}}$ (graphic from [124], ©2018 IEEE).

The ground-truth is generated by sampling episodes until convergence is reached and sampling of further episodes does not lead to a change of the policy anymore. The upper plot of figure 4.6 shows the absolute difference of the approximated values of Q-function compared to the values of the optimal Q-function. The plot in the middle of figure 4.6 shows the average number of episodes which are sampled during runtime. The lower plot of figure 4.6 shows the percentage of how likely a non-optimal action selection is. The plot shows, that all heuristics result in a significantly reduced probability of selecting a non-optimal action for a runtime of up to 500 ms. The reason for this, is that the heuristics allow to steer the search in the right direction. Nonetheless, for a longer sampling time, the heuristic approaches underperform the non-heuristic approaches. This is the case as the heuristics underestimate the potential future reward (i.e. potentially overestimating costs). This is especially the case for the constant velocity heuristic. As the heuristic must present a lower bound on the value function, the heuristic is non-optimal and prohibits exploring the right branches in the long run. It can also be seen, that the 3-step Dijkstra heuristic performs better than the n-step Dijkstra heuristic. This is the case as the 3-step Dijkstra heuristic needs less calculation time, because of the smaller optimization horizon. This allows to sample more episodes to capture the state uncertainty, which the heuristic itself is not capable of considering.

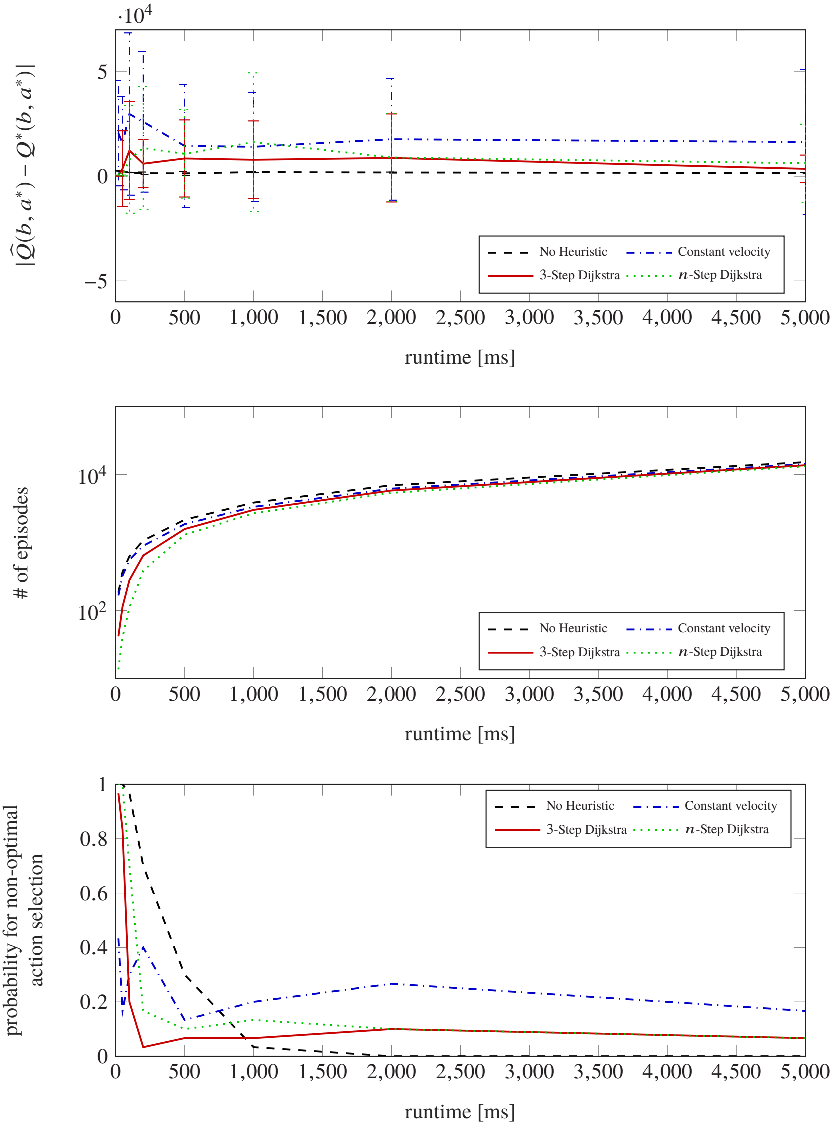


Figure 4.6: Comparison of the convergence of different heuristics for the scenario presented in figure 4.5 (graphic from [124], ©2018 IEEE).

In the following some additional remarks about convergence are given. The general assumption is, that the problem becomes exponentially harder to solver for an increasing number of other vehicles and an increasing number of potential paths, $|\mathcal{P}|$. Nonetheless, as the algorithm only searches in the reachable belief space, adding non relevant vehicles (i.e. not directly influencing the ego vehicle's reward) does not make the problem harder to solve. This is the case as sampling based POMDP solvers scale with the reachable belief space and not in general with the size of the belief space [41]. It is even noticed, that also problems with more relevant vehicles may lead to faster convergence as the reachable, free belief space is smaller because of the more possible trajectories of the other vehicles. This may lead then to a smaller exploration space of the algorithm and therefore to faster convergence. It may be summarized that the complexity of the underlying POMDP formulations varies mostly with the given micro traffic situation.

4.4.2 Policy Behavior Planning

The approximated policy presents an optimized behavior for various future, possible scenarios which may arise during the execution of the policy. This is the case, as the policy contains not only a single trajectory, but an optimized reactive action for the most probable future scenarios. This section shows the policies for different degrees of considered uncertainty. The policies for the scenario in figure 4.5 are shown in figure 4.7. The other vehicle has two possible paths to drive on (drive straight or turn right), but it is unknown at the beginning on which of both paths it is driving. The optimal behavior of the autonomous vehicle is strongly related to the future behavior of the other vehicle. The *open-loop* planner (figure 4.7a) has to slow down immediately as it is not able to incorporate future observations in the planning phase. This means, that the planner reacts to both possible future situations simultaneously. On the contrary, the POMDP planner (see figure 4.7b - e) for various considered uncertainties) is able to reason about both possible scenarios. This results in a policy that postpones the decision of crossing vs. braking to a future point in time when more observations has been recorded. It can be seen, that the introduction of further uncertainties (such as motion model uncertainty (figure 4.7c), observation uncertainty (figure 4.7d) and sensor noise (figure 4.7f) results in a more conservative policy which also contains more branches to account for the increased number of scenarios. One can also notice that the

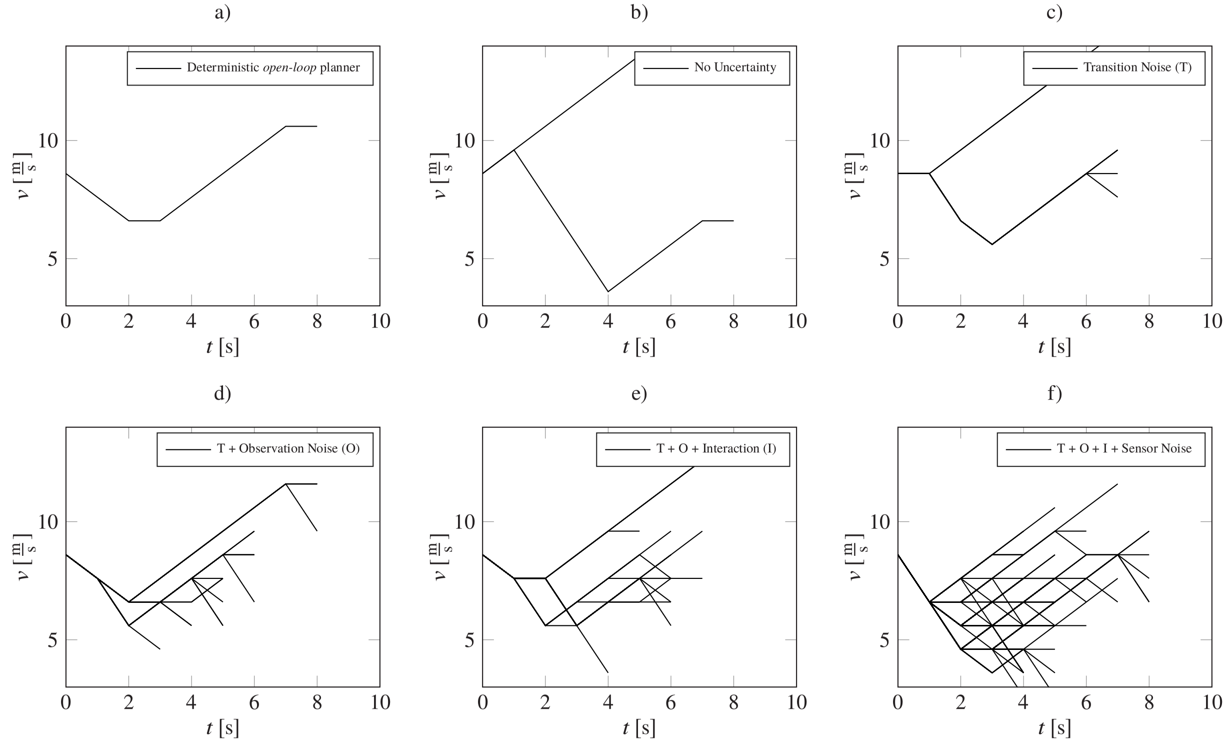


Figure 4.7: Comparison of a *open-loop* planner and POMDP for different considered uncertainties. The plots show the approximated policies. The policies are recorded with a UCT factor of 200000 to motivate broader exploration of the belief tree (graphic from [124], ©2018 IEEE).

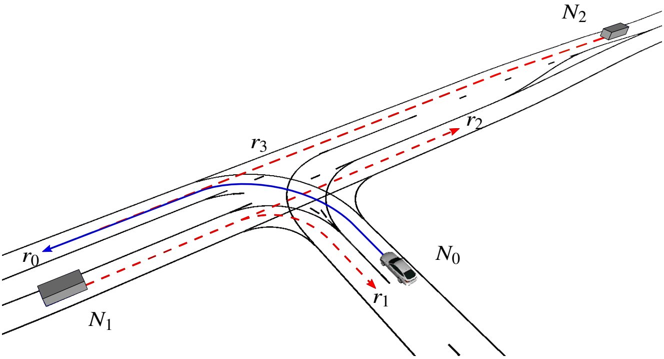


Figure 4.8: Top view of the T-junction scenario as presented in (graphic from [124], ©2018 IEEE).

introduction of the interaction model (figure 4.7e)) allows for a less conservative policy again (i.e. less braking, faster intersection approach), compared to not incorporating interaction (figure 4.7d)). This is because the planner is going to consider that the other vehicle is going to react on the actions of the autonomous vehicle, given that it is nearer to the intersection.

Simulation: Merging on a T-junction

This section presents the recorded trajectories of a simulation run for merging at a T-junction. The scenario, presented in figure 4.8, is as follows. The ego vehicle intends to do a left turn to merge into a main road at a T-junction. While the other vehicles on the main road have the right of way, the ego vehicle must yield if required. The ego vehicle has to decide whether to merge before or after vehicle N_2, which is approaching the intersection from the right. While merging before vehicle N_2 would be possible, an approaching vehicle from the left (N_1) makes the scenario more complex. This is because vehicle N_1 has two possible options and therefore its predicted behavior is not known to the autonomous vehicle. The options of vehicle N_1 are driving straight and intersect with the autonomous vehicle or turning right without any influence on the behavior of the autonomous car. In addition to the uncertainty of the chosen route of vehicle N_1, the ego vehicle also has to consider the uncertain longitudinal prediction of both vehicles which is realized by the interactive

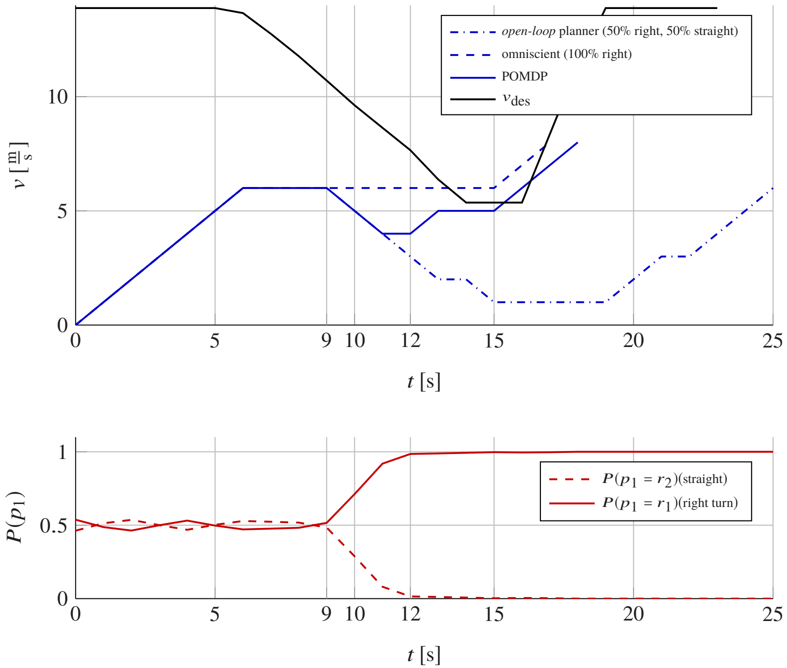


Figure 4.9: Evaluation for a T-junction scenario where vehicle 2 turns right. The upper figure compares the different trajectories for the different planners. The lower figure shows the path probabilities of vehicle 2, generated by the particle filter (graphic from [124], ©2018 IEEE).

and probabilistic motion model. This uncertainty is incorporated by adding Gaussian noise on the interactive motion model (see equation (4.8)).

The upper figure of Figure 4.9 shows the driven trajectory of the autonomous car for different planners as well as the desired reference velocity. The lower figure shows the estimated probability for each maneuver of vehicle N_2, tracked over time. The following description of the scene follows the description in the corresponding publication [124].

At the beginning, the ego vehicle accelerates up to the desired curve velocity (defined in [121]). After 9 seconds, the belief for the behavior of vehicle N_2 is still uncertain, therefore the planner starts to decelerate slightly to reduce the

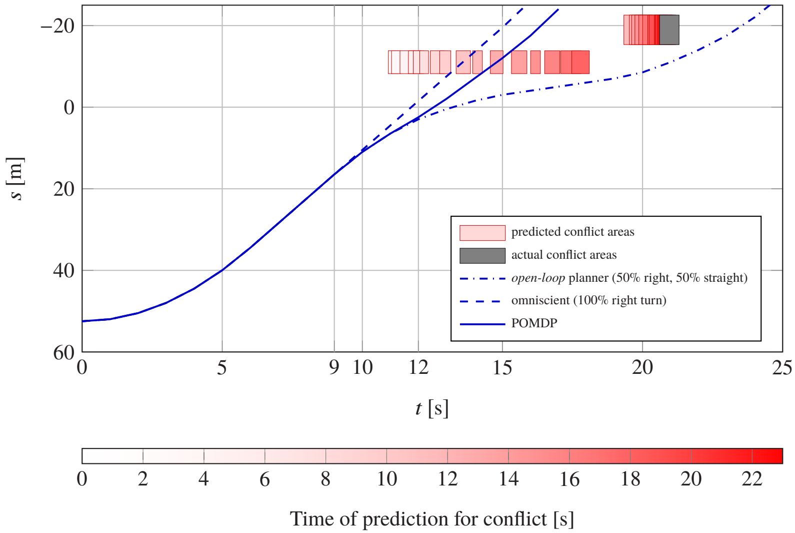


Figure 4.10: The figure shows the planned positions over time when the POMDP planning is executed with different fixed probabilities and with the probability from prediction and POMDP sampling (graphic from [124], ©2018 IEEE).

probability of a collision and to have more time to receive new measurements. Thus, the two possible options, yielding to vehicle N_1 or merging immediately, are kept open for the ego vehicle. This behavior (also known as *information gathering*) is the result of the policy because the observation model has simulated, that the next measurements will lead to a less uncertain belief state. Because of the observation model, it can even *infer* at what point in time the belief becomes less uncertain and approach the intersection accordingly. After 12 seconds, the prediction is precise enough such that the ego vehicle can cut in before vehicle N_1.

For the same scenario, Figure 4.10 plots the position over time and especially the predicted time interval during which the other vehicles occupy the areas that conflict with the path of the ego vehicle. It can be seen that the point of a conflict between the ego vehicle and vehicle N_1 is constantly postponed while vehicle 2 breaks upon the intersection, leading to even having no conflict at all when the turning behavior of vehicle N_2 becomes apparent.

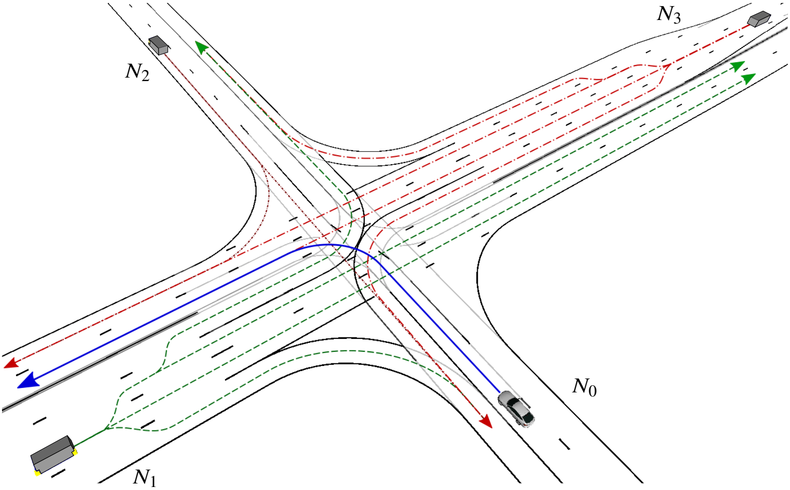


Figure 4.11: Top view of a more complex intersection scenario with three other vehicles (graphic from [124], ©2018 IEEE).

The POMDP approach is compared to different planning algorithms with either omniscient behavior (no uncertainty about the behavior of vehicle N_1) or conservative behavior (*open-loop* planner). It can be seen, that the POMDP planner performs nearly as well (it is able to merge before vehicle 1) as the omniscient approach, which has no uncertainty about the future behavior of both vehicles at all. The *open-loop* planner approach on the other hand considers all possible trajectories and does not incorporate future measurements. This results, as shown in figure 4.9, in a conservative suboptimal trajectory, such that the ego vehicle cannot merge before vehicle N_2.

Simulation: Crossing of a Complex Intersection

A key strength of the algorithm is that it can be used for various intersections because of its generic formulation. Therefore, the performance of the algorithm is also presented for a more complex scenario. The scenario in Figure 4.11, contains a larger, unsignalized intersection, with in total 10 different possible routes for the other three vehicles. The results are very similar to the previous T-

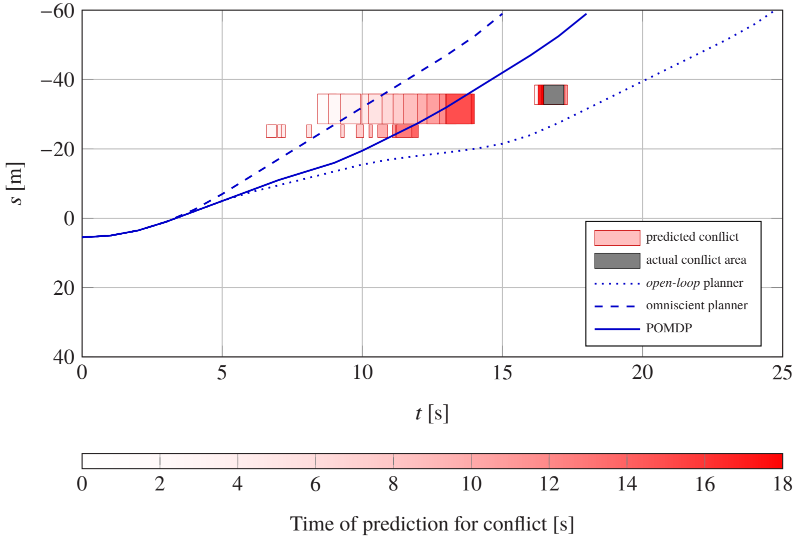


Figure 4.12: Evaluation of the complex intersection scenario. The figure shows the planned positions over time for three different planners as well as the predicted and actual conflicts. The lowest rectangles represent the conflict area with vehicle N_1, the rectangles in the middle the conflict area with vehicle N_2 and the upper rectangles the conflict area with vehicle N_3 (graphic from [124], ©2018 IEEE).

junction scenario. Again, the POMDP planner acts with very similar behavior as the omniscient approach (see figure 4.12 for the trajectories). On the other hand, the resulting behavior of the *open-loop* planner is very conservative as the planner has to consider many different predictions at the same time.

4.5 Summary

This chapter demonstrates a *closed-loop*motion planning algorithm for autonomous driving in uncertain, urban environments. The algorithm is able to retrieve an optimized policy *online* for the crossing of arbitrary intersections. To reduce the complexity of the algorithm, the behavior of the autonomous vehicle is optimized in longitudinal direction along a preplanned path. A set of path hypotheses is generated for each surrounding vehicle a priori.

The key focus of the algorithm is the incorporation of various uncertainties, namely:

- perception uncertainty
- path uncertainty
- model uncertainty
- interaction.

The POMDP formulation is solved *online* by combining Monte Carlo sampling (the ABT algorithm) with near optimal roll-out heuristics which can be calculated fast at runtime. By considering possible future observations explicitly, the algorithm is able to predict in what ways the current belief state may change in the future. This enables the postponing of decisions, such as merging before or after another vehicle, as the algorithm is able to predict that future observations will lead to a less uncertain prediction. The policy implicitly contains the different maneuvers for the different homotopy classes. It is shown in various simulation scenarios how the algorithm outperforms simpler approaches (namely *open-loop* planner), which do not consider the uncertainties explicitly. The results show, that the possibility to postpone decisions allows the algorithm to drive less conservative trajectories. These trajectories are even similar to the ones of omniscient planning algorithms which have full knowledge about the future trajectories of the surrounding vehicles.

5 Coupled 2D Planning for Interactive Merging

The algorithms in the previous chapters (chapter 3 and chapter 4) are based on the simplification of an a priori *path-velocity decomposition* [48]. By planning a path around static obstacles first, the planning problem is reduced to one spatial dimension and therefore easier to solve due to a lower dimension of the action and state space. This is a valid approach for scenarios where the path of the autonomous car is independent of the longitudinal velocity (e.g. crossing of an intersection). Nonetheless, scenarios exist, where this assumption is not valid. This is for example the case for lane changes (see [113] for a

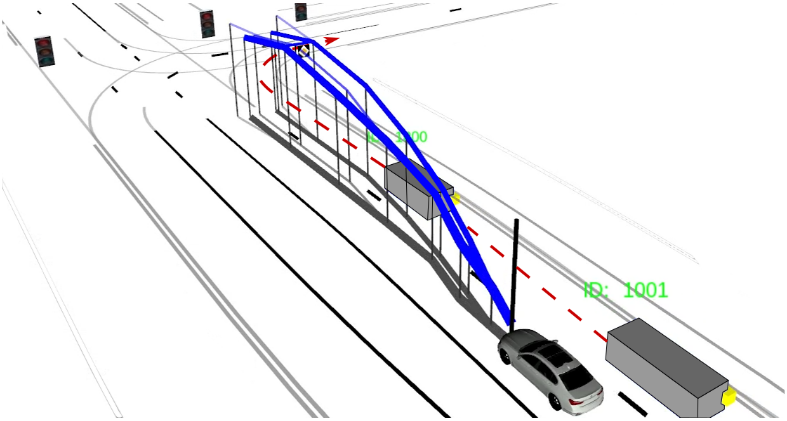


Figure 5.1: Visualization of the *closed-loop*, *online* algorithm: The planning algorithm approximates the optimal policy on a receding horizon for the most probable future scenarios *online*. The optimal policy π^* is shown in blue, plotted with its velocity over the itudinal distance and lateral offset. The policy contains reactive plans, depending on different, future observations of the environment. The plans correspond to the different, reachable homotopy classes. It can be seen, that the policy approaches the gap and merges in case of an observed yielding behavior of the other vehicle.

scenario overview) where the longitudinal position of the actual lane change is dependent on the previous longitudinal velocity. This is especially important for merging in dense traffic, where the longitudinal speed profile incorporates gap-approach and speed adaption.

This chapter extends the algorithm presented in chapter 4 to lane change scenarios. It describes an algorithm which is able to execute lane changes that are requested from the navigational layer (as described in [102]) of an autonomous vehicle. Lane changes are necessary to follow the desired route through the topological map, to circumvent obstacles or to progress faster depending on the current traffic flow information. The presented algorithm optimizes longitudinal and lateral planning in a combined manner. It incorporates potential interaction with other traffic participants (their reaction to a merge attempt) and the uncertainty of the corresponding prediction. The policy is optimized in a *closed-loop* manner by considering future observations about the behavior of the other vehicles. Especially for lane changes in congested traffic, gaps on neighboring lanes are often small, such that planning of a collision-free merge trajectory is not possible. Instead, the possibility to merge depends on an interactive, friendly behavior of another vehicle. The presented algorithm reduces the uncertainty about the friendliness of the other drivers by approaching gaps to gather information about their potential behavior. This allows for merging in congested traffic where gaps of sufficient size do not exist.

The main contributions of this chapter are as follows:

- gap selection, approach and merge are combined in one algorithm and are implicitly part of the optimal policy
- combined longitudinal and lateral optimization
- explicit modeling of the interaction with other vehicles
- trained model to predict yield behavior of the surrounding vehicles
- *online* optimization of a *closed-loop* formulation on a continuous state and belief space
- combination of MCTS with a deterministic A^* roll-out heuristic for fast convergence to the optimal policy

The chapter is based on and was previously published in [125].

5.1 Related Work

Lane changes can be divided in different stages which are: gap selection, gap approach, gap evaluation and the merge maneuver into the gap. More often than not, these tasks are separated into different algorithms instead of being solved in one generic algorithm. While this allows for a simpler algorithm design, it also reduces the space of possible, interactive solutions which may lead to suboptimal behavior. It may even lead to not finding a solution at all in critical cases. In the following, simple *gap assessment* algorithms are reviewed first and are followed by various *planning algorithms*.

5.1.1 Gap Assessment Algorithms

The first algorithms which performed lane changes in real traffic were realized during the Darpa Urban Challenge [20]. These approaches were rule-based and separated the lane change into different subtasks. For example, the winning team [107] designed an arbiter which evaluated the feasibility to merge into a certain gap based on the velocities of the surrounding drivers, the gap size and further metrics.

More advanced rule-based concepts go a step further and asses the utility of different gaps. Hereby, not only the current state, but also the measurement uncertainty as well as the predicted behavior of the other agents is used to decide for suitable gaps. Nonetheless, the lane change itself is still separated into a gap approach and a merge maneuver [3]. While such a utility based approach cannot guarantee safety, other algorithms exist which can guarantee the safety of lane changes by using verification methods based on reachability analysis [80].

Another approach which uses a POMDP formulation is presented in [105]. The authors design a high-level state space of eight states and account for sensor uncertainty of the surrounding vehicles. Limiting the horizon of the policy to two planning steps allows to solve the problem *online*.

5.1.2 Planning-Based Algorithms

Next to the *gap assessment* algorithms, various algorithms exist which generate a trajectory or policy to merge into a certain gap. The algorithms can be distinguished by their capability of combined longitudinal and lateral planning as well as the consideration of interaction and uncertainties.

Longitudinal Planning

Given a map layout in which the current lane of a certain vehicle merges into another lane, combined longitudinal and lateral planning must not be considered. This is the case as the spatial coordinates of the merge position are independent of the velocity of the autonomous car but defined a priori by the topological map (the point where both lanes start to intersect).

The authors of [27] use a Probabilistic Graphical Model (PGM) to infer from various measurements how likely a yielding behavior of the other vehicles is. Given the estimated yield probabilities, the algorithm selects a certain target for a longitudinal, model-based ACC planner. In [23], the authors introduce Multi Policy Decision Making (MPDM), which provides a fast solution for a POMDP formulation. They use a predefined set of potential policies describing high level maneuvers (such as lane following, merge, . . .). By use of *online* forward simulations, the value of each policy is estimated, given various potential behaviors of the other agents. Finally, the policy which maximizes the expected reward is chosen. The complexity of the forward simulations is reduced by using non-probabilistic motion and observation models.

To avoid such a forward simulation, the authors of [75] use passive Actor critic RL to learn the value function for a similar framework.

The algorithm presented in chapter 4 may also be used for longitudinal merging scenarios and allows to generate an optimal policy given uncertainty of prediction and interaction.

Longitudinal and Lateral Planning

A hybrid formulation for the merge problem is presented in [113]. At first, a deterministic graph search algorithm plans a trajectory into a gap of sufficient size while neglecting uncertainty and interaction. In a second step, an MPC algorithm is used to optimize the speed profile in longitudinal and lateral direction separately.

The authors of [74] use an a priori defined set of constraints to optimize a longitudinal speed profile, by use of a QP, to approach and merge in a certain, preselected gap. In a subsequent step, another QP formulation is used to optimize the lateral speed profile for the lane change. Again, the formulation does not allow to incorporate interaction and uncertainties.

The authors of [67] demonstrate how cooperative and interactive behavior may be generated for the case of lane changes on highways. The well-known MCTS algorithm is used to model the deterministic interaction between the various traffic participants. The degree of aggressiveness in the interactive, cooperative behavior is designed by weighting the costs of the other drivers in a combined cost functional. A similar approach with continuous actions is presented in [59]. The authors use a decoupled MCTS formulation and use guided search and semantic actions to allow for solving the problem on a continuous action space. While both approaches are able to model interaction and uncertainties, hidden variables are not introduced. Therefore, the algorithms do not reason over a belief state which also prohibits information gathering.

5.2 Approach

This chapter describes the design of an algorithm which generates an optimal policy for the lane change problem in congested traffic. To respect the uncertainty of the other drivers (uncertain interaction and prediction) in an optimal way, a policy is generated over a belief state. To allow for actively gathering information about the hidden states of the surrounding drivers, possible future observations are considered during planning. This leads to a *closed-loop* optimization of the optimal policy by use of a POMDP. This allows for the generation of a behavior, in which the autonomous car approaches the most promising gap given the current scene configuration, while already considering various potential future scenarios (merge or abort merge) in the optimal policy. However, using a simple vehicle model and discrete actions leads to non-optimal smoothness of the trajectory regarding higher derivatives such as jerk.

To address this limitation, the most probable trajectory is extracted from the policy. This trajectory is then optimized by a sampling-based [112] or a MPC-based trajectory planning algorithm [38], using continuous actions (see section 1.5 for further details).

The problem is solved *online* with the anytime, MCTS-based ABT algorithm (see section 2.4 for further details). The generic problem formulation as well as the *online* capability allows the application of the algorithm in various real-world scenarios.

5.2.1 State Space

To allow the modeling of interaction, the autonomous car N_0 as well as the other agents N_k are represented in the state space. An actual state $x \in \mathcal{X}$ is defined in continuous space as

$$x = [x_0, x_1, x_2, \dots, x_K]^{\mathrm{T}}. \tag{5.1}$$

Hereby, x_0 represents the state of the autonomous car and x_k, with $k \in \{1, \dots, K\}$, represents the states of the other agents N_k on the same or neighboring lanes. The position of a vehicle N_k is described by the *Frenet-Serret* formulas on a certain lane l_k at longitudinal position s_k, with longitudinal velocity v_k and lateral position d_k (see figure 5.2). The origin of the *Frenet-Serret* coordinate system is located at the beginning of the next intersection (see figure 5.2). The state of the autonomous vehicle is defined as

$$x_0 = [s_0,\ d_0,\ v_0,\ l_0]^{\mathrm{T}}, \tag{5.2}$$

while the other vehicles are described with

$$x_k = [s_k,\ v_k,\ l_k,\ m_k]^{\mathrm{T}}. \tag{5.3}$$

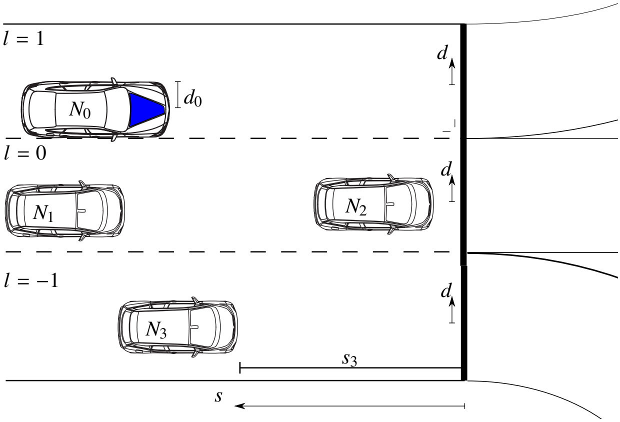


Figure 5.2: Visualization of the dimensions of the state space of the autonomous and the other cars. The desired lane of the ego vehicle is always defined as $l = 0$ (graphic from [125], ©2018 IEEE).

The variable m_k describes a hidden variable. It cannot be measured directly, but inferred via observations over time. The variable is used to describe the friendliness of the other driver i.e. if he will react by yielding ($m_k = 1$) to a merge attempt or not ($m_k = 0$).

5.2.2 Action and Transition Model

The action space $\mathcal{A}$ is defined as $\mathcal{A} = \mathcal{A}_{\text{long}} \times \mathcal{A}_{\text{lat}}$ with a set of discrete longitudinal accelerations $\mathcal{A}_{\text{long}} = \{-1\,\frac{\text{m}}{\text{s}^2}, 0\frac{\text{m}}{\text{s}^2}, 1\,\frac{\text{m}}{\text{s}^2}\}$ and a set of lateral velocities $\mathcal{A}_{lat} = \{-v_{\text{lat}}\frac{\text{m}}{\text{s}}, 0\frac{\text{m}}{\text{s}}, v_{\text{lat}}\frac{\text{m}}{\text{s}}\}$. A possible action of the ego vehicle is defined as a $a_0 = \left[a_{0,\text{long}}, v_{0,\text{lat}}\right]$. The non-holonomic kinematics (see section 1.2.2) of the autonomous car are taken into account by constraining v_{lat} via a maximum side slip angle as defined in [74]:

$$v_{\text{lat}} = \min(0.17 v_0, 0.5\,\frac{\text{m}}{\text{s}}). \tag{5.4}$$

While the action in longitudinal direction is a discrete acceleration, the action in lateral direction is a lateral velocity. The underlying assumption, that a lateral velocity may be reached immediately is valid as long as the lateral velocity is constrained on the current longitudinal velocity.

The transition model $T(x', x, a_0)$ of the autonomous car and the other vehicles is defined for discrete time with a step size of Δt as followed for the different dimensions:

$$
\begin{aligned}
s'_k &= s_k - v\Delta t - \frac{1}{2} a_{k,\text{long}} \Delta t^2 && k \in \{0, \ldots, K\} \\
v'_k &= v_k + \Delta t a_{k,\text{long}} && k \in \{0, \ldots, K\} \\
m'_k &= m_k && k \in \{0, \ldots, K\} \\
l'_k &= l_k && k \in \{1, \ldots, K\} \\
l'_k &= l_k \pm 1 && d_k + d_k v_{k,\text{lat}} \lesseqgtr \mp \frac{w_\text{lane}}{2},\ k \in \{0\} \\
d'_k &= d_k v_{k,\text{lat}} + \left(l'_k - l_k\right) w_\text{lane} && k \in \{0\}.
\end{aligned}
$$

The width of the lane is assumed to be constant, s.t.$w_\text{lane}(s_k) = w_\text{lane}$. Furthermore, the assumption is made, that the other vehicles do not change lanes and drive in the middle of their lane. While the action of the autonomous car, a_0, is part of the optimization problem, the action of the surrounding agents, $a_k(x)$, with $k \in \{0, \ldots, K\}$ is determined by a model, given the current state.

5.2.3 Motion Model of Surrounding Agents

For the behavior generation of the surrounding agents, the IDM [104] is used to realize car following behavior. It is also adapted for the case of interactive yielding.

In general, two possible behaviors must be modeled: Cooperative yielding to the merge attempt of the autonomous vehicle or non-yielding car following behavior. These two behaviors can be modeled by either using the preceding vehicle as target vehicle N_target for the IDM (not yielding) or by using the merging, autonomous car as target vehicle. If a preceding vehicle does not exist, the desired reference velocity $v_{k,\text{des}}$ is approached. The reference velocity is based on the road curvature and speed limit and extracted as described in section 3.3.2.

Table 5.1: IDM parameters

T	$0.5\,\mathrm{s}$	δ	4
a_{IDM}	$1.75\,\frac{\mathrm{m}}{\mathrm{s}^2}$	σ^2	0.1
d_{IDM}	$2\,\mathrm{m}$	Δv_k	$v_k - v_{\text{target}}$
b	$-0.8\,\frac{\mathrm{m}}{\mathrm{s}^2}$	φ_k	$s_{\text{target}} + L_{\text{target}} - s_k$

The acceleration $N_k(x)$ of another vehicle N_k used for the transition model is

$$a_k = a_{\text{IDM}} \left[1 - \left(\frac{v_k}{v_{\text{target}}} \right)^{\delta} - \left(\frac{\phi(v_k, \Delta v_k)}{\varphi_k} \right)^2 \right] + \mathcal{N}(0, \sigma^2) \tag{5.5}$$

and

$$\phi(v_k, \Delta v_k) = d_{\text{IDM}} + v_k T + \frac{v_k \Delta v_k}{2\sqrt{a_{\text{IDM}} b}}. \tag{5.6}$$

The model parameters are the desired time headway T, a comfortable decelaration b, a minumum distance d_{IDM}, a maximum acceleration a_{IDM} and the acceleration exponent δ.

The target vehicle N_{target} is set depending of the yield classification $P_{k,\text{yield}}$:

$$N_{target} = \begin{cases} N_0 & \text{if } P_{k,\text{yield}} = 1 \\ N_{k,\text{front}} & \text{otherwise} \end{cases}. \tag{5.7}$$

$N_{k,\text{front}}$ denotes the leading vehicle of vehicle N_k on the lane l_k and L_k denotes the absolute length of vehicle k.

For the case of non-existing front vehicles, Equation (5.6) is set to zero. The acceleration of the model is disturbed with Gaussian noise to account for prediction uncertainty. The parameters for the IDM are given in table 5.1.

5.2.4 Observation Model

The variable m_k is a hidden state and describes if the driver of vehicle N_k will behave cooperative or not during a merge attempt. This variable cannot be measured directly but can be inferred observations over time. The POMDP formulation allows to predict what possible future observations may be measured and in what way they are going to influence future belief states. This enables the policy to choose certain actions which lead to more precise belief states, a behavior known as *information gathering*. This means for the merge scenario, that configurations in which the autonomous vehicle is approaching a certain gap are preferred, as the potential interactive behavior can be observed. Is is assumed, that there is no measurement noise and therefore the observation $o = Z(x', a)$ is defined as follows:

$$o = [o_0, o_1, \ldots, o_K]^{\mathrm{T}} \tag{5.8}$$

with $o_0 = \left[s'_0,\ d'_0,\ v'_0,\ l'_0\right]^{\mathrm{T}}$ and $o_k = \left[s'_k,\ v'_k,\ l'_k\ \right]^{\mathrm{T}}$ for $k \in \{0, \ldots, K\}$.

5.2.5 Reward Model

The reward function $R(x, a, x')$ is the sum of different possible rewards, motivating different behaviors:

$$R(x, a, x') = R_{\text{vel}} + R_{\text{act}} + R_{\text{end\_lane}} + R_{\text{wrong\_lane}} + R_{\text{center}} + R_{\text{coll}}. \tag{5.9}$$

The different rewards are explained in more detail in the following.

Reference Velocity

The goal of R_{vel} is to realize a behavior which follows a certain reference velocity, defined for each lane. The reference velocity is defined based on the maximum speed limit, adapted by a comfortable reference speed in curvatures including an approach phase (see section 3.3.2 for more details). The reward is defined as:

$$R_{\text{vel}}(x'_0) = \begin{cases} -100 \cdot (v_{\text{ref}} - v_0)^2 & ,\text{if } v_0 > v_{\text{ref}} \\ -100 \cdot (v_{\text{ref}} - v_0) & ,\text{if } v_0 < v_{\text{ref}} \end{cases}. \tag{5.10}$$

Desired Lane

The purpose of the desired lane rewards is to motivate a lane change in general ($R_{\text{wrong\_lane}}$) and in particular when the lane is going to end ($R_{\text{end\_lane}}$). The reward for being on a non-desired lane, $R_{\text{wrong\_lane}}(x_0)$, is simply defined with a negative reward of -600 if the autonomous vehicle is not on its desired lane ($l_0 \neq 0$). The end of lane reward, $R_{\text{end\_lane}}(x_0)$, is a negative reward whose absolute value increases linearly over the last 50 meters of a lane from 0 to -1000.

Lane Center

A quadratic reward on the lane center, $R_{\text{center}}(x_0)$, is used to motivate the ego vehicle to drive in the middle of the lane, with $R_{\text{center}} = -200 \cdot d_0^2$.

Action Selection

To minimize the used accelerations the action of the ego vehicle has a reward of $R_{\text{act}}(a) = -100 \cdot \left(a_{0,long}^2 + 2 \cdot |v_{0,lat}|\right)$.

Collision

Finally, a collision reward $R_{\text{coll}}(x)$ punishes with -10^6 if the autonomous car is entering the longitudinal area of another vehicle on the lane of the other vehicle. A simple linear increasing cost map at the back of the other vehicles is used to realize following behavior after a merge, as done in section 3.3.2.

5.2.6 Learned Yielding Model

For the surrounding agents, two possible motion models exist. Yielding to a possible merge attempt or simply following the existing front vehicles. The POMDP formulation is based on the definition of realistic transition models. Therefore, the idea is to learn from recorded data what world situation is most

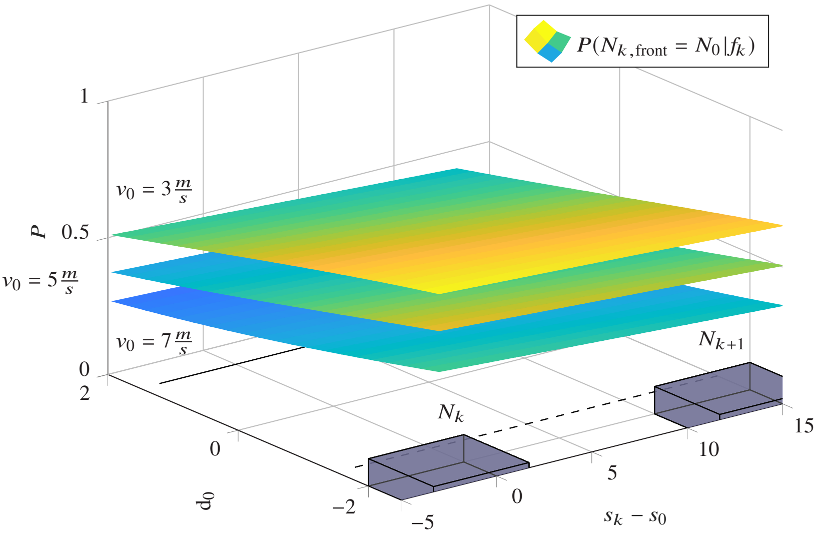


Figure 5.3: The probability $P(N_{k,\text{front}} = N_0|f_k)$ for a yield of vehicle N_k for various possible configurations of the autonomous car in the gap. The probability is drawn over feature d_0 and the feature, $s_k - s_0$. The probability is also shown for three different velocities of the autonomous vehicle v_0. The other two vehicles are driving with $v_k = v_{k+1} = 5\,\frac{\text{m}}{\text{s}}$ (graphic from [125], ©2018 IEEE).

promising to see a yield reaction of the other agent. This allows that the policy steers the autonomous car to the most promising position. A logistic regression classifier is used to determine the probability of vehicle N_k yielding to N_0, in a given scene, described by the feature vector f_k:

$$f_k = \left[1, \varphi_k, d_0, v_0, s_k - s_0, v_k, v_{k,\text{front}}\right]^{\text{T}}. \tag{5.11}$$

The longitudinal position of the ego vehicle in the gap is described with $s_k - s_0$ and the absolute gap size $\varphi_k = s_{k,\text{front}} + L_{k,\text{front}} - s_k$ and the length of the front vehicle as L_{target}. The result of the logistic regression classifier is the yield probability of vehicle N_k for a scene described by f_k:

$$P(N_{k,\text{front}} = N_0 | f_k) = \frac{1}{1 + e^{-\theta^{\mathrm{T}} f_k}}. \tag{5.12}$$

The vector θ is the trained weight vector of the logistic regression model.

The probability $P(N_{k,\text{front}} = N_0 | f_k)$ and a threshold probability β_{yield} is used to choose the front car $N_{k,\text{front}}$ of agent N_k as follows:

$$N_{k,\text{front}} = N_0, \ \textbf{if} \ P(N_{k,\text{front}} = N_0 | f_k) > \beta_{\text{yield}} \wedge m_k = 1. \tag{5.13}$$

Increasing/decreasing the threshold allows for less/more aggressive policies.

The classifier is trained with recorded data of real-world lane changes. Used training data is labeled to the two different maneuver classes, represented by using a different target vehicle for the IDM. The feature vector f is normalized and scaled by its variance such that every feature can contribute in the same way. The training data contains 4847 positive data points (vehicle yields to the merge request) and 1691 negative ones (vehicle does not yield). The accuracy of the algorithm is 84.3% in the test set.

5.2.7 Implementation

As in chapter 4, the merge planner is solved as described in section 2.4. The used parameters are presented in table 5.2.

Heuristic Function

To steer the construction of the belief tree in a promising direction, the value of newly explored belief states is estimated by use of a heuristic function section 2.4. The value estimate is obtained with a roll-out $g(x)$ from the current particle. The roll-out uses a near-optimal plan, generated by an A^* [61] graph search with three steps and a following constant velocity action up to the planning horizon. This idea is described in section 2.4.

Table 5.2: TAPIR parameters

c	20	γ	1.0
t_{hor}	10 s	Δt	1 s

The A^* [61] search itself also needs a heuristic which is realized over the reward $R_{\text{non\_des\_lane}}$. The heuristic $h_{\text{non\_des\_lane}}$ presents an estimate of the future costs by calculating how many further steps are at least on a non-desired lane:

$$h_{\text{non\_des\_lane}} = -600 \cdot \lfloor (|l_0| - 1) w_{\text{lane}} + \frac{1}{2} w_{\text{lane}} + sgn(l_0) d_0 \rfloor. \tag{5.14}$$

5.3 Results

The scenarios are set up in a simulation software at the BMW Group and the algorithm is evaluated on a system with an Intel Core i7-4910MQ CPU with 2.9 GHz. The simulation vehicles are controlled with a rule-based expert system [39] and not the IDM, which is used in the forward simulation.

5.3.1 Analysis of Belief State Policy

At first, a generated policy and its capability to plan *closed-loop* interactive behaviors is examined in detail. The policy is shown in figure 5.4 for the merge scenario presented in figure 5.1. The plot shows the subset of all sampled episodes which are part of the approximated optimal policy, which considers various possible future scenarios. The figure is split up in two parts. While the right side shows how the belief over the friendliness of the other drivers (m_1 and m_2) is predicted to change, the left side shows the actual policy. It can be seen, that the two maneuvers of the rear vehicle ((non) cooperative behavior) of the gap are present in the policy. But, the policy does not only respect the two different maneuvers but also the uncertain longitudinal prediction in each maneuver class. Additionally, it can be seen that the policy considers the current uncertainty of the belief and incorporates that more information will be available in the next time step. At $t = 1s$ the belief tree is predicted to split

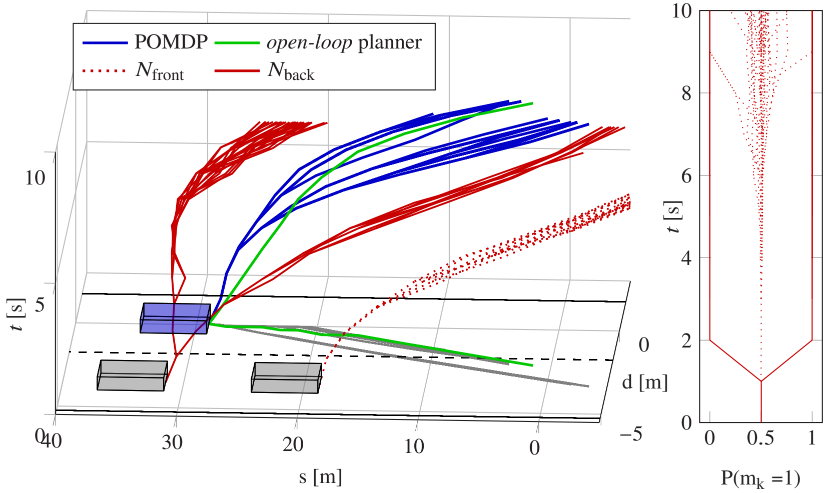


Figure 5.4: Policy of the ego car (blue) for merging onto another lane with two other vehicles (red). It can be seen that the ego car accelerates first up to the velocity of the gap (implicit gap approach). The policy contains two different future plans depending on the behavior of the other vehicle. After receiving the next observation, the ego vehicle plans to merge either before or behind the other car. The optimal policy contains 37 potential scenarios and was optimized for 2000 ms to retrieve many episodes. The POMDP policy is compared to a *open-loop* planner, which does not respect future observations but considers all predictions simultaneously. It can be seen, that the *open-loop* planner plans directly for the more conservative maneuver, i.e. merging behind N_{back} (graphic from [125], ©2018 IEEE).

in one tree representing possible cooperation and another tree representing non-cooperative behavior of the rear vehicle (figure 5.4, right side). A non-cooperating rear vehicle will lead to a merge behind it, while a cooperative behavior allows the autonomous car to merge in front. The right picture of figure 5.4 show how the estimation of m is predicted to change, given particular actions of the autonomous car. It can be seen, that the belief of the rear vehicle is assumed to be known in the next time step (by observing the reaction to the merge attempt) while it is not yet known at b^0. It is also interesting to see, that no better estimation will be available for the front vehicle, as information about the belief state can only be gathered by approaching the corresponding gap. As the autonomous car does not approach the gap in front of the first vehicle, information about its possible cooperative behavior will not be gathered.

The policy of the *closed-loop* POMDP planner is compared with the policy of an *open-loop* planner. The *open-loop* planner assumes the same initial belief state as the POMDP, but does not incorporate future measurements. This prohibits it to take into account that more observations will be perceived which allow to estimate the state m more precisely and to postpone the merge decision accordingly. Therefore, the *open-loop* planner has to plan a more conservative policy and merge behind the rear vehicle.

5.3.2 Online Simulation

Addtionally to the direct evaluation of the policy in the previous section, this section shows a qualitative evaluation of the performance of the planner in an *online* simulation. The *online* simulation is run for the scenario shown in figure 5.1 which is extended by two more vehicles, such that four vehicles occupy the neighboring lane. Given the uncertainty in the longitudinal prediction, the gaps are too small to allow for planning of a trajectory in one of them. The longitudinal position of all vehicles is shown in the upper plot of figure 5.5. It can be seen, that the autonomous car N_0 approaches the gap between vehicle N_2 and N_3 first by approaching the longitudinal position of N_3 ($t = 2s - 6s$). The autonomous car observes, that vehicle N_3 does *not* start to act cooperatively (i.e. braking to allow a merge). Its hidden state, m_3 is therefore inferred to be non-cooperative, i.e. $m_3 = 0$ which can be seen in the lower plot of figure 5.5. The autonomous car stops the merge attempt, slows down (see third subplot in figure 5.5) and approaches the other gap in front of vehicle N_4 (see the velocity during the gap approach $t = 3s - 9s$ of N_0). At time $t = 10s$, a significant observation is received, i.e. vehicle N_4 is braking for the autonomous car. This results in an inferred estimation of $m_4 = 1$ for N_4. That leads to a predicted, cooperative trajectory of vehicle N_4, such that the autonomous car starts to merge immediately. The merge friendly behavior of the other agents may be estimated in a discrete manner as the behavior of each maneuver class is very different and no measurement noise is assumed. As soon as the lane change of the autonomous vehicle has happened ($t = 11s$), the belief state is not tracked anymore.

This example shows how the different phases of a lane change: decision for a gap, approach, yield prediction and merge are handled implicitly in one optimization problem. None of these steps is needed to be modeled in a single

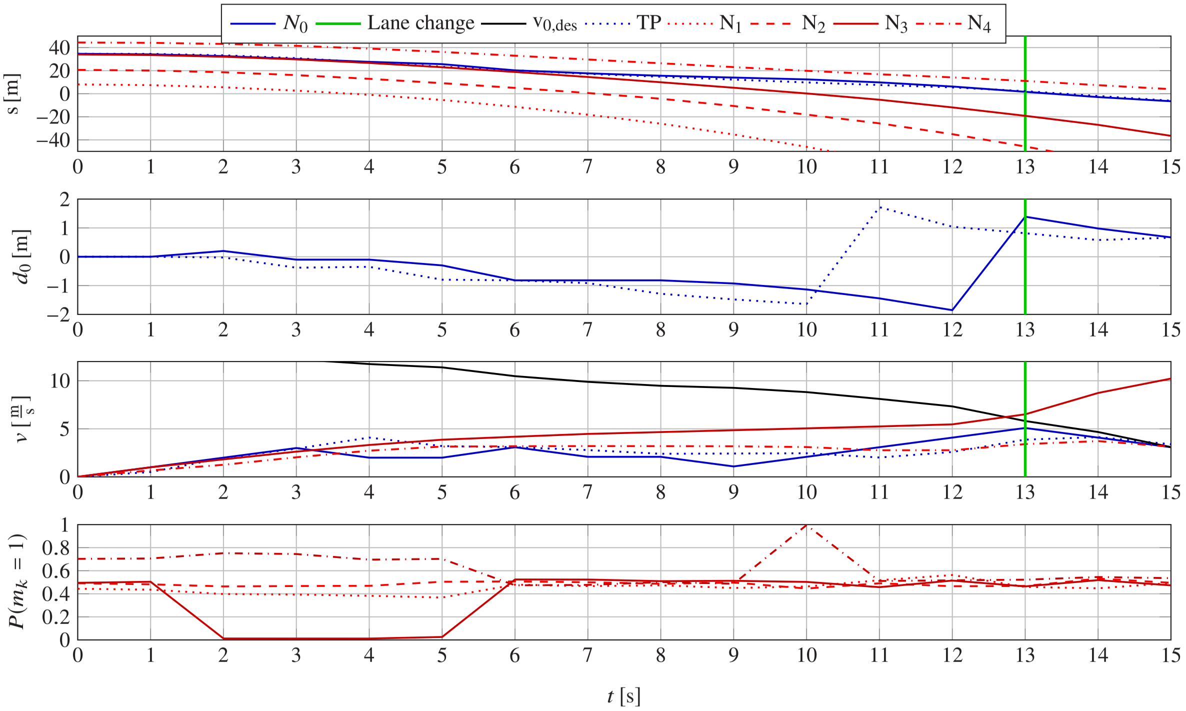


Figure 5.5: The trajectories of all agents in a *online* simulation of the scenario in figure 5.1, extended to four vehicles. TP is the trajectory of the ego vehicle, which is generated from the POMDP's reference trajectory by the trajectory planner (graphic from [125], ©2018 IEEE).

module, which is the case for the state of the art. Aside from that, one of the key strengths of a POMDP formulation show up in this simulation. The planner uses actions to reach positions (close to a potential merge gap), where it can *gather information* about the latent states of the other agents. This allows the planner to quickly asses the possibility to merge in a certain gap and to act accordingly.

One can also notice, that the trajectory planning layer provides a smoother execution of the reference trajectory, which is provided by the POMDP. This also leads to deviations to the reference trajectory, which can be seen, e.g., for the lateral position of the autonomous car (second plot in Figure 5.5) where the behavior planner expects the autonomous car to enter the other lane at $t = 13s$, while it already happened at $t = 11s$ due to a small offset.

5.4 Summary

This chapter presents a behavior planning algorithm, based on a POMDP formulation, that allows to execute lane changes in densely populated environments. It is demonstrated how the different stages of a lane change algorithm (gap selection, gap approach and merge) can be modeled in one optimization problem. The policy is optimized in a *closed-loop* manner by respecting possible future observations. These observations allow to derive information about the hidden variables of the driver models. The models respect the

- longitudinal uncertainty in the prediction
- unknown willingness for yielding.

Incorporating these uncertainties in the *closed-loop* optimization of the policy allows the autonomous car to merge in (too) narrow gaps. The formulation as a POMDP allows for a behavior, where the autonomous car approaches the gap to reduce the uncertainty about the prediction of the other vehicle (yielding to the merge attempt or not) by considering possible future observations. For every possible future observation, the policy contains a reactive plan to act accordingly. To allow for optimal approach behavior, the action of the autonomous car is optimized simultaneously in longitudinal and lateral direction. MCTS is used with domain specific heuristics to allow for solving the non-convex, probabilistic optimization problem *online*.

6 Planning under Sensor Occlusions

The previous chapters demonstrate how various uncertainties can be considered explicitly during behavior planning for autonomous vehicles. For example, the unknown intention and the likelihood for yielding/interaction of the other drivers can be modeled as hidden variables in their motion models. Nonetheless, these algorithms work under the assumption that the physical state of the environment can be fully observed. This is an invalid assumption in real-world environments as occluded objects may not be perceived.

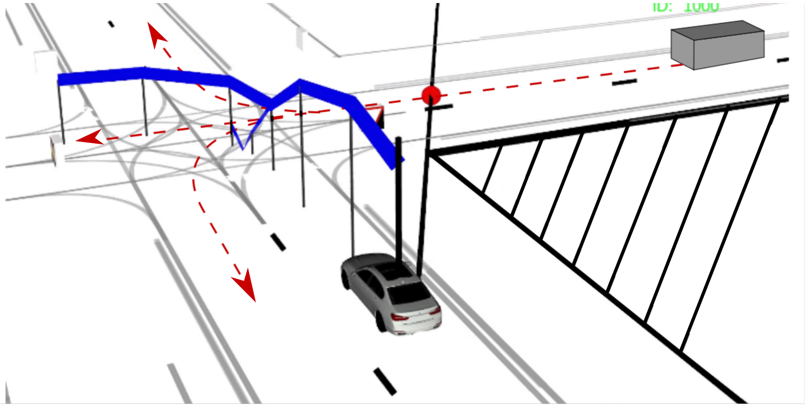


Figure 6.1: Visualization of the *closed-loop*, *online* algorithm: The goal of the autonomous car is to turn left at an intersection. Due to the constrained Field of View (FoV), other potential vehicles cannot be observed. The policy is shown in blue, with its velocity plotted over the longitudinal distance. It can be seen, that the policy contains several, *closed-loop* plans (crossing or stopping) for the point of time in the future where a sufficient FoV will exist (graphic from [123], ©2019 IEEE).

In this chapter, the behavior planning algorithm from the previous chapters is advanced. A formulation is presented which considers explicitly that other agents may be occluded and that they cannot be perceived by the sensors of the autonomous car.

Handling this existence probability in an optimal way is a non-trivial problem. Simply using a worst-case assumption for the configuration of another vehicle is not possible. This is the case, as the worst-case configuration of the other agent is dependent on the behavior of the autonomous car, which is yet to be optimized.

Standard approaches limit the longitudinal velocity of the autonomous car, such that braking before a possible conflict point is always possible if the FoV is not large enough at the current point in time. This results in planning of a trajectory which leads to a stop in front of the occlusion. This trajectory is executed until the FoV is suddenly large enough, s.t. a potential occluded vehicle is not relevant anymore. At that point in time, replanning generates a new trajectory which does not respect the occlusion anymore. This behavior is rather conservative, even a full stop, the so called *freezing robot* is possible (see [103]).

This chapter presents a less conservative approach. The presented algorithm reasons over a belief state, representing the existence probability of occluded vehicles, so called *phantom* vehicles. It considers occluded areas, created by *static* as well as by *dynamic* objects. Hereby, the planner does not only consider the current FoV but also reasons about the future change of the FoV. This is done, by simulating the FoV over the planning horizon, to model at what vehicle configurations the FoV may be sufficiently large. Such *closed-loop* planning allows for behaviors which actively explore the environment to reduce uncertainty in the belief. Potentially existing, i.e. *phantom* vehicles, are described by their reachable set instead of their configuration. This representation allows to represent an unknown number of vehicles in an occluded area by one set. By combining the FoV simulation with the simulation of possible phantom vehicles, the decision point in the policy can be determined.

The problem is formulated as a POMDP, which creates a policy containing various future plans for different future observations (see figure 6.1).

The key contributions of this chapter are as follows:

- modeling of the FoV during planning to allow for *information-gathering*
- modeling of *phantom* vehicles with their reachable sets
- evaluation in various simulations
- *online* optimization of a *closed-loop* formulation on a continuous state and belief space
- combination of MCTS with a deterministic A^* roll-out heuristic for fast convergence to the optimal policy

This chapter is based on and was previously published in [123, 130].

6.1 Related Work

A worst-case approximation of potential vehicles in an occluded area can be done by reachability analysis. The general idea of reachability analysis is to describe all possible, future configurations of a certain vehicle by a set [2]. This idea can be transfered to occlusions, by describing all possible vehicle configurations in one occlusion by one reachable set. An unknown number of objects in one occluded area can therefore be described by one reachable set only. By use of motion models and certain assumptions on the underlying parameters, all future configurations of these vehicles can also be described by one reachable set. The authors of [87] use a simple heuristic to calculate a safety distance to the occluded areas. The occluded areas itself are calculated by a geometric model. In [77], the authors provide a safety verification method for planning trajectories under occlusions. Kamm's circle [109] is used for the physical models to define the reachable set, describing all possible, future vehicle configurations, more exactly. The approach is evaluated with *static* and *dynamic* occlusions at intersections. In [40], the unobservable area at the current point in time is extracted by mapping a static grid on a topological map. As long as potential objects are not far enough away from the path of the autonomous vehicle, planning on that area of the path is not allowed.

The aforementioned approaches assume no knowledge about vehicles inside the occlusions. In [33], the authors observe vehicles entering the occlusion and track it throughout their time of being not observable. Various potential tracks are created when the vehicle enters the occluded area (for the different possible behaviors in the occluded area) and tracked throughout the intersection with a hybrid Gaussian Mixture Model. The behavior planner can therefore respect the occluded car by respecting the various hypotheses. As soon as the vehicle leaves the occlusion, the observation is matched on the most likely synthetic track, using the Kullback-Leibler divergence.

The previous approaches consider occlusions by constraining the planner with predictions or reachable sets of potentially existing vehicles. In the following, approaches which reason over the uncertainty directly during optimization of a trajectory are presented. The authors of [17] use a POMDP to respect observation uncertainty due to occlusions in their planning problem. A scenario-specific discretization of the state space is learned and a policy is approximated *offline* with the solver presented in [18]. The resulting policy contains information gathering actions which decrease the uncertainty about the positions of the other agents by optimizing the set of possible observations. It is shown that the approach is capable of merging into traffic under existence of non-trivial occlusions with an a priori known number of other vehicles

A POMDP formulation is used in [16] to cross occluded intersections and zebra crossings while considering the occlusions generated by static obstacles. The authors use a scalable approach which transfers the problem to several subproblems, containing the autonomous car and one other vehicle. After solving the POMDP for every subproblem, a common Q-function is retrieved by using the sum or minimum of the different Q-functions. The authors use a Gaussian belief over the position of occluded agents and solve the problem *offline*.

The authors of [91] use a POMDP to model occlusions at intersections. The problem formulation assumes a fixed number of agents in the environment, including in occlusions. Raytracing is used for the calculation of the FoV, which is mapped on a lanelet [10] representation. The problem is solved *offline* with use of the ABT algorithm.

A Deep Reinforcement Learning (DRL) approach for the crossing of intersections with occlusions is presented in [42]. The authors use a discretized state space, realized as a grid with color codes as input for a Deep Q-Learning (DQL) formulation. The results demonstrate that the network is able to learn explorative behavior in front of intersections. However, DRL impedes generalization to unseen scenarios and suffers from approximation errors.

6.2 Approach

This chapter presents the problem formulation for *online* decision making for an autonomous car under occlusions. The drawbacks of existing approaches are overcome by combining the strengths of reachability analysis, probabilistic reasoning and *closed-loop* planning. By representing occluded vehicles with reachable sets, all possible vehicle configurations in one occlusion can be represented by one set. The set of possibe occluded vehicles on one occluded lane is referred to as *phantom* vehicle because of their uncertain existence. The *phantom* vehicles are used within a POMDP formulation to allow for probabilistic reasoning over their existence probability. Simulating the future FoV during planning allows even for active *information-gathering*. Static and dynamic obstacles are considered during the calculation of the FoV. The focus of the approach is, that the algorithm finds implicitly an optimal behavior for the handling of occlusions.

6.2.1 State Space

The problem is defined with a state space that is a composition of the state x_0 of the autonomous car N_0, the states x_k with $k \in 1, \dots, K$ of the surrounding vehicles N_k and the states x_l with $l \in [K+1, \dots, L]$ of the so called *phantom* vehicles N_l, i.e. possibly existing vehicles in occlusions. As it is infeasible to describe all possible vehicle configurations in occlusions by particular states, the idea is to describe all possible configurations on a occluded lane by one set. This set can be described for the longitudinal modeling as one *phantom* vehicle with infinite length and a certain maximum velocity (see section 6.2.3 for more details).

The state of the environment is described as

$$x = [x_0, x_1, \dots, x_K, x_l, \dots, x_L]^{\mathrm{T}} \tag{6.1}$$

with the state of the autonomous car defined as

$$x_0 = [s_0, d_0, v_0]^{\mathrm{T}} \tag{6.2}$$

and the states of the other vehicles as

$$x_k = [s_k, v_k, p_k]^{\mathrm{T}} \tag{6.3}$$

and the states of possible phantom vehicles as

$$x_l = [s_l, p_l, g_l]^{\mathrm{T}}. \tag{6.4}$$

The autonomous vehicle N_0 follows a path p_0 which is generated collision-free regarding static obstacles and is either generated by a path planner a priori or simply retrieved from the road geometry of a topological map, s.t. $p_0 = r_0$ (see section 2.5 for an introduction to the path-velocity decomposition). The other agents drive on a certain path p_k which is extracted from the road Topology $\mathcal{R}$, s.t. $p_k = r_i \in \mathcal{R} = \{r_1, ..., r_I\}$ for $I \in \mathbb{N}$. Agents are described by its longitudinal position s and its absolute velocity v in the Frenet frame on their path p. The velocity of the *phantom* vehicles is not part of the state space as they are assumed with a certain maximum velocity. The variable g is a boolean, indicating whether there is a car in the occluded area ($g = 1$) or not ($g = 0$). Therefore, $g_l = 0$ describes a world configuration in which there is no vehicle behind the field of view on p_l. The state cannot be directly measured but only be inferred over time. Again, the policy is described over a belief state $b(x)$ which describes the probability, that another car exists in the occlusion. Simulating the *phantom* vehicles ahead in the belief tree allows to infer at what future configuration the required FoV will be reached.

6.2.2 Observation Model

The state of the environment cannot be fully observed due to the existence of hidden states. Nonetheless, the hidden states can be inferred over time by possible observations of the environment. The observation space is defined in a similar way as the state space, s.t. an observation $o \in \mathcal{O}$ is

$$o = [o_0, x_1, \ldots, x_K, x_l, \ldots, x_L]^{\mathrm{T}}. \tag{6.5}$$

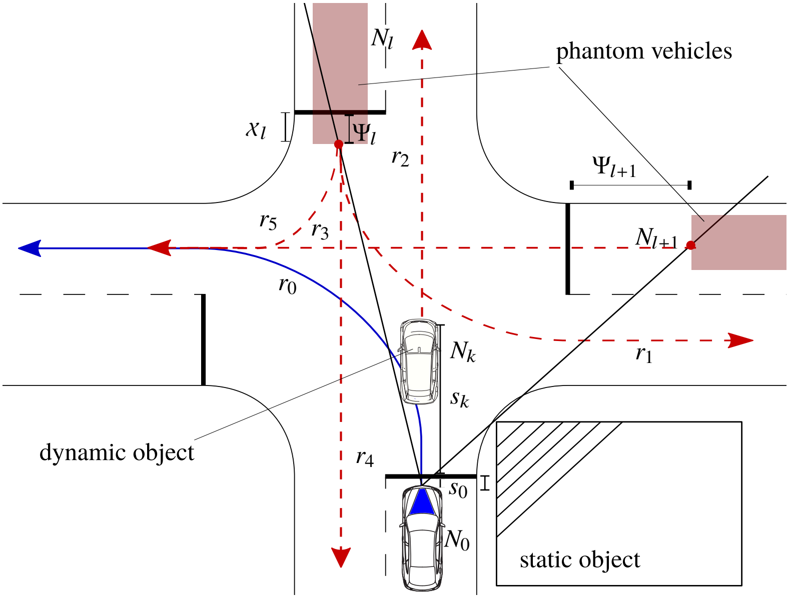


Figure 6.2: This schematic demonstrates the a state space representation advanced from chapter 4. All vehicles are encoded by their distances to the intersection on their respective path. The autonomous vehicle follows its path, $p_0 = r_0$, while the other vehicles have various path hypotheses $(r_1, \ldots, r_5)$. Crossing paths define conflict areas, which may not be occupied by the autonomous and another (phantom) vehicle at the same time. The conflict areas are not drawn for simplification issues but are explained in detail in chapter 4. Phantom vehicles are drawn as red boxes (N_l), starting at the edges of the FoV of the autonomous car N_0 which is mapped on each lane (red dashed line) of the topological map (graphic from [123], ©2019 IEEE).

The localization of the autonomous car is assumed to be noise free. The autonomous car can therefore be fully observed and the observation is defined as: $o_0 = [s_0, v_0]^{\mathrm{T}}$.

The observation of the surrounding vehicles is defined in global coordinates, as the path is a latent variable and cannot be observed directly: $o_k = [v_k, x_k, y_k]^{\mathrm{T}}$. This is done in the same way as presented in section 4.3.4.

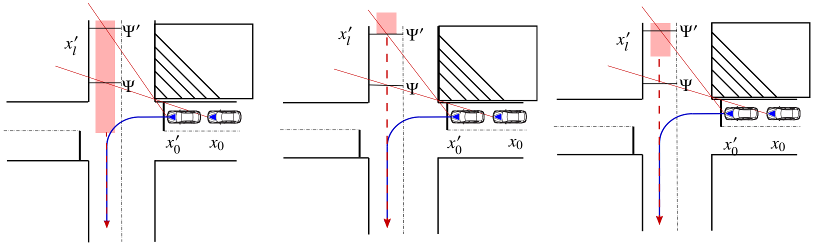
For every potential phantom vehicle, an observation is also generated. The goal is to define an observation which is independent of the unknown number of possible agents in the occlusion and that can be matched on the *phantom* vehicle. This must be the case as the number of vehicles in an occlusion is unknown. Therefore, a observation is defined by the FoV on every lane. The FoV Ψ defines the edge of the FoV on the path of phantom vehicle l (see figure 6.2), s.t. $o_l = [\Psi_l, p_l]$.

6.2.3 Representation of Phantom Vehicles

Representing all possible vehicle configurations in the occluded area is computationally infeasible. Nonetheless, the idea of using a certain worst-case configuration instead is also not viable. This is the case as a worst-case configuration cannot be calculated as it is completely dependent of the future trajectory of the ego vehicle which is yet to be optimized. This coupled problem is analytically not solvable. The idea of the problem formulation is to define a *phantom* vehicle for each occluded lane, representing all possible vehicle configurations in a certain occlusion. This is realized by placing a phantom vehicle at the start of the FoV with assumed infinite length and a velocity above the speed limit (to represent a worst-case speeding assumption), s.t. $v_{\text{phantom}} = 1.3\, v_{\max}$. If every possible, occluded vehicle configuration would be represented explicitly, a certain subset of these configurations would be discovered, when the FoV is expanded. Nonetheless, in this work, the idea is to represent all these configurations by one reachable set. Instead of splitting the set into many discretized subsets, the idea is to sample if the *phantom* vehicle is detected or not. The probability of this sampling is proportional to the revealed occluded area during the transition from x to x'. This is realized via the traffic density, defined as a uniform probability distribution in the occluded area. Given the uncertainty of the existence of another vehicle, the only safe way to cross is a sufficiently high, free FoV at the future point in time where the crossing decision has to be made. The probability $P_\phi(\Delta\Psi)$ for the existence of at least one phantom car in the revealed FoV with length $\Delta\Psi = \Psi' - \Psi$, is defined with the Bernoulli distribution:

$$P_\phi(\Delta\Psi) = \begin{cases} 0 & ,\text{for}\Delta\Psi < 0 \\ \frac{\Delta\Psi}{\omega} & ,\text{for}\Delta\Psi \geq 0 \wedge \Delta\Psi < \omega \,. \\ 1 & ,\text{for}\Delta\Psi >= \omega \end{cases} \tag{6.6}$$

The specific volume ω is defined as the average number of vehicles in an occluded area per 100 m.



(a) $g = 1 \wedge s_l > \Psi_l$: A phantom car which has been already moved out of the FoV before is set a step forward.

(b) $g = 1 \wedge \phi = 0$: A phantom car is sampled to not drive out of the occluded area.

(c) $g = 1 \wedge \phi = 1$: A phantom car is sampled to drive out of the occluded area.

Figure 6.3: The three different cases of phantom vehicles which must be handled by the probabilistic transition model (graphic from [123], ©2019 IEEE).

6.2.4 Action and Transition Model

The presented planning approach can be used with a longitudinal action set $\mathcal{A}_{\text{long}}$ or a union of $\mathcal{A}_{\text{long}}$ and an additional lateral action set $\mathcal{A}_{\text{lat}}$. While the longitudinal action set only allows the autonomous vehicle to follow its path r_0 exactly, the union with a $\mathcal{A}_{\text{lat}}$ allows the autonomous car to drive with offset to the path, to potentially increase the FoV if needed. The longitudinal action set is defined as

$$\mathcal{A}_{\text{long}} = \{-2\frac{\text{m}}{\text{s}^2}, -1\frac{\text{m}}{\text{s}^2}, 0\frac{\text{m}}{\text{s}^2}, 1\frac{\text{m}}{\text{s}^2}\}. \tag{6.7}$$

For the case of combined longitudinal and lateral actions, non-negative longitudinal actions are combined with a set of lateral velocities $\mathcal{A}_{\text{lat}} = \{-v_{\text{lat}}, 0, v_{\text{lat}}\}$, s.t.

$$\mathcal{A}_{\text{long,lat}} = \mathcal{A}_{\text{long}} \cup (\{\mathcal{A}_{\text{long}} \geq 0\} \times \mathcal{A}_{\text{lat}}). \tag{6.8}$$

In this case, the planning problem changes from a longitudinal one to a combined longitudinal and lateral planning problem. The lateral velocity actions are proportionally constrained by the longitudinal velocity of the autonomous car to guarantee kinematic feasibility. Therefore, v_{lat} is defined as $v_{\text{lat}} = \min(0.17 v_0, 0.75)$ as explained in section 5.2.2.

The motion model for the autonomous vehicle is formulated with the discrete longitudinal dynamics as

$$x_0' = \begin{bmatrix} s_0' \\ v_0' \\ d_0' \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ v_0 \\ d_0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_{\text{long}} \\ v_{\text{lat}} \end{bmatrix}. \tag{6.9}$$

For the surrounding vehicles it is defined as

$$x_k' = \begin{bmatrix} s_k' \\ v_k' \\ r_k' \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_k \\ v_k \\ r_k \end{bmatrix} + \begin{bmatrix} -\frac{1}{2}\Delta t^2 \\ \Delta t \\ 0 \end{bmatrix} a_{\text{k}}. \tag{6.10}$$

The action a_k of another agent N_k is retrieved from an extended version of the IDM which e.g. also adapts to road curvatures section 4.3.2.

For the phantom vehicles, various cases exist for the transition model. As the states are expanded during simulation, every state transition (x, x') defines a current and next FoV for every lane, i.e. $(\Psi(s_0), \Psi'(s_0))$. The transition model of the phantom vehicles is dependent on g and the amount of new exploration, i.e. $\Delta\Psi(s_0, s_0') = \Psi'(s_0') - \Psi(s_0)$. A positive $\Delta\Psi$ denotes an increased FoV and a negative $\Delta\Psi$ denotes a decreasing FoV on a certain lane.

The different cases are explained in the following:

Case 1: $g = 0$. In the case of a state which is representing a non-existing phantom vehicle, the transition is:

$$x_l' = \left[\max(\Psi_l, \Psi_l'), p_l, g_l\right]^{\text{T}}. \tag{6.11}$$

If the state represents an existing phantom vehicle, various transitions exist (see figure 6.3):

Case 2a: $g = 1 \wedge s_l > \Psi_l$. The first possibility is that a phantom vehicle moved already out of the occluded area in previous steps of the forward simulation. In that case, it is simply advanced for the passed time step:

$$x_l' = \begin{bmatrix} x_l' \\ p_l' \\ g_l' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_l \\ p_l \\ g_l \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} 1.3 \cdot v_{\max} \cdot \Delta t. \tag{6.12}$$

For the cases that an existing phantom vehicle is still at the edge of the FoV, a sample ϕ is drawn form the Bernoulli distribution, s.t. $\phi \sim P_\phi(\Delta\Psi)$.

Case 2b: $g = 1 \wedge \phi = 0$. If a phantom vehicle is sampled to not drive out of the occlusion ($\phi = 0$), it is simply set to the new field of view:

$$x'_l = \left[\max(\Psi_l, \Psi'_l), p_l, g_l\right]^{\mathrm{T}}. \tag{6.13}$$

Case 2c: $g = 1 \wedge \phi = 1$. A vehicle is sampled to drive out of the occlusion ($\phi = 1$). The min operator is needed to respect the cases in which the environment changes in a way s.t. the FoV decreases:

$$x'_l = \left[\min(\Psi_l, \Psi'_l) - 1.3\Delta v_{\max} t, p_l, g_l\right]^{\mathrm{T}}. \tag{6.14}$$

6.2.5 Reward Model

The immediate reward in POMDPs is defined for a state-action pair. To balance various objectives, the overall reward contains different terms:

$$R(x, a) = R_{\mathrm{act}}(a_0) + R_{\mathrm{vel}}(v) + R_{\mathrm{coll}}(x). \tag{6.15}$$

Accelerations are punished with a small negative reward, s.t. $R_{\mathrm{act}} = -100a_0^2$, to maximize comfort.

$$R_{\mathrm{vel}}(x) = \begin{cases} -400 \cdot |v_0 - v_{\mathrm{des}}| & , v_0 \leq v_{\mathrm{des}} \\ -400 \cdot (v_0 - v_{\mathrm{des}})^2 & , \text{otherwise} \end{cases}. \tag{6.16}$$

A crash with a (phantom) vehicle is punished with $R_{\mathrm{coll}} = -20000$. In the case of combined 2-D planning, a special lateral reward, $R_{\mathrm{center}} = R_{\mathrm{FOV}}(x_0) + R_{\mathrm{act,lat}} + R_{\mathrm{d}}(d_0)$, is added to the the reward functional. R_{FOV} motivates increasing the FoV, while $R_{\mathrm{lat,acc}}$ and R_{d} punish lateral accelerations or driving with lateral offset.

Table 6.1: Parameter values for the evaluation of the occlusion POMDP planner.

c	20000	δ_t	1 s
γ	0.8	t_{hor}	6s
ρ	$1\,\frac{\text{agent}}{100\,\text{m}}$		

6.2.6 Implementation

The algorithm is implemented by use of the ABT algorithm. It is implemented in the same way as the algorithms from chapter 4 and chapter 5. The framework and implementation is explained in detail in section 2.4. As mentioned, the algorithm is heavily based on a deterministic roll-out. The optimal behavior in the roll-out is determined with an A^* graph search using a heuristic based on the idea of ICS for collisions with ghost vehicles.

6.3 Results

The occlusion-aware planning approach is evaluated with the parameters given in table 6.1. The POMDP planner of this thesis is evaluated against two other approaches: An omniscient planner, acting with full observability of occluded objects in the scene and a baseline approach, which makes its decision based on the current FoV only. This baseline approach does not simulate the FoV ahead and does therefore not consider how the FoV changes during execution. This is an often used standard approach which is only able to solve the problem by constant replanning. The approach is evaluated in scenarios with static and dynamic occlusions.

6.3.1 Static Occlusion

The algorithm is at first evaluated in a scenario with a static occlusion (see figure 6.4 for the scenario). It is generated by the edge of a house and has the effect that the autonomous car is unable to fully observe another lane. The speed limit is reduced to $v_{\max} = 5.5\,[\frac{\text{m}}{\text{s}}]$, as this emphasizes the different planning behaviors. Results, showing the behavior of the autonomous car are

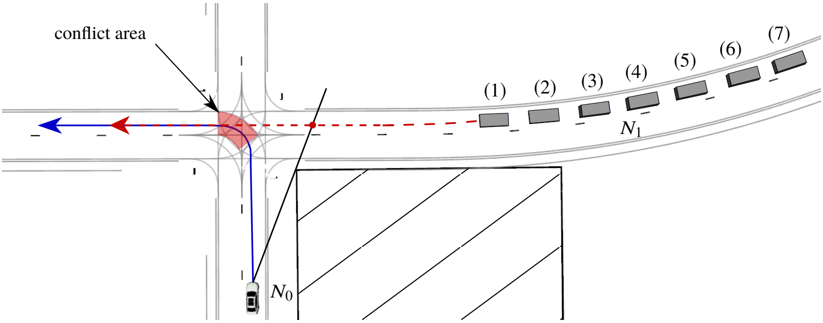


Figure 6.4: An urban scenario with a static occlusion. It is evaluated for several initial configurations of the occluded vehicle (graphic from [123], ©2019 IEEE).

presented in two parts. First, figure 6.5 compares the driven trajectories of the POMDP planner, the omniscient planner and the baseline approach during a simulation run. Secondly, the planner is evaluated in a quantitative way by comparing its performance over many simulations (table 6.2) to account for its probabilistic nature.

Qualitative Evaluation:

figure 6.5 shows the driven trajectories for the scenario demonstrated in figure 6.4 for the different planners (POMDP, omniscient, baseline). In the beginning (until $t = 4s$) all planning approaches accelerate towards the intersection. From $t = 4s$ on, POMDP and the omniscient planner are able to accelerate further, while the baseline approach already starts to decelerate because the current FoV is not yet large enough for safe crossing of the intersection. Because the baseline planner is not able to predict the FoV, it starts to brake immediately. The POMDP planner on the other hand, acts nearly as optimistic as the omniscient approach, due its capability, to already plan several options for future observations. This behavior, of planning for various future scenarios, can also be seen in figure 6.1.

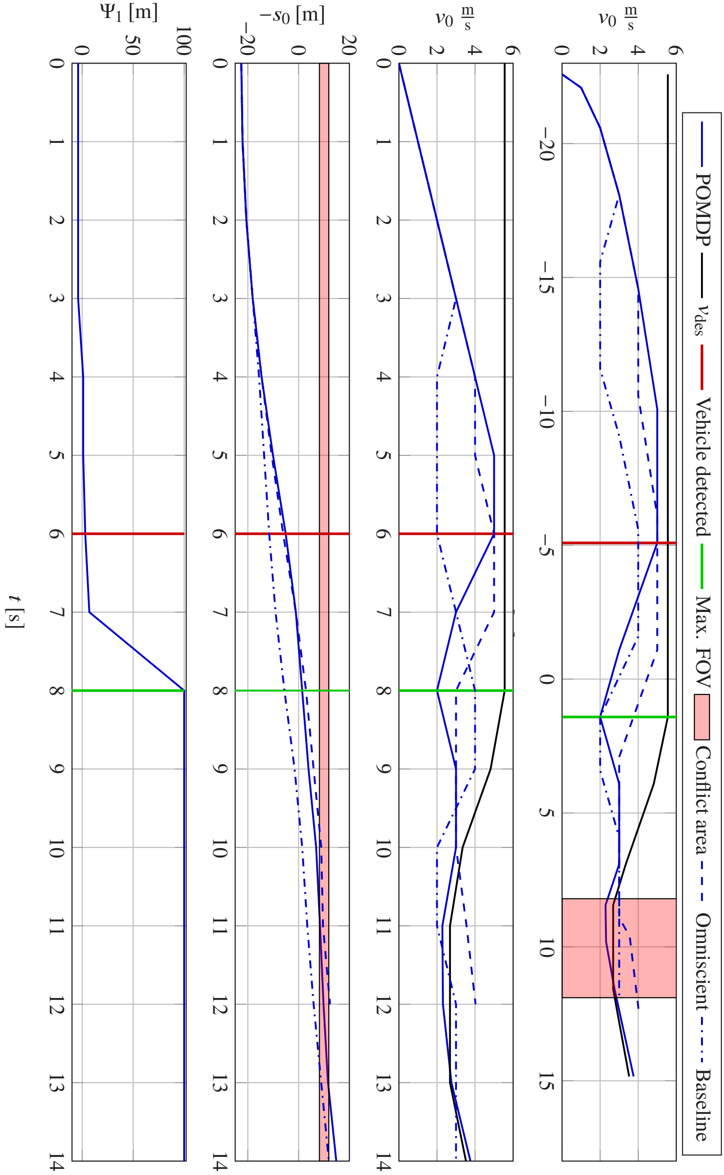


Figure 6.5: Simulation of a scenario with a static occlusion and a close oncoming vehicle (graphic from [123], ©2019 IEEE).

Table 6.2: Evaluation of the scenario with a static occlusion, described in figure 6.5 (graphic from [123], ©2019 IEEE).

Setup	Planner	t [s]	$\sum \lvert a_0 \rvert$ [$\frac{\mathrm{m}}{\mathrm{s}^2}$]
No car	baseline	13.309	10.320
	omniscient	11.237	8.880
	POMDP	11.481	8.660
setup (1)	baseline	13.355	10.560
	omniscient	11.283	8.520
	POMDP	11.632	9.390
setup (2)	baseline	13.942	11.500
	omniscient	12.803	10.670
	POMDP	13.353	13.560
setup (3)	baseline	15.316	14.090
	omniscient	13.881	14.618
	POMDP	15.384	16.490
setup (4)	baseline	16.194	13.817
	omniscient	10.249	9.816
	POMDP	10.654	9.515
setup (5)	baseline	14.262	12.326
	omniscient	10.953	8.720
	POMDP	11.334	9.357
setup (6)	baseline	13.638	11.421
	omniscient	10.946	8.700
	POMDP	11.362	8.930
setup (7)	baseline	13.775	10.540
	omniscient	10.961	8.670
	POMDP	11.358	8.870

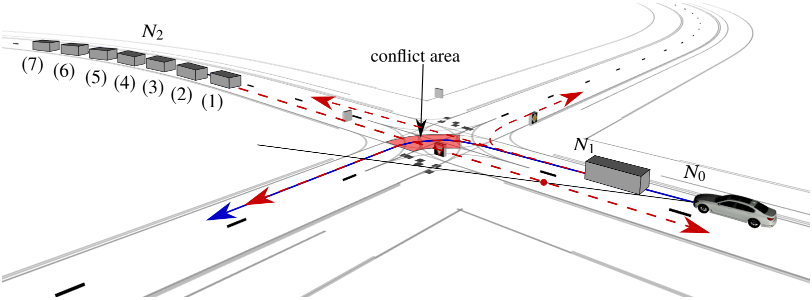


Figure 6.6: A typical scenario in which the autonomous vehicle wants to turn left, but the view is occluded by another vehicle. Different possible configurations of the other vehicle are denoted in brackets (graphic from [123], ©2019 IEEE).

Quantitative Evaluation:

As the POMDP formulation is solved by a sampling-based solver, the generated solution is not deterministic. Moreover, it is not known which configuration of the occluded car is most interesting. Therefore, each of the 7 scenarios presented in figure 6.4 is run 50 times and the results are compared in terms of average time to cross the intersection and in terms of the average, accumulated, absolute acceleration of the autonomous car. This is shown in table 6.2. It can be seen that the POMDP performs nearly as good in terms of average crossing time and comfort as the omniscient planner, due its capability of predicting the FoV. The baseline approach crosses the intersection 30% slower in average and has less comfort (i.e. accelerates/decelerates more often) due its more reactive approach.

6.3.2 Dynamic Occlusion

The planner is also evaluated in a dynamic scenario (see figure 6.6). It presents a typical situation in which the autonomous vehicle intends to turn left at an intersection while the oncoming traffic cannot be observed as it is occluded by a front vehicle of the autonomous car.

The scenario is evaluated in a quantitative way by doing 50 simulations for each planner (POMDP, omniscient, baseline). The results are shown in table 6.3.

Table 6.3: Evaluation of the scenario with a dynamic occlusion, described in figure 6.5 (graphic from [123], ©2019 IEEE).

Setup	Planner	t [s]	$\sum \lvert a_0 \rvert$ [$\frac{\text{m}}{\text{s}^2}$]
No car	baseline	9.505	8.600
	omniscient	9.116	9.100
	POMDP	9.197	9.160
setup (1)	baseline	11.382	10.100
	omniscient	11.724	11.200
	POMDP	11.637	11.071
setup (2)	baseline	13.638	12.960
	omniscient	13.573	13.406
	POMDP	13.491	12.700
setup (3)	baseline	14.274	13.074
	omniscient	10.927	10.925
	POMDP	12.247	11.789
setup (4)	baseline	9.194	9.024
	omniscient	9.087	9.030
	POMDP	9.107	8.900
setup (5)	baseline	9.430	8.745
	omniscient	9.109	9.040
	POMDP	9.201	8.930
setup (6)	baseline	9.584	8.630
	omniscient	9.105	9.090
	POMDP	9.285	8.650
setup (7)	baseline	9.638	8.660
	omniscient	9.109	9.060
	POMDP	9.382	8.730

It can be seen, that the POMDP does not outperform the other planners as strongly as for the static scenario. Only in setup three, the baseline planner is obviously outperformed by the POMDP planner. The reason for this is, that the vehicle in front of the autonomous car drives over the intersection and therefore creates the occlusion only for a short period of time. Therefore, only very specific world configurations are solved with more intelligent behavior by the POMDP planner.

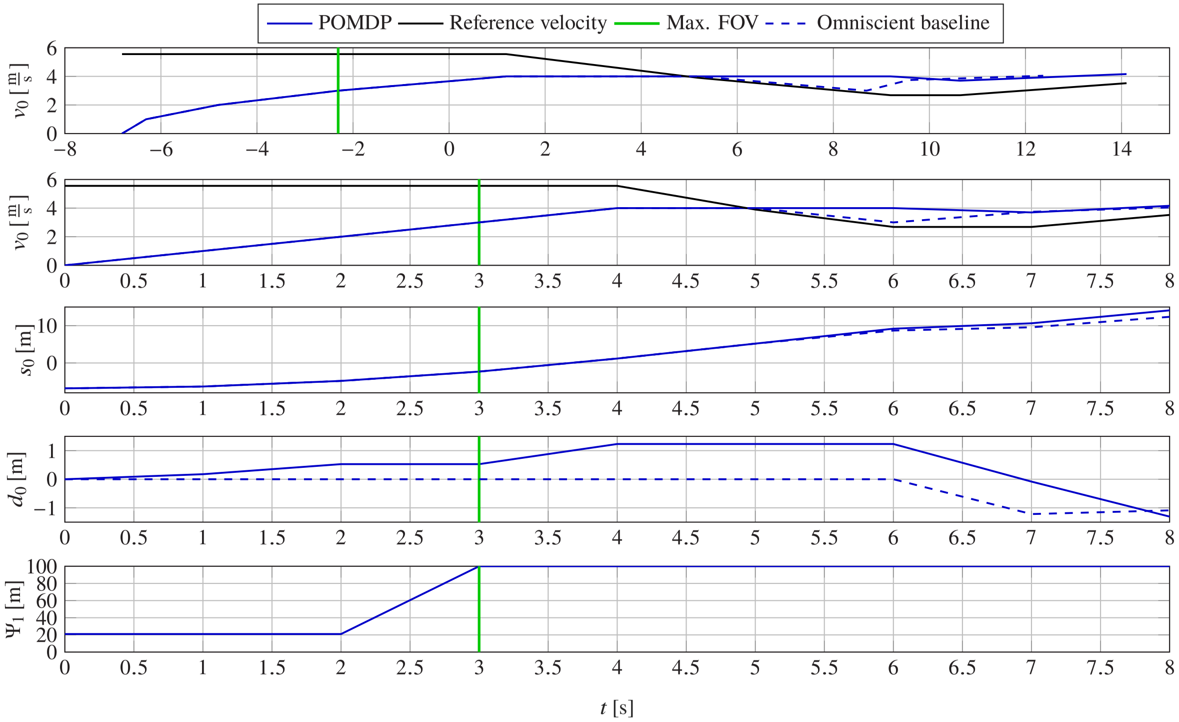
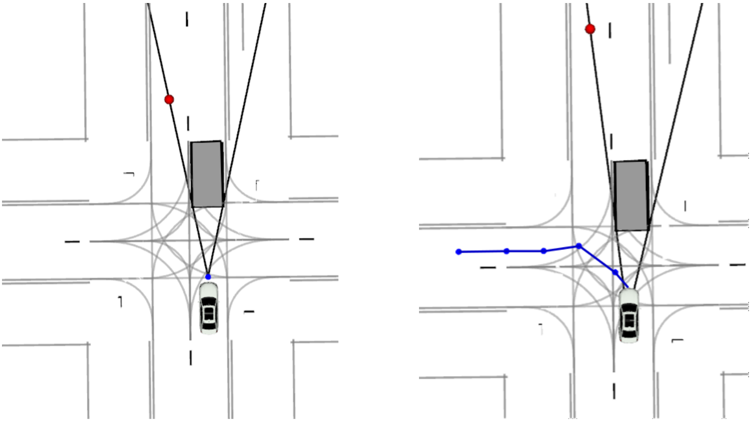


Figure 6.7: Extensive evaluation for combined lateral and longitudinal planning as shown in (graphic from [123], ©2019 IEEE).



(a) No lateral actions result in stillstand of the car, i.e. the Freezing robot problem.

(b) With lateral actions, the FoV can be increased and turning left is possible.

Figure 6.8: Presentation of a scenario where actively exploring the surrounding is necessary. The POMDP planner is once executed without and once with lateral planning (graphic from [123], ©2019 IEEE).

6.3.3 2D Motion Primitives

This sections demonstrates a scenario with the goal of showing results for a combined optimization of lateral and longitudinal optimization including information gathering behavior. The scenario is shown in figure 6.8. The scenario demonstrates an occluded left turn and is set up in a way s.t. that actively exploring the FoV is the only possibility for safe turning. In this scenario, the action vector $\mathcal{A}_{\mathrm{long/lat}}$ is selected to allow for lateral explorative behavior. As this combined optimization problem (lateral, longitudinal, information gathering) is very hard to solve *online* the additional reward R_{lat} is introduced to motivate lateral exploration (both are defined in section 4.3).

The scenario is set up in a way, that the autonomous vehicle is not able to turn left without lateral planning, as the FoV is not big enough for safe traversing.

In figure 6.8, two snapshots of the planning problem are shown, comparing the longitudinal POMDP with the POMDP formulation with $\mathcal{A}_{\mathrm{long/lat}}$. It can be seen, that the longitudinal POMDP gets stuck at the intersection, demonstrating the freezing robot problem.

Additionally, the driven trajectory of the POMDP planner is compared with the driven trajectory of the omniscient planner in figure 6.7. It can be seen that both planners drive nearly the same trajectory with the only difference of a lateral offset of the POMDP planner to increase the FoV.

6.4 Summary

This chapter extends the POMDP formulation of the previous chapters, s.t. explicit reasoning over potentially occluded objects is incorporated. An *online*, *closed-loop* planner is presented which generates optimized behaviors for scenarios with occlusions. The key focus of the algorithm is the

- propagation of the FoV over the planning horizon
- representation of occluded vehicle configurations with reachable sets
- probabilistic reasoning over the existence of occluded vehicles.

The generic POMDP formulation allows to reason over a varying number of occlusions on arbitrary map layouts, generated by *static* or *dynamic* obstacles. The policy is optimized for the existence uncertainty of vehicles in occlusions. The policy contains various, *closed-loop* plans for different future scenarios of (not) revealing vehicles in the occluded areas. Simulating the FoV over the planning horizon allows the autonomous car to choose actions which maximize the FoV for critical areas. The POMDP approach is compared in simulation scenarios to simpler approaches which make decisions based on the current FoV only. It is shown how the POMDP approach is able to outperform these approaches in terms of intersection crossing time and total, accumulated acceleration. The created behavior of the POMDP planner is even similar to the one of an omniscient planner with full knowledge about the configurations of the occluded agents. This is possible because of the capability of the planner to predict what information may be perceived in the future and to postpone the decision accordingly.

7 Conclusion

The first contribution of this thesis is the introduction of optimization based behavior planning and decision making for autonomous driving. Instead of selecting a certain behavior from an a priori defined set of potential behaviors, the planners of this thesis generate an optimal behavior itself. Hereby, the planner respects the predicted trajectories of the other agents as well as traffic rules in one cost functional. The underlying decisions are made implicitly during the non-convex optimization of the trajectory/policy instead of by a rule-based system.

The main contribution of this thesis is the presentation of such a global planner which optimizes in the space of policies instead of trajectories. It considers the uncertainty of the behavior of the other agents as well as their potentially interactive behavior explicitly by combining prediction and planning in one problem. The respected uncertainty of the other agents are namely the unknown intention to follow a certain path, their uncertain longitudinal motion models, possible interaction as well as uncertain measurements and existence uncertainty due to occlusions. The problem is formulated with discretized actions on a continuous state and observation space. The result is a policy over the current belief state, describing the optimal action of the autonomous car, given the most likely future scenarios.

Such a policy allows for less conservative behavior compared to a reactive trajectory planner. This is the case as the optimization for various future scenarios often results in a behavior which postpones decisions under the knowledge that more information about the other drivers is likely to be available in the future. It also allows for actively gathering information about the intent of the other drivers to reduce the uncertainty of the tracked belief space of the world.

Such a generic problem formulation is considered to be intractable to solve on a continuous state space. The second main contribution of this thesis is to demonstrate how such a problem formulation can be solved *online*. This is realized by combining state of the art solvers, based on MCTS, with domain specific heuristics.

Summarized, the main contributions of the thesis are:

- Formulation of behavior generation as a POMDP
- Formulation on a continuous state and observation space
- Combining a state of the art solver with domain specific near-optimal roll-outs
- Solving a intertwined planning-prediction formulation *online*
- Evaluation of all algorithms in various, *online* simulation scenarios

The capabilities of the planner are incrementally advanced and demonstrated throughout the thesis:

At first, chapter 3 presents how a sequential decision making formulation can be used to generate one behavior (i.e. a single trajectory) which is optimal given various, future, deterministic events on the road.

In chapter 4, the problem formulation is advanced to a belief state. The algorithm handles uncertain prediction as well as the interaction of the other traffic participants with the autonomous car. This allows the algorithm to act non-conservatively for scenarios in which dynamic agents with uncertain prediction are potentially crossing or merging on the path of the autonomous car. The algorithm optimizes the policy on a longitudinal path over the uncertain belief state of possible predicted trajectories of the other agents.

In chapter 5 the algorithm is advanced in two ways. At first longitudinal and lateral optimization is combined in one problem formulation. This allows the algorithm to adapt the right longitudinal speed and merge in a gap at the same time. Secondly, the algorithm is able to perfom so-called *information gathering* actions (approach a certain gap) to estimate the intent to yield of other drivers. This allows the algorithm to merge in gaps, which are initially too small. The algorithm generates a combined longitudinal and lateral policy over the belief space.

Finally, chapter 6 presents an extension to the algorithm which allows to reason over the existence uncertainty of potentially occluded objects. By sampling different scenarios, the algorithm is able to implicitly find a optimal policy for the autonomous car, with a sufficient field of view to act safely. It can implicitly predict at what point in the future enough information will be available. The resulting policy contains respective plans to act accordingly, given the corresponding observation. The algorithm generates a policy over the belief of possibly existing, occluded objects.

7.1 Future Research Directions

The presented algorithms can be further advanced in several directions.

The current problem formulation, does not allow to formally guarantee safety as it optimizes the expected reward. Changing the problem formulation to POMDPs with constraints would allow to guarantee safety on a theoretical basis. Secondly, just as the problem formulation, the approximate solver also cannot give safety guarantees. This is due to the sampling-based nature of the solver which can miss important episodes in theory. This could be overcome by extending the overall architecture, s.t. formal verification methods are applied on the optimal action before execution. These changes would allow for a formally safe architecture.

The particle filter which tracks the belief state over time is implemented as an unweighted version. This is done as the real observation of the filter is directly matched on one observation in the belief tree to keep the tree alive. As the episodes in the belief tree are unweighted itself, matching of observations is easier to fulfill by use of an unweighted particle filter. Nonetheless, the performance of the particle filter is not very efficient, due to the large step-size of the POMDP planner and the unweighted version of the filter. Further research could go in the direction on how to use weighted particle filters with a higher frequency, while simultaneously keeping the belief tree intact.

The particle filter as well as the policy optimization itself may be improved by using learned instead of hand-tuned motion models. First promising results were already realized during this thesis [128].

Further room for improvement lies in the solver itself. All state of the art solvers rely on Monte-Carlo sampling of episodes. The procedure of sampling possible episodes to construct the belief tree is predestined for parallelization to speed up the solving of POMDP formulations by magnitudes. Parallelized Monte-Carlo solvers have been published recently [21].

For the case that drastically faster solving of POMDPs becomes tractable in the future, the coupling between behavior planner and trajectory planner may be revised. Under the assumption, that the problem can be solved for a larger action space, the possible configuration space of the autonomous car can be sampled more densely. By combining this with sampling over the parameters of motion primitives that are optimized for minimum jerk/acceleration, smoother policies may be created which could make the trajectory planner obsolete. The technique of progressive widening would allow to even generate a policy on a continuous parameter/action space.

Bibliography

Literature

[1] Waymo press footage. https://waymo.com/press/ [Online. Accessed: 2019-08-23] 2019.

[2] M. Althoff, “Reachability analysis and its application to the safety assessment of autonomous cars,” PhD thesis, Technische Universität München, 2010.

[3] M. Ardelt, C. Coester, and N. Kaempchen, “Highly Automated Driving on Freeways in Real Traffic Using a Probabilistic Framework,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1576–1585, 2012.

[4] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, “Intention-aware online POMDP planning for autonomous driving in a crowd,” in *IEEE International Conference on Robotics and Automation*, pp. 454–460, 2015.

[5] Y. Bai, Z. J. Chong, M. H. Ang, and X. Gao, “An Online Approach for Intersection Navigation of Autonomous Vehicle,” in *IEEE International Conference on Robotics and Biomimetics*, pp. 2127–2132, 2014.

[6] M. Bansal, A. Krizhevsky, and A. S. Ogale, “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst,” arXiv:1812.03079 [cs.RO]. 2018.

[7] H. Banzhaf, M. Dolgov, J. Stellet, and J. M. Zollner, “From Footprints to Beliefprints: Motion Planning under Uncertainty for Maneuvering Automated Vehicles in Dense Scenarios,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1680–1687, 2018.

[8] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[9] P. Bender, O. Ş. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *IEEE Intelligent Vehicles Symposium*, pp. 1386–1392, 2015.

[10] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *IEEE Intelligent Vehicles Symposium*, pp. 420–425, 2014.

[11] S. Bhattacharya and R. Ghrist, "Path Homotopy Invariants and their Application to Optimal Trajectory Planning," in *IMA Conference on Mathematics of Robotics*, pp. 139–160, 2015.

[12] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *AAAI Conference on Artificial Intelligence*, pp. 1230–1237, 2010.

[13] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner *et al.*, "End to end learning for self-driving cars," arXiv:1604.07316 [cs.CV]. 2016.

[14] B. Bonet, "An epsilon-Optimal Grid-Based Algorithm for Partially Observable Markov Decision Processes," in *International Conference on Machine Learning*, pp. 51–58, 2002.

[15] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief State Planning for Autonomously Navigating Urban Intersections," in *IEEE Intelligent Vehicles Symposium*, pp. 825–830, 2017.

[16] M. Bouton, A. Nakhaei, K. Fujimura, and M. Kochenderfer, "Scalable Decision Making with Sensor Occlusions for Autonomous Driving," in *IEEE International Conference on Robotics and Automation*, pp. 2076–2081, 2018.

[17] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1537–1542, 2011.

[18] S. Brechtel, T. Gindele, and R. Dillmann, "Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation," in *International Conference on Machine Learning*, pp. 370–378, 2013.

[19] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas *et al.*, “A Survey of Monte Carlo Tree Search Methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[20] M. Buehler, K. Iagnemma, and S. Singh, “The darpa urban challenge: Autonomous vehicles in city traffic,” *Springer Tracts in Advanced Robotics*, 2009.

[21] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, “Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty,” arXiv:1802.06215 [cs.AI]. 2018.

[22] R. Coulom, “Efficient selectivity and backup operators in monte-carlo tree search,” in *International Conference on Computers and Games*, pp. 72–83. Springer, 2006.

[23] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, “MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving,” in *IEEE International Conference on Robotics and Automation*, pp. 1670–1677, 2015.

[24] F. Damerow and J. Eggert, “Risk-aversive behavior planning under multiple situations with uncertainty,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 656–663, 2015.

[25] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numererical Mathmatics*, vol. 1, no. 1, pp. 269–271, 1959.

[26] J. D’Onfro. ’I hate them’: Locals reportedly are frustrated with alphabet’s self-driving cars. https://www.cnbc.com/2018/08/28/locals-reportedly-frustrated-with-alphabets-waymo-self-driving-cars.html [Online. Accessed: 2019-08-23] 2018.

[27] C. Dong, J. M. Dolan, and B. Litkouhi, “Interactive ramp merging planning in autonomous driving: Multi-Merging leading PGM (MML-PGM),” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1–6, 2017.

[28] N. Du Toit and J. Burdick, “Robot motion planning in dynamic, uncertain environments,” *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2012.

[29] D. Ferguson, T. M. Howard, and M. Likhachev, “Motion planning in urban environments,” *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[30] D. Ferguson and A. Stentz, “The field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments,” *CMU Technical Report*, 2005.

[31] T. Fraichard and H. Asama, “Inevitable collision states. a step towards safer robots?” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 388–393, 2003.

[32] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, “Multi-policy decision-making for autonomous driving via changepoint-based behavior prediction,” in *Robotics: Science and Systems*, 2015.

[33] E. Galceran, E. Olson, and R. M. Eustice, “Augmented vehicle tracking under occlusions for decision-making in autonomous driving,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3559–3565, 2015.

[34] T. Gindele, D. Jagszent, B. Pitzer, and R. Dillmann, “Design of the planner of Team AnnieWAYs autonomous vehicle used in the DARPA Urban Challenge 2007,” in *IEEE Intelligent Vehicles Symposium*, pp. 1131–1136, 2008.

[35] H. Goldstein, C. Poole, and J. Safko, *Classical Mechanics*. Addison Wesley, 2002.

[36] D. González, J. Pérez, V. Milanés, and F. Nashashibi, “A review of motion planning techniques for automated vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[37] T. Gu, J. M. Dolan, and J.-W. Lee, “Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5474–5480, 2016.

[38] B. Gutjahr, L. Gröll, and M. Werling, “Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC,” *IEEE Trans-*

actions on Intelligent Transportation Systems, vol. 18, no. 6, pp. 1586 – 1595, 2017.

[39] A. Hochstädter, P. Zahn, and K. Breuer, "A comprehensive driver model with application to traffic simulation and driving simulators," in *IEEE Human-Centered Transportation Simulation Conference*, 2001.

[40] S. Hoermann, F. Kunz, D. Nuss, S. Renter, and K. Dietmayer, "Entering Crossroads with Blind Corners. A Safe Strategy for Autonomous Vehicles," in *IEEE Intelligent Vehicles Symposium*, pp. 727–732, 2017.

[41] D. Hsu, W. S. Lee, and N. Rong, "What makes some POMDP problems easy to approximate?" in *Advances in Neural Information Processing Systems*, pp. 689–696, 2007.

[42] D. Isele, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning," in *ArXiv*, 2017, arXiv:1705.01196 [cs.AI].

[43] J. Johnson and K. Hauser, "Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path," in *IEEE International Conference on Robotics and Automation*, pp. 2035–2041, 2012.

[44] J. Johnson and K. Hauser, "Optimal longitudinal control planning with moving obstacles," in *IEEE Intelligent Vehicles Symposium*, pp. 605–611, 2013.

[45] S. Julier and J. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[46] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[47] S. Kammel, J. Ziegler, B. Pitzer, M. Werling *et al.*, "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[48] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[49] K. Kaur and G. Rampersad, “Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars,” *Journal of Engineering and Technology Management*, vol. 48, pp. 87–96, 2018.

[50] D. Klimenko, J. Song, and H. Kurniawati, “Tapir: A software toolkit for approximating and adapting pomdp solutions online,” in *Proc. Australasian Conference on Robotics and Automation*, 2014.

[51] M. J. Kochenderfer, C. Amato, G. Chowdhary, J. P. How *et al.*, *Decision Making Under Uncertainty: Theory and Application*, 1st ed. The MIT Press, 2015.

[52] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *European Conference on Machine Learning*, pp. 282–293, 2006.

[53] R. Kohlhaas, T. Bittner, T. Schamm, and J. M. Zöllner, “Semantic state space for high-level maneuver planning in structured traffic scenes,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1060–1065, 2014.

[54] R. Kohlhaas, D. Hammann, T. Schamm, and J. M. Zöllner, “Planning of high-level maneuver sequences on semantic state spaces,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 2090–2096, 2015.

[55] R. Kohlhaas, T. Schamm, D. Nienhüser, and J. M. Zöllner, “Anticipatory energy saving assistant for approaching slower vehicles,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1966–1971, 2011.

[56] R. Kohlhaas, T. Schamm, D. Lenk, and J. M. Zollner, “Towards driving autonomously: Autonomous cruise control in urban environments,” in *IEEE Intelligent Vehicles Symposium*, pp. 116–121, 2013.

[57] H. Kurniawati, D. , and W. S. Lee, “SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces.” in *Robotics: Science and Systems*, 2008.

[58] H. Kurniawati and V. Yadav, “An online POMDP solver for uncertainty planning in dynamic environment,” in *International Symposium on Robotics Research*, pp. 611–629, 2013.

[59] K. Kurzer, F. Engelhorn, and J. M. Zöllner, “Decentralized Cooperative Planning for Automated Vehicles with Continuous Monte Carlo Tree Search,” in *IEEE Intelligent Vehicles Symposium*, pp. 452–459, 2018.

[60] W. H. Kwon, A. M. Bruckstein, and T. Kailath, “Stabilizing state-feedback design via the moving horizon method,” *International Journal of Control*, vol. 37, no. 3, pp. 631–643, 1983.

[61] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[62] S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” Tech. Rep., 1998.

[63] A. Lawitzky, D. Wollherr, and M. Buss, “Energy optimal control to approach traffic lights,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4382–4387, 2013.

[64] P. Le Beau. Waymo starts commercial ride-share service. https://www.cnbc.com/2018/12/05/waymo-starts-commercial-ride-share-service.html [Online. Accessed: 2019-08-23]

[65] S. Lefèvre, D. Vasquez, and C. Laugier, “A survey on motion prediction and risk assessment for intelligent vehicles,” *Robomech Journal*, vol. 1, no. 1, pp. 1–14, 2014.

[66] D. Lenz, T. Kessler, and A. Knoll, “Stochastic model predictive controller with chance constraints for comfortable and safe driving behavior of autonomous vehicles,” in *IEEE Intelligent Vehicles Symposium*, pp. 292–297, 2015.

[67] D. Lenz, T. Kessler, and A. Knoll, “Tactical Cooperative Planning for Autonomous Highway Driving using Monte-Carlo Tree Search,” in *IEEE Intelligent Vehicles Symposium*, pp. 447–453, 2016.

[68] M. Likhachev and D. Ferguson, “Planning long dynamically feasible maneuvers for autonomous vehicles,” *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[69] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, “Learning policies for partially observable environments: Scaling up,” in *International Conference on Machine Learning*, pp. 362–370, 1995.

[70] O. Madani, S. Hanks, and A. Condon, “On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems,” in *AAAI Conference on Artificial Intelligence*, 1999.

[71] S. Mandava, K. Boriboonsomsin, and M. Barth, “Arterial velocity planning based on traffic signal information under light traffic conditions,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1–6, 2009.

[72] A. Mazzalai, F. Biral, M. D. Lio, M. Darin, and L. D’Orazio, “Automated crossing of intersections controlled by traffic lights,” in *IEEE International Conference on Intelligent Transportation Systems*, 2015.

[73] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp *et al.*, “Junior: The stanford entry in the urban challenge,” *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[74] J. Nilsson, M. Brannstrom, J. Fredriksson, and E. Coelingh, “Longitudinal and Lateral Control for Automated Yielding Maneuvers,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1404–1414, 2016.

[75] T. Nishi, P. Doshi, and D. Prokhorov, “Freeway Merging in Congested Traffic based on Multipolicy Decision Making with Passive Actor Critic,” arXiv:1707.04489 [cs.AI]. 2017.

[76] A. Nunes and K. Hernandez, “Autonomous vehicles and public health: High cost or high opportunity cost?” psyarXiv:10.31234/osf.io/6e94h. 2019.

[77] P. F. Orzechowski, A. Meyer, and M. Lauer, “Tackling Occlusions & Limited Sensor Range with Set-based Safety Verification,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1729–1736, 2018.

[78] B. Paden, M. Cap, S. Yong, D. Yershov, and E. Frazzoli, “A Survey of Motion Planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, 2016.

[79] C. Papadimitriou and J. Tsisiklis, “The complexity of markov decision processes,” *Mathematics of Operations Research*, pp. 441 – 450, 1987.

[80] C. Pek, P. Zahn, and M. Althoff, “Verifying the safety of lane change maneuvers of self-driving vehicles based on formalized traffic rules,” in *IEEE Intelligent Vehicles Symposium*, pp. 1477–1483, 2017.

[81] J. Pineau, G. Gordon, and S. Thrun, “Point-based value iteration an anytime algorithm for POMDPs,” in *International Joint conference on Artificial Intelligence*, pp. 1025–1032, 2003.

[82] M. Pivtoraiko and A. Kelly, “Efficient constrained path planning via search in state lattices,” in *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005.

[83] M. Pivtoraiko, R. A. Knepper, and A. Kelly, “Differentially constrained mobile robot motion planning in state lattices,” *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[84] X. Qian, F. Altche, P. Bender, C. Stiller, and A. de La Fortelle, “Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective,” in *IEEE International Conference on Intelligent Transportation Systems*, pp. 205–210, 2016.

[85] S. Ross, J. Pineau, S. Paquet, and B. Chaib Draa, “Online planning algorithms for POMDPs,” *Journal of Artificial Intelligence Research*, vol. 32, pp. 663–704, 2008.

[86] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 2009.

[87] M. Sadou, V. Polotski, and P. Cohen, “Occlusions in Obstacle Detection for Safe Navigation,” in *IEEE Intelligent Vehicles Symposium*, pp. 716–721, 2004.

[88] SAE International, “Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems,” SAE International, Tech. Rep. SAE J 3016, 2014.

[89] J. Schulman, J. Ho, A. Lee, I. Awwal *et al.*, “Finding locally optimal, collision-free trajectories with sequential convex optimization,” in *Robotics: Science and Systems*, 2013.

[90] J. Schulz, K. Hirsenkorn, J. Löchner, M. Werling, and D. Burschka, “Estimation of collective maneuvers through cooperative multi-agent

planning," in *IEEE Intelligent Vehicles Symposium*, pp. 624–631, 2017.

[91] P. Schörner, L. Tröttel, J. Doll, and M. Zöllner, "Predictive Trajectory Planning in Situations with Hidden Road Users Using Partially Observable Markov Decision Processes," in *IEEE Intelligent Vehicles Symposium*, pp. 2299–2306, 2019.

[92] V. Sezer, T. Bandyopadhyay, D. Rus, E. Frazzoli, and D. Hsu, "Towards autonomous navigation of unsignalized intersections under uncertainty of human driver intent," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3578–3585, 2015.

[93] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[94] Z. Shiller, "Off-Line and On-Line Trajectory Planning," *Motion and Operation Planning of Robotic Systems*, vol. 29, pp. 29–62, 2015.

[95] D. Silver, A. Huang, C. J. Maddison, A. Guez *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[96] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, pp. 2164–2172, 2010.

[97] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," *NHTSA Traffic Safety Facts*, 2015.

[98] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *AUAI Conference on Uncertainty in artificial intelligence*, pp. 520–527, 2004.

[99] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems*, pp. 1772–1780, 2013.

[100] E. J. Sondik, "The optimal control of partially observable markov processes," PhD thesis, Stanford University, 1971.

[101] S. Sontges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *IEEE Intelligent Vehicles Symposium*, pp. 160–166, 2017.

[102] C. Stiller, G. Färber, and S. Kammel, "Cooperative cognitive automobiles," in *IEEE Intelligent Vehicles Symposium*, pp. 215–220, 2007.

[103] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 797–803, 2010.

[104] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical observations and Microscopic Simulations," in *Physical Revue E Interdisciplinary Topics*, vol. 62, no. 2, pp. 1805–1824, 2000.

[105] S. Ulbrich and M. Maurer, "Probabilistic online POMDP decision making for lane changes in fully automated driving," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 2063–2067, 2013.

[106] C. Urmson. Ted: How a driverless car sees the road. https://www.youtube.com/watch?v=tiwVMrTLUWg [Online. Accessed: 2019-04-01] 2015.

[107] C. Urmson, J. Anhalt, D. Bagnell, C. Baker *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[108] P. Vack. Self-drive cars and you: A history longer than you think. https://www.velocetoday.com/self-drive-cars-and-you-a-history-longer-than-you-think/ [Online. Accessed: 2019-08-23] 2014.

[109] E. Velenis, "Analysis and control of high-speed wheeled vehicles," PhD thesis, Georgia Institute of Technology, 2006.

[110] M. Werling, "Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien," PhD thesis, Karlsruher Institute für Technologie, 2010.

[111] M. Werling and L. Gröll, "Low-level controllers realizing high-level decisions in an autonomous vehicle," in *IEEE Intelligent Vehicles Symposium*, pp. 1113–1118, 2008.

[112] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenet Frame," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 987–993, 2010.

[113] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-Partitioned Environmental Representation and Planning Architecture for On-Road Autonomous Driving," in *IEEE Intelligent Vehicles Symposium*, pp. 632–639, 2017.

[114] D. Zhao, "Supervised adaptive dynamic programming based adaptive cruise control," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 318–323, 2011.

[115] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha - a local, continuous method," in *IEEE Intelligent Vehicles Symposium*, pp. 450–457, 2014.

[116] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1879–1884, 2009.

[117] J. Ziegler, P. Bender *et al.*, "Making Bertha Drive - An Autonomous Journey on a Historic Route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[118] J. R. Ziehn, M. Ruf, D. Willersinn, B. Rosenhahn *et al.*, "A tractable interaction model for trajectory planning in automated driving," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1410–1417, 2016.

[119] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko *et al.*, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

Publications of the Author

[120] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, "A Combined Model- and Learning-Based Framework for Interaction-Aware Maneuver Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1538–1550, 2016.

[121] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1010–1016, 2016.

[122] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *IEEE Intelligent Vehicles Symposium*, pp. 1671–1678, 2017.

[123] C. Hubmann, N. Quetschlich, J. Schulz, J. Bernhard *et al.*, "A POMDP Maneuver Planner For Occlusions in Urban Scenarios," in *IEEE Intelligent Vehicles Symposium*, pp. 2172–2179, 2019.

[124] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated Driving in Uncertain Environments: Planning with Interaction and Uncertain Maneuver Prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 5–17, 2018.

[125] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller, "A belief state planner for interactive merge maneuvers in congested traffic," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1617–1624, 2018.

[126] J. Schulz, C. Hubmann, J. Löchner, and D. Burschka, "Interaction-aware probabilistic behavior prediction in urban environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3999–4006, 2018.

[127] J. Schulz, C. Hubmann, J. Lochner, and D. Burschka, "Multiple Model Unscented Kalman Filtering in Dynamic Bayesian Networks for Intention Estimation and Trajectory Prediction," in *IEEE International Conference on Intelligent Transportation Systems*, pp. 1467–1474, 2018.

[128] J. Schulz, C. Hubmann, N. Morin, and J. Loechner, "Learning Interaction-Aware Probabilistic Driver Behavior Models from Urban Scenarios," in *IEEE Intelligent Vehicles Symposium*, pp. 1326–1333, 2019.

Supervised Theses

[129] M. Becker, "Decision making for autonomous driving at urban intersections under intention-uncertainty of other vehicles," Master's thesis, fortiss GmbH and Robotics and Embedded Systems Group, Technische Universität München, 2016.

[130] N. Quetschlich, "Handling occlusions in urban scenarios: A belief state planner for autonomous driving," Master's thesis, fortiss GmbH and Robotics and Embedded Systems Group, Technische Universität München, 2018.

Schriftenreihe
Institut für Mess- und Regelungstechnik
Karlsruher Institut für Technologie
(1613-4214)

Band 001 Hans, Annegret
Entwicklung eines Inline-Viskosimeters
auf Basis eines magnetisch-induktiven
Durchflussmessers.
ISBN 3-937300-02-3

Band 002 Heizmann, Michael
Auswertung von forensischen Riefenspuren
mittels automatischer Sichtprüfung.
ISBN 3-937300-05-8

Band 003 Herbst, Jürgen
Zerstörungsfreie Prüfung von Abwasserkanälen
mit Klopfschall.
ISBN 3-937300-23-6

Band 004 Kammel, Sören
Deflektometrische Untersuchung spiegelnd
reflektierender Freiformflächen.
ISBN 3-937300-28-7

Band 005 Geistler, Alexander
Bordautonome Ortung von Schienenfahrzeugen
mit Wirbelstrom-Sensoren.
ISBN 978-3-86644-123-1

Band 006 Horn, Jan
Zweidimensionale Geschwindigkeitsmessung
texturierter Oberflächen mit flächenhaften
bildgebenden Sensoren.
ISBN 978-3-86644-076-0

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 007 Hoffmann, Christian
Fahrzeugdetektion durch Fusion monoskopischer Videomerkmale.
ISBN 978-3-86644-139-2

Band 008 Dang, Thao
Kontinuierliche Selbstkalibrierung von Stereokameras.
ISBN 978-3-86644-164-4

Band 009 Kapp, Andreas
Ein Beitrag zur Verbesserung und Erweiterung der Lidar-Signalverarbeitung für Fahrzeuge.
ISBN 978-3-86644-174-3

Band 010 Horbach, Jan
Verfahren zur optischen 3D-Vermessung spiegelnder Oberflächen.
ISBN 978-3-86644-202-3

Band 011 Böhringer, Frank
Gleisselektive Ortung von Schienenfahrzeugen mit bordautonomer Sensorik.
ISBN 978-3-86644-196-5

Band 012 Xin, Binjian
Auswertung und Charakterisierung dreidimensionaler Messdaten technischer Oberflächen mit Riefentexturen.
ISBN 978-3-86644-326-6

Band 013 Cech, Markus
Fahrspurschätzung aus monokularen Bildfolgen für innerstädtische Fahrerassistenzanwendungen.
ISBN 978-3-86644-351-8

Band 014 Speck, Christoph
Automatisierte Auswertung forensischer Spuren auf Patronenhülsen.
ISBN 978-3-86644-365-5

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 015 Bachmann, Alexander
Dichte Objektsegmentierung in Stereobildfolgen.
ISBN 978-3-86644-541-3

Band 016 Duchow, Christian
Videobasierte Wahrnehmung markierter Kreuzungen mit lokalem Markierungstest und Bayes'scher Modellierung.
ISBN 978-3-86644-630-4

Band 017 Pink, Oliver
Bildbasierte Selbstlokalisierung von Straßenfahrzeugen.
ISBN 978-3-86644-708-0

Band 018 Hensel, Stefan
Wirbelstromsensorbasierte Lokalisierung von Schienenfahrzeugen in topologischen Karten.
ISBN 978-3-86644-749-3

Band 019 Carsten Hasberg
Simultane Lokalisierung und Kartierung spurgeführter Systeme.
ISBN 978-3-86644-831-5

Band 020 Pitzer, Benjamin
Automatic Reconstruction of Textured 3D Models.
ISBN 978-3-86644-805-6

Band 021 Roser, Martin
Modellbasierte und positionsgenaue Erkennung von Regentropfen in Bildfolgen zur Verbesserung von videobasierten Fahrerassistenzfunktionen.
ISBN 978-3-86644-926-8

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 022 Loose, Heidi
Dreidimensionale Straßenmodelle für Fahrerassistenzsysteme auf Landstraßen.
ISBN 978-3-86644-942-8

Band 023 Rapp, Holger
Reconstruction of Specular Reflective Surfaces using Auto-Calibrating Deflectometry.
ISBN 978-3-86644-966-4

Band 024 Moosmann, Frank
Interlacing Self-Localization, Moving Object Tracking and Mapping for 3D Range Sensors.
ISBN 978-3-86644-977-0

Band 025 Geiger, Andreas
Probabilistic Models for 3D Urban Scene Understanding from Movable Platforms.
ISBN 978-3-7315-0081-0

Band 026 Hörter, Marko
Entwicklung und vergleichende Bewertung einer bildbasierten Markierungslichtsteuerung für Kraftfahrzeuge.
ISBN 978-3-7315-0091-9

Band 027 Kitt, Bernd
Effiziente Schätzung dichter Bewegungsvektorfelder unter Berücksichtigung der Epipolargeometrie zwischen unterschiedlichen Ansichten einer Szene.
ISBN 978-3-7315-0105-3

Band 028 Lategahn, Henning
Mapping and Localization in Urban Environments Using Cameras.
ISBN 978-3-7315-0135-0

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 029 Tischler, Karin
Informationsfusion für die kooperative Umfeldwahrnehmung vernetzter Fahrzeuge.
ISBN 978-3-7315-0166-4

Band 030 Schmidt, Christian
Fahrstrategien zur Unfallvermeidung im Straßenverkehr für Einzel- und Mehrobjektszenarien.
ISBN 978-3-7315-0198-5

Band 031 Firl, Jonas
Probabilistic Maneuver Recognition in Traffic Scenarios.
ISBN 978-3-7315-0287-6

Band 032 Schönbein, Miriam
Omnidirectional Stereo Vision for Autonomous Vehicles.
ISBN 978-3-7315-0357-6

Band 033 Nicht erschienen

Band 034 Liebner, Martin
Fahrerabsichtserkennung und Risikobewertung für warnende Fahrerassistenzsysteme.
ISBN 978-3-7315-0508-2

Band 035 Ziegler, Julius
Optimale Trajektorienplanung für Automobile.
ISBN 978-3-7315-0553-2

Band 036 Harms, Hannes
Genauigkeitsuntersuchung von binokularen Normalenvektoren für die Umfeldwahrnehmung.
ISBN 978-3-7315-0628-7

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 037 Ruhhammer, Christian
Inferenz von Kreuzungsinformationen aus Flottendaten.
ISBN 978-3-7315-0721-5

Band 038 Stein, Denis
Mobile laser scanning based determination of railway network topology and branching direction on turnouts.
ISBN 978-3-7315-0743-7

Band 039 Yi, Boliang
Integrated Planning and Control for Collision Avoidance Systems.
ISBN 978-3-7315-0785-7

Band 040 Schwarze, Tobias
Compact Environment Modelling from Unconstrained Camera Platforms.
ISBN 978-3-7315-0801-4

Band 041 Knorr, Moritz
Self-Calibration of Multi-Camera Systems for Vehicle Surround Sensing.
ISBN 978-3-7315-0765-9

Band 042 Rabe, Johannes
Lane-Precise Localization with Production Vehicle Sensors and Application to Augmented Reality Navigation.
ISBN 978-3-7315-0854-0

Band 043 Weiser, Andreas
Probabilistische Vorhersage von Fahrstreifenwechseln für hochautomatisiertes Fahren auf Autobahnen.
ISBN 978-3-7315-0794-9

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 044 Tian, Wei
Novel Aggregated Solutions for Robust Visual Tracking in Traffic Scenarios.
ISBN 978-3-7315-0915-8

Band 045 Gräter, Johannes
Monokulare Visuelle Odometrie auf Multisensorplattformen für autonome Fahrzeuge.
ISBN 978-3-7315-0935-6

Band 046 Spindler, Max
Ferromagnetische Inhomogenitäten zur berührungslosen Bestimmung der Geschwindigkeit und gleisselektiven Position von Schienenfahrzeugen.
ISBN 978-3-7315-1000-0

Band 047 Sons, Marc
Automatische Erzeugung langzeitverfügbarer Punktmerkmalskarten zur robusten Lokalisierung mit Multi-Kamera-Systemen für automatisierte Fahrzeuge.
ISBN 978-3-7315-1029-1

Band 048 Hubmann, Constantin
Belief State Planning for Autonomous Driving: Planning with Interaction, Uncertain Prediction and Uncertain Perception.
ISBN 978-3-7315-1039-0

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Schriftenreihe

INSTITUT FÜR MESS- UND REGELUNGSTECHNIK
KARLSRUHER INSTITUT FÜR TECHNOLOGIE (KIT)

048

This work presents a behavior planning algorithm for automated driving in urban environments with an uncertain and dynamic nature. The uncertainty exists for two main reasons: First of all, the perception of the surrounding objects is based on noisy sensor measurements and impeded by occlusions. Secondly, the future behavior of the surrounding traffic cannot be measured directly but only estimated in a probabilistic fashion and is also dependent on the action of the autonomous car. Robust and non-conservative decision making in such environments is only possible by taking these uncertainties and interdependencies explicitly into account. Solving this problem exactly is considered intractable why it is avoided in favor of simpler approaches that result in non-optimal behavior.

This work presents how this planning problem can be modeled in a generic way as a Partially Observable Markov Decision Process (POMDP) and how a sufficiently good approximation of the solution can be found online via sampling of the most likely trajectories of all agents. The solution to the POMDP is a policy over belief states, that provides the optimal action given the current uncertainty and all resulting possible future scenarios. The global formulation allows the planning algorithm to make the decision for certain high-level maneuvers implicitly. The explicit consideration of uncertainties allows for intelligent behavior such as postponing decisions by hedging against future possible events. Also, actions are selected to gain more information in the future by reducing occlusions or testing the cooperation of other drivers.

The work demonstrates how this approach outperforms simpler approaches in simulated scenarios such as unprotected turns, lane changes and reasoning under occlusions.

ISBN 978-3-7315-1039-0

9 783731 510390 >

ISSN 1613-4214
ISBN 978-3-7315-1039-0

Gedruckt auf FSC-zertifiziertem Papier