Dynamic Decision-making in Continuous Partially Observable Domains: A Novel Method and its Application for Autonomous Driving

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Sebastian Brechtel

aus Germersheim

Tag der mündlichen Prüfung:	6. Juli 2015
Erster Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann
Zweiter Gutachter:	Prof. Dr. Wolfram Burgard

Abstract

Decision-making is a crucial challenge on the way to autonomous systems, including robots and self-driving vehicles. In real-life tasks, dynamics play a critical role and the ability to anticipate and assess future consequences is essential for making robust decisions. However, this is aggravated by two factors: uncertainty and the continuous nature of the environment. Uncertainty arises because information is limited. The development of situations is stochastic and can only be partially observed. Uncertainty also characterizes the task of autonomous driving in traffic. The behavior of human drivers can only be predicted with uncertainty. Further, most aspects of situations remain hidden because, for example, the view is blocked. Even with good visibility, the precise state of the environment is never known due to inevitable measurement errors. Decision makers are forced to reason about possible developments as well as their own ability to acquire knowledge about these developments. The already high complexity of this task is further increased for continuous space problems, where every situation potentially develops in an infinite number of ways. Additionally, from the viewpoint of the decision maker, every situation can manifest in an infinite number of measurements. Partially observable Markov decision processes (POMDPs) are a principled mathematical framework for modeling sequential decision problems under uncertainty. However, due to their complexity, no sufficient methods for solving complex, high-dimensional continuous POMDPs such as driving in traffic exist.

In this work, we advance the state-of-the-art in artificial intelligence and machine learning by developing a novel, general method for solving continuous POMDPs approximately. In contrast to previous work, we do not impose fundamental prior restrictions on the POMDP's models. Instead, we find adaptive approximations by integrating value-directed machine learning into continuous Monte Carlo value iteration. This way, we simultaneously solve the POMDP and learn a problem-specific, discrete representation for the policy. In addition, we contribute to research in robotics and autonomous driving with a general approach to decision-making for self-driving vehicles. We model the driving problem as a continuous POMDP and solve it with the presented method. Previous approaches either simplify the problem, for instance, by neglecting uncertainties, or are restricted to specific tasks, such as highway driving. Although our approach is not limited to a certain task, it anticipates uncertainties and complex situation developments, including cooperative interaction between drivers.

Zusammenfassung

Die Fähigkeit in der realen Welt Entscheidungen zu treffen ist essentiell für alle Arten von autonomen Systemen, jedoch gleichzeitig höchst komplex. Für eine rational begründete Entscheidungsfindung in dynamischen Systemen müssen die Konsequenzen von Handlungen antizipiert und abgewogen werden. Dies wird dadurch erschwert, dass sich Entwicklungen nur mit Unsicherheit vorhersagen lassen und gleichzeitig nie vollständig und exakt wahrgenommen werden können. Diese Eigenschaften sind auch für den Straßenverkehr prägend: Die Intentionen anderer Verkehrsteilnehmer können nicht direkt wahrgenommen und daher ihre Verhalten nicht genau prädiziert werden. Aus dieser Unsicherheit resultiert eine enorme Vielfalt an möglichen Situationsverläufen. Gleichzeitig erfassen Sensoren immer nur Bruchteile der Umgebung, da zum Beispiel Häuser die Sicht verdecken. Selbst wenn ein anderes Fahrzeug direkt wahrnehmbar ist, unterliegt jede Messung Fehlern. Um dennoch sichere Entscheidungen treffen zu können, muss diese Unkenntnis des genauen Zustandes der Welt explizit behandelt werden. Dazu gehört nicht nur die Vorhersage von Situationsverläufen, sondern auch das Antizipieren, ob und wie genau zukünftige Situationen wahrgenommen werden können. Die Komplexität der Entscheidungsfindung erhöht sich dadurch noch zusätzlich. Hinzu kommt, dass die reale Welt kontinuierlich ist. Dadurch kann sich jede Situation potenziell auf unendlich viele Arten weiterentwickeln. Der Raum der Wahrscheinlichkeitsverteilungen über dem kontinuierlichen Zustandsraum, welcher unsichere Situationen beschreibt, ist sogar unendlich-dimensional. Naives, erschöpfendes Planen wird damit unmöglich und es müssen sinnvolle Vereinfachungen des Problems gefunden werden.

Partially Observable Markov Decision Processes (POMDP) bieten ein allgemeines, mathematisch fundiertes Rahmenwerk, um sequentielle Entscheidungsprobleme mit Unsicherheiten in der Prädiktion sowie in der Wahrnehmung zu modellieren. Diskrete POMDP von mittlerer Größe können heute bereits approximativ gelöst werden. Autonomes Fahren ist jedoch, wie die meisten realen Probleme, ein Prozess im kontinuierlichen Raum. Für hoch-dimensionale oder gar kontinuierliche POMDP, wie das autonome Fahren, existieren bisher keine ausreichenden Lösungsverfahren. Wie für die meisten Probleme existiert auch keine allgemeingültige, ausreichend niedrigdimensionale Diskretisierung des Problems. Wie geeignet eine Diskretisierung für ein Entscheidungsproblem ist, hängt dabei von der genauen zugrundeliegenden Pro-

blemstellung ab und kann je nach Situation variieren. Menschen können in solchen Systemen nur agieren, indem sie beim Planen Heuristiken anwenden, vereinfachen und problemspezifisch abstrahieren.

Wissenschaftlicher Beitrag Zwei Beiträge dieser Arbeit können hervorgehoben werden. Zum einen wird in der vorliegenden Arbeit eine neuartige, allgemeine Methode zum effizienten approximativen Lösen von kontinuierlichen POMDP mittels *Value Iteration* entwickelt und umgesetzt. Die zentrale Grundidee des Verfahrens ist, dass nur Zusammenhänge repräsentiert werden müssen, welche sich während der Planung als relevant für die Lösung des Problems erweisen. Ähnlich wie der Mensch kann die Planung so von unwichtigen Details abstrahieren und die Berechnung vereinfachen. Um diese Idee zu realisieren, wird eine problemspezifische diskrete Repräsentation des kontinuierlichen Raumes automatisch, als Teil des Planungsprozesses, gelernt. Hierfür wurde das Lernen einer effizienten, diskreten Repräsentation in ein kontinuierliches Bellman α-Funktions-Backup integriert.

Die beiden grundlegenden Fortschritte dieser Methode gegenüber bisherigen Verfahren sind, dass die Repräsentation nicht a priori festgelegt wird und dass als Kriterium für den nicht-parametrischen Lernprozess der *Value*, also die in der Zukunft zu erwartende Belohnung, verwendet wird. Die Repräsentation wird demnach mit dem Ziel gelernt, die Lösung des gegebenen Problems möglichst optimal darzustellen und nicht, wie in bisherigen Ansätzen, alle Situationen. In der experimentellen Evaluation konnten mit diesem problemspezifischen, nicht-parametrischen Lernen 12-dimensionale kontinuierliche Räume in realistischen Problemstellungen auf weniger als 1000 diskrete Zustände reduziert werden.

Der zweite Beitrag dieser Arbeit ist ein allgemeiner Ansatz für die Entscheidungsfindung von autonomen Fahrzeugen. In diesem Ansatz wird die Entscheidungsfindung für autonomes Fahren als kontinuierlicher POMDP modelliert und automatisch mithilfe der entwickelten Methode gelöst. Bisherige Arbeiten zur Entscheidungsfindung für kognitive Automobile vereinfachen die zugrundeliegende Problemstellung, indem sie beispielsweise Unsicherheiten ignorieren oder den eigentlich kontinuierlichen Verkehrsraum a priori diskretisieren. In dieser Arbeit wird gezeigt, dass gängige Vereinfachungen nicht für alle möglichen Situationen und Szenarien geeignet sind und sogar gefährliche Konsequenzen haben können. Alternative Ansätze beschränken sich auf Teilprobleme des autonomen Fahrens, wie zum Beispiel Autobahnfahren oder verleihen dem Fahrzeug durch manuell erstellte Regelsätze Autonomie. Dieser Herangehensweise sind durch die hohe Situationsvielfalt und Problemkomplexität insbesondere von innerstädtischem Fahren jedoch Grenzen gesetzt. Vorausschauendes Fahren erfordert begründetes Handeln auch in unbekannten Situationen. Aus diesem

Grund ist ein generischer Ansatz zu bevorzugen, der nur ein fixes Ziel und fixe Modelle benötigt, um für jede individuelle Situation selbsttätig eine Lösung abzuleiten.

In dieser Arbeit wird daher das Konzept verfolgt, das Problem zunächst als kontinuierlichen POMDP zu formulieren, ohne dabei die Räume und Modelle grundlegend zu vereinfachen oder zu beschränken. Eine Vereinfachung des Problems erfolgt anschließend automatisch und adaptiv für das spezielle Verkehrsszenario durch das Repräsentationslernen des entwickelten POMDP Lösungsverfahrens. Dadurch können komplexe Sachverhalte wie Unsicherheiten und partielle Beobachtbarkeit durch Sichtverdeckung, die Interaktion von Verkehrsteilnehmern untereinander und auch ihre Reaktionen auf Aktionen des autonomen Fahrzeugs dargestellt werden. Außerdem kann Hintergrundwissen, wie zum Beispiel Straßenkarten, integriert werden. Die Evaluation in mehreren Verkehrsszenarien zeigt, dass dieser Ansatz nicht auf bestimmte Fahrszenarien beschränkt ist.

Im Folgenden sind die wissenschaftlichen Beiträge der Arbeit zur Forschung in den Bereichen Künstliche Intelligenz, Maschinelles Lernen und Robotik aufgelistet und gegliedert:

1. Neuartige Methode zum effizienten Lösen von kontinuierlichen POMDP.
    a) *Point-Based Monte Carlo Value Iteration* (PBVI) ohne grundsätzliche Beschränkung der Räume und Modelle.
    b) *Bellman α-Funktions-Backup*, welches temporales Schließen mit dem Lernen einer geeigneten Raumrepräsentation verbindet.
    c) Automatische Abstraktion und Generalisierung der Planung im kontinuierlichen Raum.
    d) Effiziente Implementierung der Methodik, die in der Lage ist, hochdimensionale, realistische Probleme zu lösen und in Experimenten schneller bessere Entscheidungsstrategien erzeugt als existierende Verfahren.
2. Allgemeiner Ansatz für die taktische Entscheidungsfindung von autonomen Fahrzeugen.
    a) Repräsentation von Entscheidungsproblemen im Straßenverkehr als kontinuerlichen POMDP.
    b) Durchgehend probabilistische Formulierung.
    c) Anwendung der kontinuierlichen Lösungsmethodik für autonomes Fahren in Kreuzungs- und Einfädelsituationen mit nicht-trivialen Sichtverdeckungen.

Danksagung

Zunächst möchte ich Professor Rüdiger Dillmann für seine Betreuung sowie die jahrelange gute Zusammenarbeit danken. Er lehrte mich wichtige Grundlagen zur guten Mitarbeiterführung, zum selbständigen wissenschaftlichen Arbeiten und nicht zuletzt gewährte er mir die nötige Freieit zu forschen. Ich bedanke mich bei Professor Wolfram Burgard für sein großes Interesse und seine herausfordernden Gedanken und Fragen zu meiner Arbeit. Ich bedanke mich außerdem für die inspirierenden und lehrreichen Gespräche und Diskussionen mit den Professoren am KIT im Vorfeld meiner Arbeit. Genannt seien hier unter anderem Marius Zöllner, Hannes Hartenstein, Jürgen Beyerer und Ralf Reussner. Diese Arbeit wäre nicht möglich gewesen ohne das Engagement meiner Studenten und ohne meine Kollegen am *Humanoids and Intelligence Systems Lab*. Ein ganz besondereres Danke geht an meinen Kollegen und guten Freund Tobias Gindele. Ohne seinen außergewöhnlichen Optimismus, Motivation, Neugier und Fachkenntnisse wäre meine Arbeit sicherlich nicht die, die sie heute ist. Ich danke ihm für lehrreiche Zusammenarbeit und schöne gemeinsame Zeit. Hervorheben möchte ich zudem Rainer Jäkel, Pascal Meißner, Sven Schmidt-Rohr, Alexander Kasper, Stefan Ulbrich sowie Joachim Schröder. Ein großes Dankeschön auch an Christine Brand, Diana Kreidler und Isabelle Wappler für ihre Unterstützung in allen bürokratischen und vor allem menschlichen Fragen. Ich danke Professor Tamim Asfour und Peter Steinhaus für ihre selbstlose Unterstützung auch in schwierigen Zeiten.

Ich danke meinen Studienkollegen und Freunden Florian Faion, Sebastian Wirkert, Andreas Geiger und Christian Hirsch. Unsere allwöchentliche “Mittwochsrunde” war immer eine Quelle von Ablenkung, Entspannung, aber auch Inspiration.

Ich danke meinen Eltern Ingrid und Willi, die immer ihr Möglichstes getan haben, um mich bei der Verwirklichung meiner Träume und Entwicklung meiner Fähigkeiten zu fördern. Danke auch an meinen Bruder Tobias, den ich immer und in allen Lebensfragen an meiner Seite weiß. Ich danke meiner Tochter Nova Auguste, die mich in der kurzen Zeit, in der sie da ist, schon so viel über das Leben und die Lebensfreude gelehrt hat.

Ich widme diese Arbeit meiner Lebenspartnerin Sarah Helen Kächele. Du hast die besondere Fähigkeit, mir neue Blickwinkel auf die Welt zu eröffnen, die vieles in einem anderen Licht erscheinen lassen. Damit ergänzt Du mich. Danke für unsere wunderbare gemeinsame Zeit.

Contents

Chapter 1

Introduction

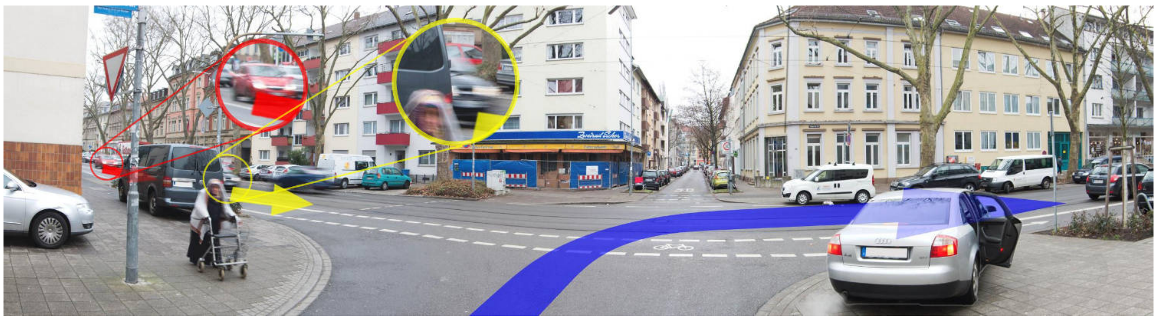
Driving a vehicle in traffic requires the cognitive ability to constantly make the right decisions even in complex situations. However, this task is made difficult because the drivers' knowledge about the environment and its development are uncertain: the behavior of other road users cannot be predicted with certainty and their intentions are hidden to the decision maker. Even the physical state of the world can only be perceived inaccurately and the majority of the environment remains completely hidden. The complexity induced by these uncertainties is even aggravated by the fact that the traffic state is continuous and high-dimensional. In conclusion, an infinite number of possible developments have to be taken into account and even representing a single situation can require infinite resources.

Motivation

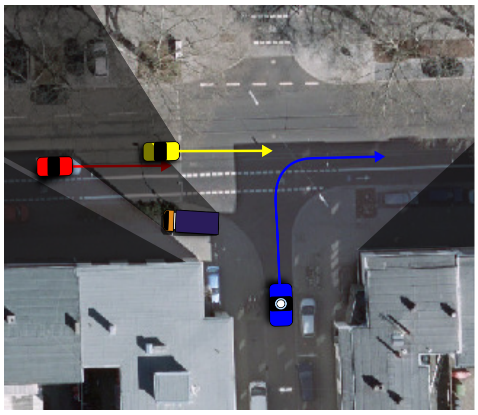
The scenario in Figure 1.1 illustrates the inherent challenges of decision-making. The first image shows an intersection scene from the driver's perspective. His goal is to turn right and merge onto the road. However, the line of sight to the incoming cars is blocked by a house and a parked truck. He can only gather some information by peaking through the gap between them.

In order to make safe decisions in this situation, the positions, orientations, and velocities of the incoming cars have to be estimated from noisy measurements and their potential motion has to be predicted. Eventually, the future effects of driving actions have to be assessed, even if the cars are hidden for some time.

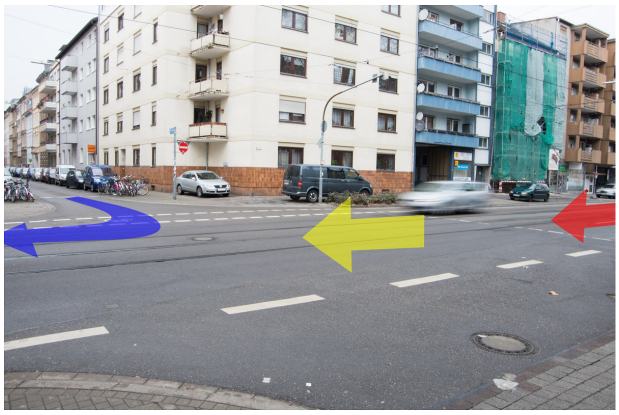
In fact, there is even more to it in this scene. In the first place, it is essential to stop early to get a good view on the traffic. Making such a decision requires awareness of one's own ignorance. Ultimately, a decision maker must anticipate what he can see, what he cannot see and what the consequences of knowing and not knowing might be. In this thesis, a method is presented that is capable of autonomous decision-making with partial information. The scenario in Figure 1.1 will be revisited and we show that the new method is able to automatically generate a safe driving policy.



(a) Driver's perspective.



(b) Birdseye view with lines of sight. Map Data [City of Karlsruhe].



(c) Perspective from other side of the road.

Figure 1.1: Urban traffic scenario from different points of view. The blue line indicates the driving goal, the red and yellow lines the incoming traffic.

Dynamic Decision-making Under Uncertainty

Decision-making is the cognitive process of choosing between several alternatives in order to achieve goals. The real world is usually dynamic so that future developments have to be considered. The development of the world in turn depends on future decisions. Thus, the result of decision-making must not only be a single, static decision. It is a policy that determines the present and all future decisions even for unexpected developments. Decision-making in real life settings is usually aggravated by uncertainties arising through the stochastic dynamics of the environment and incomplete information about the state of the world. Probability theory provides a mathematically sound basis to model these uncertainties. However, often the state of the world must be described with (infinitely many) continuous values because there is no generally sufficient symbolic representation. In consequence, exact planning is infeasible. Human planners presumably apply heuristics and other forms of simplification to find solutions that satisfy the given goal criteria [Botvinick and Toussaint, 2012]. Inspired by the human ability to generalize thoughts and decisions, the main goal of this thesis is to combine the ideas of learning and planning in order to automatically find a simplified representation that is suited for the specific decision problem. The derived general method can be utilized for various applications including decision-making in traffic.

Dynamic Decision-making for Autonomous Driving

The road-traffic domain in general and especially the task of autonomous driving is a very good example for decision-making under uncertainty. The idea of self-driving cars has been around since the invention of motor vehicles in the late 19th century. Progress in research suggests that what has been an utopian vision for almost a century is on the verge of becoming reality. Currently, universities, public research institutes, car manufacturers and even the search engine giant *Google* are putting considerable efforts into researching the technology (see Figure 1.2). For safe driving, decision-making is one of the key capabilities. Despite traffic regulations, road markings, etc., traffic still is a real-world and mainly uncontrolled environment. The arising uncertainties and the sheer variety of possible situations render decision-making in traffic very difficult.

Driving is a highly dynamic process. For good driving decisions, possible outcomes have to be anticipated and assessed. In order to avoid drawing the wrong conclusions, it is very important to consider that behaviors of road users are complex and highly coupled. Drivers make space, give way or simply brake (e.g., if a another car is blocking their lane). In this process, two main sources of uncertainty have to be considered: the first source stems from the behavior of other road users. Their reactions are to some extent rational and predictable, but their precise behavior remains stochastic and cannot be predicted with absolute certainty. The reason for the second source of uncertainty is that most parts of the environment are unknown. The intentions of other road users are always hidden to sensors and can only be estimated over time from measurable aspects, such as their poses or velocities. However, even physical properties usually cannot be perceived, for example because objects block the view of the sensors. When they are visible to the sensors, the observation of road users still underlies significant noise. Because of these uncertainties, the long-term consequences of driving decisions are not definite and driving can never be absolutely safe. Unfortunately, quite the contrary is the case. Even the best decision can only minimize the probability for accidents while still reaching the driving destination.

For rational decision-making in the domain of driving, not only potential developments of the world have to be predicted. The development of the decision maker's state of information and the process of his information gathering has to be anticipated as well. Because of the arising combinatorial complexity, manually modeling of decisions, for example, with rule-based systems, is difficult and error-prone. The number of situations that have to be distinguished is exceedingly high. Often, situations only

(a) *CoCar* by [FZI]. (b) *INTELLIGENT DRIVE* by [Mercedes-Benz]. [1.1] (c) Self-driving car by [Google]. [1.2]

Figure 1.2: Prototypes for autonomous driving.

differ in nuances and still require different reactions. Also, it is very difficult for human experts to assess probabilities objectively and draw the right conclusions for decision-making [Hogarth, 1975]. This is even intensified for sequential decision processes. In this case, human experts would have to deal with chains of conditional probability distributions.

Continuous Partially Observable Markov Decision Process (POMDP)

Based on probability theory, POMDPs provide a principled and powerful framework for modeling sequential decision problems under uncertainties [Sondik, 1971; Kaelbling et al., 1998]. In POMDPs, the decision maker estimates the state of the world by making observations. Formulating the decision problem as a POMDP, enables to judge and compare policies objectively. Advances in research made it possible to solve discrete POMDPs of medium complexity in reasonable time. However, driving, like most other real-world applications, has continuous and thus uncountably infinite state and observation spaces. As a consequence, the belief space that represents the knowledge about the state is not only high- but infinite-dimensional and known approaches for discrete state POMDPs are not directly applicable. A common approach is transforming the continuous POMDP to a discrete POMDP. Finding a suitable discretization, however, is very difficult, as the quality of a representation varies heavily depending on the specific task. Naive approaches to discretization, such as dividing the space into equidistant regions fail for more complex and higher dimensional decision tasks due to the *curse of dimensionality*. The resulting discrete space representation is, in some places, too coarse to model the necessary detail. In other places it is too fine for planning with limited computational capacities. Current research on continuous POMDPs intends to eliminate the restriction to discrete spaces. The prospect of being able to efficiently plan with complex continuous and uncertain decision processes is very promising. It would open a much wider field of application, including

[1.1]Copyright Daimler AG. All Rights Reserved.
[1.2]Google and the Google logo are registered trademarks of Google Inc., used with permission.

robotics, health care and operations research. However, due to the exceedingly high complexity, solving continuous POMDPs certainly is a challenge. In addition, one of the main operations, when solving POMDPs, is computing continuous integrals which do not have a closed-form solution, in general.

1.1 Thesis Statement

> For decision-making in real-life environments, not only given information, but also the lack of such must be considered. Approximate planning in dynamic, continuous and partially observable environments can be made feasible, if inductive learning is incorporated to realize the cognitive abilities of generalizing thoughts and putting selective attention on relevant information.

To support this statement, we propose a value iteration method for continuous POMDPs that implements the stated cognitive abilities. We evaluate this method for synthetic tasks as well as realistic traffic scenarios, where safe decisions can only be derived, if uncertainties are considered.

1.2 Problem Statement

This thesis is concerned with the general task of making rational decisions under the constraint that only uncertain and incomplete knowledge about the situation can be obtained. It focuses on planning for dynamic decision processes where an *agent* acts in a continuous environment in order to maximize a real-valued, predefined optimization criterion over time. Special emphasis is put on the task of tactical decision-making for autonomous driving.

1.2.1 General Decision-making

The following information is given to the decision maker:

> **Spaces:** It is assumed that the known spaces are sufficient to solve the decision process. This includes the *state space* representing situations and the *observation space* representing the measurements the agent can make with his sensors. Both spaces can be either of discrete, continuous, or hybrid nature. In this work, the term *situation* denotes the agent's knowledge about the state of the world. This knowledge can be given in form of a probability distribution over the state

space called *belief*.[1.3] The *action space* is a discrete set of decision choices that the agent can select from.

Models: We assume that sufficient knowledge about the decision process is given in form of models that are put as conditional distributions. The agent knows the capabilities and limitations of his sensors through the *observation model*. He also has knowledge about the uncertain world dynamics and the influence he has on it by choosing an action encoded in the *transition model*.

Reward: The *reward function* implicitly defines the goal of decision-making by assigning rewards to states. As the decision process is probabilistic, the future development and consequently the future reward cannot be determined with certainty. The goal of the optimization is therefore to maximize the expected reward over the future states (called the *value*).

Initial belief: Knowledge about the current situation is given by the *initial belief*. This initial distribution is the starting point for planning.

The result of the decision-making system is a *policy* and a *value*. The policy tells the agent which action to choose in the initial and the possible following situations. The value is the reward that he can expect when executing the policy. Such a decision problem can be represented as continuous POMDP. Finding a policy then equals solving the POMDP.

1.2.2 Tactical Decision-making for Autonomous Driving

For autonomous driving, the decision process can be specified. In Figure 1.3 a typical hierarchical system architecture for an autonomous car is sketched. It resembles the basic structure of a *rational agent* [Russell and Norvig, 1995] that interacts with the world.

The main input to tactical decision-making is the current situation, a belief-distribution over object poses and velocities. Information for the situation is gathered by sensors (e.g., from odometry, and GPS for internal states and cameras or LIDAR-sensors for external states). The perception system extracts abstract observations, such as poses and velocities of objects, from these physical measurements. *State estimation and interpretation* extracts the situation: it solves the data association problem, fuses observations over time (e.g., multitarget tracking with Bayesian filtering) and sets them into relation with the traffic context (e.g., with lanes). Additionally to the situation, the agent has background knowledge about the environment, such as

[1.3] The belief is sufficient because we assume the Markov-property to hold. Alternatively, a situation can be given by the *history* of observations and actions.

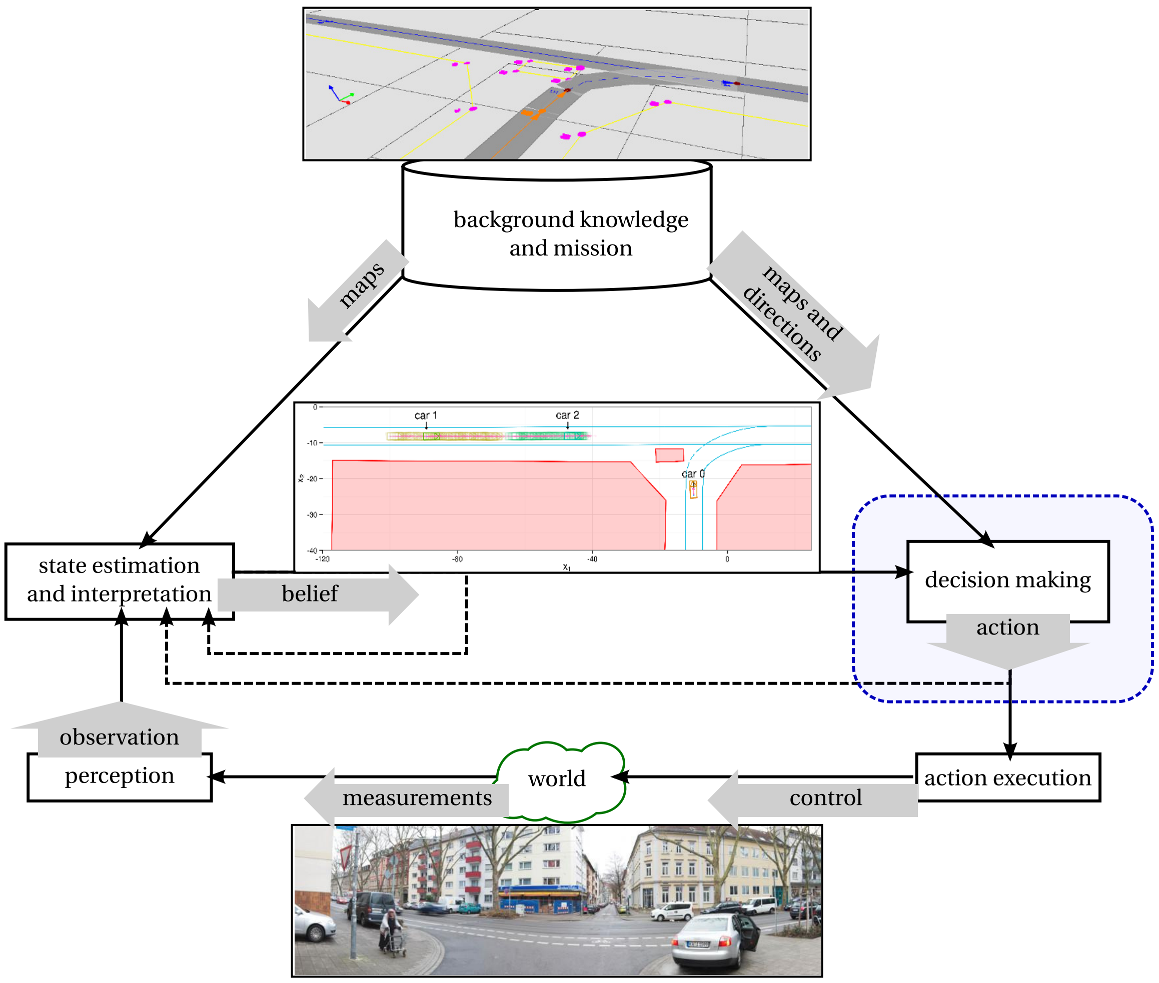


Figure 1.3: Embedding of decision-making into the system architecture of an autonomous car. Arrows illustrate the flow of information. Dashed lines indicate feedback loops over time.

road maps and maps with static environmental objects that can block the view of the sensors. The background knowledge also includes models for sensors and driver's behaviors. The *mission* determines the goal of decision-making, similar to a turn-by-turn navigation system giving directions.

The decision-making system selects high-level actions with a duration of 1 to 3 s, such as *accelerate* or *change lane*. These actions parametrize the *action execution*, a lower-level control system that eventually influences the development of the world by setting the steering angle and operating the accelerator pedal or the brake. This lower-level control system can also consist of several layers, such as *motion planning* and *trajectory-following control*. It operates with shorter reaction times (< 100 ms) to enable reactive maneuvers, such as emergency braking.

The task of tactical decision-making for autonomous driving is of special interest for this work for two reasons. On the one hand, decision-making for driving represents a big challenge because the traffic environment exhibits most properties that make sequential decision-making so difficult. It is continuous, partially observable, highly dynamic, and the dynamics are complex and indeterministic due to human interaction. On the other hand, traffic regulations and the underlying road geometry and topology induce some degree of structure to the problem. This restricts the number of possible situation developments so that there is a realistic chance of solving the problem. To our knowledge, no general approach to this problem has been proposed to date that satisfies the requirements of self-driving cars. Solutions that neglect these properties might create sufficient results for many, if not most driving situations. But they do not pose a generally safe solution. In situations where these properties are predominant they will fail. Hand-build solutions can consider these properties. However, the sheer complexity of the task makes it impossible to predefine a reaction in every situation. In unseen situations, this approach lacks the cognitive ability to generalize to new situations.

We define the following requirements for the resulting behavior policies, from which we draw conclusions for the concept of this thesis:

Driving Goal: The policy should pursue multiple driving goals and find an appropriate trade-off, if these are in conflict. It should lead the vehicle safely to its destination and thereby respect secondary aspects, such as efficiency and comfort.

Anticipation: Decision-making should be able to predict possible developments several timesteps ahead into the future, assess their impact and probability and, finally, draw objective conclusions about the current decision. Special focus needs to be put on the anticipation of mutual interactions of road users and also their reactions to actions of the self-driving vehicle.

Information Awareness: The policy should account for its own ignorance regarding the current and future states.

Information Gain: The policy should actively and intentionally acquire information, if this information is important for future decision-making.

Generality: The decision-making approach should be able to generalize to different situations without severe manual modification.

1.3 Concept and Contributions

Today, the usual approach for tactical decision-making for autonomous driving is modeling decision policies manually. In contrast to this, we propose an automatic optimization approach based on models for the traffic dynamics and the way the autonomous car can perceive it. Together with a reward function they define a decision process that can be solved automatically to obtain a policy. We argue, that defining or learning the task is easier and yields superior policies than manually defining the optimal solution for every situation. Another important difference to most existing works, is that we offer an end-to-end probabilistic formulation of driving. From the observations of the environment through the prediction of other drivers to the results of applied control actions, we utilize probability theory as unifying language to account for uncertainties on all levels of abstraction.

The concept of this thesis is to model decision-making problems as continuous POMDPs and then solve them automatically. To be able to consider complex transition and observation models with interaction and background knowledge, we model the POMDP in form of a dynamic Bayesian network (DBN). As a main contribution, a general method for approximately solving POMDPs with continuous state and observation spaces is presented. It realizes the idea to simultaneously solve a continuous POMDP and learn an efficient, discrete state space representation for every particular POMDP problem. A novel value iteration step is presented that integrates inductive learning of an efficient discrete representation into an Monte Carlo (MC) Bellman [Bellman, 1957a] backup. From a mathematical view-point, this allows computation of the continuous integrals but the integration of learning also enables two human abilities: to focus on relevant details and to generalize sparse planning results over the infinite space. The presented continuous POMDP solver yields superior performance compared to previous methods. It allows to solve higher-dimensional continuous POMDPs without inducing principle restrictions for the models of the POMDP (e.g., linear transition models). Due to this capability, it can be used for many other applications besides driving.

For the application of decision-making on a tactical abstraction level for autonomous driving, we present a general continuous POMDP, which meets all the requirements defined in Section 1.2.1. By solving this POMDP formulation for a specific situation, decisions can be derived automatically, without manual intervention.

This thesis contributes to research in robotics, machine learning and artificial intelligence as follows:

1. Novel method for solving continuous POMDPs efficiently.
    a) Point-based value iteration (PBVI) without restrictions of spaces or models.
    b) Bellman α-function backup that integrates representation learning and temporal inference.

   c) Automatic abstraction and generalization of planning results over the continuous space.
   d) Efficient implementation of the method that is able to solve high-dimensional POMDPs and shows state-of-the-art exceeding performance in experiments.

2. Approach to tactical decision-making for driving.
   a) General representation of driving as a decision process.
   b) End-to-end probabilistic formulation.
   c) Evaluation of autonomous intersection handling and merging with non-trivial occlusions.

1.4 Document Outline

This document is structured as follows. First, we give an overview of related work in the context of decision-making for autonomous driving in Chapter 2. In Chapter 3, we give a detailed introduction to decision-making under uncertainty with dynamic decision processes, namely MDPs and POMDPs. We discuss the complexity of solving discrete (PO)MDPs and present ideas and methods, in particular, approximate value iteration for POMDPs. The background provided in this chapter is helpful for understanding the method for continuous POMDPs that is developed in this work. While in Chapter 3 only discrete-space problems are considered, in Chapter 4, we generalize the discrete problem to continuous state and observation spaces and discuss the consequences for the difficulty of the problem. In this chapter, preliminaries and details are given that are the basis of the novel algorithm for solving continuous POMDPs that we develop in this work.

Chapter 5 describes the main contribution of this work. It presents the following contents: MC algorithms for continuous-belief prediction and continuous value iteration, integrating a discrete representation of the continuous space for efficient computation of the continuous integrals, and machine learning a problem-specific representation of the continuous value function.

In Chapter 6, the second contribution, a general approach for decision-making for autonomous driving is developed. Therefore, we express the driving task as continuous decision process. By solving the decision process, uncertainty-aware policies can be generated. We explain how the decision process can be solved automatically with discrete MDP value iteration or the novel method for continuous POMDPs.

In Chapter 7, we evaluate this line of action by solving automated highway driving using the discrete MDP and urban driving scenarios with the presented continuous

POMDP. Further, the continuous POMDP is analyzed and compared with state-of-the-art methods.

Notation We begin every chapter with a brief outline of its contents and how it integrates into the document. Further, we give a detailed outline of the structure of the chapter and conclude every chapter with a discussion of the insights. Text passages with a special meaning are highlighted:

> Text passages that make *design decisions* explicit are highlighted by a green bar on the left (like this text passage).

> *Conclusions* are highlighted with a blue bar on the left (like this text passage).

Throughout the document, we often use the name *ego vehicle* for the self-driving car. In illustrations of traffic scenarios, it can be differentiated from other vehicles by the circular shaped sensor on the rooftop (see for example Figure 1.1b).

Chapter 2

Related Work on Decision Making for Driving

This chapter gives an overview of approaches to decision-making and related tasks in the context of autonomous driving. The qualities and limitations of existing methods are discussed, assessed, and conclusions are drawn. A detailed introduction of general decision processes is given in the next chapter.

Tactical decision-making for autonomous driving creates a policy (or a plan) that selects high-level driving actions in every situation with the main goal to navigate the autonomous vehicle safely to its destination (see Section 1.2.2). Approaches to this task can be distinguished by the following criteria:

- **Policy generation:** Are decisions programmed by hand or are they generated automatically (e.g., through learning or planning)?
- **Utility:** Does the decision-making have a notion of *utility* or *value* that assesses how well a decision satisfies the goals of driving? Is it based on heuristics or derived from models?
- **Consideration of dynamics:** Does the decision-making consider the dynamic development of situations and can it predict them? Does it consider non-linear models, interaction, and cooperation between traffic participants and the self-driving vehicle? Are vehicles assumed to be independent from each other?
- **Consideration of uncertainty:** Are uncertainties in the dynamics and partial observability considered at all? To which extent are they simplified? Common simplifications are assuming a *worst case* scenario or conservatively estimating probabilities. Often, probabilities are restricted, e.g., to uni-modal distributions.
- **Information basis and representation:** Which aspects of situations are considered for decision-making and how are they represented internally? For example, discrete (often symbolic) and continuous representations can be differentiated.
- **Generality:** Is the approach limited to certain tasks or driving scenarios, such as highway driving, or can it generalize to new situations and scenarios?

Chapter Overview In Section 2.1, we start with a short history of research in autonomous driving in the light of tactical decision-making and discuss the current state-of-the-art. Then, in Section 2.2, a brief outline of other related problems is given. In Section 2.3, we conclude with a discussion and assessment of the presented methods and draw conclusions for the approach developed in this work.

2.1 Tactical Decision-making

Already back in the 90ies, impressive results were achieved with autonomously driving prototypes even in real traffic. Dickmanns et al. built a prototype vehicle *VaMoRs* that was capable of rudimentary autonomous highway driving with manual interventions [Dickmanns et al., 1994]. The purely reactive system *ALVINN* used Neuronal Nets to directly train a control using camera images as input [Pomerleau, 1991]. This system was originally designed for military purposes and basically realized road following in unstructured terrain [Pomerleau, 1994]. At that time, the research focus of these prototypes was on the architecture, control, vision, and practical integration rather than decision-making. However, the importance and difficulty of decision-making for driving was recognized early. Forbes et al. developed a general approach considering dynamic uncertainties as well as partial observability for the *BATmobile* project [Forbes et al., 1995] that led the way for many of the current decision-making systems.

In 2004 and 2005, the two *DARPA Grand Challenges* brought autonomous driving back into the focus of the public and research [Buehler et al., 2007]. Being funded by the Defense Advanced Research Projects Agency (DARPA), the focus was put on military applications and driving in off-road and unstructured terrain, such as deserts, where decision-making plays a minor role.

Decision-making became more important in the succeeding *DARPA Urban Challenge* in 2007 [Buehler et al., 2009], which moved the focus from unstructured to urban environments. In contrast to the previous competitions, the vehicles had to navigate safely in an environment with other traffic and obey traffic regulations. They faced complex tasks such as merging into moving traffic. However, compared to real-life traffic and, in particular, inner city traffic, decision-making was significantly facilitated and uncertainties could be mostly neglected or simplified: no objects blocked the visibility, e.g., at intersections, and the other drivers, being instructed professional drivers, behaved relatively predictably. Further, road users other than cars, such as pedestrians or cyclists, were neglected and the types of situations were limited. In consequence, no general solution to the problem was required and hand-coded models sufficed. Although the results of the *Urban Challenge* indicated that urban autonomous driving is feasible, real-life traffic is much more complex and requires more sophisticated decision-making methodologies.

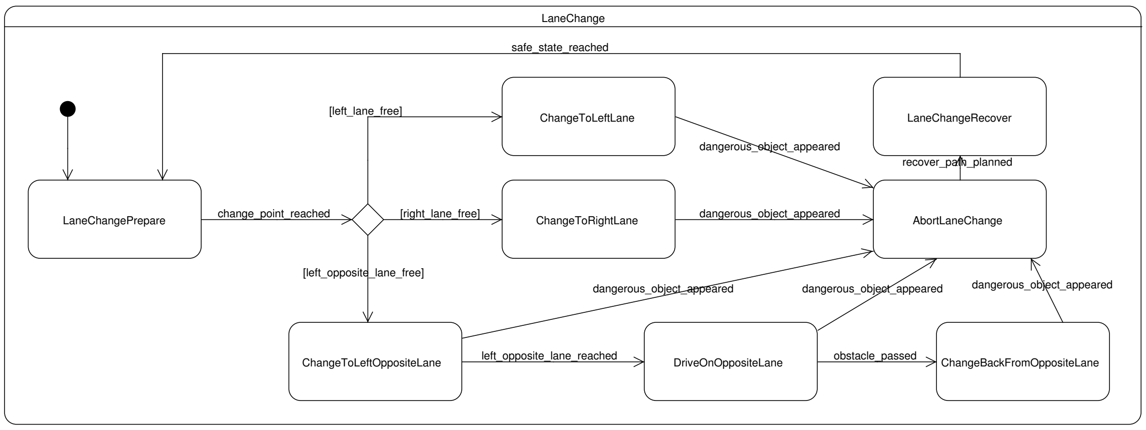


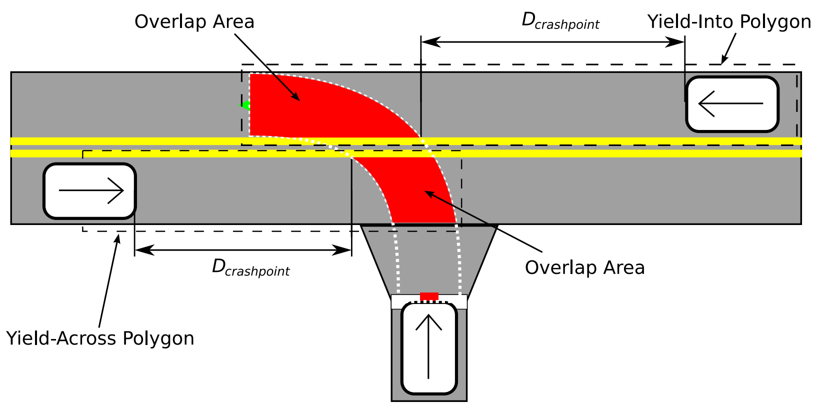
Figure 2.1: Sub-state of *AnnieWAY*'s state machine for lane changes [Gindele et al., 2008].

Current research in decision-making for cars mostly focuses on finding more general and robust solutions to sub-problems, such as merging into traffic or highway driving. Especially automated highway and freeway driving are popular topics, presumably due to their, compared to urban driving, manageable complexity. Thus, there is a high chance that this functionality is ready for series production in the next years. The long-term goal, however, is fully autonomous driving in all situations. Important projects today include the *Google Self-Driving Car*, the *PROUD* project by *VisLab* around Alberto Broggi that conducted experiments in Italy, and *INTELLIGENT DRIVE* by Mercedes Benz that autonomously drove the historic Bertha Benz route [Ziegler et al., 2014b].

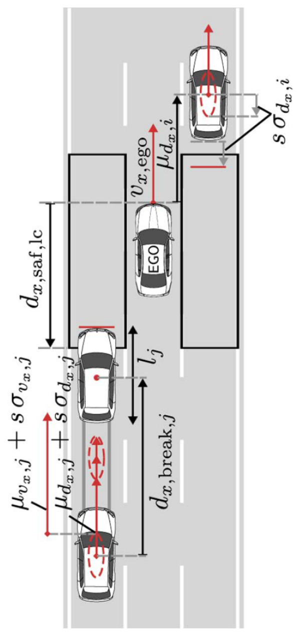
2.1.1 Manual Decision Programming

The presumably most common approach to decision-making, today, is to model reactions to situations by hand. In the *Urban Challenge*, finite-state machines emerged as the preferred representation for interpreting situations and making decisions in the same framework [Kammel et al., 2008; Bacha et al., 2008; Miller et al., 2008; Bohren et al., 2008; Montemerlo et al., 2008; Urmson et al., 2008]. Being well understood and tested, they posed a good solution for the limited number of scenarios in the Urban Challenge, especially in view of the limited development period. One example is the hierarchical state machine used by Team *AnnieWAY* [Gindele et al., 2008]. In their formulation, states represent driving behaviors and transitions are triggered by manually modeled conditions. In Figure 2.1, the sub-state that executes lane changes is shown.

Substantial differences of the state machines can be found in the specific implementations for traffic scenarios. For example, *BOSS*, the winner of the Urban Challenge, performs several checks based on manually defined metrics [Urmson et al., 2008]. For merging into a lane, first, a yield polygon is constructed to determine relevant cars (see Figure 2.2a). Then, the time of arrival is estimated as basis for detecting yield clearance.



(a) Yield polygon construction (*BOSS*) [Baker and Dolan, 2008]. ©2008 IEEE



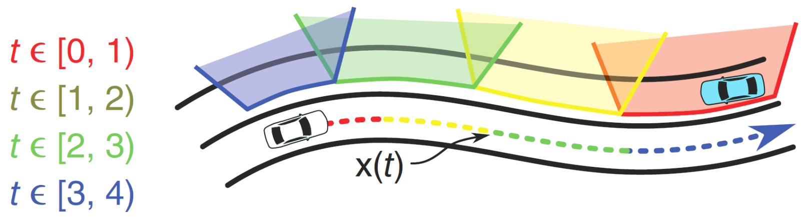
(b) Lane change feasibility [Ardelt et al., 2012]. ©2012 IEEE

Figure 2.2: Examples of manually modeled metrics for merging and highway-driving.

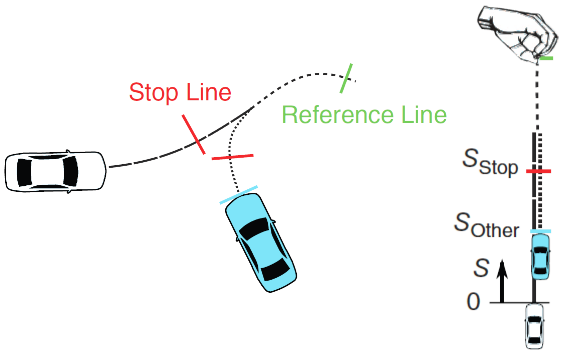
Uncertainties are treated in a rudimentary way by assuming *worst case* velocities for the arrival estimation. To handle the case that the sensors lose the other car momentarily, *BOSS* requires the yield window to be constantly open for 1 s. Another example is team *Junior* that uses a threshold test, based on velocity and proximity, to find *critical zones* where the vehicle has to wait [Montemerlo et al., 2008].

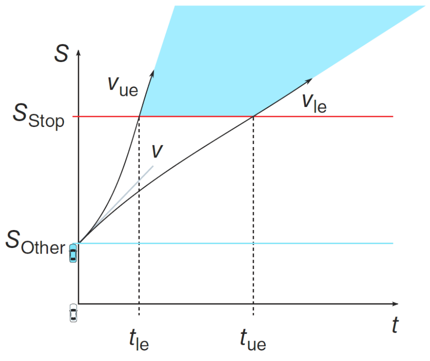
Apart from the *Urban Challenge*, manual programming is still frequently applied, for example, in [Ardelt et al., 2012], for generating automated lane change decisions for freeway driving. Therefore, they manually model criteria based on longitudinal predictions of the other vehicles. Figure 2.2b shows a scenario where the criterion states that no lane changes are feasible. Interaction between drivers is considered to a small extent, for example, by assuming that the incoming driver on the fast lane would react on a lane change to the left by braking. Similarly, uncertain dynamics are treated by defining worst case scenarios. Another example is presented in [Girault, 2004]. They propose a finite state machine to switch between different control laws for tasks such as entering a highway, merging, or yielding.

Manual programming is a very efficient way to create acceptable policies for simpler scenarios with limited complexity. Additionally, the methods are easily understandable and, for small problems, traceability of the software is very good. It can be doubted, however, that the fully manual approach scales to general autonomous driving in urban scenarios. With this approach, individual tasks such as following a leading



(a) Constraints for incoming vehicle in different time-spans from $t = 0$ to 4.





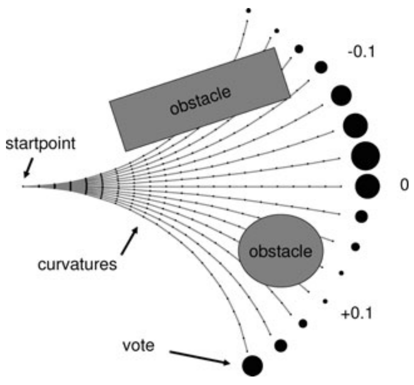
(b) Merge scenario and 1D projection. (c) Constraint for merge.

Figure 2.3: Spatiotemporal constraints for incoming traffic and for merging [Ziegler et al., 2014b]. ©2014 IEEE

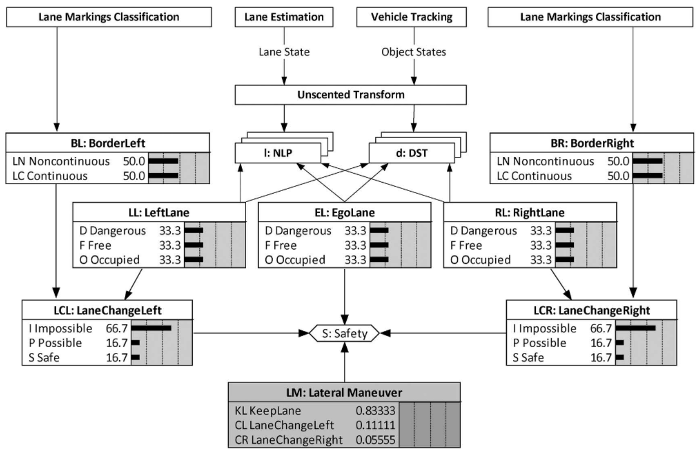
vehicle or merging into traffic, but also variations of these tasks need tailored solutions. Due to the number of possible situations and the difficulty to treat situations with several road users and complex uncertainties manually, the programmed models become too complex and the advantages (simplicity, traceability, etc.) get lost.

Structuring the program flow with manually implemented architectures Nevertheless, manual programming poses a sensible option to manage resources and structure the program flow. For instance, planning can then be used to eventually make decisions in the specific situation. Schröder et al. propose a biologically-motivated process to switch driving behaviors based on *motivations* (e.g., *road following*, *speed control* and *collision avoidance*). While the behavior network is build by hand, the final driving is conducted by the vehicle control. Therefore, several concurrent behaviors are fused using a *driving corridor* that is handed over to the vehicle control. In a later work, dynamic *risk maps* that discretize the continuous state into a grid are proposed as representation for fusing behaviors [Schröder, 2009].

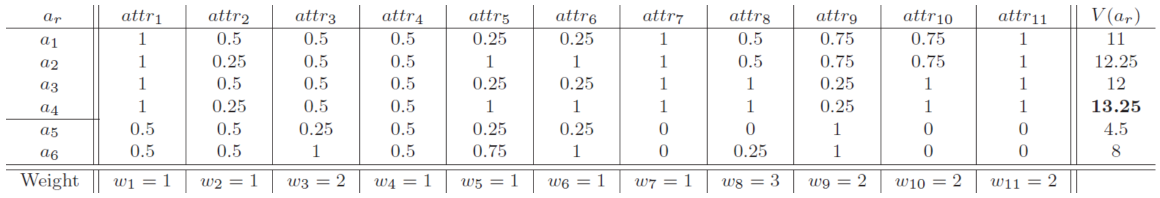
For mastering the Bertha Benz route, Ziegler et al. realized a similar idea using a hierarchical, concurrent state machine architecture to generate spatiotemporal (i.e., time and space) constraints [Ziegler et al., 2014b] (see Figure 2.3 for illustration). When, for example, conducting a yield and merge maneuver at the same time (similar to our initial example scenario in Figure 1.1), the concurrent states for yield and merge simultaneously generate constraints by assuming worst case behaviors for each. What we call *tactical decision-making* is skipped in their approach and these constraints are di-



(a) *DAMN* curvature voting for *Caroline* [Rauskolb et al., 2008].



(b) Decision network (DN) for lane change decisions [Schubert, 2012]. ©2012 IEEE



a_r	$attr_1$	$attr_2$	$attr_3$	$attr_4$	$attr_5$	$attr_6$	$attr_7$	$attr_8$	$attr_9$	$attr_{10}$	$attr_{11}$	$V(a_r)$
a_1	1	0.5	0.5	0.5	0.25	0.25	1	0.5	0.75	0.75	1	11
a_2	1	0.25	0.5	0.5	1	1	1	0.5	0.75	0.75	1	12.25
a_3	1	0.5	0.5	0.5	0.25	0.25	1	1	0.25	1	1	12
a_4	1	0.25	0.5	0.5	1	1	1	1	0.25	1	1	**13.25**
a_5	0.5	0.5	0.25	0.5	0.25	0.25	0	0	1	0	0	4.5
a_6	0.5	0.5	1	0.5	0.75	1	0	0.25	1	0	0	8
Weight	$w_1 = 1$	$w_2 = 1$	$w_3 = 2$	$w_4 = 1$	$w_5 = 1$	$w_6 = 1$	$w_7 = 1$	$w_8 = 3$	$w_9 = 2$	$w_{10} = 2$	$w_{11} = 2$	

(c) Table with heuristic utility function definitions for 6 actions and 11 attributes (i.e., objectives) [Furda and Vlacic, 2011]. ©2011 IEEE

Figure 2.4: Voting-based (2.4a) and utility-based decision approaches (2.4b and 2.4c).

rectly given to a subsequent trajectory planning module. However, uncertainties and, in particular, partial observability cannot be taken into account correctly by the proposed trajectory planning. In fact, to our knowledge, no current method exists that is capable of planning with partial observability and uncertain dynamics on the abstraction level of trajectory planning or even continuous vehicle control. Consequently, a scenario with constraint visibility, such as the one we presented in the introduction, cannot be solved by the approaches proposed in [Schröder, 2009] and [Ziegler et al., 2014b].

2.1.2 Utility- or Value-based Decision-making

In difference to the methods in the previous section, where decisions are encoded directly by hand, in utility-/value-based methods the decisions are determined indirectly by selecting the action that best satisfies a criterion. The precise notion of this criterion differs between approaches, but they all share the same basic idea: the criterion quantifies the expected success of executing an action in a situation with respect to a defined goal. The goal for driving typically is a combination of safety, target-reaching, efficiency, traffic rule compliance, and comfort. It is important to understand that for dynamic decision making, possible future consequences of actions have to be taken into account, in order to determine the criterion. For example in Markov decision processes (MDPs), the *value* is formally defined as the sum of rewards that is expected to be received in the future (see Section 3.1.1 for details). In contrast to the value, the *reward* (i.e., negative *cost*) is immediately received and only constitutes how well the a situation currently satisfies the goals (e.g., if the self-driving vehicle is at this very moment involved in a collision). Note that the terms *value* and *utility* usually are synonymously used, depending on the area of research.

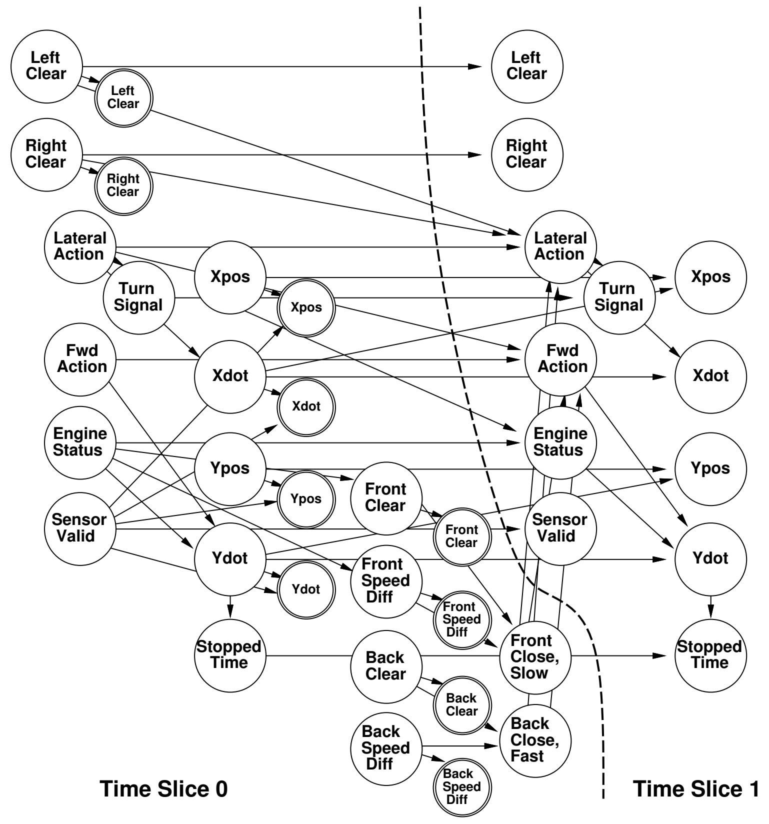
First, we present approaches that manually model the utility. Then, we present approaches that derive the utility automatically by learning or planning. We lay special focus on how partial observability can be taken into account.

Manual Utility Definition

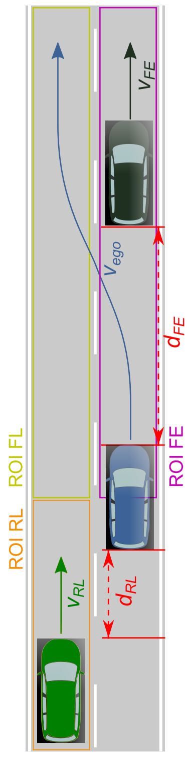
The distributed voting-based behavior architecture (DAMN) used by team *Caroline* was an exception to the otherwise dominating state machines in the Urban Challenge [Rauskolb et al., 2008]. The DAMN architecture [Rosenblatt, 1997] shares many similarities with utility-based decision-making. For a discrete number of steering angles and speeds, the resulting driving arc is computed and evaluated against *behaviors*, such as *avoid obstacles*, *follow way points*, or *stay on roadway* to collect *votes* (see Figure 2.4a). The votes are weighted and the steering angle and speed that generated the driving arc with the highest score is selected. For handling more complex situations, such as stop lines, the original DAMN architecture is extended with *interrupts*.

In [Furda and Vlacic, 2011; Furda, 2010], a multiple criteria-based approach is presented. They propose to, first, determine feasible maneuvers using Petri nets and then select the best maneuver based on multiple criteria decision-making (MCDM). In their examples, they name 11 MCDM attributes, e.g., *keep distance to front vehicle*, *keep distance to left boundary*, or *drive around obstacles*, to define the driving objectives. They create a table that determines the utility for all driving actions with respect to all these attributes (see Figure 2.4c). Then, a one-shot decision is made by the MCDM that balances the utilities. They argue that multiple potentially conflicting objectives need to be considered for driving. However, eventually, they restrain to a weighted sum for combining the multiple criteria.

Schubert presents a utility-based approach that is able to consider partial observability [Schubert, 2012]. He proposes decision networks (DNs), an extension to Bayesian networks (BNs), for building a probabilistic model of the decision problem (see Figure 2.4b). The utility is manually defined in a large table for every possible combination of situations and decisions. This approach is able to consider partial observability, but unlike partially observable Markov decision processes (POMDPs) it is



(a) Dynamic probabilistic network for situation prediction [Forbes et al., 1995].



(b) Features and regions of interest [Ulbrich and Maurer, 2013]. ©2013 IEEE

Figure 2.5: Approaches for lane change decisions considering uncertainty.

not able to consider the development of situations over time. To be able to model the prediction of traffic, the DN model would have to be extended to a dynamic decision network (DDN).

It can be doubted that the general approach of modeling the decision criterion by hand scales to more complex scenarios exhibiting uncertainties. With this approach, in total, |actions| × |situations| values have to be set. For every of these values, the probability and consequences of all relevant future developments must be assessed by a human expert. It is more promising to derive the utility or value yielded by actions in situations automatically from an immediate reward (i.e., cost, goal, etc.). Next, we present approaches to this based on planning and learning.

Model Predictive Control

A frequently applied method for one- and multi-lane highway driving that directly works in the continuous space is model predictive control (MPC). MPC sets the current control by anticipating future events using a model of the dynamics. MPC originally is a control method with continuous action space. However, it can be extended with discrete lane-change actions for decision-making. In [Nilsson and Sjoberg, 2013], driving on two-lane one way roads is realized using MPC with a mixed integer program formulation. For the prediction, other road users are assumed to hold their velocity and to keep lanes. A similar approach for, in theory, arbitrary numbers of lanes is proposed in [Bahram et al., 2014]. While still assuming constant velocities for other vehicles, they use non-linear MPC to incorporate a prediction of other driver's lane changes. These approaches are to some extent capable of managing *disturbances* in the system, but predictions are in general assumed to be accurate and the state to be observable. Therefore, with this approach, uncertainties cannot be treated correctly. Further, presumably to keep complexity manageable, only a single time step of prediction is considered, which prohibits planning of several successive actions.

Reinforcement Learning

More sophisticated driving behaviors can be achieved by learning policies from experience. An early approach by Forbes et al. is based on a dynamic probabilistic network, which essentially resembles a dynamic Bayesian network (DBN), for predicting highway traffic [Forbes et al., 1995]. The proposed network is shown in Figure 2.5a. It uses symbolic states to encode, for example, if the left or the front of the vehicle is clear or occupied. The symbolic states are assumed to be directly measured. Due to the rough discretization of the state space, only rudimentary predictions can be made with this model. Forbes et al. propose three decision-making approaches using this probabilistic model as basis: manual decision rules encoded in a decision-tree, lookahead planning by extending the network to a DDN, and reinforcement learning for the full POMDP. Although Forbes et al. found that avoiding manual programming and considering partial observability improves results, their proposed approaches were ahead of their time. Sufficient methods and computers for solving the full POMDP or utilizing a more detailed state space did not exist. However, many of the presented ideas are adopted by today's approaches.

Ngai et al. present model-free reinforcement learning of overtaking decisions for freeway driving [Ngai and Yung, 2007, 2011]. They learn Q-values for different driving goals, such as *target seeking* and *collision avoidance*. For the latter, interaction between road users is considered by double action Q-Learning (DAQL). For decision-selection, they combine the (DA)Q-values for the different goals using a weighted sum. Besides the inability to consider partial observability, this quite general approach has some practical flaws: e.g., for Q-learning the (originally continuous) input of the reinforcement needs to be discretized and the weights for the combining the goals are set according to a manually defined table. Further, many aspects that should in our opinion be automatically derived (e.g., what is a safe following distance), are incorporated into the reward functions. Unfortunately, this reduces the generality of the approach and requires more manual programming and parameter tuning.

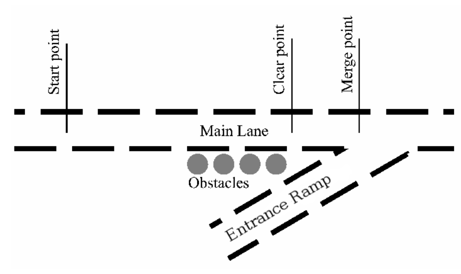
When learning from demonstrations, not even the immediate reward needs to be specified by an expert. Kuderer et al. propose learning the cost function for highway driving from demonstration [Kuderer et al., 2015]. They employ inverse reinforcement learning to capture individual driving-styles, based on features. These include accelerations, velocities, lane curvature, relative lane position, and the distance to a leading vehicle. With the learned models, trajectories can be generated for autonomous driving by optimizing the learned cost function. Therefore, other vehicles are predicted assuming that they hold velocities and lane. This method is not meant for making driving decisions. However, for example, a lane change can be executed with this method by generating trajectories with respect to the desired lane.

Planning with Partial Observability

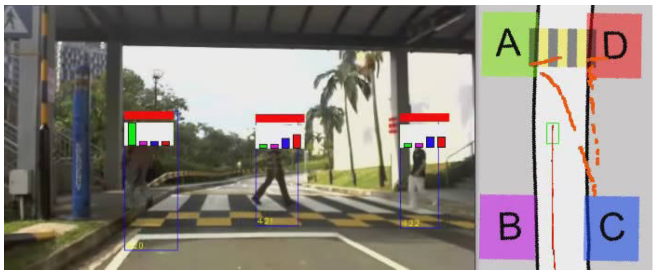
A few approaches have been proposed that consider partial observability for planning. However, due to the excessive complexity, usually the problem is significantly simplified.

Ulbrich et al. show how a POMDP with a discrete state space can be used to make lane change decisions [Ulbrich and Maurer, 2013]. They use a small set of only 8 manually defined states to describe highway situations. The 8 states are composed of 3 symbolic binary states: *lane change possible*, *lane change beneficial*, and *lane change in progress*. To observe these abstract states, Ulbrich et al. model a complex signal processing based on the velocities and distances of vehicles in the three regions of interest shown in Figure 2.5b, *rear left (RL)*, *front left (FL)* and *front ego (FE)*. The matrices defining the transition model and observation model were defined manually. While this approach keeps the complexity of solving the POMDP manageable, the tailored representation restricts the application to highway lane change decisions. Due to the very limited state space representation, the capabilities of the resulting policies are limited even for this application. More complex dynamics or information gain, e.g., necessary when trucks disturb visibility of other vehicles, cannot be considered in this POMDP.

Wei et al. [Wei et al., 2011] propose QMDP-based planning for single-lane driving that is able to take into account transition uncertainties and, to a limited extend, partial observability. In this approach, a continuous state space is considered that is composed of the distance to the lead vehicle and its velocity as well as the velocity, acceleration, and jerk of the autonomous vehicle. Monte Carlo (MC) sampling is applied for computing cost functions and for integrating over the continuous state space in order to compute the expected cost. They show that considering uncertainty in the behav-



(a) Ramp scenario [Wei et al., 2011]. ©2011 IEEE



(b) Pedestrian crossing [Bandyopadhyay et al., 2013a]. With permission of Springer.

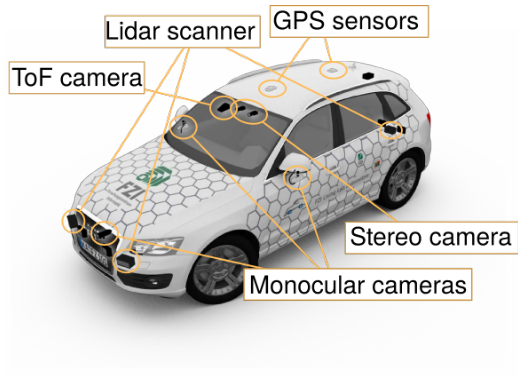
Figure 2.6: Examples of planning with partial observability.

ior of the leading vehicle as well as limitations of the perception improves robustness. The QMDP model assumes that the current state is only partially observable, but in the next time step, the state of the world will be be fully observable. QMDPs have a significantly reduced complexity compared to the full POMDP. However, they underestimate the limitations of the sensors. Further, the state space in this approach is restricted to single-lane driving and only dynamics in direction of driving are considered. Despite that, the authors realize good results in a ramp-scenario, where other cars on a priority lane merge into the lane of the ego vehicle while the visibility is obstructed by *obstacles* (see Figure 2.6a). However, to realize this, manual programming is required such as switching the reasoning algorithm depending on a predefined *clear point* and *merge point*, or determining worst cases for the other car's initial distribution.

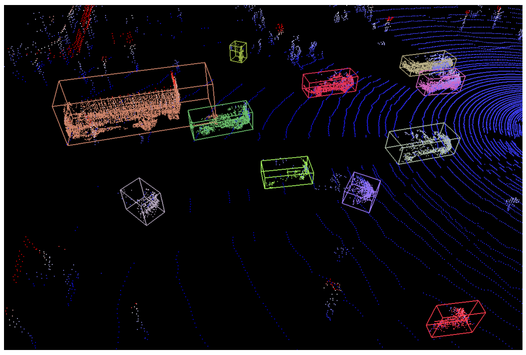
Bai et al. solve an intersection-like scenario with a single other vehicle that is observed with partial observability by considering a full POMDP [Bai et al., 2014]. Their model is not really usable for driving. Instead, it should be viewed as a showcase for the POMDP solver developed by Bai et al.. Velocities and inertia were not considered in their example and, thus, the state space of the problem is relatively low-dimensional compared to real-life driving.

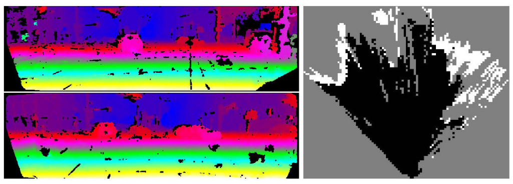
Bandyopadhyay et al. suggest that for navigation tasks the intentions of other road users should be considered [Bandyopadhyay et al., 2013a]. They present a mixed observable Markov decision process (MOMDP) approach where other road users' intentions are hidden variables. It is tested for a pedestrian crossing as shown in Figure 2.6b. Additionally, due to the computational complexity, multiple pedestrians are treated as independent entities. To reduce the complexity of the problem, the decision maker only reacts to a single pedestrian, which is selected using a manually defined safety metric. In [Bandyopadhyay et al., 2013b], the same approach is tested in an intersection example with a single other car whose driver's intentions are assumed to be hidden. To be able to apply their model, they first discretize the continuous vehicle positions and velocities in 2.5 m and 1 m/s steps. To compensate for the coarse dis-

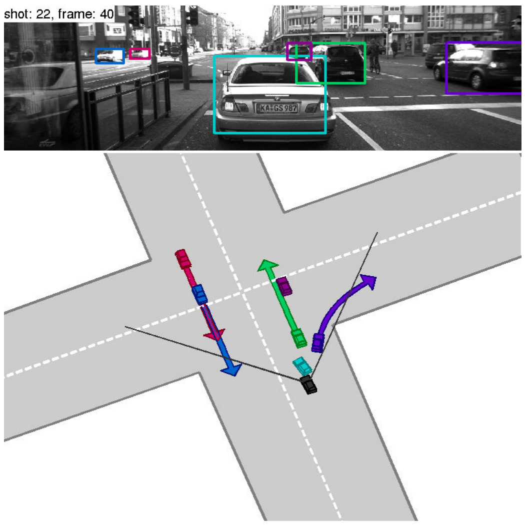
(a) Self-driving car with LIDAR [Google].[2.1]



(b) *CoCar's* sensors [FZI].


(c) LIDAR-based detection and tracking [Moosmann, 2013].


(d) Occupancy map from stereo-cameras [Geiger, 2013].



(e) Camera-based detection and prediction [Geiger, 2013].

Figure 2.7: Sensors and perception.

cretization, they are forced to assume higher transition uncertainty (see the conclusion for more information on this problem). This aside, their approach has two crucial issues. First, the physical state of other road users is assumed to be fully observable during planning. Thus, it performs suboptimal under measurement noise and can completely fail, when the view is blocked. The second issue is that it only considers one road user at a time. However, a safe policy for all examples can only be computed, when multiple road users are considered (see our discussion of the related work in Section 2.3 for an example).

[2.1] Google and the Google logo are registered trademarks of Google Inc., used with permission.

2.2 Related Research Topics

In this section, we briefly outline the components that are necessary to realize self-driving vehicles and also related domains. As can be seen in Figure 1.3 of our introduction, decision-making requires all of these components, either as input or to execute the selected actions.

2.2.1 Perception

Various sensors are used for this task ranging from radar, trough (stereo) cameras to laser scanners with 360° field of view (see Figure 2.7). Several aspects of the environment are of interest, including self-localization (e.g., [Grisetti et al., 2007]), other road users and objects (e.g., [Moosmann, 2013]), and infrastructure such as traffic signs (e.g., [Nienhüser, 2014]) or the geometry and topography of intersections (e.g., [Geiger, 2013]).

2.2.2 Object Tracking

The perceived information can be processed in multiple ways. The most common is to track all potentially relevant objects in the environment (i.e., estimate their state over time). Usually, state estimation is solved by iteratively integrating observations and predictions of the state in a probabilistic framework, for instance, a Bayes Filter [Thrun et al., 2005]. For tracking multiple targets, the information which object generated which measurement is crucial. This problem is commonly known as *data-association* (see [Bar-Shalom, 2000]).

Alternatively to representing every objects' state, a grid-based discretization of the space can be used that classifies every cell into *occupied* or *not-occupied* [Elfes, 1989; Thrun, 2002]. For *occupancy grids*, the object detection and data-association are facilitated. Simple measurement models can be found for every type of sensor, which enables robust and conservative estimation of the free-space for navigation tasks (see, e.g., Figure 2.7d). A sophisticated variant of this idea is a Bayesian occupancy filter (BOF) that has a probabilistic representation of occupancy and can be extended with velocities [Coue et al., 2006; Tay et al., 2008]. Usually, the grid-cells are measured and predicted independently. Due to this and the missing object notion of these systems, they are less suited for long term predictions than object-based systems.

We showed in a collaborative work that prediction accuracy is significantly improved, when the cells are not assumed to be fully independent and context information, such as map data is integrated [Gindele et al., 2009]. In [Brechtel et al., 2009], multiple interacting models are proposed so that the type of the object which causes

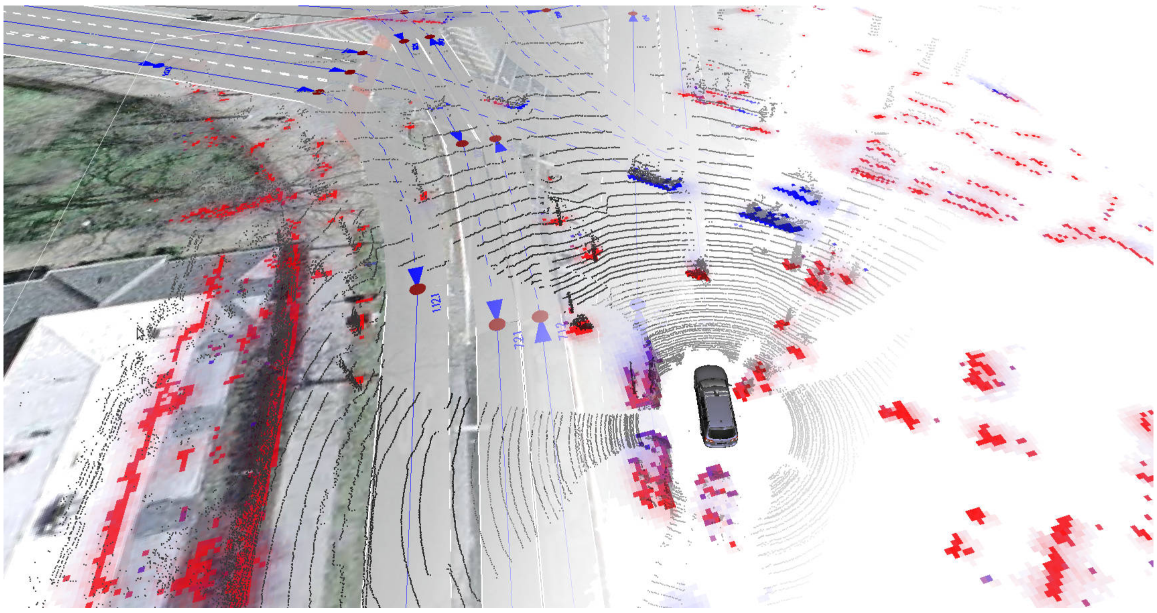


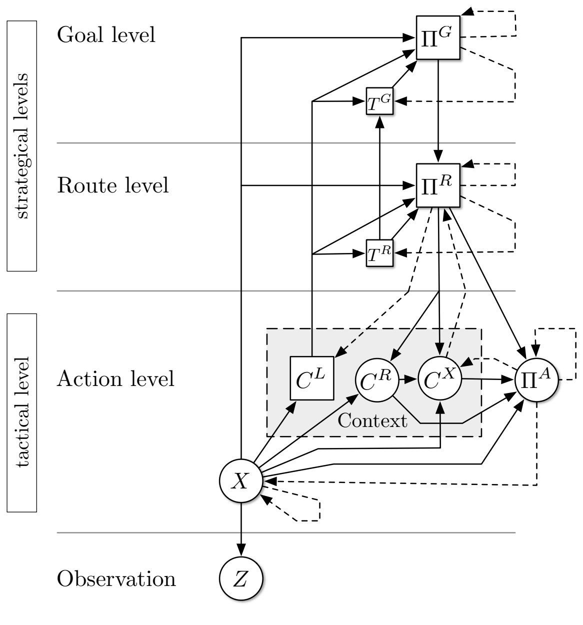
Figure 2.8: Bayesian occupancy grid with multiple models. Occupancy classified as vehicles are indicated in blue, other objects in red [Brechtel et al., 2009].

the occupancy can be estimated. This way, different motion models, e.g., for pedestrians and cars, can be applied, which further improves the quality of the prediction. Using approximate inference, the computational complexity is still practically feasible [Brechtel et al., 2010]. Interestingly, a representation similar to occupancy grids is often used for decision-making, even though with a much reduced grid resolution that covers only a few regions around the vehicle and is often aligned to lanes (e.g., [Forbes et al., 1995; Brechtel et al., 2011; Ulbrich and Maurer, 2013]). For many applications, however, information about the object entities is vital.

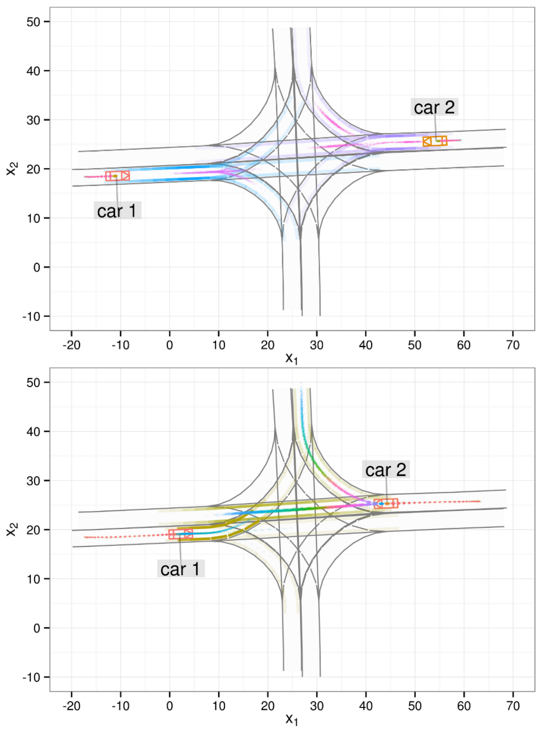
2.2.3 Situation Interpretation and Prediction

Sophisticated vehicle tracking methods utilize map data to improve prediction (e.g., [Petrich et al., 2013; Alin et al., 2013]). In most cases for object tracking it suffices to assume that road users are independent of each other. However, for decision-making and planning, not only the current state of the environment must be estimated. Accurate long-term predictions are necessary that can only be achieved when interpreting situations and taking into account road users' *contexts* and estimating their *intentions* over time.

Multiple related approaches have been proposed to find out drivers' intentions, e.g., for behavior estimation [Zhang and Rössler, 2009; Kasper et al., 2011], lane prediction [Lefèvre et al., 2011], and plan recognition [Bui, 2003]. Semantic, ontology-based representations of traffic situations can alleviate situation analysis [Kohlhaas et al., 2014; Ulbrich et al., 2014].



(a) Hierarchical DBN model.



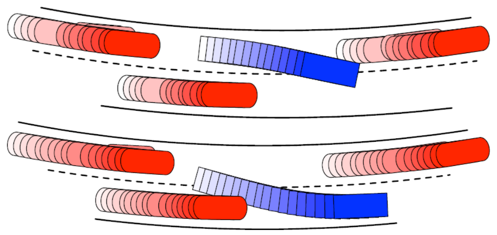
(b) Prediction in intersection scenario.

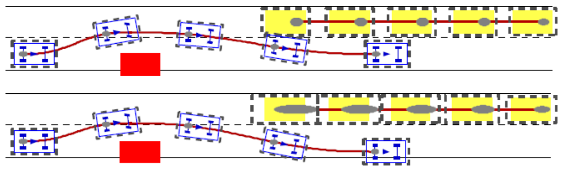
Figure 2.9: General method for interpreting and predicting traffic situations [Gindele, 2014].

Probabilistic framework for estimation and prediction In [Schamm, 2014], interpretation of situations based on object-oriented probabilistic relational models (OPRML) is presented that puts road users into relation. However, dynamics are not considered. In [Dagli et al., 2003], a dynamic probabilistic network is proposed for inferring and predicting motivations and goals of drivers in highway scenarios. Interaction between vehicles is considered by taking the time gap to the nearest neighbor of the estimated vehicle into account.

In a collaborative work with Gindele et al., we developed a hierarchical dynamic Bayesian network (DBN) representation of traffic situations and their development over time [Gindele et al., 2010]. The goal of the approach is to predict behaviors of road users as accurately as possible. The basic idea is to take the perspective of road users, in order to comprehend and estimate their goals, plans and actions (see Figure 2.9a). Therefore, information and models from the abstraction level of measurable physical states to abstract road users' goals are integrated, in a single probabilistic framework. Mutual relationships between road users are considered as well as relations over time. On the lowest level, physical driving models are employed.

In contrast to physical models, the behavior models that predict which control a driver applies given a certain context, are very difficult to model by hand. Therefore, in [Gindele et al., 2013], we present a method to learn them from noisy and incomplete observations of traffic scenes. In Figure 2.9b, an example of the method's capabilities


(a) Two episodes of planning in a spatiotemporal latice search graph [McNaughton et al., 2011]. ©2011 IEEE


(b) Considering uncertainty in the dynamics using linear-quadratic Gaussian (LQG). Results with low and high variance. [Xu et al., 2014]. ©2014 IEEE

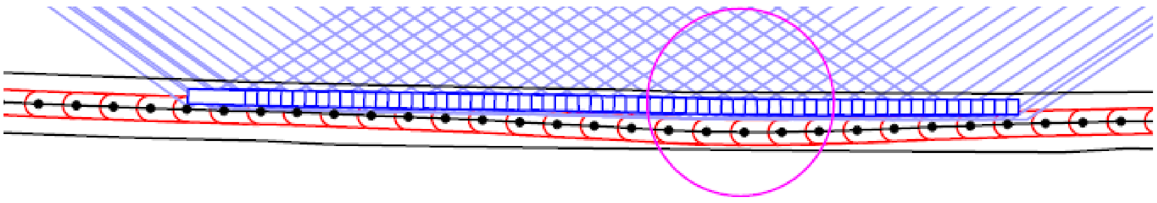

(c) Locally optimal motion planning. Car drives from the left to the right [Ziegler et al., 2014a]. ©2014 IEEE

Figure 2.10: Motion planning.

is shown. The first image shows two cars approaching an intersection and their predicted position in 6 s. The second image shows the trajectory prediction for several time steps in a later situation. The system concluded in this scenario that *car* 1 has the goal to make a left turn, as it slowed down slightly and there is a second car that has priority. For this reasons, other options such as, turning right or going straight were discarded over time. The illustration shows that such abstract conclusions manifest in (potentially multi-modal) predictions of the continuous trajectory of the vehicles.

This method is very general and not limited to certain scenarios. Further, it can be the basis for numerous autonomous driving tasks (see [Gindele et al., 2015]). It provides the foundation for estimation, interpretation and, most importantly, prediction for the decision-making approach presented in this thesis.

2.2.4 Motion Planning and Vehicle Control

The high-level actions selected by the decision-making, usually are realized by lower-level modules that generate the continuous control signals for steering and acceleration. These systems must work with an higher update rate and in the continuous state space, in order to react quickly and precisely on sudden unexpected developments, for instance, by avoiding obstacles or emergency braking. Typically, motion planning is used to find eligible trajectories under consideration of traffic and dynamics of the autonomous vehicle (e.g., trajectory planning [Werling et al., 2010; Ziegler et al., 2014a; McNaughton et al., 2011], or model-free reinforcement learning [Kuderer et al., 2015]).

In a few approaches uncertainties are considered (e.g., with LQG [Xu et al., 2014]). Examples are displayed in Figure 2.10. Stable continuous controls for the actuators are then generated by a trajectory control system (e.g., MPC [Werling et al., 2011]). An additional safety verification for the planned trajectory can be interposed between these two levels (e.g., [Althoff and Dolan, 2014]). Alternative realizations of the vehicle control waive trajectory planning and directly create control actions (e.g., model-free reinforcement learning of steering control for holding lanes [Riedmiller et al., 2007]).

2.2.5 Cooperative Driving

In the future, vehicles probably will be connected with each other and the infrastructure. For all of the tasks we presented above, cooperative solutions exists. The in the following presented decision-making related examples all assume a central unit that controls all involved vehicles. Cooperative decision-making based on fuzzy logic is proposed in [Raimondi and Melluso, 2008]. In [Schwarting and Pascheka, 2014], an approach to cooperative highway driving is presented that is based on behavior prediction. It anticipates conflicts between vehicles and a recursively resolves them. In [Frese and Beyerer, 2011], different motion planning algorithms (elastic bands, tree search, mixed-integer linear programming, and a priority-based approach) are compared for cooperative collision avoidance.

2.2.6 Robot Navigation

The task of navigating autonomous robots in environments shared with humans exhibits many similarities with autonomous vehicle operation and, thus, methods are to some degree interchangeable. Similar to driving, estimating and predicting the intentions and behaviors of humans is a central part of this task [Kuderer et al., 2012; Bandyopadhyay et al., 2013a; Kretzschmar et al., 2014]. In both tasks, partial observability plays a crucial role and, due to the continuous space, complexity poses a problem. In [Foka and Trahanias, 2007; Theocharous, 2002], therefore hierarchical POMDP models were developed to make computations feasible. Stochastic optimal control is proposed (e.g., in [Van Den Berg et al., 2012]) for robot motion planning with uncertainty and even continuous POMDP methods were developed with robot navigation in mind (e.g., [Roy et al., 2005]). More information on this topic is given in Section 4.2.

Despite their similarities, robot navigation also differs in many ways from navigating an autonomous car in traffic. On the one hand, the motion of vehicles is usually guided by lanes and traffic regulations apply. Further, the motion of vehicles is significantly restricted by their kinematics. On the other hand, dynamics are more important for driving, which again complicates the problem. Also, safety is a bigger issue, because errors and collisions are much more fatal due to the higher speeds and masses involved.

2.3 Discussion of Related Work

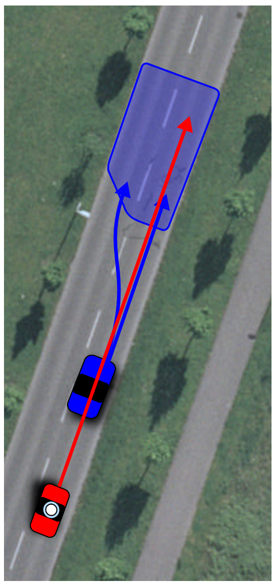
Tactical decision-making, today, emerges as one of the remaining challenges for autonomous driving. The possible reasons for this are manifold. The first reason is that decision-making is located at the highest-level of the processing chain. A decision-making system can only be utilized, tested, and demonstrated, if all other skills on lower abstraction levels are realized sufficiently. To make safe decisions, well-functioning sensors and perception of the environment, self-localization, precise maps, tracking and prediction of road users, and interpretation of situations are required. To execute the decisions, motion-planning, vehicle-control, and actuators are required.

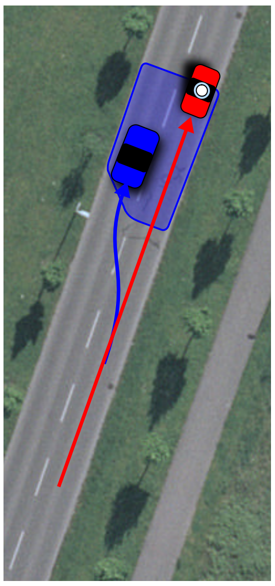
The second reason that we identified is that for long-term decision-making, ultimately, all information that the vehicle can obtain potentially has to be considered. Additionally, the number of possible situations and developments that a car can exhibit is very high. This is aggravated by the uncertainties in the process. In specific situations or scenarios, however, only a small subset of the information is relevant. Many scenarios, such as highway driving or special junction scenarios without uncertainties (as the DARPA Urban Challenge indicated), can be sufficiently covered in a sweeping way with manually programmed systems. For *general* decision-making in all situations, however, oversimplifying the task is potentially dangerous.

We discuss frequently applied simplifications and their consequences. Then, we draw conclusions for our approach and put it into relation to the state-of-the-art.

2.3.1 Common Simplifications to Reduce Complexity

The full tactical decision problem for driving with uncertainties can be modeled as a continuous partially observable Markov decision process (POMDP). However, due to the daunting complexity of continuous POMDPs (see Chapter 3), multiple simplifications of the decision problem for driving have been proposed in the literature. These apparently work in many situations. However, they fail in other scenarios or types of situations. This renders them unsuitable for the general autonomous driving task. We give examples of these simplifications that have counter-intuitive and non-trivial consequences. We further elaborate on the topic in Section 6.1.4, where we draw consequences for the models utilized in our approach.


(a) Initial situation with reachable set prediction.


(b) First development.

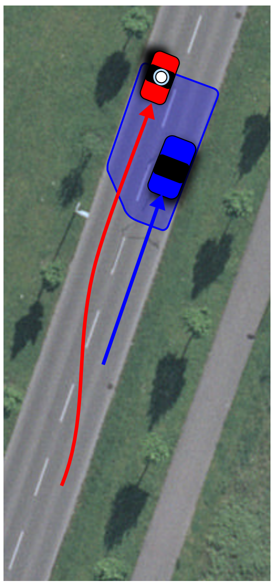

(c) Second development.

Figure 2.11: Common simplifications of uncertainty and potential consequences. Map Data [City of Karlsruhe].

Simplifications of Uncertainty

The most frequently neglected property is the uncertainty in both, the prediction and the perception. It is relatively obvious that underestimating or not considering uncertainty at all, e.g., by only taking into account the most likely assumption, is dangerous. Underestimating uncertainty means disregarding potential developments—against better knowledge. Interestingly, overestimating uncertainties is potentially dangerous, too.

Worst case assumption In nearly every manual approach and many planning approaches, *worst case* scenarios are defined. However, despite the fact that in complex situations no single unique worst case scenario can be identified, another problem arises: if the policy is optimized only for the *worst case* scenario, it can fail for *normal* scenarios.

Conservative estimate An often applied simplification to counteract this, is estimating probabilities conservatively. For example, Althoff et al. realize this by analyzing, if road segments are *reachable* for road users from an initial situation [Althoff and Dolan, 2014]. *Reachable sets* determine all possibly *occupied* road sections for every time step, similar to the Bayesian occupancy filter (BOF) presented before. These reachable sets are then used for verifying the safety of planned trajectories. Similarly, many motion planning systems use safety margins that are larger than the real vehicle structure to improve robustness against measurement noise. Additionally, gradually increasing safety margins when checking collisions farther away in the future can improve robustness against uncertain predictions (see, e.g., [Werling et al., 2011]).

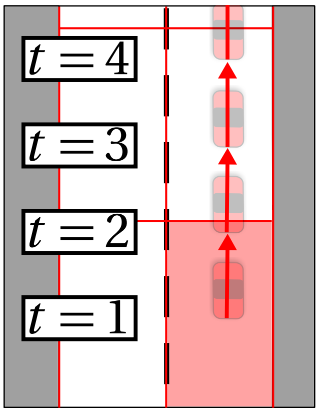
Formally, both simplifications are equivalent to assuming an unobservable Markov decision process (UMDP)[2.2] model and, in addition, considering all possible states. Therefore, every state with probability higher than zero is assumed to be true. Unobservable Markov decision processes (UMDPs) describe a *blind* agent that cannot gather any information in the future [Hauskrecht, 2000]. Thus, they drastically underestimate the perception capabilities of the agent. The big advantage of UMDPs is that they, compared to POMDPs, only marginally increase the complexity of the planning problem. The reason for this is that UMDPs basically consider a POMDP with a single observation that provides no information. Consequently, there is only a single possible future development of the world under a fixed behavior policy (see Figure 3.5 for further illustration). Further, in combination with the zero threshold for the probability, they create a lower bound (i.e., conservative) estimate on the value. If a trajectory (or a policy) is safe with respect to a conservative reachability estimate, it can be guaranteed that it is really safe.

But how can a conservative estimate result in dangerous decisions? The answer is simple: finding the best (or even any safe) policy can be prevented because the cost of safe decisions is overestimated. An example is shown in Figure 2.11a, where the red self-driving car approaches a slower blue car. The set that is reachable by the blue car after some seconds is indicated by the blue area. When planning with the reachable set, the only viable option is to brake hard. However, as both displayed developments show, this is not necessary. In all cases, the red car would have noticed what the driver in the blue car does and could react, e.g., with a lane change. Another example is merging onto a highway with an initially blocked view on the traffic. Under the assumption that all reachable states for the other vehicles are *true*, a planning algorithm would infer that merging is impossible. The self-driving vehicle could slow down and enter the ramp with dangerously low speed or even stop. Human drivers, however, know that they will be able to see the traffic after having reached the ramp.

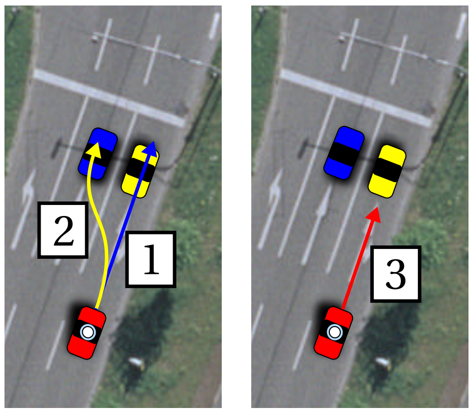
Simplification of the Representation of the Problem

Besides simplifying uncertainties, the originally continuous space is one of the main reasons for the complexity of decision-making (see Chapter 4) and, thus, often simplified. A common approach is defining symbolic states (e.g., [Forbes et al., 1995]) or naive discretizing of the state space (e.g., [Brechtel et al., 2011; Ngai and Yung, 2011]).

[2.2]A detailed introduction to decision processes is given in Chapter 3.



(a) Too coarse discretization of a two-lane road.



(b) Policies when considering only a single road user or both.
Map Data [City of Karlsruhe].

Figure 2.12: Frequently applied simplifications of the state space.

Often parts of the solution are implicitly integrated into the states. This limits the quality of the achievable policy (e.g., [Ulbrich and Maurer, 2013]). The level of detail of the discrete space is often even insufficient for making simple predictions, as we visualized in Figure 2.12a. The example shows the continuous motion of a car from time step $t = 1$ to 4. However, as its velocity is too short and the time step too long, using the discrete state space, the car is predicted to not leave its initial state (highlighted in red). Solutions to this problem are either increasing the resolution and with it the complexity of the problem or overestimating the uncertainty (e.g., [Bandyopadhyay et al., 2013a]), which decreases the quality of the solution and can also negatively affect the difficulty of the problem when using approximate methods (see Section 3.5.2 for details).

Another aspect is that the state space grows exponentially with the number of considered road users. An obvious simplification is to create a policy for all of them independently and then combine the policy (e.g., [Bandyopadhyay et al., 2013a]). However, the example in Figure 2.12b shows that a safe policy when considering all road users often is not even part of the policies created for a subset of road users. In the example, holding lane (policy $\boxed{1}$) is safe with with respect to the blue vehicle and changing lane (policy $\boxed{2}$) is safe with respect to the yellow vehicle. In reality, both policies will guide the vehicle into a collision. Stopping (policy $\boxed{3}$) is the best solution for this situation, but can only be found when considering both cars.

2.3.2 Conclusion

In Table 2.1, we assess the presented state-of-the-art algorithms and rate their *generality* (that is, if the approach is limited to certain scenarios or if it can generalize to new, before unseen situations). Some existing approaches for decision-making, such as

	Prediction[a]				Partial observability[a]		Space			
	linear	non-linear	interaction	probabilistic	simplified	arbitrary	utility/value	discrete	continuous	generality
Manual programming, e.g., [Urmson et al., 2008]	×	×	×	×	×	×	×	✓	×	(−−)
Utility-based, e.g., [Schubert, 2012]	×	×	×	×	(✓)	(✓)	✓	×	✓	(−)
Model-free Learning, e.g., [Ngai and Yung, 2011]	(✓)[b]	(✓)[b]	(✓)[b]	(✓)[b]	×	×	✓	✓	(✓)	(+)
Model predictive control, e.g., [Bahram et al., 2014]	✓[c]	(✓)[c]	✓[c]	×[c]	×	×	✓	(✓)	✓	(−)
Trajectory Planning, e.g., [Ziegler et al., 2014b]	✓	✓	✓	×	×	×	✓	✓	✓	(++)
QMDP [Wei et al., 2011]	✓	✓	✓	✓	✓	×	✓	×	(✓)[d]	(−)
Discrete MDP (our approach)	✓	✓	✓	✓	×	×	✓	✓	×	(++)
Continuous POMDP (our approach)	✓	✓	✓	✓	✓	✓	✓	✓	✓	(+++)

"(✓)" means that the feature is not supported in a native way but can be integrated.
[a] Manual uncertainty handling, e.g., worst case definition, is not accounted for.
[b] Implicit prediction.
[c] Prediction is limited to a single time step into the future.
[d] Tailored for merging and highly simplified.

Table 2.1: Comparison of exemplary approaches for tactical decision-making for driving.

[Ziegler et al., 2014b], are already highly developed. Nevertheless, none of the reviewed approaches is able to solve problems that exhibit non-trivial visibility and stochastic, non-linear interaction between road users. The reason for this is quite simple: today, no sufficient methods exist to solve higher-dimensional continuous decision processes with these uncertainties. All presented approaches apply critical simplifications to the original decision-making problem in order to make computation feasible. As a consequence, none of the presented approaches could find a safe policy in the intersection scenario presented in our motivation in Figure 1.1.

Not being able to solve only a single situation or even a class of situations is enough to render an approach inappropriate for autonomous driving. For fully autonomous driving, a general approach is necessary that is capable of solving any situation that the car might face.

In this thesis, we develop a novel methodological solution that can be applied to the full problem: a continuous POMDP. The developed method adaptively applies simplifications suitable for the specific problem, instead of imposing fundamental prior restrictions to the decision process. We build a general model for driving problems and apply the presented method for solving this model automatically. This approach generalizes to unseen situations by planning and learning and is less constrained than any previously proposed approach.

Chapter 3

Background on (Partially Observable) Markov Decision Processes

In this chapter, the basics of dynamic decision-making under uncertainties for discrete state, observation, and action spaces are covered. This chapter helps to understand the complexity of the task as well as the motivations, ideas, and methods behind the novel approach developed in Chapter 5.

A discrete time Markov decision process (MDP) models the sequential decision process of an agent acting in a dynamic environment with uncertain dynamics [Howard, 1960; Bellman, 1957b]. MDPs can be viewed as an extension of Markov chains to an optimization problem: adding actions enables the agent to influence the process. Rewards (or costs) define an optimization criterion. Figure 3.1 shows the interaction of an MDP agent with its environment. Drake introduced partially observable Markov decision processes (POMDPs), where in addition to the uncertain dynamics of an MDP the state of the world is only partially observed through a noisy channel [Drake, 1962]. Traditionally, decision processes are restricted to discrete spaces.

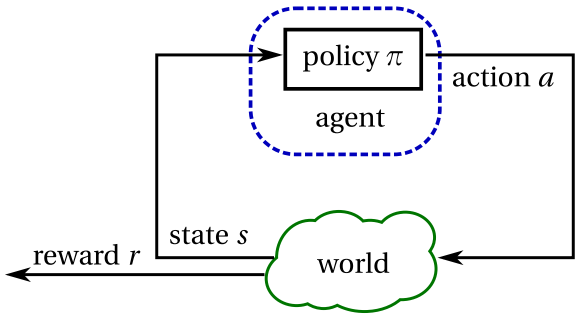


Figure 3.1: Interaction of an MDP agent with the world.

Chapter Overview In Section 3.1, we provide preliminaries and thereby establish the notations used in this work. The second part of this chapter (Section 3.2) is concerned with methods for solving decision processes. We lay the focus on discovering the origins of the complexity of solving POMDPs and on state of the art methods to counteract excessive complexity. The methods that are directly used in the later chapters

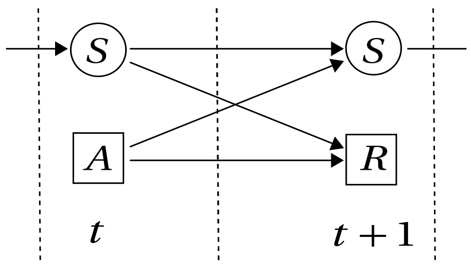


Figure 3.2: MDP as dynamic decision network (DDN) with random variable S for the state. The action A can be set by the decision maker and the reward R is a deterministic function.

are MDP value iteration in Section 3.3 and point-based α-vector value iteration in Section 3.5.1. In Section 3.6, interesting applications are explained to give an impression of the abilities and the versatility of POMDPs in practice.

3.1 Definitions and Preliminaries

We first introduce MDPs and POMDPs. Then, we explain how a POMDP can be cast as a belief state MDP.

3.1.1 Markov Decision Process (MDP)

A discounted MDP with infinite horizon is specified by the tuple

$$(\mathbb{S}, \mathbb{A}, T, r, s_0, \gamma)\,. \tag{3.1.1}$$

In the following, these elements will be explained. In case of discrete MDPs, $\mathbb{S}$ denotes the set of discrete states

$$s \in \mathbb{S}. \tag{3.1.2}$$

The world is assumed to always be in exactly one of these states and the agent knows this true state at any point in time. In robotics tasks, the state usually comprehends the agent's intrinsic state as well as his environment. $\mathbb{A}$ is the set of actions. The agent can partly control the process by deciding for one action

$$a \in \mathbb{A}. \tag{3.1.3}$$

MDPs are stochastic dynamic processes that fulfill the Markov property of order one. Hence, the next state $s' := s_{t+1}$ at time step $t+1$ only depends on the current state $s := s_t$ and the current action a. States are temporally related through the *transition model* T, which is a conditional probability mass function. In Figure 3.2, this relation is depicted as DDN, a dynamic Bayesian network (DBN) extended with rewards and actions. T describes the probability that the world evolves from state s to s', when the agent chooses action $a \in \mathbb{A}$ following

$$T(s', a, s) = p(s'|s, a) = p(s_{t+1}|s_t, a) \,. \tag{3.1.4}$$

The agent can only influence the evolution of the world and not fully control it. A real-valued reward function

$$r : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R} \,. \tag{3.1.5}$$

implicitly defines the goal of the decision process. It basically states how 'desirable' a state-action pair is for the agent

A policy $\pi : \mathbb{S} \rightarrow \mathbb{A}$ defines the behavior of the agent by mapping every state $s \in \mathbb{S}$ to an action. The aim of the MDP optimization is to find an optimal policy π^* that maximizes the *value* V_π for an initial state s_0 so that

$$\pi^* = \arg\max_\pi V_\pi(s_0) \,. \tag{3.1.6}$$

The value under a policy π is defined as expected future reward when executing the policy by

$$V_\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))\right] . \tag{3.1.7}$$

Due to the discount $\gamma \in [0, 1)$, the value is finite and the decision problem well-defined. Instead of introducing a discount factor, a limited time horizon could be used.

> In this work, only discounted decision processes are discussed. Assuming a limited horizon can introduce discontinuities in the value function over time for infinite horizon problems such as driving.

3.1.2 Partially Observable Markov Decision Process (POMDP)

A discounted infinite horizon POMDP is an extension of an MDP to partial observability. See Figure 3.3 for a representation of a POMDP as DDN (i.e., a DBN extended with rewards and actions). A POMDP is represented by the tuple

$$(\mathbb{S}, \mathbb{A}, \mathbb{O}, T, \Omega, r, b_0, \gamma\,). \tag{3.1.8}$$

In contrast to MDPs, the state in POMDPs is a latent (or hidden) variable that has to be estimated using inference. Like in hidden Markov models, the agent can only perceive

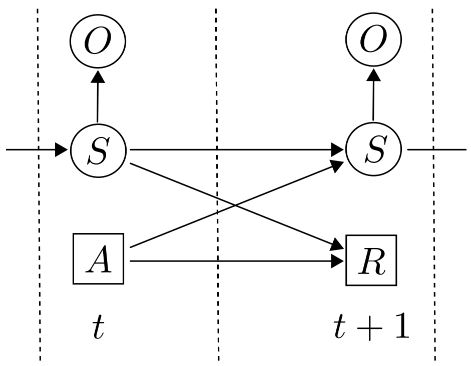


Figure 3.3: POMDP as DDN with random variable S for the state and O for the observation. The action A can be set by the decision maker and the reward R is a deterministic function.

the state partially through indirect observations of the world. In addition to the MDP's states s and actions a, the observations o are introduced with

$$o \in \mathbb{O} . \tag{3.1.9}$$

The agent can infer about the state s by making observations o following the *observation model* Ω. It defines the probability for making an observation o when the system reaches a state s by

$$\Omega(o, s') = p(o|s') := p(o|s_{t+1}) . \tag{3.1.10}$$

In contrast to MDPs, the agent does in general not know the state of the world exactly. Instead, he has to consider the whole *history* of actions and observations. Alternatively, he can obtain an estimate of the state by temporal fusion of the recent action and observation with his prior knowledge about the state. The embedding of this state estimation process is outlined in Figure 3.4. The estimate that the agent obtains in this process is given in form of a probability distribution over the states and denoted as *belief*. The belief space $\mathbb{B}$ contains all n-dimensional probability distributions over a state space S with n discrete states so that

$$b \in \mathbb{B} , \qquad b : \mathbb{S} \to \mathbb{R}_{\geq 0} , \qquad \text{and} \qquad \sum_{s \in \mathbb{S}} b(s) = 1 . \tag{3.1.11}$$

Figure 3.5 sketches the possible course of beliefs. Analogously to Bayesian filtering, the belief can be updated using the transition and observation model. The posterior belief

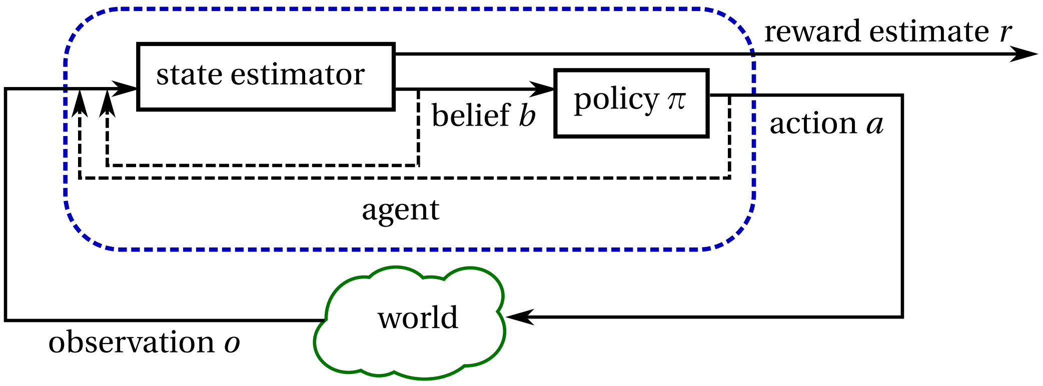


Figure 3.4: Interaction of a POMDP agent with the world. Arrows with dashed lines indicate feedback loops over time.

$b'_{b,a,o}$ can be computed from the current belief b. The *belief propagation*[3.1] depends on the action a that the agent chooses and the observation o he makes by

$$b'_{b,a,o}(s') = \frac{p(o|s')}{p(o|b,a)} \sum_{s \in \mathbb{S}} p(s'|s,a)b(s)\,. \tag{3.1.12}$$

Like in MDPs, the goal for the agent's *policy* π is to maximize the *value* V. However, because the agent never knows the state of the world, values and policies cannot be defined on the state as in MDPs. For POMDPs the past history of actions and observations has to be considered. However, as for MDPs, the Markov property applies so that the belief holds sufficient information about the history of actions and observations to plan optimally [Astrom, 1969; Bertsekas, 2007]. Consequently, it is sensible to define the values and policies on the belief space $\mathbb{B}$, i.e.,

$$V : \mathbb{B} \to \mathbb{R}, \qquad \text{and} \qquad \pi^* : \mathbb{B} \to \mathbb{A}\,. \tag{3.1.13}$$

The basic reward is, as in MDPs, defined on the state space and not on the belief space because only the true underlying state should be rewarded/penalized and not the 'believed' state. The expected immediate reward for a belief b when conducting action a is

$$E[r(s,a)] = \sum_{s \in \mathbb{S}} b(s) r(s,a). \tag{3.1.14}$$

The agent aims to find an optimal policy $\pi^* : \mathbb{B} \to \mathbb{A}$ that maximizes the value and act accordingly. The value is defined as the expected sum of rewards for future beliefs b_t, discounted by $\gamma \in [0,1)$, when starting in an *initial belief* b_0 and following the policy π

$$V_\pi(b_0) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(b_t))\right]. \tag{3.1.15}$$

[3.1] In this thesis, the term *belief propagation* denotes the transition from one belief to the next. This is not to be confused with the message passing algorithm for inference in graphical models.

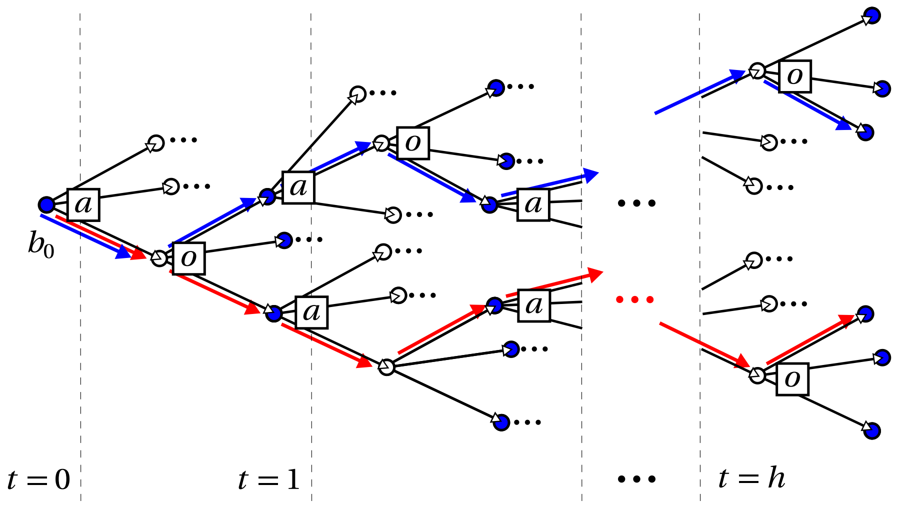


Figure 3.5: Possible belief developments in a POMDP starting in the initial belief b_0 with three actions a and three distinct observations o. Two POMDP histories (red and blue) of length h are highlighted.

For optimal decision making, when being in belief b, the potential follow up beliefs for the different actions and observations for all future time steps have to be assessed. See Figure 3.5 for an impression of the complexity of possible developments.

3.1.3 POMDP Formulated as a Belief State MDP

On the one hand, POMDPs are generalizations of MDPs. On the other hand, any POMDP can be cast as an MDP with a continuous state space that represents the beliefs of the POMDP. Solving this *belief state MDP* (or *information-state MDP*) solves the original POMDP. This means that the task of solving POMDPs is highly related to solving (continuous) MDPs and insights and methods for MDPs can be transferred to some degree.

We indicate the belief state MDP transition model and reward function by $(\cdot)_b$ to differentiate them from the original POMDP or MDP. The transition model $p_b(b'|b,a)$ in a belief state MDP is derived from the POMDP following the belief propagation in Equation 3.1.12. The probability of the transition from a belief state b to the next belief state $b'_{b,a,o}(s')$ is given by the probability of observing o in b so that

$$p_b(b'_{b,a,o}|b,a) = p(o|b,a)\,. \tag{3.1.16}$$

The reward model of the belief state MDP is defined as expected immediate reward [3.2]

$$r_b(b,a): \mathbb{B} \to \mathbb{R}, \qquad \text{with} \qquad r_b(b,a) = \sum_{s\in\mathbb{S}} b(s)r(s,a)\,. \tag{3.1.17}$$

[3.2] This reward definition might appear elusive because the agent is rewarded for *believing* to be in states with high rewards. Because the belief update in Equation 3.1.12 is a passive process, however, he cannot deliberately decide to believe something that is untrue.

3.2 Solving Decision Processes

The objective of solving a (PO)MDP is to find an *optimal policy* π^* which maximizes the expectation for the reward sum over the future time steps. The value V_π is the basis to assess and compare policies π. If the optimal value is known, a policy that yields this value can be easily derived. Vice versa, the value that a policy yields can also be quite easily computed. The problem of solving a (PO)MDP thus can be reduced to either finding an optimal policy or finding the optimal value. Recall that POMDPs and MDPs are directly related by the idea of belief state MDPs. We use "(PO)MDP", when referring to both problems.

3.2.1 History of Related Work

Solutions to optimizing (PO)MDPs can be found in different areas of research. If the models of the (PO)MDP [3.3] and the reward function are known, a policy can be found by planning. This procedure mainly has its origin in the area of operations research with groundbreaking work by Sondik et al. [Smallwood and Sondik, 1973; Sondik, 1971]. In case that the models are not fully known a priori they can be learned while the agent interacts with the environment and receives rewards. This procedure mainly has its origin in the area of reinforcement learning [Sutton and Barto, 1998]. Rather than just imitating the reactions of an agent, reinforcement learning aims to explore the structure of the problem through experience.

Nevertheless, it is difficult to draw a clear line between planning and reinforcement learning. Approaches from both areas typically follow the idea of *dynamic programming* [Bellman, 1957a] and apply (PO)MDP models. Research in both areas is very much inspired by each other. As a consequence, the methods are hard to distinguish: they often combine both ideas. The two most popular ways of utilizing the idea of dynamic programming for planning MDPs and POMDPs are *policy iteration* and *value iteration* (see, e.g., [Sondik, 1971; Puterman, 1994; Hansen, 1998]).

Policy iteration (or *policy search*) searches directly in a restricted class of controllers. With every iteration the policy for the complete temporal horizon is updated. In finite-state MDPs, a linear system needs to be solved to first evaluate a fixed policy and then improve it. In policy iteration the policy needs to be represented explicitly. While in MDPs this can be simply accomplished by decision lists, for POMDPs often finite state controllers are used as representation [Hansen, 1998; Poupart and Boutilier, 2003]. Another option is to apply gradient ascent for searching the policy space for locally optimal solutions [Meuleau et al., 1999; Bartlett and Baxter, 2001]. Contrary to policy iteration, value iteration improves the value function in every iteration, by planning

[3.3]The transition model T for MDPs and, additionally, the observation model Ω for POMDPs.

one time step into the future. The policy is then eventually derived from the optimal value. In value iteration, instead of the policy the value needs to be represented. In MDPs, this can be done using a simple vector over the state space. In POMDPs, the property that the value function is piecewise-linear and convex (PWLC) over the belief space allows to represent it as a finite set of vectors over the state space.

3.2.2 Complexity of Solving MDPs and POMDPs

Despite their close relationship, solving a POMDP is considerably more difficult than solving the corresponding MDP. While discrete-state MDPs can be solved in polynomial time, solving discrete POMDPs exactly is provably *intractable* [Papadimitriou and Tsitsiklis, 1987]. The problem of solving the most general case of discrete POMDPs is even *undecidable* as shown in [Madani et al., 1999]. In [Littman, 1996], it is proven that the general infinite horizon, stochastic POMDP problem for boolean rewards is EXPTIME-hard. The solving time is exponential for an infinite number of instances. Some other simplifications of POMDPs, including deterministic transition and observation, completely unobservable Markov decision processes (UMDPs) and POMDPs with a limited time horizon, or combinations of these, can be less complex to solve than the general case. It is out of the scope of this thesis to give a more detailed complexity analysis. Instead, we summarize important results from the literature and explain and outline the reasons why POMDPs are so hard to solve. Understanding complexity is the most important preliminary for creating efficient methods.

In POMDPs, not only the initial state is uncertain. There are possible uncertainties in the information about the underlying states in all time steps of the process. This property is the main origin of the high complexity. Structural properties that can be found and exploited in many MDPs are typically destroyed for the corresponding POMDP by these uncertainties about the current state [Monahan, 1982]. Two intuitive challenges when solving POMDPs can be identified: the *curse of dimensionality* [Kaelbling et al., 1998] and the *curse of history* [Pineau et al., 2003].

Curse of Dimensionality

A POMDP with $|\mathbb{S}|$ states can be converted to a belief state MDP (see Section 3.1.3). An MDP solver can then be applied to solve the POMDP, but the solver must reason about an $|\mathbb{S}|$-dimensional continuous belief space $\mathbb{B} \in \mathbb{R}^{|\mathbb{S}|}$.

The belief space scales exponentially with the number of states in the problem. The meaning of this can be easily seen in a practical example from the domain of drivers assistance systems: when, for instance, considering just the velocity and the distance of the car driving in front of us with just 10 discrete states per dimension for an adaptive cruise control (ACC), already a $(10 \times 10) = 100$-dimensional belief has to be considered. Naively iterating all thinkable belief states in a belief state MDP is virtually impossible because there might be an uncountably infinite number of them [Kaelbling et al., 1998]. Using a naive approximation approach, such as discretizing the belief space scale with again just 10 steps, results in the astronomic number of 10^{100} belief states. As the complexity of solving an MDP is polynomial in the number of states, this naive discretization approach is practically infeasible. At the same time, even if this belief state MDP could be solved, numerous and potentially critical approximation errors are introduced because the discretization could be insufficient for the problem.

Curse of History

For optimal decision making in a partially observable environment an agent must consider not only the current observation, but also the *history* of past actions and observations. A POMDP agent can in every time step decide between $|\mathbb{A}|$ actions. He then perceives the results of his decisions by making one of $|\mathbb{O}|$ observations (see Section 3.1.2). Every sequence of actions and observations encodes a history that needs to be mapped to the optimal decision. A single history can encode a decision rule. It tells the agent which action to chose in the specific situation with this history. The combination of several histories can encode a policy.

The possible number of histories is exponential in the length t of the history: $(|\mathbb{A}| \times |\mathbb{O}|)^t$. For exhaustive planning, every single history has to be considered and assessed. For finite horizons the number of histories is also finite. In infinite horizon problems, the possible number of histories is infinite, but can at least be enumerated (in contrast to the uncountably infinite number of possible beliefs). For illustration, recall the example in Figure 3.5 with three actions and three observations.

The curse of history and the curse of dimensionality are coupled to some extend: if the dimensionality of a problem is very high, there is more room for histories to develop differently. However, in real world problems, often this is not the case. Sometimes there is a small number of histories that only covers tiny fragments of a large or even continuous state space. In driving, e.g., physical limitations, constrain the motion of road users from one time step to the next. Further, the perception of the state through the sensors and as a result, the number of distinct observations is limited. Because of this, the number of relevant belief states and policies is highly limited, too. The method for POMDPs with continuous state spaces that we present in this thesis exploits this property in several ways to make computation practically feasible.

3.3 Value Iteration for MDPs

Value iteration algorithms compute a sequence of value functions starting from an initial value function. Therefore *Bellman backups* steps are repeated until convergence [Bellman, 1957b]. In MDPs, the value for all states s needs to be computed. Basically, the value function V^{n-1} at iteration step $n-1$ for the possible future states s' is propagated back in time in order to obtain the new nth step value function[3.4] V^n

$$V^0(s) = r(s,a) \tag{3.3.1}$$

$$V^n(s) = \max_{a\in\mathbb{A}} V_a^n(s) \tag{3.3.2}$$

$$V_a^n(s) = r(s,a) + \gamma \sum_{s'\in\mathbb{S}} p(s'|a,s) V^{n-1}(s') \,. \tag{3.3.3}$$

If the backup step is repeated for every state, the value function converges monotonically to the optimal value V^* after a finite number of iterations [Bertsekas, 1987]. In discrete MDPs, the value can be efficiently represented by a single vector over s. The running time for each iteration is $O(|\mathbb{A}||\mathbb{S}|^2)$. The value iteration backup is a *contraction.* Every time it is executed the value function converges to the optimal value function. Value iteration converges in polynomial time, depending on the (fixed) discount γ [Littman et al., 1995]. If no optimal solution is needed, value iteration can be stopped early. An approximate solution is called ϵ-optimal, if

$$\forall s \in \mathbb{S} : |V^*(s) - V^{n+1}(s)| \le \epsilon \,. \tag{3.3.4}$$

This condition can be ensured (and value iteration stopped), if the *Bellman residual* [Williams and Baird, 1993] is

$$\max_{s\in\mathbb{S}} |V^n(s) - V^{n+1}(s)| \le \epsilon \frac{1-\gamma}{2\gamma} \,. \tag{3.3.5}$$

The policy π^n in the nth iteration is easily derived from the value function by

$$\pi^n(s) = \arg\max_{a\in\mathbb{A}} V_a^n(s) \,. \tag{3.3.6}$$

Using this equation, also the *optimal* policy π^* can be obtained from the optimal value function in V^*.

3.4 Value Iteration for POMDPs

As stated before, the problem of finding the optimal value for a general infinite horizon POMDPs is uncomputable. Nevertheless, it can be approximated to an arbitrarily small ϵ by a finite (but sufficiently long) horizon POMDP [Sawaki, 1978; Sondik, 1978].

[3.4]In reinforcement learning literature, V_a^n is frequently denoted *Q-function*, Q_a^n.

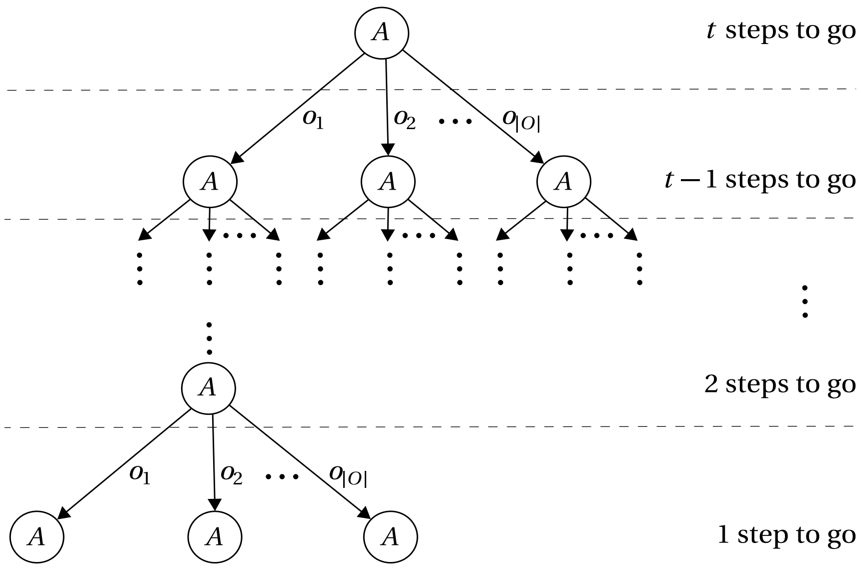


Figure 3.6: Policy tree of length t. The nodes A denote which actions should be taken. The edges denote the observations (or outcomes) $o_1, \ldots, o_k$ of the actions (inspired by [Kaelbling et al., 1998]).

The idea of value iteration for POMDPs intuitively utilizes that the number of histories is finite for finite horizons. As a consequence, finite value function representations exist.

First, we introduce *policy trees* to illustrate the Bellman recursion for POMDPs. Then, *α-vectors* are introduced, which pose a finite representation of value functions in POMDPs. With these preliminaries, POMDP value iteration can be derived by converting the POMDP to a belief state MDP and then applying MDP value iteration. We proof the linearity of the Bellman backup operator, which enables α-vector backups with gradient information. Finally, approximations for exact POMDP value iteration are presented with a focus on point-based algorithms. These are the basis for our continuous POMDP method.

3.4.1 Policy Tree

Reactions to action and observation histories can be represented by a *policy tree* [Littman, 1996]. The policy tree represents a full t-step behavior with a *plan* that deterministically tells the agent with which action to react to received observations (see Figure 3.6 for an illustration). In many ways, policy trees are equivalent to decision trees[3.5]. Note that a policy tree does not encode a POMDP policy because it can only encode a single *plan*. Encoding a policy requires several different plans that the agent can chose from, depending on his belief.

[3.5]Here, we refer to the term as it is usually used in decision-making: a graph to model decisions and their possible consequences.

On the one hand, the policy tree representation is not very efficient and for this reason not necessarily directly used to implement POMDP solvers.[3.6] On the other hand, it intuitively implies the recursive formulation of POMDPs as well as an effective value function representation that is used by many dynamic programming approaches: every policy tree induces an α-vector. Thus, policy trees give an intuitive understanding of α-vectors, which will be formally introduced in the next section.

Let s be the true initial state of an agent. If this agent behaves according to the policy tree PT for a time t, he can expect receiving the value $\alpha(s)$. This value can be computed by recursively descending the policy tree

$$\alpha_{\mathrm{PT}}(s) = \underbrace{r(s,a)}_{\text{current reward}} + \underbrace{\gamma \sum_{s' \in \mathbb{S}} p(s'|s,a) \sum_{o \in \mathbb{O}} p(o|s') \alpha_{\mathrm{PT}(o)}(s')}_{\text{expected future reward}} , \tag{3.4.1}$$

where the choice of a is given by the policy tree PT. The future values $\alpha_{\mathrm{PT}(o)}(s')$ for the next states s' are determined by the subtrees PT(o) that are conditioned by the received observation o.

As the true state s is not known but estimated in form of a belief $b(s)$, the t-step value for an initial belief b_0, as defined in Equation 3.1.15, is of interest. Since the value is the expectation of the world's states, it can be derived from the states' values $\alpha_{\mathrm{PT}}(s)$ for any belief b as follows

$$V_{\mathrm{PT}}(b) = \sum_{s \in \mathbb{S}} b(s) \alpha_{\mathrm{PT}}(s) . \tag{3.4.2}$$

3.4.2 Value Function Representation with α-vectors

The belief and the value function can be interpreted as belief and α-vectors respectively so that this value definition can be rewritten to a more compact form

$$b = \begin{pmatrix} b(s_1) \\ \vdots \\ b(s_{|\mathbb{S}|}) \end{pmatrix}, \qquad \alpha_{\mathrm{PT}} = \begin{pmatrix} \alpha(s_1) \\ \vdots \\ \alpha(s_{|\mathbb{S}|}) \end{pmatrix}, \qquad \text{and} \qquad V_{\mathrm{PT}}(b) = \langle b, \alpha_{\mathrm{PT}} \rangle_d , \tag{3.4.3}$$

where $\langle \cdot, \cdot \rangle_d$ is the discrete *expectation operator* that equals a *dot product.* Intuitively, every α-vector represents the values that can be expected when following a policy tree PT . The values $\alpha_{\mathrm{PT}}(s)$ predict the value that can be expected for every state when following this policy tree.

The above equations show that the value function V_{PT}, which is induced by a policy tree PT, is linear in b. For optimal behavior, it is necessary to follow different policy

[3.6] A forest of policy trees can be transformed to a *finite state controller* (also denoted *policy graph* or *plan graph*). They pose a compact policy representation that is frequently used for policy search methods (see,e.g., [Kaelbling et al., 1998; Hansen, 1998]).

trees/plans depending on the current belief. The optimal value function V_t for a belief b for the horizon t can be expressed as the best of all policy trees PT

$$V_t(b) = \max_{\mathrm{PT}} V_{\mathrm{PT}}(b) \,. \tag{3.4.4}$$

If the horizon t is finite, the number of possible policy trees is finite, too. For this reason, V_t can only consist of a finite number of (linear) facets. Combining Equation 3.4.2 and 3.4.4, we can represent value functions with finite horizons using a set $\Gamma = \{\alpha_1, \alpha_2, \ldots\}$ of α-vectors

$$V(b) = \max_{\alpha \in \Gamma} \sum_{s \in \mathbb{S}} b(s)\alpha(s) \overset{(3.4.3)}{=} \max_{\alpha \in \Gamma} \langle b, \alpha \rangle_d \tag{3.4.5}$$

The policy trees PT are no longer necessary. They are fully replaced by α-vectors that store the value that PT could receive.As a result, the t-step value function forms a piecewise-linear and convex (PWLC) *Simplex*. Figure 3.7 shows an example of such a function for a two-dimensional belief space. The convexity can be intuitively justified. Beliefs on the extremes of the simplex mean that the agent knows for sure that he is in the according state. In these beliefs he can opt for directly maximizing the reward. In contrast to that, the belief b_u, exactly in the middle between the extreme-states, is a uniform distribution: here, the agent first has to obtain the knowledge which state the world is in. This lack of information leads to a lower value. In some cases, it might be not even worth gathering the information. However, having more knowledge always pays out or in other words: information is always worth something. A proper trade-off between acquiring information and exploiting information to receive rewards can only be found when considering the full POMDP.

Resulting from the curse of history, the theoretical number of policy trees and thus the number of facets describing the value function can become very high. It is doubly exponential in the horizon. Let t be the length of the horizon, then the number of policy trees is $|\mathbb{A}|^{\frac{|\mathbb{O}|^t - 1}{|\mathbb{O}| - 1}}$ [Littman, 1996]. In practice, however, only a few of these theoretical policy trees and with them α-vectors have an impact on the optimal policy.

3.4.3 Belief State Value Iteration for POMDPs

As described in Section 3.1.3, a POMDP can be cast as belief state MDP. The optimal policy for this belief state MDP is also optimal for the original POMDP. However, because the belief state space is continuous rather than discrete, the MDP cannot be solved directly. We derive value iteration for POMDPs from value iteration for the belief state MDP. The resulting POMDP backup is a linear operator, which justifies using α-vectors as finite representation.

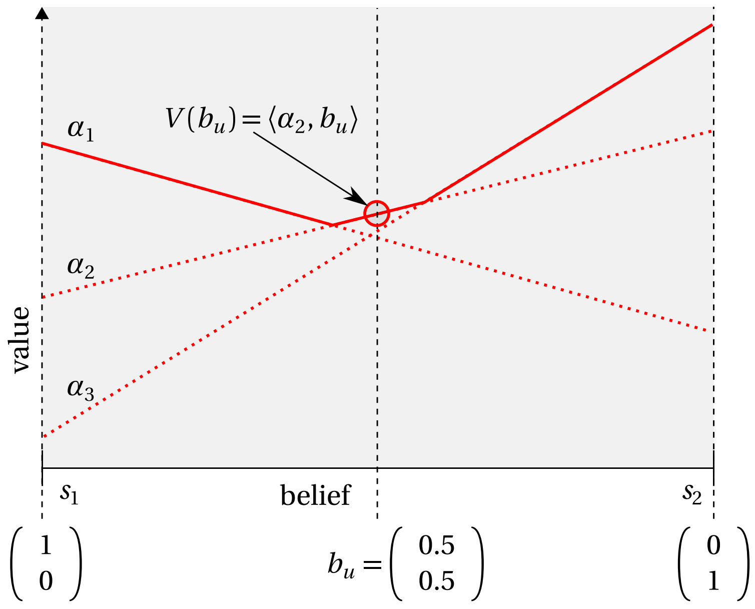


Figure 3.7: Value function V in a state space with two states s_1 and s_2. V is defined as maximum of the set of α-vectors $\Gamma = \{\alpha_1, \alpha_2, \alpha_3\}$. For the belief b_u, α_2 yields the best value.

The nth Bellman backup of the value function for the belief state b can be computed based on the $n-1$th value function V^{n-1}

$$V^0(b) = r_b(b,a) \tag{3.4.6}$$

$$V^n(b) = \max_{a\in\mathbb{A}} V_a^n(b) \tag{3.4.7}$$

$$V_a^n(b) = r_b(b,a) + \gamma \sum_{b'\in\mathbb{B}} p_b(b'|b,a)\; V^{n-1}(b')\,, \tag{3.4.8}$$

where r_b and p_b are the belief state MDP reward and transition functions, respectively (see Section 3.1.3).

The next belief b' is fully conditioned by the action a, observation o and the current belief b following Equation 3.1.12. Thus, we denote it $b'_{b,a,o}$. As $b'_{b,a,o}$ is conditioned by o, the sum over the next beliefs b' can be replaced by the sum over the observations o. Additionally, the conditional probability of the next belief $p_b(b'|b,a)$ can be replaced with the conditional probability of the next observation $p(o|b,a)$.

Following Equation 3.4.5, the $n-1$th step value function is PWLC and represented by vectors α in the set Γ^{n-1}. For the value computation of the next belief $V^{n-1}(b'_{b,a,o})$, we select the $\alpha^{n-1}_{b,a,o}$ which yields the best value for this belief

$$V^{n-1}(b'_{b,a,o}) = \sum_{s'\in\mathbb{S}} b'_{b,a,o}\,\alpha^{n-1}_{b,a,o} = \left\langle b'_{b,a,o}, \alpha^{n-1}_{b,a,o} \right\rangle_d\,, \tag{3.4.9}$$

$$\text{where } \alpha^{n-1}_{b,a,o} := \operatorname*{arg\,max}_{\alpha\in\Gamma^{n-1}} \left\langle b'_{b,a,o}, \alpha \right\rangle_d\,. \tag{3.4.10}$$

The belief state MDP backup in Equation 3.4.8 for action a can be rewritten using the POMDP reward $r(s,a)$, transition $p(s'|s,a)$ and observation functions $p(o|s')$, instead of the belief state MDPs reward $r_b(b,a)$ and transition function $p_b(b'|b,a)$:

$$V_a^n(b) = r_b(b,a) + \gamma \sum_{o\in\mathbb{O}} p(o|b,a) V^{n-1}(b'_{b,a,o}) \tag{3.4.11}$$

$$\overset{(3.4.9)}{=} r_b(b,a) + \gamma \sum_{o\in\mathbb{O}} p(o|b,a) \sum_{s'\in\mathbb{S}} b'_{b,a,o}(s') \alpha^{n-1}_{b,a,o}(s') \tag{3.4.12}$$

$$\overset{(3.1.12)}{=} r_b(b,a) + \gamma \sum_{o\in\mathbb{O}} \sum_{s'\in\mathbb{S}} \cancel{p(o|b,a)} \frac{p(o|s')}{\cancel{p(o|b,a)}} \alpha^{n-1}_{b,a,o}(s') \sum_{s\in\mathbb{S}} p(s'|s,a) b(s) \tag{3.4.13}$$

$$\overset{(3.1.17)}{=} \sum_{s\in\mathbb{S}} \underbrace{\left(r(s,a) + \gamma \sum_{o\in\mathbb{O}} \sum_{s'\in\mathbb{S}} p(o|s') p(s'|s,a)\, \alpha^{n-1}_{b,a,o}(s') \right)}_{:=\alpha^n_{b,a}(s)} b(s)\,. \tag{3.4.14}$$

It turns out that the value function after the Bellman backup remains convex. The backup result V_a^n is represented by a set of linear combinations $\alpha^n_{b,a}$ of $n-1$th step vectors $\alpha^{n-1}_{b,a,o} \in \Gamma^{n-1}$ by

$$V^n(b) = \max_{a\in\mathbb{A}} V_a^n(b) = \max_{a\in\mathbb{A}} \left\langle b, \alpha^n_{b,a} \right\rangle_d\,. \tag{3.4.15}$$

Consequently, if the value function in the previous step $n-1$ is PWLC, this property is maintained for the value function in the nth step (for the original proof see [Sondik, 1971]). Note that the belief-propagation in Equation 3.1.12 is not linear because of the marginalization denominator $p(o|b,a)$. This denominator is canceled out in the POMDP backup as the expectation over all observations o is computed. For this reason, the backup is independent from the current belief (aside from selecting the next α-vectors). This gives rise to performing α-vector backup as described in the next section.

Analogously to MDPs, using the above backup computation, a policy can be derived that yields the nth value function's results following

$$\pi^n(b) = \operatorname*{arg\,max}_{a\in\mathbb{A}} V_a^n(b)\,. \tag{3.4.16}$$

However, direct application of this value iteration scheme is impossible. The belief state space is continuous and thus of uncountably infinite size. Naively iterating all belief states is thus not an option (see Section 3.2.2). Various methods exist that approximate the belief space by discretization. The backup in Equation 3.4.11 is then only performed in the finite set of belief points . The values of the beliefs that are not part of this set can be obtained following an interpolation-extrapolation rule.

Different schemes have been proposed for belief point selection. Fixed regular grids as proposed in [Lovejoy, 1991a] suffer directly from the exponential growth of the belief space with its dimensionality. Some procedures exploit that for many problems the most important beliefs are the extremes of the belief simplex where the agent is certain to be in one of the states. Non-regular grid approximations using these *extreme points* with additional belief points have been proposed that can be, e.g., gained by forward simulation [Hauskrecht, 2000] or by adding intermediate points [Brafman, 1997]. With this procedure ϵ-optimality can be achieved [Bonet, 2002]. However, the belief space must be sampled very thoroughly, which is why this general approach is vulnerable to the *curse of dimensionality*. The next section shows, how the special properties of belief state MDPs can be exploited to generate value functions for the complete belief space using gradient information.

3.4.4 Exact Value Iteration for POMDPs Using α-vector Backups

In contrast to the infinite number of belief states, there is only a finite set of possible policy trees, i.e., derived α-vectors, for finite horizon problems. Thus, the nth step value function can be represented as a finite set Γ^n of α-vectors (see Equation 3.4.5).

This idea, in combination with the important discovery that the value iteration backup is a linear operator, gives rise to α-vector backups. In contrast to belief state Bellman backups, which only compute values for one out of an infinite number of belief states, α-vector backups define a value function over the complete belief simplex.

To obtain an nth step α-vector for action $a \in \mathbb{A}$ the backup has to be computed for all states $s \in \mathbb{S}$ following

$$\alpha_a^n(s) = r(s,a) + \gamma \sum_{o\in\mathbb{O}} \sum_{s'\in\mathbb{S}} \alpha^{n-1}(s') p(o|s') p(s'|s,a), \quad \text{with} \quad \alpha^{n-1} \in \Gamma^{n-1} \quad . \tag{3.4.17}$$

Unlike the direct belief state backup in Section 3.4.3, this computation covers the whole belief space and not just a single point in the belief space. Instead, all choices of $\alpha^{n-1} \in \Gamma^{n-1}$ could dominate in one of all possible next beliefs. To obtain a full backup that eventually approximates an exact solution arbitrarily closely, α-vector backups with all combinations of observations and α-vectors in Γ^{n-1} for all actions $a \in \mathbb{A}$ have to be computed. If all created α-vectors from all combinations form Γ^n, the value function is provably improved (if it has not already been optimal before). However, with this naive approach the number of α-vectors increases exponentially with the size of the observation set $|\mathbb{O}|$ in every iteration $|\Gamma^n| = |\mathbb{A}||\Gamma^{n-1}|^{|\mathbb{O}|}$.

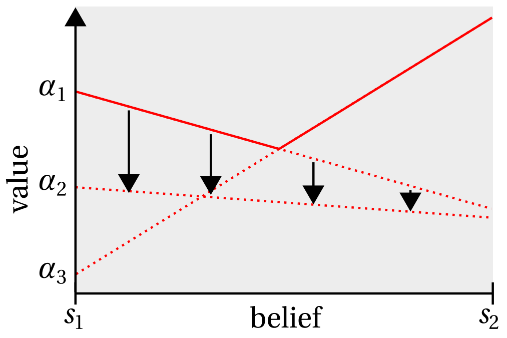


Figure 3.8: α-domination: α_1 dominates α_2 because it is superior over the whole belief space. Hence, α_2 can be safely neglected for the optimal policy.

3.4.5 α-vector Domination

Usually only a small subset of α-vectors is important for the solution. This *parsimonious* set *dominates* the other α-vectors (see Figure 3.8). A vector α_1 dominates another vector α_2, if it yields superior values for every point in the belief space according to

$$\forall b \in \mathbb{B} : \langle b, \alpha_1 \rangle_d \geq \langle b, \alpha_2 \rangle_d \, . \tag{3.4.18}$$

Dominated α-vectors do not carry useful information for the optimal value. There is no situation (or belief), in which they should be selected, if the best behavior is wanted. Dominance can be checked by executing a linear program.

Many exact value iteration methods aim to find a parsimonious set. Some methods are based on the idea of pruning the dominated α-vectors like Monahan's algorithm [Monahan, 1982] and Lark's Filtering [White III, 1991]. "Incremental pruning" is more effective because it discards vectors earlier [Zhang and Liu, 1996, 1997; Cassandra et al., 1997]. Similarly, Sondik's original "One-Pass"-Algorithm [Sondik, 1971; Smallwood and Sondik, 1973] and the "Witness"-Algorithm [Littman et al., 1996; Kaelbling et al., 1998] aim to directly build a parsimonious set.

Despite all this research effort, exact value iteration in POMDPs is only feasible for tiny problems (e.g., the two state "Tiger"-problem from [Cassandra et al., 1994]). Value iteration proceeds in many ways similar to a breadth-first search in the belief state space [Pineau et al., 2003]. For this reason, performing exact α-vector backups directly runs into the *curse of history*. Even using the best exact algorithms, only POMDPs with very few states and/or short horizons can be exactly solved in reasonable time. This renders the exact approach unusable for practical problems and limits its applications to rather academic toy problems. Next, we elaborate on more practical approximate methods.

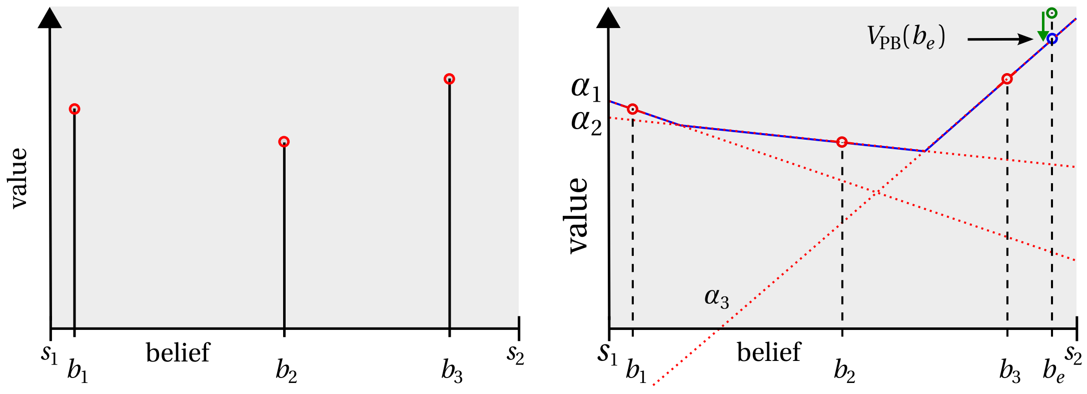


Figure 3.9: Point-based backup in beliefs b_1, b_2 and b_3 without and with α-vector backup.

3.5 Approximate Point-based Value Iteration for POMDPs

The computational infeasibility of exact solutions inhibited the practical application of POMDP models for many years, even though partial observability is very relevant for real-world applications. Theoretically, POMDPs promise huge improvements over current methods and approximate POMDP policies suffice for most applications. In fact, even rough approximations are usually superior to solutions that ignore uncertainty and partial observability.

At the end of the 20th century, POMDPs with up to a few dozens of states could be solved exactly. The rapid development of approximate solvers in the last decade, now allows tackling POMDPs with thousands of states. This performance boost suddenly rendered POMDPs applicable for many tasks. The development was mainly driven by the insights about the origin of the theoretical complexity of solving POMDPs. Most approximation algorithms tackle either the *curse of dimensionality*, the *curse of history*, or both.

The idea of performing backups only at a number of points in the belief space instead of the whole simplex is supposedly the one that recently had the biggest success (see [Shani et al., 2013] or [Smith and Simmons, 2012] for an overview). The general approach is based on exact value iteration with α-function backups presented in Section 3.4.4. The exact approach creates new α-vectors for all combinations of actions, observations, and old α-vectors in every iteration. For this reason it falls victim to the *curse of history*. In contrast to this, a point-based backup only creates one new α-vector for the belief point it was executed in. This procedure does not only allow polynomial-time backups instead of the exponential-time exact backups, but more importantly avoids the exponential growth of the α-vector set.

Figure 3.9 shows a comparison of point-based α-vector backups with belief state backups. Without the gradient information of α-vectors, the results cannot be directly generalized to other beliefs. We illustrate, exemplary for the belief b_e, how the point-

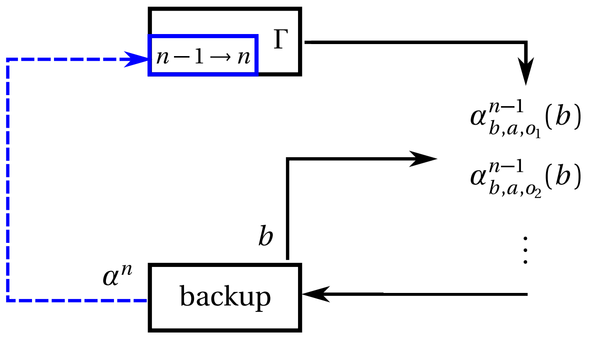


Figure 3.10: Point-based value iteration (PBVI) flow diagram.

based value V_{PB} can be obtained from the set of α-vectors for any belief. The value yielded by a belief state Bellman backup (indicated by the green color) might be higher than the point-based approximation. However, the values for the belief points of the backup (in this case b_1, b_2 and b_3) are always exactly as good as the nth belief state backup in the beliefs.

3.5.1 Point-based α-vector Backup

Every α-vector backup is performed in a belief point b. Intuitively, the computation aims to improve the behavior in this belief. In most cases, this belief is somehow representative for many other beliefs. The α-vector created in belief point b, thus, also contains important information for other beliefs.

The α-vector backup in the belief point b only differs from the exact one in Equation 3.4.17 in the choice of the α^{n-1}

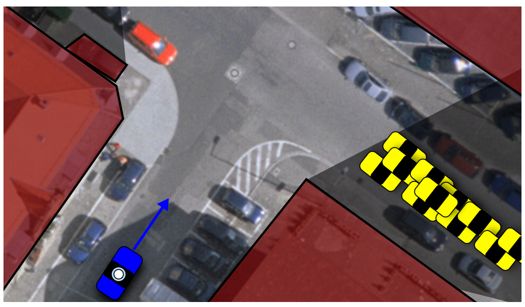
$$\alpha^n_{b,a}(s) = r(s,a) + \gamma \sum_{o\in\mathbb{O}} \sum_{s'\in\mathbb{S}} p(o|s')\alpha^{n-1}_{b,a,o}(s')p(s'|s,a)\,, \tag{3.5.1}$$
$$\text{with } \alpha^{n-1}_{b,a,o} = \arg\max_{\alpha} \left\langle b'_{b,a,o}, \alpha \right\rangle_d .$$

In the point-based backup, the vectors $\alpha^{n-1}_{b,a,o}$ are chosen to be those α-vectors that dominate the successor beliefs $b'_{b,a,o}$ of the belief point b (see the belief propagation in Equation 3.1.12). The point-based α-vector backup could be derived from the belief state backup of b by separately computing the values for all states in b following the last part of the conversion in Equation 3.4.11. The created α-vector optimizes the result for the belief point it is executed in.

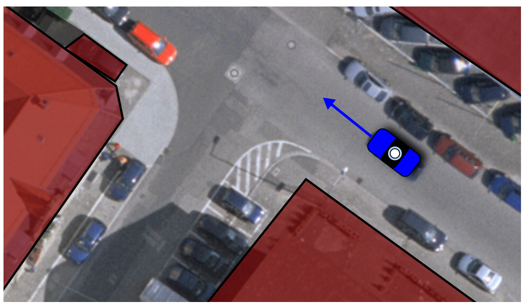
Evaluating the new α-vector for b yields the same value as the $n-1$th value function so that

$$\left\langle \alpha^{n-1}_{b,a,o}, b \right\rangle_d = V^{n-1}_a(b). \tag{3.5.2}$$

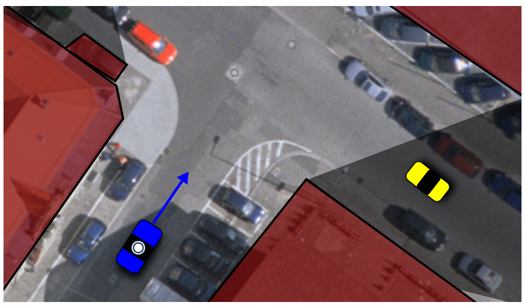
where $\alpha^{n-1}_{b,a,o}$ is the *gradient* of the value function V^{n-1} in the point b. Figure 3.10 sketches the point-based α-vector recursion. The old α-vector set Γ^{n-1} is used to



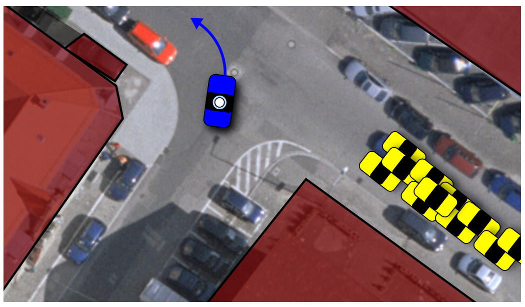
(a) Initial belief.



(b) State physically not reachable.



(c) Belief not reachable because the view of the other vehicle is blocked.



(d) Belief not reachable because the view of the other vehicle is good.

Figure 3.11: Illustration of reachability on the state and the belief level.
Map Data [City of Karlsruhe].

evaluate the α-values for the predicted beliefs. The Bellman Backup creates a new α-function that is added to Γ and increments its version from $n-1 \rightarrow n$.

Early algorithms using this idea aimed at solving the POMDP exactly [Sondik, 1971; Cheng, 1988; Zhang and Zhang, 2001] and still fell victim to the complexity. It turned out that point-based α-vector backups are a good basis for approximate methods because they also backup the gradient of the value and still make statements for the complete belief space. This is also the reason why all these algorithms are *anytime* capable, meaning that they can provide a valid (if not optimal) solution for any belief at any point of computation.

Belief Point Selection

For the choice of belief points, similar approaches as for belief state value iteration without gradient information have been proposed, including fixed grids [Lovejoy, 1991a,b] and forward simulation [Hauskrecht, 2000]. In more recent work, a large number of heuristics is proposed to obtain a representative set of beliefs that improves efficiency. We identified the following concepts.

Reachability: The beliefs in the set should be reachable from the initial belief. This includes the reachability on a state level: many states are (e.g., physically) not reachable from a previous state. Beliefs where the probability for such states is greater zero can be neglected. An example is the state in Figure 3.11b that could in principle be

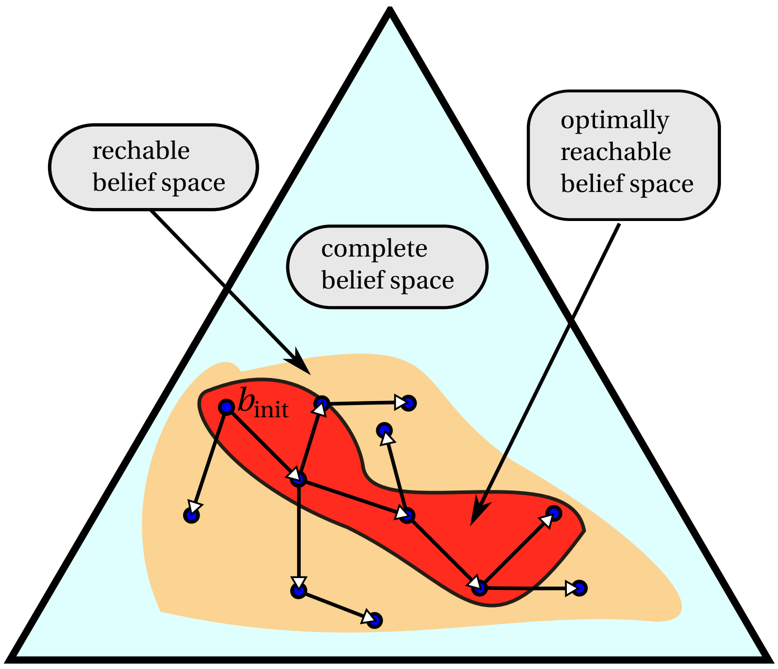


Figure 3.12: Belief space and subspaces. The reachable belief space can be explored by forward simulation from the initial belief. The optimally reachable belief space contains only beliefs that can be reached when executing the optimal policy.

reached, but not from the initial belief. The reachability on belief-level additionally includes that some information can or cannot be obtained. The belief in Figure 3.11c is not reachable because the exact pose of the hidden car cannot be known. In contrast to this, in Figure 3.11d, the other car is not hidden and thus, the knowledge about the other car's pose cannot be that uncertain. The reachable subset of the belief can be constructed by forward simulation, starting at the initial belief (see Figure 3.12).

Optimality: All beliefs in the set should reflect trajectories of optimal solutions. Beliefs that are part of non-optimal solutions are only reachable under suboptimal policies. When executing the optimal policy, they are irrelevant.

Information Gain: To avoid redundant computations, belief points should add as much information as possible about the value and consequently about the policy. Formally, beliefs do not add new information, if they create the same α-vectors. Intuitively, beliefs that are far away from each other and represent different situations add more information. If beliefs are close, often their α-vectors will be similar. Note that this is only a heuristic and not necessarily true. Sometimes similar beliefs can lead to different behaviors (e.g., at the point where the policy of a car changes from *merge onto the lane* to *brake and wait for the next gap*).

Information Propagation: As illustrated in Figure 3.13, α-vectors propagate the information of the Bellman backup over the belief space. As α-vectors contain the gradient of the value, they can in theory contribute to the value of any belief in the belief space, no matter how distant the belief is. However, the backup operation in the belief that

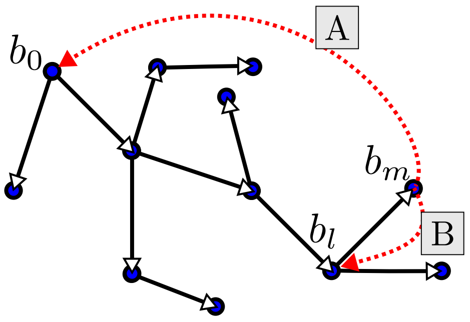


Figure 3.13: Propagation of information in a POMDP. The α-vector created in belief b_m unlikely directly contributes to the policy in b_0 (path $\boxed{\text{A}}$). In contrast to this, it definitely has influence on the backup of its predecessor belief b_l (path $\boxed{\text{B}}$).

is directly reachable from the previous belief definitely contributes to its value. This discovery can improve the efficiency of value iteration by performing backups in the right order.

State-of-the-art Point-based Algorithms

The PBVI [Pineau et al., 2003] algorithm was the first to guarantee convergence under certain conditions. To obtain a representative set of belief points, it uses forward simulation in combination with a heuristic that aims to select those beliefs that are most distant from the existing set in every step. Because distance in the belief space often does not translate into information gain, a modified version of PBVI has been proposed that chooses the belief that maximizes the gain of value [Pineau et al., 2006].

The PERSEUS algorithm [Porta et al., 2006; Spaan and Vlassis, 2005] creates a reachable belief set using random simulation. Instead of filtering this set itself, it only performs backups in a subset of these belief points in every step. This imposes another way to reduce the growth of the α-vector set Γ.

The heuristic search value iteration (HSVI) [Smith and Simmons, 2004] relies on the idea of goal-directed forward simulation. Additionally, in difference to previous methods, the beliefs are not just seen as an unordered set. The perhaps most important innovation of HSVI is that it utilizes that the backup order can be very important for the speed of the convergence. HSVI maintains the history of explored beliefs in a *belief tree* that is related to the idea of policy trees (see Figure 3.6). Other than in policy trees, in belief trees, nodes are not labeled with actions but with the belief that the agent would have in that node. All choices of actions are available in the belief tree (see Figure 3.14) so that they pose an efficient way to store the history of explored beliefs. HSVI performs a depth-first exploration that samples deeply into the belief tree following a heuristic. After this deep-sampling step, backups are performed in the reverse order, starting from the most future belief and ending in the initial belief. This way, the effect of the discoveries in the future beliefs can be pointedly propagated back to the beliefs

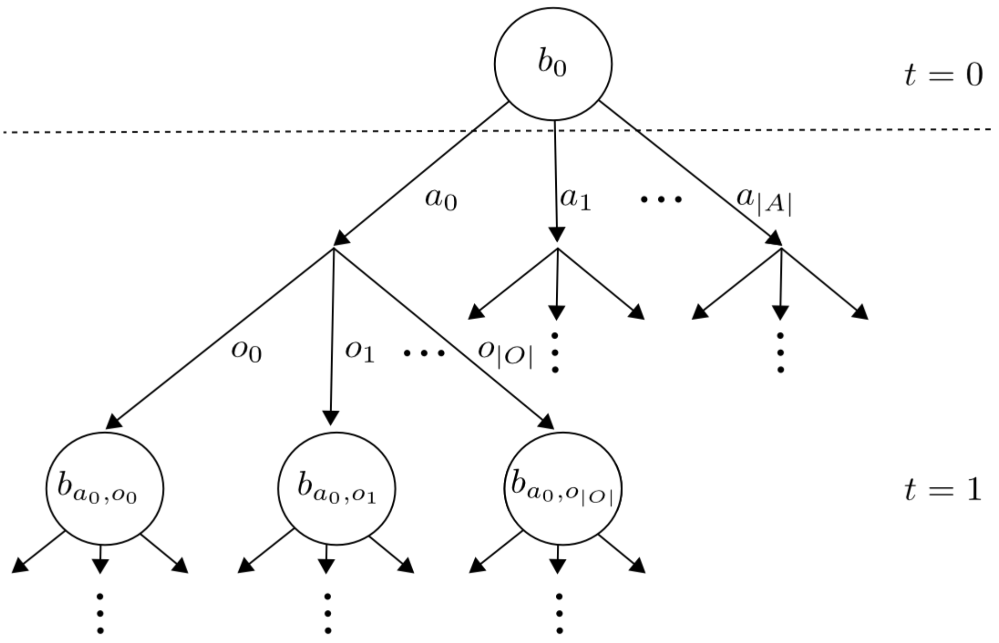


Figure 3.14: Time step 0 and 1 of a belief tree. The nodes are labeled with the beliefs.

that they are interesting for, thereby avoiding redundant computations. HSVI maintains a lower as well as an upper bound of the value to implement this heuristic and to obtain an estimate for the precision of the current policy. The most promising paths are explored first by greedily sampling the action a which currently has the highest upper bound value. This heuristic can assure convergence because if the current upper bound was too loose and promised too high values, it will eventually decrease, once it is explored. The exploration heuristic of HSVI chooses the observation that promises the highest information gain. Therefore, the *excess uncertainty* is introduced. It quantifies the remaining uncertainty in the value function estimate. It is defined using the difference of the upper and lower value bound in the belief. The heuristic chooses the belief with the highest excess uncertainty. It is clear that this heuristics assures convergence as the uncertainty in a belief can only decrease once explored. The excess uncertainty also poses a good termination criteria. If it is small, either because the belief is too far in the future or because it has been sufficiently explored, further exploration of this path can be stopped.

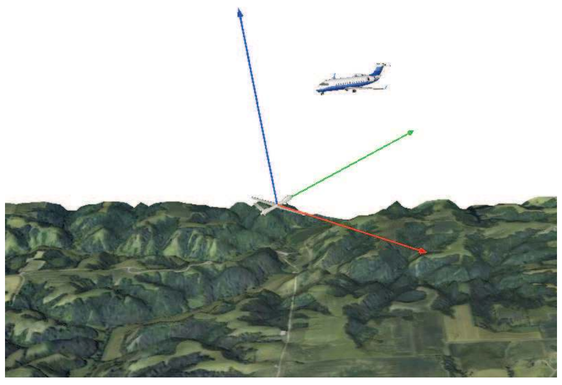
Due to the huge success of this method, a large number of modifications of the original HSVI have been proposed. We will briefly present the most important of them. FSVI [Shani et al., 2007] guided the exploration using an MDP. HSVI2 improved the performance by using tighter bounds for the initialization, avoiding the calculation of linear programs and making better use of sparsity [Smith and Simmons, 2012]. In order to exploit sparse beliefs more efficiently, they proposed to neglect α-vectors for the computation of the value of a belief, if there are non-zero probability states in the belief that have not been covered by the sparse backup which created the α-vector. This was realized by *masking* the α-vectors. SARSOP aims to find tighter approximations of the optimal belief space by value prediction through learning as well as pruning beliefs that have been found to be suboptimal [Kurniawati et al., 2008]. The GapMin-algorithm [Poupart et al., 2011] replaced the sawtooth upper bound representation and update. Additionally, it replaced the depth-first search with a prioritized breadth-first search in order to find tighter upper bounds. The PGVI (packing-guided value iteration) [Zhang et al., 2014] uses the idea of the *covering number* (see next section) to alleviate the curse of history.

3.5.2 Difficulty of Approximating POMDPs

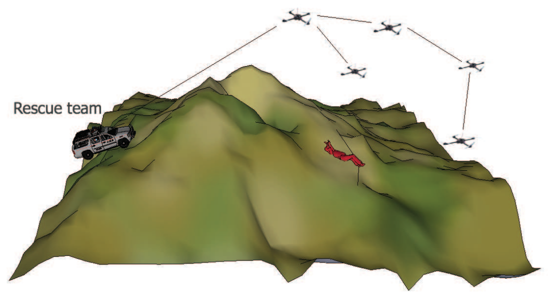
The presented POMDP solvers that are based on the idea of point-based value iteration are among the fastest existing methods for discrete spaces today. Every solver has advantages and disadvantages that prevail depending on the given problem. The size of the history and the dimensionality are intuitive sources of the complexity of POMDPs. However, when approximating POMDPs with algorithms that tackle exactly these two problems, they become overshadowed by other challenges. It turns out that the underlying complexity of a problem is inadequately represented just by the dimension of the belief space or the size of history. The dimensionality, for instance, can just be a result of a poor representation of the problem.

Uncertainty plays a dominating role for the underlying complexity of POMDPs. However, the influence of uncertainty is far from being uni-dimensional. On the one hand, uncertainty results in higher entropy beliefs, which makes them considerably more inefficient to approximate with MC methods. Additionally, uncertain transition probabilities can result in a higher number of future observations that has to be considered. On the other hand, uncertainty can in some cases also simplify computation. If, for example, the observation of the situation is very uncertain, only a small number of next beliefs with low entropy has to be considered. In the extreme case of a UMDP with a blind agent only a single next belief exists. This significantly reduces complexity. Absolute certainty about the state simplifies the problem up to an MDP. If only some variables are completely observable a factored representation can be build as in mixed observable Markov decision processs (MOMDPs) [Ong et al., 2010, 2009]). Certainty that some states are not true, leads to sparse beliefs. This property can also be computationally exploited.

The practical complexity stemming from the history is also more difficult to discover than just numbering the combinatorial possibilities. Despite the fact that only a small subset of all histories might be relevant for the solution (analogously to the idea of the optimally reachable belief space), especially the influence on the performance of dynamic programming is difficult to quantify. Some problems have a long horizon and



(a) Collision avoidance [Temizer et al., 2010].



(b) Search and Rescue [Waharte and Trigoni, 2010]. ©2010 IEEE

Figure 3.15: POMDPs for unmanned air vehicles.

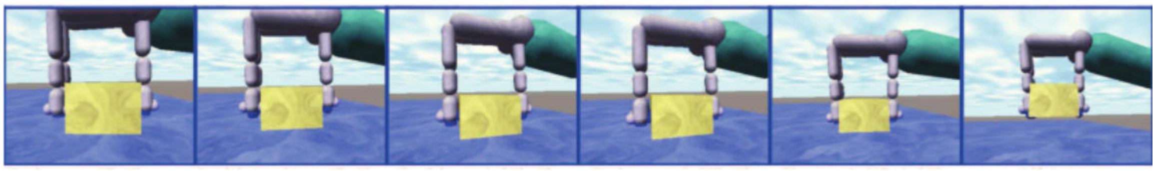
a large theoretical history, but if the agent often faces the same or similar situations the relevant history can be much smaller. If beliefs recur, DP approaches can reuse the results. Algorithms performing α-vector backups can even benefit from beliefs that are similar in terms of their future value in the states. One extreme case is induced by the *terminal state* that is introduced in Section 6.1. It states that the decision process is *over*: the agent is trapped in the terminal state.

In [Hsu et al., 2007] the *coverage* of the belief space is proposed as metric that is able to comprise some of the properties that make POMDPs difficult to solve. But it is still difficult to discover the underlying complexity of a POMDP and which approximation method is best for a given problem. Finally, it is important to state that the real complexity is usually much lower than the theoretical complexity. This is the only reason why large discrete POMDPs and especially continuous POMDPs, which this thesis is mainly interested in, can be solved at all.

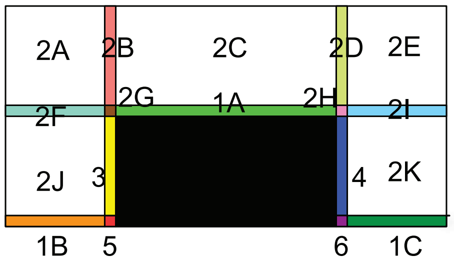
3.6 POMDP Applications

POMDPs pose a very general formulation of sequential decision problems. For this reason, applications are very wide spread. POMDPs are always useful, if a sequence of decisions is wanted rather than a single decision and if major uncertainties are involved in the process. We briefly introduce interesting examples to give an impression for which kind of application considering partial observability comes in handy.

Originally, the topics that researchers had in mind when developing POMDPs were mainly industrial applications. Early research was concerned with machine maintenance [Smallwood and Sondik, 1973; White III, 1991]. The idea is to decrease cost for repair, replacement, inspection or loss of production by optimizing maintenance using a POMDP. Therefore, the state of a machine (or an abstract unit) is modeled as



(a) Simulated grasping.



(b) State space abstraction for POMDP models.

Figure 3.16: Robot grasping with a POMDP policy [Hsiao et al., 2007]. ©2007 IEEE

hidden variable. The wide-spread potential of POMDPs was discovered early (see [Cassandra, 1998] for an overview). Fields of application include autonomous robots, business, health-care, operations-research, social applications, and military applications.

Human-Computer Interaction POMDPs are very useful when interacting with humans. A typical example are dialog systems and especially spoken dialog systems [Young et al., 2013; Williams and Young, 2007; Young, 2006]. The latter are becoming more and more important, for example to reduce costs in call-centers. More recently, speech interfaces, such as Apple's "Siri" and Google's "OK, Google", are becoming the prevalent communication mode with mobile devices, cars, and other technologies. However, there is always a high detection error-rate and uncertainty what the user really said and meant. This problem can be tackled with POMDPs by modeling the users intent as hidden variable so that it can be estimated and pointedly found out.

Another application of POMDPs are systems that assist people with dementia to perform important actions like hand-washing, while at the same time annoying them as little as possible [Boger et al., 2005; Hoey et al., 2007]. For education, the use of POMDPs tutoring systems have been proposed that aim to find optimal teaching actions for individual student models [Rafferty et al., 2011; Folsom-Kovarik et al., 2010]. In this context, the learning progress is modeled as hidden variable.

Information Gain The capability of implicitly performing information gain, if necessary, is important for many applications. Finding a good trade-off between information gain and its cost becomes especially important in medical diagnostics and treatment. Medical systems that combine diagnosis and treatment planning and scheduling have been proposed [Hauskrecht and Fraser, 1998a,b]. The underlying disease of a patient is only partially observable through diagnostic and investigative actions. Additionally, it can be derived from the patients reaction to treatments. For this reason, diagnostics is a highly dynamic process and the cost for gaining information (e.g., by performing a treatment with side effects) can be severe. In [Ayer et al., 2012], an approach to apply POMDPs to design a mammography screening is presented, depending on the personal risk characteristics of women. This way, costs for mammograms as well as false-positives can be reduced.

Autonomous Robots Despite these interesting applications, autonomous robots are one of the main applications of POMDPs. Robots that navigate in the real world often suffer from incomplete and noisy perception. This is also true for flying robots. POMDPs have been proposed for collision avoidance for unmanned aircraft [Temizer et al., 2010; Bai et al., 2011] (see Figure 3.15a). The information gain capability is very useful for "search and rescue" tasks. In [Roy et al., 2006], a POMDP approach for finding and escorting people efficiently in health-care facilities is presented. [Waharte and Trigoni, 2010] found that POMDPs have high potential for supporting rescue operations with unmanned air vehicles (see Figure 3.15b for illustration). In [Hsiao et al., 2007] an POMDP approach for grasping under uncertainty is proposed. To handle the complexity, an abstract symbolic state space is build that separates the state into several regions for observations, rewards and transitions (see Figure 3.16).

3.7 Conclusion

In this chapter, we introduced decision processes and presented methods for solving MDPs and POMDPs with discrete spaces. We identified the size of the history and the continuity of the belief space as the main origins of complexity and presented methods to counteract.

A short selection of applications has been presented that gives an impression of the humongous variety of thinkable applications for POMDPs. Nevertheless, due to their complexity, they have, in practice, only been applied to a few special problems. Today, applications are still restricted to tasks where a small, representative state space can be hand-build or automatically derived prior to solving the POMDP. Especially for autonomous robots that navigate in the real-world this is usually not possible.

One of the main motivations of this work is to open POMDPs to a wider field of applications. In the next chapter a method for solving continuous POMDPs is developed. The ability to solve POMDPs with continuous spaces is essential to get rid of the manual step of pre-tailoring a discrete representation, which prevents using POMDPs in many fields of application, today.

Chapter 4

Continuous Partially Observable Markov Decision Processes

The methodological foundations of the method for continuous POMDPs in Chapter 5 are established in this section. We introduce continuous POMDPs, elaborate on the differences to discrete POMDPs, both, formal and practical and discuss related work. We derive requirements and draw conclusions for the method developed in this thesis.

In Section 3.2, we presented research from the last 50 years that eventually led to methods that can approximate relatively large discrete POMDPs with up to thousands of states. This methodical breakthrough enabled the application of POMDPs in a number of domains. The real world is of continuous rather than discrete nature, though. In general, this has severe consequences on the difficulty of solving a POMDP.

The usual approach to apply POMDPs for real-world problems is to build a discrete representation that is suitable for the specific task. Such hand-build representations are mostly symbolic and based on human language. Frequently, existing models that have been used before for the problem (e.g., deterministic state charts) are adapted. If quantities become important, developers often resort to naive discretization, for example, with equidistant grids. This procedure is in many ways suboptimal. It is costly because it involves experts that model the state space. It is also prone to errors because finding a good representation is not trivial. The representation must be able to express all relevant correlations between states, temporal as well as spatial. Even for the same application, the quality of one and the same representation varies depending on the precise current situation. At the same time it has to be compact enough so that the POMDP can be solved in reasonable time. After all, a predefined discrete representation will most certainly yield inferior policies as it cannot represent every important detail.

For problems like driving, which are more naturally modeled in their original, continuous space, using a hand-build representation is by no means the best solution. Solving the *continuous POMDP* directly is potentially superior. However, considering continuous spaces takes the already utterly difficult problem of solving discrete POMDPs to a whole new level.

Chapter Overview We first introduce continuous POMDPs and point out the differences to discrete POMDPs in Section 4.1. Then, in Section 4.2, we provide an insight into the wide spectrum of work that is concerned with or related to solving continuous POMDPs, and draw conclusions from their results. In Section 4.3, we then explain point-based value iteration with gradient information for continuous POMDPs in detail, as this is the basis for the novel method presented Chapter 5.

4.1 Preliminaries on Continuous POMDPs

Analogously to the definition of a discrete POMDP in Section 3.1.2, continuous POMDPs are defined as tuple

$$(\mathbb{S}, \mathbb{A}, \mathbb{O}, T, \Omega, r, b_0, \gamma)\,. \tag{4.1.1}$$

However, instead of assuming discrete states $s \in \mathbb{S}$, observations $o \in \mathbb{O}$, and actions $a \in \mathbb{A}$, *continuous POMDPs* are based on continuous random variables.

In discrete POMDPs, the *transition model* $T(s', s, a) = p(s'|s, a)$, *observation model* $\Omega(o, s') = p(o|s')$ and *reward function* $r(s, a)$ can be efficiently represented by matrices and vectors, respectively. In continuous POMDPs, the conditional probability densities and the reward function are usually represented by parametrized functionals.

A POMDP is usually denoted *continuous*, if at least one of the sets $\mathbb{S}$, $\mathbb{O}$, and $\mathbb{A}$ is continuous. In the following, we introduce different versions of continuous POMDPs and assess their capabilities of representing tasks as well as the difficulties that they induce to solving the POMDP.

4.1.1 Continuous-state POMDPs

In contrast to discrete-state POMDPs, in *continuous-state POMDPs*, the world is modeled by an infinite number of N-dimensional states rather than a finite number of states[4.1]

$$s \in \mathbb{S} \subseteq \mathbb{R}^N\,. \tag{4.1.2}$$

The sums in the definitions for discrete POMDPs in Section 3.1.2 are replaced with integrals. The continuous belief is a probability distribution defined on the continuous space with

$$b : \mathbb{S} \to \mathbb{R}_{\geq 0} \qquad \text{and} \qquad \int_{s \in \mathbb{S}} b(s)\, \mathrm{d}s = 1\,. \tag{4.1.3}$$

[4.1]The continuous state space can be extended by discrete states. However, this case is w.l.o.g. covered by the given definition.

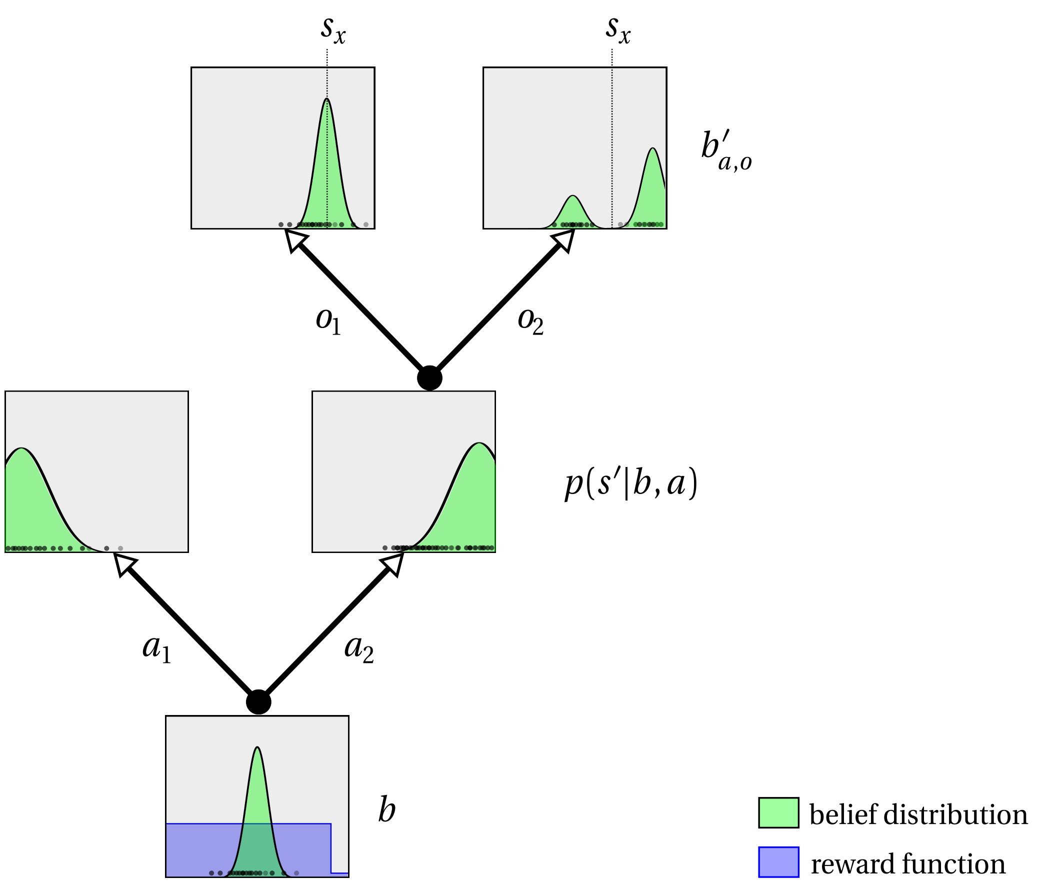


Figure 4.1: Sketch of one time step in a POMDP. The state space is 1D continuous. There are two discrete actions and two observations.

To propagate the belief under the action a and the observation o, the integral over the states s at the previous time step needs to be computed (compare with the discrete belief propagation in Equation 3.1.12)

$$b'_{b,a,o}(s') = \frac{p(o|s')}{p(o|b,a)} \int_{s\in\mathbb{S}} p(s'|s,a)b(s)\mathrm{d}s\ . \tag{4.1.4}$$

The sketch in Figure 4.1 illustrates one time step of the development of a 1D continuous-state POMDP example. The belief b shows that initially the state is only known with variance. The two actions are *move left* and *move right*. As usual in real-world systems, the prediction after executing these actions adds uncertainty. The continuous next beliefs $b'_{b,a,o}$ are the posterior distributions after updating with the observation o. The two observations in this example are o_1 *(approximately at position s_x)* and o_2 *(approximately not at s_x)*.

4.1.2 Continuous-observation POMDPs

In some cases, it can be sufficient to combine continuous-states with discrete observations. For instance, static sensor measurements can be described by a finite measurement space. Practical examples are measurements from a proximity sensor (*the object is near a fixed position*) or from a light barrier (*the object is somewhere on a fixed line in the continuous state space*).

In many dynamic environments such as driving, however, the state (e.g., positions, orientations and velocities) is directly measured. Sometimes, indirect measurements of the state space are obtained. A well-known example are indirect position estimates that are obtained from measurements from multiple distance sensors that can be triangulated. Similar to the state space, naive-discretization of the observation space can induce significant errors and inefficiency. Thus, for most continuous-state systems, it is sensible to consider continuous observations. Let M be their dimension, then[4.2]

$$o \in \mathbb{O} \subseteq \mathbb{R}^M \; . \tag{4.1.5}$$

Having an infinite number of observations aggravates the curse of history: for exhaustive planning, an agent has to take an infinite number of observations into account in every planning step. Additionally, to calculate the expectation over all observations, an integral has to be computed instead of a discrete sum over the finite set of observations.

4.1.3 Continuous-action POMDPs

For some tasks, the action space has to be assumed continuous

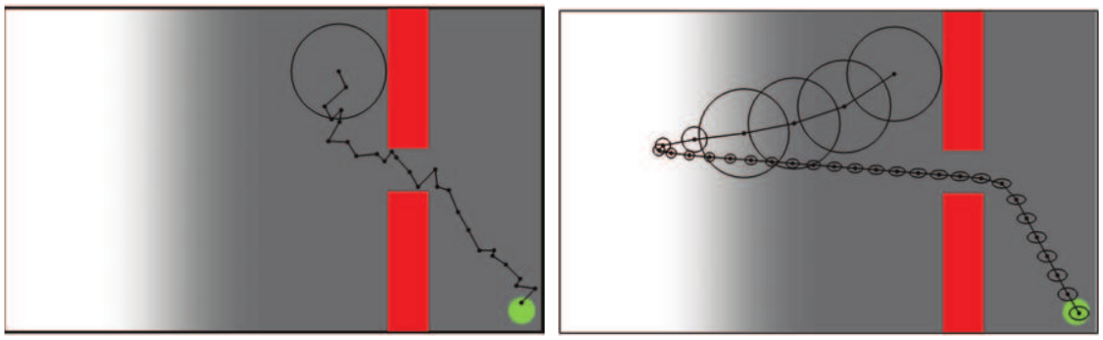
$$a \in \mathbb{A} \subseteq \mathbb{R}^L \; . \tag{4.1.6}$$

This is especially relevant for low-level tasks, where stable and precise control is the goal. While continuous actions greatly add to the generality of the POMDP, they also add to the complexity: in every step the agent has to take into account an infinite number of control-actions.

4.2 Related Work on Solving Continuous POMDPs

If the state space is continuous, the belief space is not only high- but infinite-dimensional. For this reason, known approaches for discrete state POMDPs cannot be applied directly. To solve POMDPs, expected values over the continuous state space need to be calculated. In continuous POMDPs, these are defined by continuous integrals, for which, in general, no closed form exists. Most research on solving continuous POMDPs aims at finding a finite representation of either the policy or the value function. In general, what makes a good representation is not known a priori and naive

[4.2]The continuous observation space can be extended by discrete states. However, this case is w.l.o.g. covered by the given definition.



(a) Ignoring partial observability. (b) Considering partial observability.

Figure 4.2: Stochastic optimal control using iterative application of LQG in a 2D scenario [Van Den Berg et al., 2012].

approaches, such as equidistant discretization of the space, are not feasible for problems with higher-dimensional continuous state spaces.

Related work on solving continuous POMDPs stems from different areas of research. We briefly introduce and assess a selection of methods.

4.2.1 Stochastic Optimal Control

A vast amount of related research stems from the area of *stochastic optimal control*. Most control approaches are derived from the basic linear-quadratic Gaussian (LQG) formulation [Bertsekas, 2007]. LQG can be extended with noise as shown in [Todorov and Li, 2005]. However, in this approach, it remains limited to linear systems with quadratic cost functions, purely additive white Gaussian noise on the motion model, and full observability. Further developments consider uncertain perception, but simplify the POMDP significantly by neglecting obstacles or assuming maximum-likelihood observations (e.g., [Platt Jr et al., 2010; Erez and Smart, 2010]). Recent attempts to overcome the latter limitation, for instance, combine algorithms from motion planning like rapidly-exploring random trees (RRTs) with stochastic optimal control [Bry and Roy, 2011]. Another interesting approach from this area executes approximate value iteration along a trajectory through belief space to find a local optimum [Van Den Berg et al., 2012]). As illustrated in the example in Figure 4.2 in a car-like scenario, stochastic optimal control is well-suited for low-level control tasks where all spaces are continuous and the temporal update rate is high. If the agent in the figure moves to the lighter areas, it can self-localize with higher precision. The example shows that considering partial observability enables the agent to understand that he can cross the passage more safely, if he first moves to the light area to localize himself with higher certainty.

Stochastic optimal control approaches usually use quadratic approximations of the value and linearizations of the belief dynamics. Hence, they cannot be safely be ap-

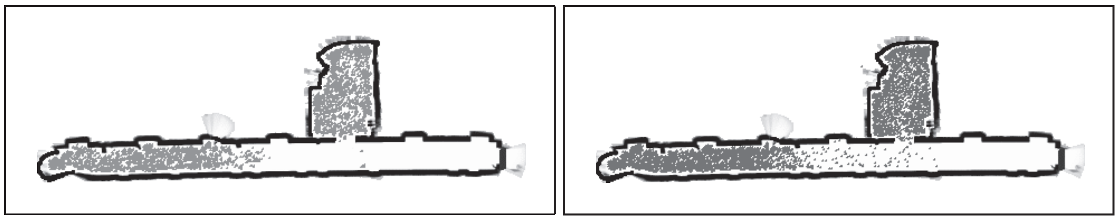

(a) Original particle distribution. (b) E-PCA compression using 6 bases.

Figure 4.3: Learned belief space representation of a person's 2D position in a person finding task [Roy et al., 2005].

plied in POMDPs with, e.g., multimodal beliefs, complex non-linear dynamics or complex reward functions. These are all properties of the higher-level tactical decision-making problem discussed in this thesis. Due to the severe simplification of the POMDP, complex behaviors of other road users or non-trivial limitations of the perception of the autonomous car could not be handled by stochastic control.

4.2.2 Parameteric Representations for General Continuous POMDPs

More general approaches aim to solve the full POMDP. Frequently, sufficient statistics are applied for belief representation. E.g., in [Indelman et al., 2014], the transition and observation model are linearized such that any predicted belief is always represented by a single Gaussian. Similarly, the *parametric POMDP* method [Brooks et al., 2006] uses Gaussian-distributions to parametrize beliefs. The result is a belief propagation similar to that of an extended Kalman filter (EKF). The authors argue that this choice of representation is reasonable because it has been successfully applied in many tracking tasks. They observe, however, that *parametric POMDP* policies often fail when there are sharp discontinuities in the value function, for example, at the edges of obstacles. Moreover, even if using more suitable parametric representations, this approach has a crucial shortcoming: the value function is hardly PWLC in the parameter-space. For this reason, the algorithm cannot make use of the value function gradient, for example, through α-function Bellman backups.

A parametric representation that is more suitable for the problem can be found by analyzing the specific problem. The approach presented in [Roy and Gordon, 2002; Roy et al., 2005] learns a low-dimensional subspace of the belief space from sampled data. It minimizes the Kullback-Leibler (KL) divergence between the sampled data and an exponential family representation by applying Exponential family principle component analysis E-(PCA). Figure 4.3 shows the quality of the compressed belief representation. Similarly, in [Zhou et al., 2010] it is proposed to learn a density projection that minimizes the KL divergence to simulated belief samples. Following this idea, the structure in the belief space can be found and a significant compression can be obtained. However, the learned transformation in both approaches is highly non-linear. This breaks convexity of the value function, which prohibits α-function backups that use gradient information. Additionally, the learned transformations might be good representations of the belief space, but they are not necessarily suitable for representing the policy and the value function. The compressions are learned before solving the POMDP, based on simulated data points. As we show in the next chapter, it is not possible to find an optimal belief space representation before having the POMDP solved.

In the Monte Carlo (MC) POMDP approach [Thrun, 2000], beliefs are represented by particle sets (analogously to sequential Monte Carlo methods). A belief state value backup is derived using importance sampling. On the one hand, this allows to approximately represent arbitrary beliefs. On the other hand, it is difficult to generalize the sample-based value results to other beliefs—a basic requirement for dynamic programming. In MC POMDP a value approximator for a belief based on k-nearest belief neighbors is proposed. To compute the distance between sample-based beliefs, these are first converted to continuous densities using Gaussian kernels. Then KL divergence is used to determine the 'distance' between beliefs. The value of a belief is then approximated by averaging the value of the k-nearest beliefs. In our opinion this straight forward solution hardly scales to more complex problems due to the following reasons. The computational effort for determining the belief-distances (one of the most frequently repeated operations of the algorithm) can become excessive. Also, the approach does not allow for utilizing value function gradient information. Further, this value function approximation requires specifying many parameters, such as the number of relevant neighbors k, the maximum distance for neighbors, and the Gaussian kernels which have significant influence on the performance. If, e.g., the kernels are chosen too small, the performance of dynamic programming deteriorates. If they are too big, conflicting contradicting value-estimates for beliefs can be the result. In practice, different kernels could be optimal for different regions of the same POMDP.

4.2.3 α-function Bellman Backup with Gradient Information

The above methods are not able to exploit gradient information using α-backups. In theory, however, there is no barrier for continuous POMDP value iteration algorithms with α-function backups. The value function in continuous POMDPs is convex and for the special case of discrete observations and actions also PWLC, as is proven in [Porta et al., 2006]. They also show that the continuous POMDP Bellman recursion is a contraction. As a consequence, α-functions are the continuous equivalent of α-vectors. In [Porta et al., 2005], α-function backup is realized using Gaussian mixture models (GMM) as belief and α-function representation. As alternative belief representation, particle sets are proposed in [Porta et al., 2006]. A significant limitation of this choice is that the GMM representation of the α-functions only poses an exact representation, if all models in the POMDP are also Gaussian-based. Further, the number of components in the GMM representation of the α-functions grows exponentially in every value iteration step. Thus, the need for a condensation of the components that shrinks their number arises. Component condensation is not only computationally expensive. It inevitably induces errors that can lead to a degradation of the value function.

4.2.4 Policy Graph Representation and Policy Search

Finite state controllers or policy graphs, as often used in discrete policy search, pose an alternative policy representation, which does not rely on an explicit representation of beliefs or value functions. Policy graphs are closely related to policy trees (see Section 3.2). Analogously to policy trees, nodes denote actions and arcs observations, but policy graphs allow cycles and express policies rather than plans. In [Bai et al., 2010], a combination of Monte Carlo sampling and policy graphs is used (in a similar manner as in [Kurniawati et al., 2008]) to perform continuous value iteration. However, policy graphs cannot represent infinite observation spaces. In [Bai et al., 2014; Bai, 2014] a classification-based solution to this problem is proposed, but it induces approximation errors. Further, this approach heavily relies on simulating the POMDP. A GPU-based variant is presented in [Lee and Kim, 2013] to exploit the parallelism of the backup. Notwithstanding these improvements, the algorithm performs redundant computations because it has to simulate polices to compute values. As a consequence, scalability is an issue for longer horizon problems or POMDPs with computationally expensive models.

Some policy search methods are able to cope with continuous spaces. Examples are PEGASUS [Ng and Jordan, 2000], which simulates belief trajectories, and stochastic gradient ascent-based approaches, such as [Bartlett and Baxter, 2001]. However, they are restricted to particular classes of policies, rely on excessive simulation and are often prone to local optima.

4.2.5 Value-directed Space Representation

Research in MDPs shows the importance of state abstraction in decision problems [Munos and Moore, 1999, 2002]. In Figure 4.4, an illustration of an MDP value function for a continuous 2D problem is given. Frontiers which require to be represented are displayed. The results show that for MDPs, areas in the state space where action or value changes happen are especially important. In [Munos and Moore, 2002], splitting the state space similar to a k-d tree is proposed to obtain a variable resolution grid that expresses the borders precisely, but does not waste much processing power on repre-

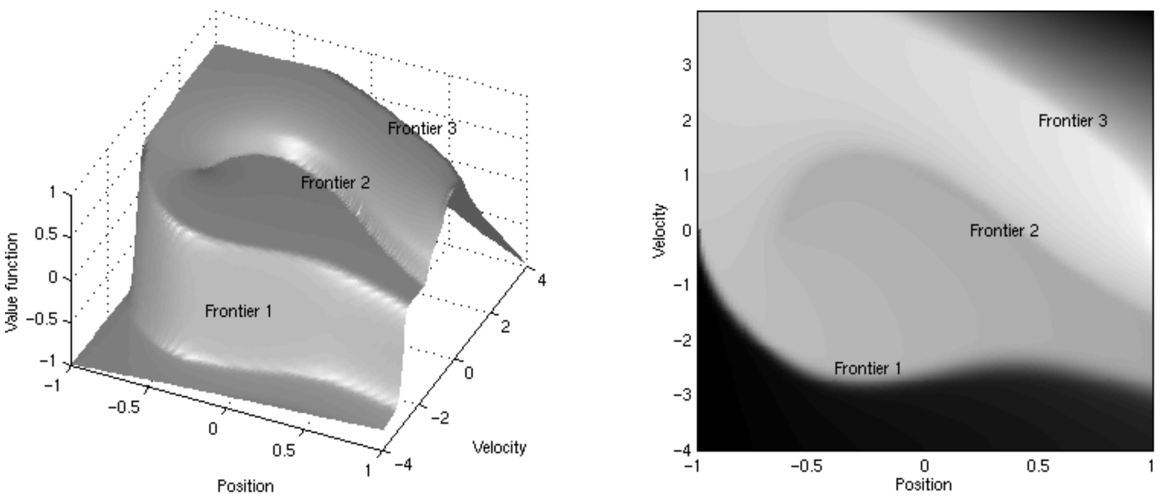


Figure 4.4: Car-on-a-hill MDP value function with policy and value borders [Munos and Moore, 1999].

senting uninteresting areas. This way, they were able to improve the performance of dynamic programming significantly compared to regular grids. Unfortunately, this approach is not capable of considering partial observability nor does it provide a scalable integration into value iteration. Nevertheless, the discovery that some areas are important when solving MDPs and others are not and the important role of the value does also hold for POMDPs and establishes a basis for finding efficient representations.

According to [Poupart and Boutilier, 2002], a representation of the state space of a POMDP is called lossless, if it preserves enough information to select the optimal policy. This is the criterion that must be upheld when finding a compression of the state space. In [Smith et al., 2007], an approach to finding conditionally irrelevant variables by analyzing the POMDP models is presented. However, for many problems this approach is too conservative. Additionally, it relies on a factored representation where no significant dependencies between the uncertain variables are allowed. Another option for state space reduction is proposed in [Feng and Hansen, 2004]: in discrete POMDPs, states with the same value can be aggregated without influencing the policy (similar to the results for MDPs in [Munos and Moore, 2002]). Because the belief is defined on the state domain, this leads to a problem specific compression of the belief space that is refined during the solving process. Solver performance can be significantly improved by utilizing these ideas, but the discrete space approaches cannot be applied to POMDPs with continuous (and thus uncountably infinite) state spaces directly.

4.3 Continuous Value Iteration

In this section, we will briefly introduce and derive the properties that we utilize analogously to discrete value iteration, described in section Section 3.4.

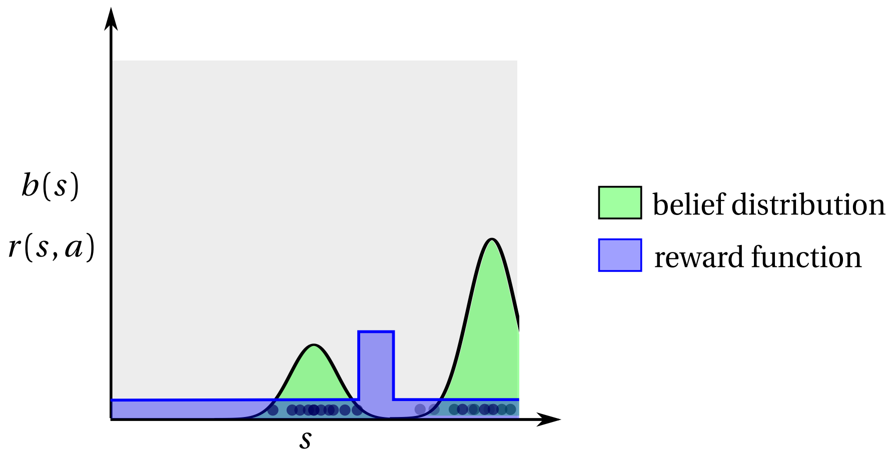


Figure 4.5: Example of a continuous 1D belief and a reward function.

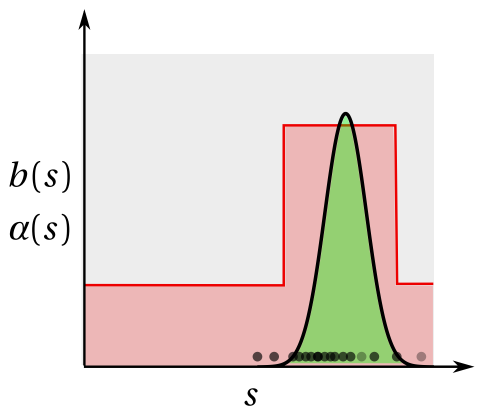
We focus on value iteration for several reasons apart from the fact that approximate discrete value iteration algorithms have been very successful in the past. The first is that finds a globally optimal policy and can be implemented as an anytime algorithm. Its calculations are directly based on the value, which is good criterion for finding a low-dimensional discrete representation of the continuous state space. As we show in Chapter 5, the idea of representation learning can be naturally embedded into value iteration. While considering continuous instead of discrete spaces in general adds to the complexity of the POMDP, it can also be beneficial: the continuous space naturally implies an inductive bias for learning the value. Further, many important properties found for belief state MDPs with discrete state spaces can be transferred to continuous value iteration.

The domain of the value function V as well as the policy π in continuous-state POMDPs is the infinite-dimensional continuous belief space $\mathbb{B}$ so that

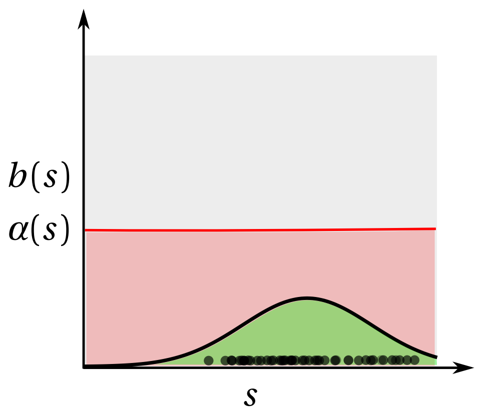
$$V : \mathbb{B} \times \mathbb{A} \to \mathbb{R} \qquad \text{and} \qquad \pi : \mathbb{B} \times \mathbb{A} \to \mathbb{A} \,. \tag{4.3.1}$$

The reward for a belief b is defined as expectation over the reward for the states

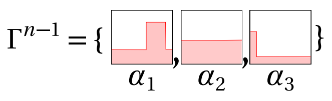
$$r_b(b,a) = \int_{s\in\mathbb{S}} b(s) r(s,a) \,\mathrm{d}s \,. \tag{4.3.2}$$



(a) Best $n-1$th α-function for belief b_1.



(b) Best $n-1$th α-function for belief b_2.



(c) Set of α-functions representing V^{n-1}.

🟩 belief distribution
🟥 α-function

Figure 4.6: Representation of a continuous value function using a set of α-functions.

4.3.1 Continuous Belief-state Bellman Backup

The nth Bellman recursion for POMDPs with continuous state, action, and observation space is given by the following belief state MDP backup

$$V^0(b) = r_b(b,a) \tag{4.3.3}$$

$$V^n(b) = \sup_{a \in \mathbb{A}} V_a^n(b) \tag{4.3.4}$$

$$\text{with } V_a^n(b) = r_b(b,a) + \gamma \int_{o \in \mathbb{O}} p_b(o|b,a)\, V^{n-1}(b'_{b,a,o})\, \mathrm{d}o\ , \tag{4.3.5}$$

where $b'_{b,a,o}$ is the next belief, if action a is executed and o is observed (see the belief propagation in Equation 4.1.4). The belief state reward r_b and transition p_b are defined analogously to the discrete belief state MDP in Section 3.1.3. The initial value function V^0 is simply defined as the expected immediate reward that is directly received in the belief b. To compute V^n in the recursion, the $n-1$th step values V^{n-1} for the next beliefs $b'_{b,a,o}$ have to be obtained. Notice that in difference to the discrete action case, as the action space in general can be an infinite continuum, the maximum over the set of actions has to be replaced with a supremum.

4.3.2 Continuous α-function Representation and Bellman Backup

It is very difficult to represent the value function directly because its domain is the infinite-dimensional belief space. Analogously to α-vectors in discrete POMDPs, the

$n-1$th value function is assumed to be expressed by a set Γ^{n-1} of α-functions that have the finite-dimensional, continuous state space as domain.

Analogously to the discrete expectation operator $\langle\cdot,\cdot\rangle_d$, which essentially is a dot-product, we introduce the continuous expectation operator $\langle\cdot,\cdot\rangle$ to improve readability

$$\left\langle x,y\right\rangle = \int_{s\in\mathbb{S}} x(s)y(s)\,\mathrm{d}s\ , \qquad \text{where} \qquad x,y:\mathbb{S}\to\mathbb{R}\,. \tag{4.3.6}$$

The $n-1$th continuous value function V^{n-1} in any belief b is expressed as the supremum of the set Γ^{n-1} of continuous α-functions by

$$V^{n-1}(b) = \sup_{\alpha\in\Gamma^{n-1}} \int_{s\in\mathbb{S}} b(s)\alpha(s)\,\mathrm{d}s = \sup_{\alpha\in\Gamma^{n-1}} \langle b,\alpha\rangle\ , \qquad \text{where} \qquad \alpha:\mathbb{S}\to\mathbb{R}\,. \tag{4.3.7}$$

Figure 4.6 shows an illustration of the α-function representation in a 1D continuous example. Recall that, intuitively, the α-value at a state $s\in\mathbb{S}$ represents the future reward that can be expected in this state when following the plan represented by the α-function. In belief b_1 of Figure 4.6, the agent knows the state of the world relatively certainly. Hence, he can exploit this knowledge to receive high values. The α-function $\alpha_1\in\Gamma^{n-1}$ dominating in this belief reflects this: it predicts high values in the area around the position of the belief b_1. In belief b_2, the agent has quite uncertain knowledge about the state. Thus, it is better, if he resorts to a more conservative α_2, until he obtained enough information. Although there is a chance that he receives the high value of α_1, in the expectation, the risk is too high. The third α-function, $\alpha_3\in\Gamma^{n-1}$, reflects a plan that yields high values when being in the left corner of the state space. However, neither b_1 nor b_2 have high probability to be there.

With this representation, the $n-1$th value for the next belief $V^{n-1}(b'_{b,a,o})$ of the value iteration Bellman backup in Equation 4.3.5 can be obtained by finding the α-function $\alpha^{n-1}_{b,a,o}$ that yields the highest value for the next belief

$$\alpha^{n-1}_{b,a,o} = \operatorname*{arg\,sup}_{\alpha\in\Gamma^{n-1}} \left\langle b'_{b,a,o},\alpha\right\rangle \qquad \text{and} \qquad V^{n-1}(b'_{b,a,o}) = \left\langle b'_{b,a,o},\alpha^{n-1}_{b,a,o}\right\rangle\ . \tag{4.3.8}$$

Notice in the above definition that V^{n-1} is defined as supremum of linear functions and thus convex. We can now rewrite the continuous nth step Bellman belief state backup in the belief b for action a

$$V_a^n(b) = r_b(b,a) + \gamma \int\limits_{o\in\mathbb{O}} \left\langle b'_{b,a,o}, \alpha^{n-1}_{b,a,o} \right\rangle \, p(o|b,a)\,\mathrm{d}o \tag{4.3.9}$$

$$= \int\limits_{s\in\mathbb{S}} \underbrace{\left(r(s,a) + \gamma \int\limits_{s'\in\mathbb{S}} p(s'|s,a) \int\limits_{o\in\mathbb{O}} p(o|s')\alpha^{n-1}_{b,a,o}(s')\,\mathrm{d}o\,\mathrm{d}s' \right)}_{=\alpha^n_{b,a}(s)} b(s)\,\mathrm{d}s \tag{4.3.10}$$

It can be seen that the continuous value iteration backup is linear. New nth step α-functions $\alpha^n_{b,a}$ are constructed as sums of linear transformations of the $n-1$th step α-functions $\alpha^{n-1}_{b,a,o}$. The notation $\alpha_{b,a}$ accounts for the belief point b which the α-function is created in and the used action a.

The updated value function V^n for a belief b is convex and given as supremum of these new α-functions

$$V^n(b) = \sup_{a\in\mathbb{A}} V_a^n(b) = \sup_{a\in\mathbb{A}} \left\langle b, \alpha^n_{b,a} \right\rangle \,. \tag{4.3.11}$$

For POMDPs with discrete action and observation spaces, and with finite horizon the value function is PWLC, as shown in [Porta et al., 2006]. In this case, the number of α-functions is finite and V^n can be exhaustively represented as finite set Γ^n. Hence, V^n is piecewise-linear. Analogously to the discrete case presented in Section 3.4.4, this proof enables exact value iteration.

Further, Porta et al. show that if all equations are well-defined, the continuous POMDP backup is a *contraction* [Porta et al., 2006]. Thus, it converges to a single fixed point: the optimal value function. The backup is also *isotonic*. Both properties together ensure that the continuous Bellman recursion *converges monotonically*. The *isotonic* property also gives rise to maintaining and updating a *lower bound* on the value function. It assures that a backup step always increases a lower bound. The same is shown for the belief state backup, when executed for an *upper bound*. Thus, backupping a bound maintains its bound property.

These results show that it is perfectly legal to represent continuous value functions by a set of α-functions Γ and perform value iteration using continuous α-function Bellman backups.

4.3.3 Point-based α-function Bellman Backup

Because of the size of the infinite-dimensional belief space, it is sensible to perform point-based backups only in those belief points that probably are (or will be) of interest. This way, the growth of the set Γ of α-functions can be slowed down and controlled. Notice that this is the kind of continuous value iteration that we realized in the presented POMDP solver.

In contrast to exact value iteration, we only create a single α-function $\alpha^n_{b,a}$ for every belief point b that we backup. To be sure to improve the value for the belief point b, the backup is computed for the action a that yields the highest value[4.3]

$$\forall a_l \in \mathbb{A} : V^n_{a_l}(b) \leq V^n_a(b) \,. \tag{4.3.12}$$

The nth α-function is generated based on the $n-1$th value representation Γ^{n-1}. Besides the immediate choice of action in b, additionally, the choice of $\alpha^{n-1}_{b,a,o}$ needs to reflect the values yielded by the best plan known in the $n-1$th step policy for the belief $b'_{b,a,o}$. Due to the linearity of the backup, yielding the highest values for b is equivalent to choosing those α-functions $\alpha^{n-1}_{b,a,o}$ from Γ^{n-1} for the backup that yield the best values for the beliefs reached from b, that is,

$$\alpha^{n-1}_{b,a,o} = \underset{\alpha \in \Gamma^{n-1}}{\arg\sup} \left\langle b'_{b,a,o}, \alpha \right\rangle \,, \tag{4.3.13}$$

where $b'_{b,a,o}$ are the next beliefs that are reached, if the agent, starting in belief b, executes action a and observes o (see Equation 4.1.4).

The nth step α-function Bellman backup in the belief point b when executing action a is then defined by

$$\alpha^n_{b,a}(s) = r(s,a) + \gamma \int\limits_{s' \in \mathbb{S}} p(s'|s,a) \int\limits_{o \in \mathbb{O}} p(o|s') \alpha^{n-1}_{b,a,o}(s') \, \mathrm{d}o \, \mathrm{d}s' \,. \tag{4.3.14}$$

Finally, to obtain the full nth step value function representation Γ^n, we extend Γ^{n-1} with the α-function created in the belief $b \in \mathbb{B}$ by

$$\Gamma^n = \Gamma^{n-1} \cup \alpha^n_{b,a_m} \,. \tag{4.3.15}$$

Figure 4.7 outlines these steps of a continuous point-based α-backup.

Repeatedly applying this backup operation ensures convergence of continuous value iteration. In discrete POMDPs, computing the α-vector backup is unproblematic, as it is defined by sums over finite spaces. In continuous POMDPs, the α-function backup defined in Equation (4.3.14) involves the computation of integrals over the continuous state space. In general, these integrals can only be solved approximately. When using a sampling-based MC approximation for the Bellman backup, the α-functions are only computed for a small subset of the state space. Naive application of MC backup will not yield valid results, since it is infinitely unlikely that the samples of

[4.3] If several actions yield the same value, the choice of action can be randomized.

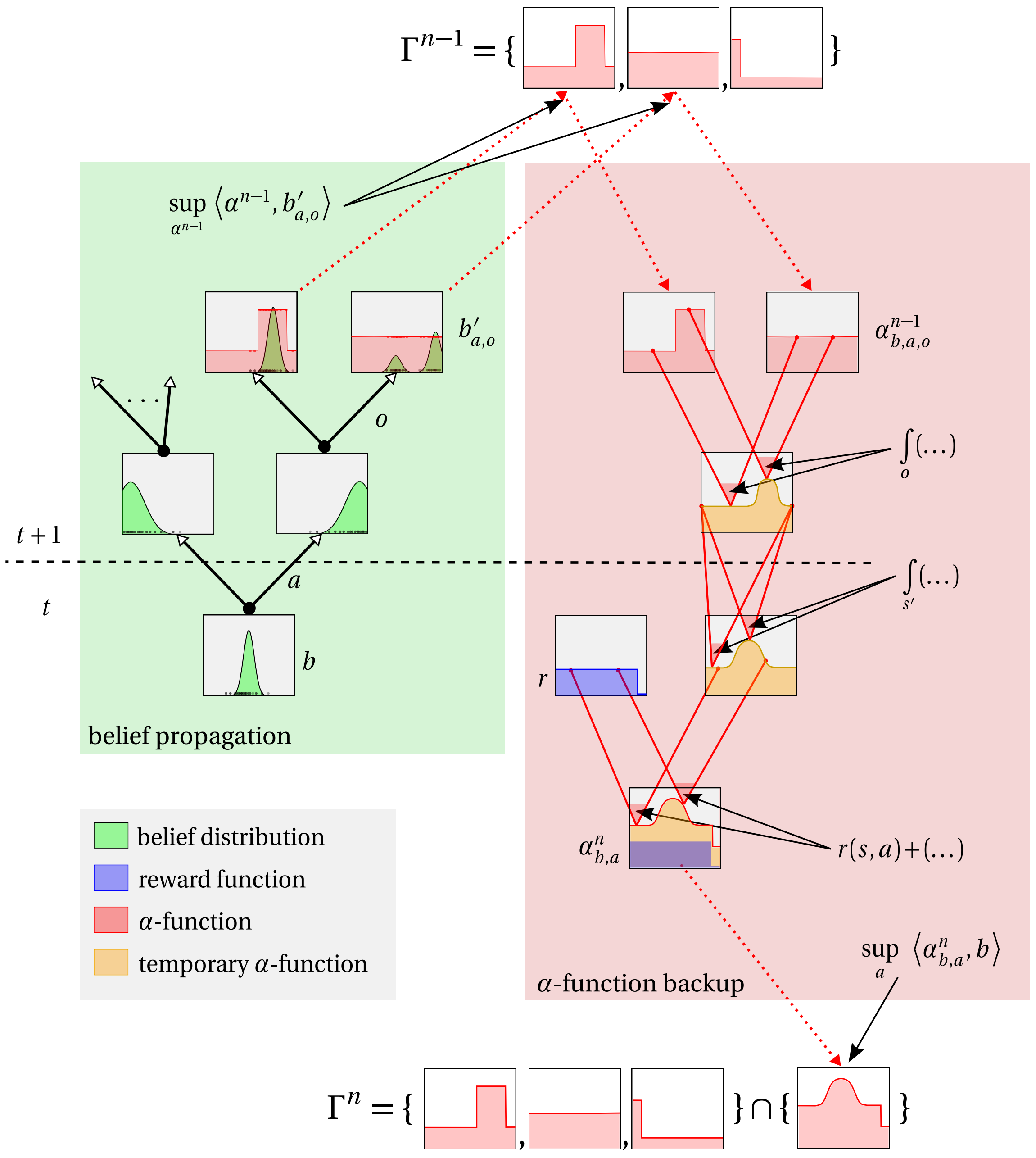


Figure 4.7: nth step continuous point-based α-function backup. The left part (highlighted green) shows the prediction of the belief point b and the selection of the best $n-1$th step α-functions for the belief. The right part (highlighted red) illustrates the Bellman backup of these α-functions. At the end of the backup, Γ^{n-1} is extended with a new α-function from the action that yields the highest values in b.

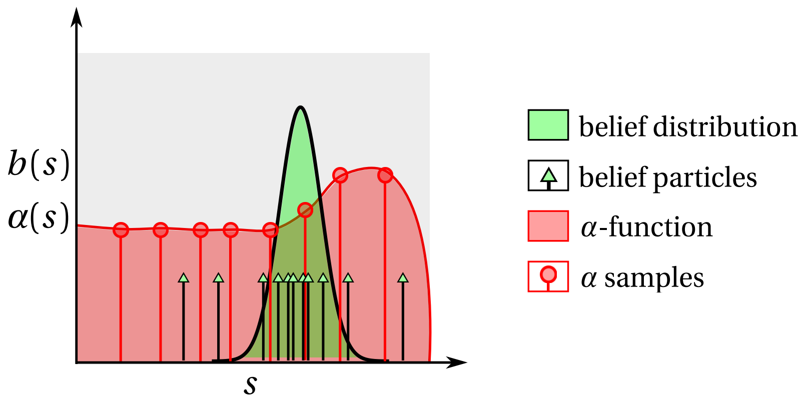


Figure 4.8: Sample-based representation of belief and α-function: direct α-backup is prohibited.

the belief overlap with the samples that represent $\alpha^{n-1}_{b,a,o}$. An example of this is shown in Figure 4.8 [4.4].

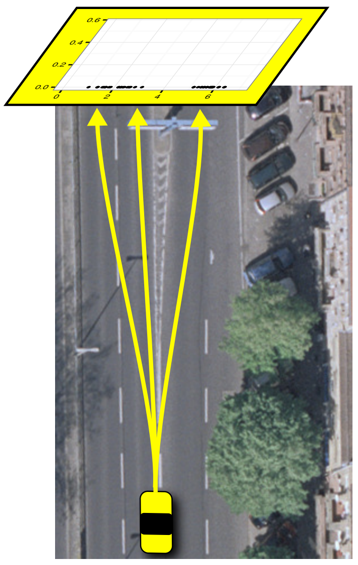
A closed form for Equation (4.3.14) can only be found for some special or restricted problems. If, for example, all models in the POMDP are linear combinations of Gaussians, the α-functions can also be represented as a linear combinations of Gaussians [Porta et al., 2006]. In this special case, the α-function backup can be computed exactly. Unfortunately, even with this severe simplification, approximations are required. The number of components grows exponentially in every value iteration step. To keep the number of components limited, Porta et al. propose an additional *condensation* algorithm that modifies the α-functions after the backup. This condensation step can introduce significant errors into the otherwise exact computations. These results shows that, in practice, even for restricted POMDP problems, approximations are inevitable.

4.4 Requirements for the Developed Method for Continuous POMDPs

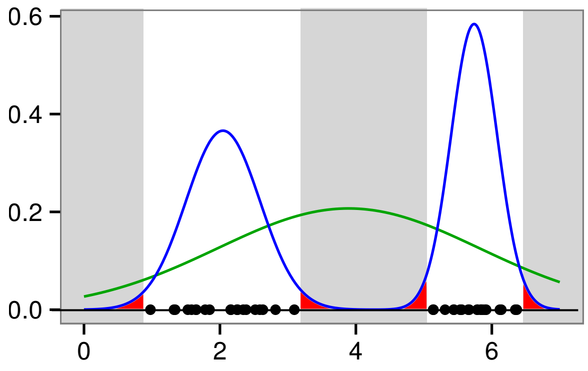
We derive requirements for the continuous POMDP solver developed in this thesis. Therefore, we summarize our insights from the state-of-the-art in the light of tactical decision-making for autonomous driving (see Section 1.2.2).

Requirement 1: Do not impose severe prior restrictions on belief distributions and value functions The belief and value representation is in general not trivial. Even if simplifications, such as predefined statistics, are suitable for tracking tasks, they usually lack expressiveness for decision-making. In Figure 4.9, we show an illustrative example with sharp borders in the state space and multimodality. The unimodal model is the basis for most stochastic optimal control approaches (see LQG in Section 4.2). However, it is neither well-suited for tracking nor tactical decision-making in traffic. It predicts the highest likelihood around position 4: directly into the obstacle between

[4.4] Computing a distance in the belief-space (e.g., KL divergence, as proposed by Thrun et al. [Thrun, 2000]), only works for belief-backups. It fails for α-backups.



(a) Prediction of vehicle position. Map Data [City of Karlsruhe].



(b) Lateral projection of the position prediction. A unimodal Gaussian approximation is indicated green, a bimodal GMM blue. Grey areas indicate the end of the road.

Figure 4.9: Predefined statistics for decision policies in a driving scenario. The black samples are drawn from the true distribution.

the roads. The bimodal distribution gives a better estimate in terms of KL divergence of the belief. However, as the red areas indicate, it cannot represent the sharp edges of the true distribution that are induced by obstacles or the end of the road. Hence, it is also inappropriate for representing the value function,.

Requirement 2: Do not constrain the POMDP model in way that the world dynamics can not be captured sufficently For long-term, decision-making on a tactical level, belief dynamics are often complex. Interaction between road users needs to be covered as well as complex perception, e.g., when road users are hidden behind other objects. The prediction, observation and reward model should not be limited to, for example, a linear or quadratic form. Note that this and the first requirement are highly coupled.

Requirement 3: Discrete action space As a an exception to the otherwise unconstrained models, we do not require the action space to be continuous. Our analysis of related work suggests that, in practice, continuous control imposes severe restrictions on the models, value function and belief representations and the choice of algorithms. In our opinion, a finite set of intelligent macro actions suffices for tactical decision-making in the context of driving. By using intelligent control mechanisms that execute maneuvers and automatically react on the context, the set of actions can be be kept relatively small. For instance, a *lane change* is a difficult control maneuver that depends on the current context, e.g., the curvature of the road, width of the lane and possibly involved obstacles. It can be executed in infinitely many slightly different ways. For deciding whether to execute this maneuver or another, these nuances can be neglected. It can be assumed that an underlying control mechanism will execute the lane change appropriately. In our opinion, the computational disadvantages of continuous actions prevail for our application.

Requirement 4: Perform simultaneous planning of the POMDP and learning of a state space representation. Finding a good representation and solving the POMDP are related problems An optimal and goal-directed representation of a problem cannot be learned prior to solving the POMDP. The task itself implies what is an efficient and at the same time sufficient representation. Only if the optimal policy is known, an optimal representation for it can be found. Vice versa, the optimal policy can only be computed if its representation suffices and is compact enough.

Requirement 5: Preserve PWLC property and apply α-function backups Generalization over the belief space is beneficial. The performance gains by making use of the value functions gradient through α-functions are tremendous because this enables to generalize value functions over the belief space in a mathematically sound way.

Requirement 6: Apply machine learning to efficiently represent and generalize the value function over the state space. The continuity of the state space can be exploited. In contrast to discrete state spaces, continuous state spaces often naturally imply distance metrics. This gives rise to machine learning by assuming an *inductive bias.* Note that previous approaches, using, e.g., sufficient statistics are implicitly based on a similar bias.

Requirement 7: Exploit locality of the belief space as well as sparseness of beliefs Only subspaces are of interest for the optimal policy. In practice, it is sufficient and necessary to sample a subset of the infinite-dimensional belief space. Further, only a small subset of the infinite state set usually is of interest for solving the POMDP. As a consequence, beliefs and α-functions can be represented sparsely.

Requirement 8: Search for a globally optimal policy An optimal POMDP policy is often fundamentally different to the solution yielded when ignoring uncertainty This is easily shown using the introducing example in Figure 1.1. The optimal POMDP policy is to stop in front of the intersection first, to gain information about the gap. This policy would never be the result of an algorithm that assumes full observability because it could always avoid the obstacle without having to stop. Generating policies using a deterministic planner and then selecting one of them by evaluating them with the POMDP is, thus, not an option. Finding a locally optimal policy from this starting point (e.g., using gradient-descent) is also suboptimal.

4.5 Conclusion

In order to meet these requirements, we propose a general value iteration approach that iteratively combines temporal reasoning (planning) with inductive reasoning (learning) to simultaneously find a globally optimal policy and a suitable discrete state space representation. We focus on continuous-state, continuous-observation and discrete-action POMDPs.

Other than imposing discrete actions, we do not fundamentally constrain the spaces or models of the POMDP a priori. This way, the underlying POMDP is general enough to suffice in most applications, including traffic. However, as related work shows, even in the rare cases where a closed form for the Bellman backup exists, for the sake of computational feasibility, approximations are inevitable. Every form of approximation can prevent value iteration from converging to the optimal result. In practice, a diverging value function leads to nonsense policies that can lead to dangerous behaviors.

We conclude that approximations are necessary, but must be applied carefully and should keep the *error in the value function* as small as possible. The quality of the value function representation poses an objective mathematical criterion for the consequences of approximation errors.

We propose two approximations to realize these requirements: First, we propose MC belief propagation and an MC α-function Bellman backup using importance sampling. This choice constrains the underlying models as little as possible. Further, we can control the focus of computation through the biased distribution that the samples are drawn from. Secondly, to close the gap between the sample-based belief representation and the sample-based α-representation (recall Figure 4.8), we learn a discrete state space representation that aims to accurately represent the value function. By assuming an inductive bias based on proximity we can utilize inductive machine learning to generalize the value function over the state space.

Chapter 5

Continuous Value Iteration with Representation Learning

We develop a novel method for solving continuous POMDPs. The central idea is to focus computations on relevant information. Therefore, learning a good, problem-specific state space representation is integrated into continuous value iteration with α-function Bellman backups.

In discrete POMDPs the dimensionality of the belief space equals the size of the state space. This is the origin of the *curse of dimensionality*. In continuous POMDPs, this problem is even aggravated: planning takes place in a belief-space with uncountably infinite dimensions. While at first sight this property appears to render continuous POMDPs absolutely infeasible, we showed in previous chapters that the underlying decision problem often is much simpler than its representation suggests. This can be exploited by smarter algorithms. Littman concluded in 1996 [Littman, 1996]:

"As the complexity results [...] show, POMDPs are simply too difficult to solve. However, they are also too important to ignore. Perhaps a resolution of this difficulty will come when researchers begin to explore applications of POMDPs to important real-world problems."

Littman referred to discrete POMDPs and he was proven right by recent approximation methods. His statement is in our opinion also true for continuous POMDPs. It is not impossible to solve continuous POMDPs approximately, but computations need to be concentrated on aspects which are relevant to the specific decision problem.

Basic Ideas The method we present in this chapter is inspired by the task of decision-making for autonomous driving. Results from an early approach, which relied on equidistant space discretization (see Section 7.2.3) indicated that the underlying patterns of a policy can be automatically learned. In this way, an efficient discrete representation, specific to every POMDP, can be derived from the original continuous representation.

The presented method solves continuous-state POMDPs efficiently without theoretically constraining the process model, observation model or the reward function.

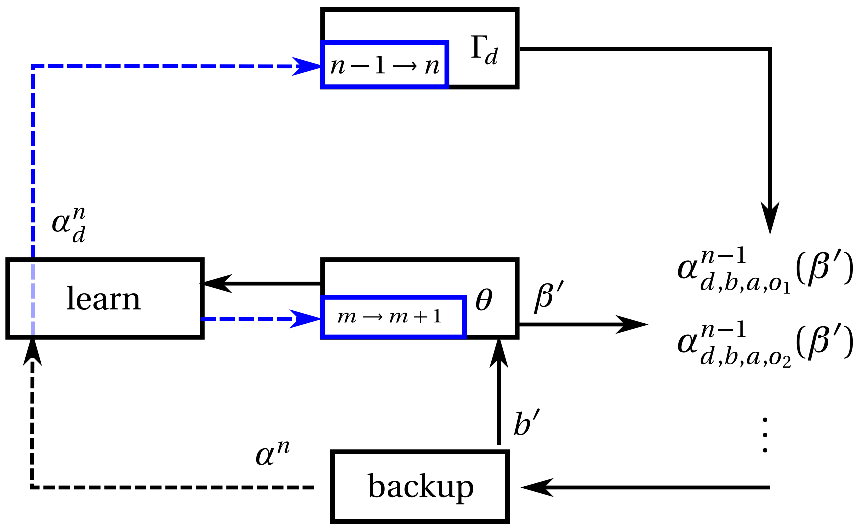


Figure 5.1: Flow diagram for point-based value iteration with representation learning. The recursion simultaneously updates the value function representation Γ_d from version $n-1 \rightarrow n$ and the discrete state space representation θ from $m \rightarrow m+1$.

To achieve this, an efficient representation of the POMDP credentials and especially of the value function and the policy has to be found. What makes a good representation depends on the specific POMDP that has to be solved. It can vary fundamentally, if the models or the situation (e.g., given by the initial belief), vary. Although being inspired by the autonomous driving scenario, the method presented in this chapter is general and can be applied to many other applications with similar properties.

Our approach clearly sets itself apart from previous work by not imposing a mathematical representation of the state space prior to solving it. In fact, finding a representation and finding a policy cannot be viewed as separate problems. The presented algorithm iteratively learns a space representation that is suited for representing the policy that solves the given POMDP. Further, using this representation, local results can be generalized via inductive learning. The basis of this representation learning process is the insight that in contrast to non-metric discrete spaces, the continuous state space naturally implies an inductive bias: spatial proximity in the state space usually translates to similar expected values. Hence, we are able to apply machine learning for finding an efficient, but at the same time sufficient, representation of the state space. Difference in value serves as mathematically sound criteria for *relevance*.

Figure 5.1 sketches our point-based value iteration (PBVI) recursion with representation learning. Analogously to the normal point-based α-function recursion in Figure 3.10, the backup in every step creates a new function α^n. Also, the backup computation uses the functions $\alpha_{b,a,o}^{n-1}$ from the old set Γ^{n-1} that dominate the predicted beliefs in the observations o. In difference to the normal backup, intermediate steps for learning and using the discrete representation θ are added. To evaluate the $n-1$th version α-functions for computing the backup, the predicted beliefs b' are first discretized to the vector β'. When the Bellman backup is finished, a learning step converts the new α-function to the discrete α_d^n and at the same time updates the representation θ from $m \to m+1$. The set Γ_d does not hold continuous domain α-functions, but their vector representations α_d.

Note that parts of the POMDP method in this chapter are presented in [Brechtel et al., 2013], however, in significantly less detail.

5.1 Conceptual Overview

The method is divided into modules. For each module, we formulate the underlying ideas and derive a mathematical framework. Then, we discuss possible implementations and propose an efficient algorithmic realization. To make clear dependencies and the sequence of execution, pseudocode for the algorithms is given.

MC POMDP Simulation and Bellman Backup: Two main abilities for planning in POMDPs are to anticipate the situation by predicting beliefs and to assess likely outcomes by Bellman backups. In continuous POMDPs, there is in general no closed form for both tasks. In Section 5.2, we propose MC algorithms to approximate them arbitrarily close, regardless of the underlying models. By avoiding redundancy in these computations, we ensure computational efficiency. These MC simulations can be viewed as the backbone of planning because they provide the connection between two time steps.

Discrete Representation of Continuous Space: The MC algorithms can compute Bellman backups and approximate the resulting continuous α-functions for any state $s \in \mathbb{S}$. However, α-functions can only be computed for a finite set of samples, (i.e., the α-function approximation remains undefined for an infinite number of states $s \in S$). To overcome this gap and enable efficient dynamic programming, the sparse α-functions (i.e., samples in s with values $\alpha(s)$) must be generalized over the complete state space.

Since the continuous beliefs as well as the continuous α-functions are defined on the same continuous state space, we propose a low-dimensional, discrete state space representation for beliefs as well as α-functions. In Section 5.3, we put this idea into a general theoretical concept and derive the mathematical framework that is necessary to carry out Bellman backups.

Iterative Representation Learning: To find a low-dimensional, discrete representation, we apply machine learning (see Section 5.4). Low-dimensional representations of the continuous state space have been proposed in previous works (see Section 4.2). These differ mainly in two aspects from our representation learning approach: they aim to represent the beliefs as accurately as possible and fix the representation prior to solving the POMDP. In contrast to this, we aim to learn a representation that minimizes the error of the value function representation. This line of action accepts a lossy compression of beliefs, but aims at lossless compression of the value and the policy.

The value function, however, is not known before the POMDP is solved. Instead of using an a priori fixed representation, we propose to apply inductive machine learning iteratively to refine the representation as part of value iteration. We propose an iterative refinement approach that can be seamlessly integrated into value iteration, allowing the representation to constantly adapt to new knowledge from the planning. Lastly, we propose an implementation of this idea using decision trees and top-down-induction.

Algorithmic Realization and Program Flow: Finally, in Section 5.5, we put the components together. A deep-sampling algorithm for exploring the belief space and carrying out the α-function backups to propagate the information of the backups efficiently is provided in Section 5.5.2. A concept for treating approximation errors is introduced in Section 5.5.3: utilizing the monotonicity of α-function backups, we formulate a criterion to detect overly optimistic generalizations and propose an algorithm to correct errors. The detailed interaction of the components is presented in Section 5.5.4.

Notation In the following, we highlight sample-based approximations of an exact computation $(\cdot)$ by $\tilde{(\cdot)}$ (e.g., for a belief $\tilde{b}$). Samples that are part of approximations are indicated by $\hat{(\cdot)}$ (e.g., $\hat{s}$ for samples from the state space).

Also, we use the indexes $\hat{s}_i, \hat{s}_j, \hat{o}_k = 1, \dots, Q$ to denote the samples in approximations. To improve readability, the number of indexes is w.l.o.g. fixed to Q for the approximations. It is possible to use different sizes of particle sets without noteworthy changes to the equations or algorithms. Sets of current state samples $\hat{s}$ are denoted $\mathbb{I}$, sets of next state samples $\hat{s'}$ are denoted $\mathbb{J}$ and sets of observation samples $\hat{o}$ are denoted $\mathbb{K}$.

In algorithmic descriptions with pseudocode, coefficients of stored vectors are addressed by $[\cdot]$. For example, the ith coefficient of the vector v is $v[i]$.

5.2 Monte Carlo (MC) POMDP Simulation and Bellman Backup

In general, there is no closed form of the integral in the continuous belief prediction problem in Equation 4.1.4 that is the basis of of planning in a POMDP. In order to evaluate the likelihood of the next state $b'(s')$, a continuous integral over the previous states $s \in \mathbb{S}$ needs to be solved. This part of the belief prediction equals the problem of sequentially estimating a systems state, often referred to as *recursive Bayesian estimation* or *Bayes filter*. Sequential Monte Carlo (SMC) methods are a well-known technique for Bayesian filtering with arbitrary transition and observation models [Doucet et al., 2001]. SMC methods approximate the estimation problem numerically by iteratively applying importance sampling (IS). The basic idea behind is to use a finite set of weighted samples to evaluate the continuous integral. Samples with their according weights are usually denoted *particles*. In contrast to, e.g., Kalman filtering, which requires linear models and uni-modal distributions, SMC methods are not limited to models and distributions with certain properties [MacKay, 2003]. Their capabilities come at the price of higher computational effort and suboptimal solutions, especially when dealing with density functions with high variance. For highly non-linear or even multi-modal transition and observation models, however, the general applicability of SMC methods outweigh their disadvantages.

Planning in POMDPs involves not only the prediction of the agents belief. Additionally to the filtering problem, the expected future consequences of actions have to be assessed and quantified. Therefore the expectation over histories has to be computed. Executing the continuous belief state value function backup in Equation 4.3.9 requires solving continuous integrals over the current states s, the next states s' and all observations o. The continuous α-function backup in Equation 4.3.14 lacks the integral over the currents states s. However, to evaluate an α-function for a belief, the expectation over all $s \in \mathbb{S}$ has to be computed, too. Similarly to Bayesian filtering, IS can be applied to approximate the integrals in the Bellman backup. This has been proposed in previous work for approximating belief state backups [Thrun, 2000] and α-function backups [Bai et al., 2010].

The task of predicting the successor beliefs $b'_{b,a,o}$ of a belief b and performing point-based value iteration Bellman backups in b are highly related. In fact, for both tasks it is important to base the computation on samples that approximate b and $b'_{b,a,o}$ accurately.

We apply MC approximations to simulate and assess the behavior of the world.

Three specific tasks can be identified: belief propagation[5.1], belief state Bellman backups, and α-function Bellman backups. In difference to previous approaches, we exploit that these three tasks are highly related. To avoid redundant sampling from or evaluation of conditional distributions, we propose using a shared sample set for all three tasks. Intelligent application of IS enables us to weight and recombine the samples in different ways, depending on the task. The sample set can be interpreted as a temporary discretization of the continuous space.

To improve clarity of presentation, the MC value and α-function Bellman backups are derived for a reward function $r(s,a)$. See Appendix A.1 for the modifications when using a reward function $\mathring{r}(s,a,s')$, which additionally depends on the next state s'.

[5.1] In this thesis the term *belief propagation* denotes the transition from one belief to the next and is not to be confused with the message passing algorithm for performing inference on graphical models.

Next, we explain, how the sample sets for the approximations are generated.

5.2.1 Drawing Sample Sets

A sample set $\mathbb{I}_b$, with b serving as proposal (i.e., biased) distribution, is drawn for every belief b where point-based backups are performed in. As b is initially given as a set of particles, this essentially is a *resampling* to avoid degradation of the particle set. The procedure is known from the sequential importance resampling (SIR) algorithm for particle filtering. The samples $\hat{s}$ in the set $\mathbb{I}_b \subset \mathbb{S}$ are sampled from the current belief b such that

$$\hat{s} \sim b(s)\,. \tag{5.2.1}$$

The current belief b is then approximated by a mixture of Dirac deltas δ with uniform weights

$$\tilde{b}(s) = \frac{1}{|\mathbb{I}_b|} \sum_{\hat{s} \in \mathbb{I}_b} \delta(s - \hat{s})\,. \tag{5.2.2}$$

The current belief b is independent from the action a. This is not true for its prediction. For this reason the two sample sets $\mathbb{J}_{b,a}$ and $\mathbb{K}_{b,a}$, representing the prediction and observation, respectively, are conditioned by the belief b and the action a. The set $\mathbb{J}_{b,a} \subset \mathbb{S}$ approximates the prediction of the state when conducting action a in the belief b so that

$$\hat{s}' \sim p(s'|b,a) = \int\limits_{s \in \mathbb{S}} p(s'|s,a) b(s)\, \mathrm{d}s\,. \tag{5.2.3}$$

The set $\hat{o} \in \mathbb{K}_{b,a} \subset \mathbb{O}$ approximates how likely an observations o is perceived after conducting action a in belief b so that

$$\hat{o} \sim p(o|b,a) = \int\limits_{s' \in \mathbb{S}} p(o|s') p(s'|b,a)\, \mathrm{d}s' = \int\limits_{s' \in \mathbb{S}} p(o|s') \int\limits_{s \in \mathbb{S}} p(s'|s,a) b(s)\, \mathrm{d}s\, \mathrm{d}s'\,. \tag{5.2.4}$$

For a belief b, Algorithm 1 obtains a total of Q samples of s and $Q \times |\mathbb{A}|$ samples of s' and o.[5.2]

In Figure 5.2a a visualization of the procedure is given. First, $\hat{s}$ is drawn from $b(s)$. Then, for every a, $\hat{s}$ is predicted and $\hat{s}'$ is drawn from the predicted distribution $p(s'|\hat{s},a)$. Finally, the observation $\hat{o}$ of the next state sample $\hat{s}'$ is drawn from $p(o|\hat{s}')$.

The usual procedure for MC approximating α-function backup $\alpha(s)$ is to sample a new set of $\hat{s}'$ for every $\hat{s}$. This procedure can entail serious computing effort: e.g., the

[5.2]It can be easily modified to obtain different numbers of samples for the sets.

Algorithm 1 Forward sampling.

```
function DRAW SAMPLES($b$)
    $\mathbb{I}_b \leftarrow \emptyset, \forall a \in \mathbb{A} : \mathbb{J}_{b,a}, \mathbb{K}_{b,a} \leftarrow \emptyset$
    for $Q$ times do
        $\hat{s} \leftarrow$ DRAW FROM($b(s)$)
        $\mathbb{I}_b \leftarrow \mathbb{I}_b \cup \hat{s}$
        for all $a \in \mathbb{A}$ do
            $\hat{s}' \leftarrow$ DRAW FROM($p(s'|\hat{s}, a)$)        ▷ Sample p(s'|b,a)
            $\mathbb{J}_{b,a} \leftarrow \mathbb{J}_{b,a} \cup \hat{s}'$
            $\hat{o} \leftarrow$ DRAW FROM($p(o|\hat{s}')$)        ▷ Sample p(o|b,a)
            $\mathbb{K}_{b,a} \leftarrow \mathbb{K}_{b,a} \cup \hat{o}$
        end for
    end for
    store $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$
end function
```

total number of created $\hat{s}$ is $|\mathbb{A}| \times Q^2$. However, for every $\alpha(\hat{s})$ computation, only Q samples are used.

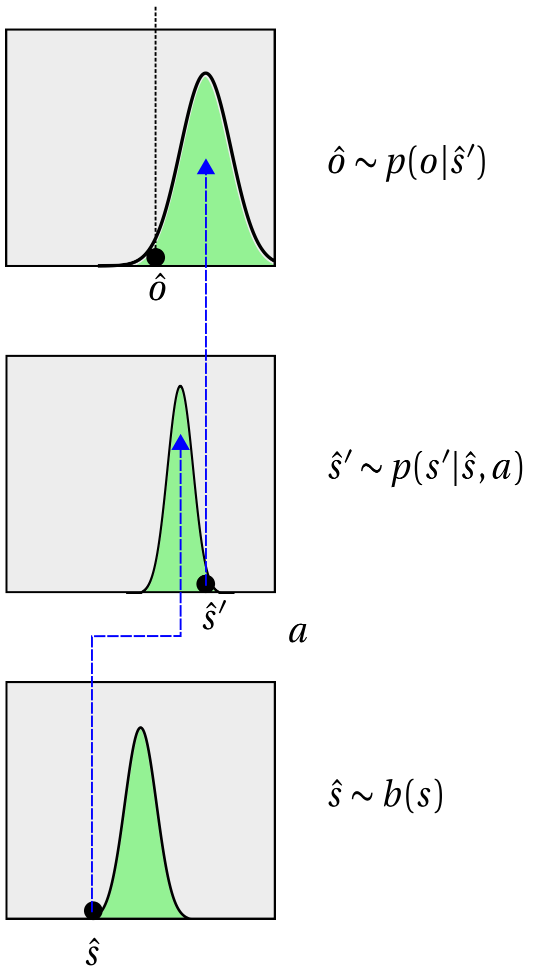
By associating the particles of the current belief $\hat{s} \in \mathbb{I}_b$ with all predicted samples $\hat{s}' \in \mathbb{J}_{b,a}$ and the observation samples $\hat{o} \in \mathbb{K}_{b,a}$, the influence of every sample $\hat{s}$ of the original belief b on the value $V(b)$ can be determined. The resulting value of all samples is the basis for all value computations. Compared to the usual procedure, we also use Q samples $\hat{s}'$ for approximating the Bellman α-function backup $\alpha(\hat{s})$, but require only $|\mathbb{A}| \times Q$ samples in total.

The motivation behind this is that state samples that are close in the state space are in practice often predicted and observed similarly. In Figure 5.2b, an example of this is given. The two state samples $\hat{s}_1$ and $\hat{s}_2$ are close in the state space. In the example, due to their proximity, their predicted probabilities $p(s'|\hat{s}_1, a)$ and $p(s'|\hat{s}_2, a)$ overlap to a great extend. For this reason, the precision can be improved, if not only the sample $\hat{s}_1'$ (stemming from $\hat{s}_1$) but also $\hat{s}_2'$ (from $\hat{s}_2$) are used to approximate the value in $\hat{s}_1$.

For the given reasons, cross-combining the particle sets is usually computationally beneficial.[5.3] Especially in POMDPs, reusing the observation particles makes sense. While for state estimation only a single observation has to be considered, a POMDP planner has to consider all possible observations(or at least a sufficient subset). In practice, often several observation samples are close to each other.

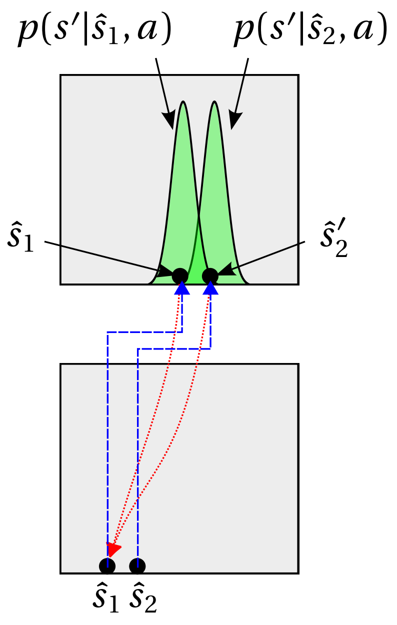
To utilize the sample sets for the three different tasks, belief propagation, belief state Bellman backups, and α-function Bellman backups, importance sampling can be em-

[5.3] In some cases, the particle recombination can have computational disadvantages: e.g., when the predicted and observed densities have little or no overlap. However, this can be alleviated by exploiting sparseness of the transition model in the implementation.



$\hat{o} \sim p(o|\hat{s}')$
$\hat{o}$
$\hat{s}' \sim p(s'|\hat{s}, a)$
$\hat{s}'$
a
$\hat{s} \sim b(s)$
$\hat{s}$

(a) Forward sampling.



(b) Overlapping predicted distributions.

Figure 5.2: MC sampling and sample recombination.

ployed to correct their bias. The basis for the importance weights are the conditional density functions of the transition and the observation for all combinations of samples

$$\forall a \in \mathbb{A}, \hat{s} \in \mathbb{I}_b, \hat{s}' \in \mathbb{J}_{b,a} : p(\hat{s}'|\hat{s}, a)\,, \quad \text{and} \quad \forall a \in \mathbb{A}, \hat{s}' \in \mathbb{J}_{b,a}, \hat{o} \in \mathbb{K}_{b,a} : p(\hat{o}|\hat{s})\,,$$

and the reward function for all state samples $\forall a \in \mathbb{A}, \hat{s} \in \mathbb{I}_b : r(\hat{s}, a)$.

Computing these weights can be expensive, if the POMDP has a complex reward function, transition models, and observation models (e.g., simulating human behaviors or complex physics). In the literature, mostly POMDP examples with simple models (e.g., linear with additive Gaussian distributed noise) are used. The computational cost for simulating the POMDP are for this reason often underestimated as being very low. However, in reality, high-level decision-making can require more complex models. In our realization of the driving task in Chapter 6, several time steps of (highly non-linear) interaction of multiple road users with each other and the road network have to be computed for evaluating a single transition probability $p(\hat{s}'|\hat{s}, a)$. For such

POMDPs, minimizing redundant evaluation of, or sampling from these conditional probabilities can be well worth the effort.

5.2.2 MC Belief Prediction

To plan in a POMDP, it is essential to foresee what could happen. Thus, the future course of beliefs must be predicted. This can be separated into two questions: What is the distribution of the next belief $b'_{b,a,o}(s')$ and how likely is it to get there.

The former question resembles one step of sequential Bayesian filtering. Having a particle-based belief representation it equals particle filtering. We use the sample sets $\mathbb{I}_b$, $\mathbb{J}_{b,a}$, and $\mathbb{K}_{b,a}$ similarly to SMC. The samples $\hat{s}' \in \mathbb{J}_{b,a}$ serve as basis for the approximation of the next beliefs $b'_{b,a,o}$. Because they are drawn from the proposal distribution $p(s'|b,a)$, they have to be weighted with the likelihood $p(o|s')$ to account for the observation.

The normalized importance weights $w^{(\hat{s}')}_{b,a,o}$ of the sample in $\hat{s}' \in \mathbb{J}_{b,a}$ for the belief posterior estimate after observation o are given by

$$w^{(\hat{s}')}_{b,a,o} = \frac{\bar{w}^{(\hat{s}')}_{b,a,o}}{\sum\limits_{j\in\mathbb{J}_{b,a}} \bar{w}^{(j)}_{b,a,o}}\,, \qquad \text{where} \qquad \bar{w}^{(\hat{s}')}_{b,a,o} = p(o|\hat{s}')\,. \tag{5.2.5}$$

The resulting belief is approximated by a mixture of Dirac deltas δ. We write

$$\tilde{b}'_{b,a,o}(s') = \sum_{\hat{s}'\in\mathbb{J}_{b,a}} w^{(\hat{s}')}_{b,a,o}\,\delta(s'-\hat{s}')\,. \tag{5.2.6}$$

Executing action a in the current belief b, the posterior belief $b'_{b,a,o}$ is now approximated by the samples $\hat{s}' \in \mathbb{J}_{b,a}$ weighted with $w^{(\hat{s}')}_{b,a,o}$. The accuracy of this approximation is arbitrarily improved, when the number of samples Q goes to infinity.

While this approximates the posterior belief distribution for any observation o, it does not tell us how probable this belief distribution is. The second part of the belief prediction is to predict what are possible next beliefs and how probable they are. Since $p(b'_{b,a,o}|b,a)$ is given by $p(o|b,a)$, every sample $\hat{o} \in \mathbb{K}_{b,a}$ conditions a posterior belief $b'_{b,a,\hat{o}}$. The belief tree can thus be expanded simply by computing the next beliefs $b'_{b,a,\hat{o}}$ for all samples $\hat{o} \in \mathbb{K}_{b,a}$ as sketched in Algorithm 2.

This kind of belief prediction resembles particle filtering with SIR. In difference to the normal filtering problem, however, not only a single observation o is taken into account. Instead a belief tree with all sampled observations $\hat{o}$ and according beliefs $b'_{b,a,\hat{o}}$ is constructed.

In [Porta et al., 2006], belief simulation for POMDP planning using an *auxiliary particle filter* [Pitt and Shephard, 1999] is proposed. In our approach, we restrain to the simpler SIR. Auxiliary particle filtering takes the observation likelihood into account

Algorithm 2 Belief tree construction.

```
Require: $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$                                   ▷ See Algorithm 1
1: function EXPAND BELIEF TREE($b, T_{\mathbb{B}}$)  ▷ add next beliefs $\tilde{b}'_{b,a,\hat{o}}$ for $b$ to belief tree $T_{\mathbb{B}}$
2:     for all $a \in \mathbb{A}$ do
3:         for all $\hat{o} \in \mathbb{K}_{b,a}$ do
4:             $\tilde{b}'_{b,a,\hat{o}} \leftarrow$ CREATE BELIEF($b, a, \hat{o}$)          ▷ following Equation 5.2.6
5:             $T_{\mathbb{B}} \leftarrow$ APPEND BELIEF TREE($T_{\mathbb{B}}, \tilde{b}'_{b,a,\hat{o}}$)
6:         end for
7:     end forreturn $T_{\mathbb{B}}$
8: end function
```

during resampling. It is useful, if the sampled particles do not fit the observation well. This can be the case, if the observation is far off the prediction (i.e., the prior $p(s'|s,a)$ only has a small overlap with the posterior $b'_{b,a,o}(s')$). Another problem in SIR occurs, if the observation model is very certain and the prediction model very uncertain. The likelihood then is very peaked and only a small set of particles is in the significant support of the likelihood. These shortcomings pose no problem, if the observations are simulated (as is the case in POMDP planning).

Our approach exploits that in SIR, the next states $\hat{s}' \in \mathbb{J}_{b,a}$ used to represent $b'_{b,a,\hat{o}}$ are independent from the observations $\hat{o} \in \mathbb{K}_{b,a}$ and thus can be shared between all next beliefs $b'_{b,a,o}$. This choice allows to increase the number of samples without deteriorating the computational performance of the POMDP backup too much. To be able to use more samples outweighs the potentially better convergence properties of auxiliary particle filtering.

5.2.3 MC Belief Value Backup

Since all samples in $\mathbb{I}_b$, $\mathbb{J}_{b,a}$ and $\mathbb{K}_{b,a}$ are originally drawn from the belief b, they are a good basis to calculate the Bellman backup for b. We can also make use of the previously computed posterior beliefs $b'_{b,a,\hat{o}}$. These are conditioned by the observation samples $\hat{o} \in \mathbb{K}_{b,a}$, which are also biased towards the belief b. For this reason, the belief state value backup is computationally inexpensive (evaluation of the next beliefs' values left aside).

The immediate reward is defined as expected reward for b (see Equation 4.3.2). It can be approximated simply by summing all rewards for the particles in $\mathbb{I}_b$, since these are sampled from b.[5.4] The future value is computed by iterating over the sampled observations $\hat{o}$ and summing the future values for the beliefs $V^{n-1}(b'_{b,a,\hat{o}})$. Again, due to the correct biasing of the samples $\hat{o} \in \mathbb{K}_{b,a}$, they can be directly used without applying weights.

[5.4]See Appendix A.1 for $\mathring{r}(s,a,s')$

Algorithm 3 Value function backup in a belief point.

```
Require: samples $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$                    ▷ see Algorithm 1
 1: function MC BELIEF VALUE BACKUP($b, \Gamma_d^{n-1}$)  ▷ approximate Bellman value backup
    for belief $b$
 2:     for all $a \in \mathbb{A}$ do
 3:         COMPUTE BELIEF VALUES($b, a, \Gamma_d^{n-1}$)     ▷ get $V_{\alpha_d}[b'_{b,a,\hat{o}}]$ (see Algorithm 6)
 4:         $\tilde{V}_a^n(b) \leftarrow 0$
 5:         for all $\hat{s} \in \mathbb{I}_b$ do
 6:             $V_a^n(b) + \leftarrow r(\hat{s}, a)$                          ▷ immediate reward
 7:         end for
 8:         for all $\hat{o} \in \mathbb{K}_{b,a}$ do                               ▷ iterate beliefs
 9:             $\tilde{V}_a^n(b) + \leftarrow \gamma \max_{\alpha_d \in \Gamma_d^{n-1}} V_{\alpha_d}[b'_{b,a,\hat{o}}]$         ▷ future value
10:         end for
11:         store $\tilde{V}_a^n(b)$
12:     end for
13:     $\tilde{V}^n(b) \leftarrow \max_{a \in \mathbb{A}} V_a^n(b)$
14:     return $\tilde{V}^n(b)$
15: end function
```

The nth value backup approximation of the exact backup in Equation 4.3.2 for the belief b results to

$$\tilde{V}^n(b) = \max_{a \in \mathbb{A}} \tilde{V}_a^n(b)\,, \tag{5.2.7}$$

$$\text{with } \tilde{V}_a^n(b) = \left(\sum_{\hat{s} \in \mathbb{I}} r(\hat{s}, a) \right) + \gamma \sum_{\hat{o} \in \mathbb{K}_{b,a}} V^{n-1}(b'_{b,a,\hat{o}^{(k)}})\,. \tag{5.2.8}$$

At this point, we assume that we can compute the continuous value function for the next beliefs $V^{n-1}(b'_{b,a,\hat{o}})$. In Section 5.3, we propose using a learned discrete representation for this. In Algorithm 3 the detailed MC backup procedure is shown.

5.2.4 Point-based MC α-function Backup

Additionally to the belief value backup, we perform point-based α-function backups to determine the value gradient for the whole belief space.

Every α-function backup maintains or improves a lower bound and the policy represented by it for all beliefs. The choice of action and future α-functions (plans), however, optimizes the value in the belief point b that it is executed in. See Section 4.3.3 for details. The challenge, when performing continuous α-function backups, is to compute the new $\alpha_{b,a}^n(s)$ from the $n-1$th step $\alpha_{b,a,o}^{n-1}$ functions following Equation 4.3.14. This computation involves integration over the continuous future states $s' \in \mathbb{S}$ and observations $o \in \mathbb{O}$. Intuitively spoken, the α-function backup assesses the influence a state s has on the expected future rewards. In theory, it must be computed for all states in $\mathbb{S}$. Recall the exact definition of the point-based α-function backup

$$\alpha^n_{b,a}(s) = r(s,a) + \gamma \int_{s' \in \mathbb{S}} p(s'|s,a) \int_{o \in \mathbb{O}} p(o|s')\, \alpha^{n-1}_{b,a,o}(s')\, \mathrm{d}o\, \mathrm{d}s' \,, \tag{5.2.9}$$

$$\text{with } \alpha^{n-1}_{b,a,o} = \operatorname*{arg\,sup}_{\alpha \in \Gamma^{n-1}} \left\langle b'_{b,a,o}, \alpha \right\rangle .$$

As we perform point-based α-function backups, we first need to select the best α-functions for the next beliefs $b'_{b,a,\hat{o}}$ from the $n-1$th α-function set Γ^{n-1}. Because the integral over the observations is approximated with the samples $\hat{o}$, we only need to find α-functions for the beliefs $\tilde{b}'_{b,a,\hat{o}}$ conditioned by these sampled observations. The beliefs $b'_{b,a,\hat{o}}$ are given from the belief prediction (see Section 5.2.2) in form of particle-based beliefs $\tilde{b}'_{b,a,o}$ and the best α-functions $\alpha^{n-1}_{b,a,\hat{o}}$ are given by

$$\alpha^{n-1}_{b,a,\hat{o}} = \operatorname*{arg\,max}_{\alpha \in \Gamma^{n-1}} \left\langle \tilde{b}'_{b,a,\hat{o}}, \alpha \right\rangle . \tag{5.2.10}$$

Due to the sifting property of Dirac deltas, the expectation computation reduces to a sum

$$\left\langle \tilde{b}'_{b,a,\hat{o}}, \alpha \right\rangle = \int_{s' \in \mathbb{S}} \tilde{b}'_{b,a,\hat{o}}(s') \alpha(s')\, \mathrm{d}s' \tag{5.2.11}$$

$$\overset{(5.2.6)}{=} \int_{s' \in \mathbb{S}} \sum_{\hat{s}' \in \mathbb{J}_{b,a}} w^{(\hat{s}')}_{b,a,\hat{o}} \delta(s' - \hat{s}') \alpha(s')\, \mathrm{d}s' \tag{5.2.12}$$

$$= \sum_{\hat{s}' \in \mathbb{J}_{b,a}} w^{(\hat{s}')}_{b,a,\hat{o}} \int_{s' \in \mathbb{S}} \delta(s' - \hat{s}') \alpha(s')\, \mathrm{d}s' \tag{5.2.13}$$

$$= \sum_{\hat{s}' \in \mathbb{J}_{b,a}} w^{(\hat{s}')}_{b,a,\hat{o}}\, \alpha(s') \,. \tag{5.2.14}$$

Here, we assume that $\alpha \in \Gamma^{n-1}$ is defined for all $s \in \mathbb{S}$. This is realized by learning a discrete representation that generalizes the sparse sample-based computations (see Section 5.3).

The same sample sets $\mathbb{I}_b$, $\mathbb{J}_{b,a}$, and $\mathbb{K}_{b,a}$ (see Algorithm 2) can also be used for the MC α-function backup. We evaluate the new α-function only for the samples $\hat{s} \in \mathbb{I}_b$. Hence, the new $\tilde{\alpha}^n_{b,a}$ function is only defined at these points $\hat{s}$ in the state space. The integral over the future states is approximated using $\hat{s}' \in \mathbb{J}_{b,a}$. Finally, the integral over the observation space is evaluated using $\hat{o} \in \mathbb{K}_{b,a}$. In difference to the belief value backup, this computation is independent from b. Thus, this time we need to compensate that the samples are biased towards b. Therfore, we employ IS with the following importance weights.

We need to compensate that the samples $\hat{s}'$ in $\mathbb{J}_{b,a}$ do not directly reflect the probability of the next state $p(s'|s,a)$ coming from s. Instead, they are biased towards b. In the limit, the biased distribution is

$$\hat{s}' \sim p(s'|b,a) \overset{(5.2.3)}{=} \int\limits_{s\in\mathbb{S}} p(s'|s,a)b(s)\,\mathrm{d}s\,. \tag{5.2.15}$$

Knowing the sample set $\hat{s} \in \mathbb{I}_b$ used to generate $\hat{s}'$, the real biased distribution can be determined more precisely by

$$\tilde{p}(s'|b,a) = \sum_{\hat{s}\in\mathbb{I}} p(s'|\hat{s},a)\; b(\hat{s})\,. \tag{5.2.16}$$

We obtain the importance weight $u_{b,a}^{(\hat{s}'\leftarrow\hat{s})}$ by dividing the target distribution by the biased distribution so that

$$u_{b,a}^{(\hat{s}'\leftarrow\hat{s})} \sim \frac{p(\hat{s}'|\hat{s},a)}{\tilde{p}(\hat{s}'|b,a)}\,. \tag{5.2.17}$$

When using the sampled observations for approximating the expectation over all posterior beliefs, we also need to compensate their bias. In the limit, it is given by

$$\hat{o} \sim p(o|b,a) \overset{(5.2.4)}{=} \int\limits_{s'\in\mathbb{S}} p(o|s') \int\limits_{s\in\mathbb{S}} p(s'|s,a)b(s)\,\mathrm{d}s\,\mathrm{d}s'\,. \tag{5.2.18}$$

Knowing the sampling procedure and the samples sets the real biased distribution is determined by

$$\tilde{p}(o|b,a) = \sum_{\hat{s}'\in\mathbb{J}_{b,a}} p(o|\hat{s}') \sum_{\hat{s}\in\mathbb{I}} p(\hat{s}'|\hat{s},a)\; b(\hat{s}) \overset{(5.2.16)}{=} \sum_{\hat{s}'\in\mathbb{J}_{b,a}} p(o|\hat{s}')\; \tilde{p}(\hat{s}'|b,a)\,. \tag{5.2.19}$$

The weight $v_{b,a}^{(\hat{o}\leftarrow\hat{s}')}$ results to [5.5]

$$v_{b,a}^{(\hat{o}\leftarrow\hat{s}')} \sim \frac{p(\hat{o}|\hat{s}')}{\tilde{p}(\hat{o}|b,a)}\,. \tag{5.2.20}$$

Analogously to Equation 5.2.5, the weights $v_{b,a}$ and $u_{b,a}$ need to be normalized so that

$$\sum_{\hat{o}\in\mathbb{K}_{b,a}} v_{b,a}^{(\hat{o}\leftarrow\hat{s})} = 1 \qquad \text{and} \qquad \sum_{\hat{s}'\in\mathbb{J}_{b,a}} u_{b,a}^{(\hat{s}'\leftarrow\hat{s})} = 1\,. \tag{5.2.21}$$

Finally, having these weights to correct the sampling bias, we can formulate the MC α-function backup approximation of Equation 5.2.9.

[5.5] It might be difficult to comprehend for the reader that the bias $\tilde{p}(s'|b,a)$ actually appears twice, in $v_{b,a}$ and $u_{b,a}$. This is due to the fact that not only the selection of the beliefs (given by $\hat{o}$) but also the samples $\hat{s}'$ that represent these beliefs have this bias.

Algorithm 4 Point-based Monte Carlo α-function Bellman backup.

```
1:  function MC ALPHA BACKUP($b, a, \Gamma_d^{n-1}, \mathbb{I}, \mathbb{J}, \mathbb{K}$)
    ▷ compute $\alpha$ for all $\hat{s} \in \mathbb{I}$, for $a$, and point-based in $b$ with samples $\mathbb{J}, \mathbb{K}$
2:      COMPUTE BELIEF VALUES($b, a, \Gamma_d^{n-1}$)
        ▷ get $\alpha_{d,a,\hat{o}}^{n-1}[\hat{s}']$ and $V_{\alpha_d}[b'_{b,a,\hat{o}}]$ (see Algorithm 6)
3:      for all $\hat{s} \in \mathbb{I}$ do
4:          $\tilde{\alpha}_{b,a}^n[\hat{s}] \leftarrow r(s,a)$
5:          for all $\hat{o} \in \mathbb{K}$ do                              ▷ iterate next beliefs
6:              $\alpha_{d,a,\hat{o}}^{n-1} \leftarrow \arg\max V_{\alpha_d}[\tilde{b}'_{b,a,\hat{o}}]$     ▷ select best alpha for next belief
7:              for all $\hat{s}' \in \mathbb{J}$ do
8:                  $\tilde{\alpha}_{b,a}^n[\hat{s}] + \leftarrow \gamma\, u_{b,a}^{(\hat{s}' \leftarrow \hat{s})} v_{b,a}^{(\hat{o} \leftarrow \hat{s}')} \alpha_{d,a,\hat{o}}^{n-1}[\hat{s}']$
9:              end for
10:         end for
11:     end for
12:     return $\tilde{\alpha}_{b,a}^n$                  ▷ $n$th step continuous $\alpha$-function approximation
13: end function
```

The reward function $r(s,a)$ can be evaluated for any s and a without applying any form of approximation [5.6]. The nth α-function value $\tilde{\alpha}_{b,a}^n$ when executing action a for a state s is given by

$$\tilde{\alpha}_{b,a}^n(s) = r(s,a) + \gamma \sum_{\hat{s}' \in \mathbb{J}_{b,a}} u_{b,a}^{(\hat{s}' \leftarrow \hat{s})} \sum_{\hat{o} \in \mathbb{K}_{b,a}} v_{b,a}^{(\hat{o} \leftarrow \hat{s}')}\, \alpha_{b,a,\hat{o}}^{n-1}(\hat{s}')\,. \tag{5.2.22}$$

With this equation, we can approximately evaluate $\tilde{\alpha}_{b,a}^n$ for any state sample in $\hat{s} \in \mathbb{I}_b$. This sample-based approximation of α is in the following represented using the set of tuples

$$\left\{\left\langle \hat{s}, \bar{\alpha}_{\mathbb{I}_b}(\hat{s})\right\rangle : \hat{s} \in \mathbb{I}_b\right\} \qquad \text{so that} \qquad \bar{\alpha}_{\mathbb{I}_b}(s) = \begin{cases} \tilde{\alpha}_{b,a}^n & \text{if } s \in \mathbb{I}_b \\ \text{undefined} & \text{otherwise} \end{cases}. \tag{5.2.23}$$

This is sufficient to compute the Bellman recursion and, in theory, also sufficient to approximately solve the POMDP. However, in practice, the α-function recursion would have to be fully unrolled because the sample representation of the system prohibits direct application of dynamic programming (in this context, recall Figure 4.8 of the previous chapter). In the following sections, we show how machine learning can bridge this gap.

5.2.5 Avoiding Sampling Redundancy

Redundancy can have negative influence on the efficiency of the presented MC algorithms. For example, when performing resampling of particle-based beliefs, there is a chance that two current state samples are drawn which are equal: $\hat{s}_1 = \hat{s}_2$. Also next

[5.6]See Appendix A.1 for $\mathring{r}(s,a,s')$

states can be equal, if the process model joins states $\hat{s}'_1 = \hat{s}'_2$. The problem gets even more evident for POMDPs with discrete observations. Due to the finite number of discrete observations, there is a high chance, that the two equal observation samples are drawn $\hat{o}_1 = \hat{o}_2$.

Iterating the (equal) samples separately during value iteration slows down computation significantly. This problem can be mitigated by constructing *distinct* sample sets. The importance weights then must be adapted because the biased distribution is effectively changed. When, for example, R equal observations are merged to one observation o_m, the corresponding weights must be multiplied by R. For instance for Equation 5.2.5, we yield a modified likelihood weight

$$\bar{w}^{(\hat{s}')}_{b,a,o} = p(o_m|\hat{s}')R \; . \tag{5.2.24}$$

5.2.6 Continuous Observation Spaces

When dealing with continuous observations, another performance problem arises. In contrast to domains with discrete observation space, there can be an infinite number of continuous observations. We cover this by sampling a finite set of continuous observations $\hat{o} \in \mathbb{K}$ during the MC belief prediction. The number of (distinct) observations $\hat{o}$ and corresponding beliefs $b'_{b,a,\hat{o}}$ that has to be sampled for good results can be very high, however. This can affect performance of the MC α-function backup in Equation 5.2.22 because an excessive number of best α-functions $\alpha_{b,a,\hat{o}}$ for the next beliefs has to be processed.

We prevent this by applying an idea presented in [Hoey and Poupart, 2005] for discrete POMDPs. The linearity of the backup allows to cluster all observations $\hat{o}$ that result in beliefs, which are dominated by the same α-function in Equation 5.2.10 without affecting the result

$$\mathbb{K}_\alpha = \left\{ \hat{o} : \; \alpha = \underset{\alpha \in \Gamma^{n-1}}{\arg\max} \left\langle \tilde{b}'_{b,a,\hat{o}}, \alpha \right\rangle \right\} \; . \tag{5.2.25}$$

The MC α-function backup can then be rewritten as follows

$$\tilde{\alpha}^n_{b,a}(s) = r(s,a) + \gamma \sum_{\hat{s}' \in \mathbb{J}_{b,a}} u^{(\hat{s}' \leftarrow \hat{s})}_{b,a} \sum_{\alpha \in \Gamma^{n-1}} v^{(\mathbb{K}_\alpha \leftarrow \hat{s}')}_{b,a} \, \alpha(\hat{s}') \, , \tag{5.2.26}$$

$$\text{where } v^{(\mathbb{K}_\alpha \leftarrow \hat{s}')}_{b,a} = \sum_{\hat{o} \in \mathbb{K}_{b,a}} v^{(\hat{o} \leftarrow \hat{s}')}_{b,a} \; . \tag{5.2.27}$$

In comparison to Equation 5.2.22, the sum over the sampled observations was replaced by a sum over (usually much fewer) α-functions. The same can be done for the belief value Bellman backup.

5.3 Discrete Representation of Continuous Space

In Section 5.3.1, we first give a general definition of the *discrete representation* concept. In Section 5.3.2, we then formulate the central assumption that there exists a discrete α-function representation and derive a discrete representation of continuous beliefs. Having a discrete representation of both, beliefs and α-functions, we can perform continuous value iteration in the discrete space. The continuous integrals in the continuous expectation operator can be efficiently computed by evaluating a finite dot-product, analogously to discrete POMDPs.

These general results can be specified for sample-based representations of continuous distributions and functions: we propose efficient algorithms to discretize particle-based beliefs (see Section 5.3.3) and sample-based continuous α-functions (see Section 5.3.4).

No specific assumptions about the implementation of the representation are made in this section, yet. However, it is assumed to be known and fix. In the following sections we elaborate on how the representation can be implemented and iteratively learned.

5.3.1 Definition of a discrete representation

We define a *discrete representation* as tuple of a set of discrete states $\mathbb{S}_d$ and a function θ

$$\langle \mathbb{S}_d, \theta \rangle \qquad \text{with} \qquad \theta : \mathbb{S} \times \mathbb{S}_d \to \mathbb{R} \,. \tag{5.3.1}$$

The discrete state set is w.l.o.g. chosen to be a subset of the natural numbers with size M that numbers the discrete states

$$s_d \in \mathbb{S}_d \subset \mathbb{N} \qquad \text{and} \qquad \mathbb{S}_d = \{1, ..., M\} \,. \tag{5.3.2}$$

The states $s_d = 1, 2, \ldots, M$ remain abstract numbers until a *meaning* is assigned to them in the continuous state space $\mathbb{S}$. Therefore, the function θ links the continuous states $s \in \mathbb{S}$ of the continuous POMDP with the discrete state $s_d \in \mathbb{S}_d$. Note that in contrast to previous work, the codomain of θ is the real numbers including positive-, negative- and zero-values.

The mapping between the continuous and discrete states does not have to be unique. On the one hand, a discrete state can be linked with several continuous states. Thus, it is allowed that for two continuous states

$$s_1, s_2 \in \mathbb{S} : \; s_1 \neq s_2 \;\wedge\; (\theta(s_d, s_1) \neq 0) \;\wedge\; (\theta(s_d, s_2) \neq 0) \,. \tag{5.3.3}$$

This is an essential capability for generalization over the continuous space. On the other hand, a continuous state s can be represented by a combination of discrete states so that for two discrete states it is allowed that

$$s_{d,1}, s_{d,2} \in \mathbb{S}_d : \; s_{d,1} \neq s_{d,2} \;\wedge\; (\theta(s_{d,1}, s) \neq 0) \;\wedge\; (\theta(s_{d,2}, s) \neq 0)\,. \tag{5.3.4}$$

Next, a more specific meaning in the context of POMDPs will be assigned to θ .

5.3.2 Assumptions and Discrete Backup Derivation

One of the core operations of the Bellman backup in Equation 5.2.9 is the evaluation of $n-1$th step α-functions in order to assess the expected future value. This computation is involved in the most often repeated computations and thus has major impact on the performance of value iteration. Additionally, for efficient dynamic programming, $\alpha(s)$ must be defined for every state $s \in \mathbb{S}$. Therefore, the sparsely computed α-function backups (see Section 5.2.4) must be generalized over the continuous space $\mathbb{S}$.

With these goals in mind, we design the discrete representation. We replace the continuous expectation operator $\langle b, \alpha \rangle$, which requires integrating over the state space $\mathbb{S}$, by a finite computation using solely the discrete representation. We show that, if the discrete computation accurately resembles the continuous expectation operator, convergence of value iteration to the optimal value is provably maintained.

In order to avoid errors in the policy representation, α-functions should be accurately represented using the discrete representation.

Therefore, we assume that the continuous function α is exactly[5.7] represented by the function α_d and a representation θ, both defined on the discrete space $\mathbb{S}_d$ (see Section 5.3.1) so that

$$\alpha(s) = \sum_{s_d \in \mathbb{S}_d} \theta(s, s_d)\alpha_d(s_d) \tag{5.3.5}$$

$$\text{with } \; \alpha_d : \mathbb{S}_d \to \mathbb{R}\,. \tag{5.3.6}$$

Th function α_d can be interpreted as vector. In other words, continuous α-functions in $s \in \mathbb{S}$ are represented by a linear combination of M basis functions $\theta(s,1), \ldots, \theta(s,M)$ (i.e., components) weighted by vectors $\alpha_d(1), \ldots, \alpha_d(M)$.

[5.7]In later sections, we relax this assumption.

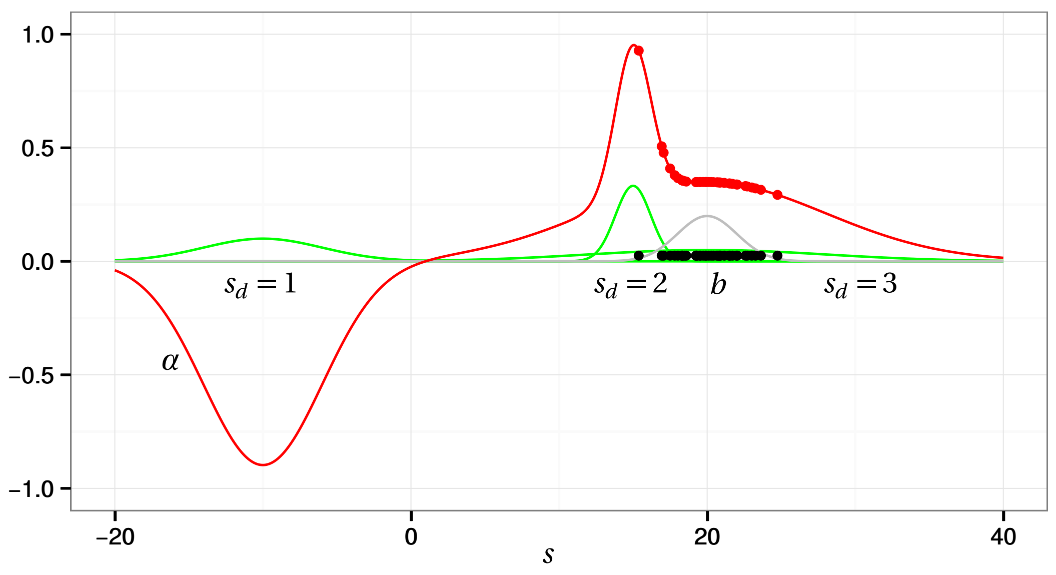


Figure 5.3: Continuous α-function (red) represented by three Gaussian distributions $s_d = 1, 2, 3$ (green). The α-vector is $\alpha_d = (-9, 2, 7)^\top$. The belief b (black) is transformed to the discrete vector $\beta \approx (0.0, 0.017, 0.048)^\top$ (i.e., there is negligible overlap with $s_d = 1$, the left part of b is associated with $s_d = 2$, and the center and right part with $s_d = 3$). The final result of $\langle b, \alpha \rangle = \left\langle \beta, \alpha_d \right\rangle_d \approx 0.37$. The α-function values $\alpha(\hat{s})$ (red dots) that sampled states $\hat{s}$ (black dots) of the belief b yield are also shown.

To improve readability, we write

$$\alpha(s) = \sum_{s_d \in \mathbb{S}_d} \theta_{s_d}(s) \alpha_d(s_d) \tag{5.3.7}$$

$$\text{with } \theta(s, s_d) := \theta_{s_d}(s)\,, \tag{5.3.8}$$

where θ_{s_d} explicitly denotes the s_dth basis function.

In Figure 5.3 an exemplary α-function represented as a linear combination of Gaussian distributions is shown. Note, however, that the basis functions are not restricted to distributions. Figure 5.4 demonstrates the flexibility of this representation by using different types of basis functions to represent an α-function.

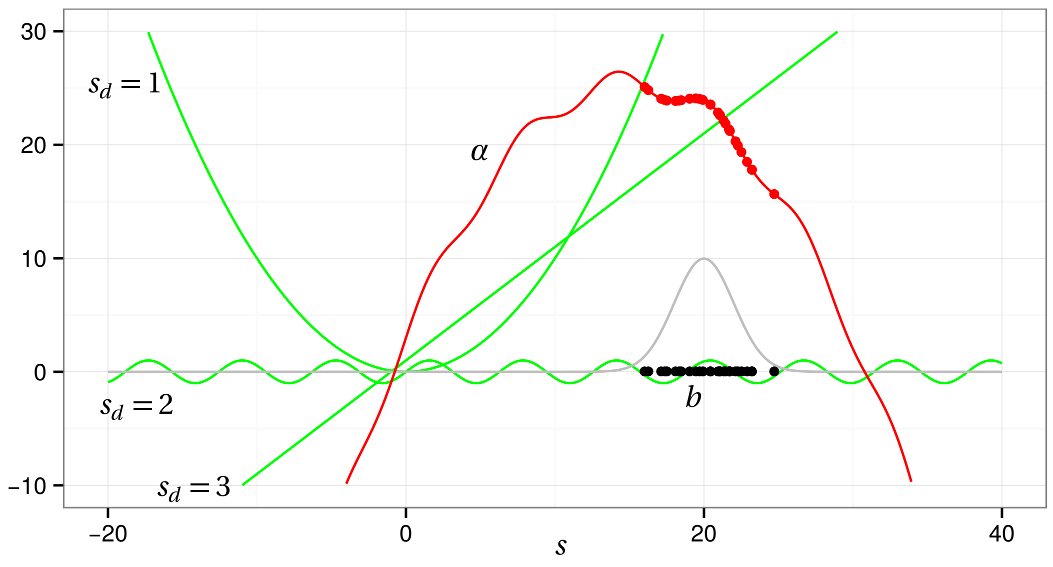


Figure 5.4: Continuous α-function (red) represented by a quadratic function ($s_d = 1$), a sine function ($s_d = 2$) and a linear function ($s_d = 3$). The basis functions are indicated in green and weighted by $\alpha_d = (-1, 1, 3)^\mathsf{T}$. The continuous belief is transformed to $\beta \approx (40.4, 0.1, 20.9)^\mathsf{T}$. The final result of $\langle b, \alpha\rangle = \langle \beta, \alpha_d\rangle_d \approx -216$. The α-function values $\alpha(\hat{s})$ (red dots) that the sampled states $\hat{s}$ (black dots) of the belief b yield are also shown.

The following Lemma 1 guarantees that the convergence of value iteration is not affected by the discretization.

Lemma 1. *Let α_d represent α (see Equation 5.3.7). The continuous expectation operator can be reduced to a dot product of two discrete vectors.*

Proof.

$$\langle b, \alpha\rangle = \int_{s\in\mathbb{S}} b(s)\alpha(s)\,\mathrm{d}s \tag{5.3.9}$$

$$\overset{(5.3.7)}{=} \int_{s\in\mathbb{S}} \sum_{s_d\in\mathbb{S}_d} \theta_{s_d}(s)\, b(s)\, \alpha_d(s_d)\,\mathrm{d}s \tag{5.3.10}$$

$$= \sum_{s_d\in\mathbb{S}_d} \alpha_d(s_d) \underbrace{\int_{s\in\mathbb{S}} \theta_{s_d}(s) b(s)\,\mathrm{d}s}_{:=\beta(s_d)} \tag{5.3.11}$$

$$= \sum_{s_d\in\mathbb{S}_d} \alpha_d(s_d)\beta(s_d) = \langle \beta, \alpha_d\rangle_d \tag{5.3.12}$$

□

We conclude that if the discrete vector β is chosen as follows, it poses an efficient discrete representation of the continuous belief b that allows to compute the expectation operator without loss of accuracy.[5.8] The discrete representation β of the continuous belief b is given by

$$\beta : \mathbb{S}_d \to \mathbb{R} \qquad \text{and} \qquad \beta(s_d) = \int\limits_{s \in \mathbb{S}} \theta_{s_d}(s) b(s)\, \mathrm{d}s\,. \tag{5.3.13}$$

Lemma 1 is not trivial because, in general, the vector β does not hold enough information to reconstruct the original continuous belief without loss. This is intended in order to neglect solution irrelevant features of the belief.

If the discrete vectors α_d and β are precomputed, the continuous integral can be transformed into a simple discrete dot-product, which can be evaluated efficiently

$$\langle b, \alpha \rangle = \left\langle \beta, \alpha_d \right\rangle_d\,. \tag{5.3.14}$$

This result indirectly implies that a continuous state $s \in \mathbb{S}$ is also represented by β_s defined on the discrete space

$$\beta_s : \mathbb{S}_d \to \mathbb{R} \qquad \text{and} \qquad \beta_s(s_d) = \theta_{s_d}(s)\,. \tag{5.3.15}$$

The function β_s can be interpreted as a vector representation of the continuous belief. Then, the α-function value of the state s for the discrete α-function representation α_d can be evaluated by

$$\alpha(s) = \left\langle \beta_s, \alpha_d \right\rangle_d\,, \tag{5.3.16}$$

analogously to the discrete dot-product of α-vectors and belief vectors in discrete POMDPs. With the above, a continuous POMDP policy can be represented by a set of discrete α-vectors

$$\alpha_d \in \Gamma_d^n\,. \tag{5.3.17}$$

This discretization can be used to efficiently implement the α-function Bellman recursion. Under the assumptions that the representation of the continuous α-functions is *lossless* and the integral that determines β can be computed exactly, convergence of the Bellman recursion is not violated, as shown in [Porta et al., 2006]. The α-function recursion then is an isotonic contraction that provably converges to the optimal value.

Finding an efficient representation that realizes Equation 5.3.7 without any or only small errors is not easy. However, highly related problems have been covered by a vast amount of research in the area of machine learning. This topic is covered in Section 5.4.

[5.8] This result is similar to the result in [Porta et al., 2006]. However, we did not restrict $b(s)$ to a specific representation. Further, we did not assume the function θ to be normed for $s \in \mathbb{S}$ (see Figure 5.4 for an example with such basis functions).

Algorithm 5 Discretization of continuous states and beliefs.

```
Require: $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$                          ▷ see Algorithm 1
Require: Discrete Representation $\langle \mathbb{S}_d, \theta \rangle$
 1: function DISCRETIZE STATES($b, a$)
 2:     for all $\hat{s}' \in \mathbb{J}_{b,a}$ do
 3:         store $\forall s_d : \beta_{\hat{s}'}[s_d] \leftarrow \theta_{s_d}(\hat{s}')$          ▷ following Equation 5.3.15
 4:     end for
 5: end function
 6: function DISCRETIZE BELIEFS($b, a$)
 7:     for all $\hat{o} \in \mathbb{K}_{b,a}$ do
 8:         $\tilde{\beta}'_{b,a,\hat{o}} \leftarrow$ SPARSE ZERO VECTOR()
 9:         for all $\hat{s}' \in \mathbb{J}_{b,a}$ do
10:             $\beta_{\tilde{b}'_{b,a,\hat{o}}} + \leftarrow w^{(\hat{s}')}_{b,a,o} \cdot \beta_{\hat{s}'}$       ▷ discrete sum following Equation 5.3.23
11:         end for
12:         store $\tilde{\beta}'_{b,a,\hat{o}}$
13:     end for
14: end function
```

5.3.3 Discretization of Particle-based Beliefs

Due to the MC approach presented in Section 5.2, beliefs (in particular the next beliefs b') are represented by particles. Let $\langle \mathbb{S}_d, \theta \rangle$ be a discrete representation and $\tilde{b}(s) \overset{(5.2.6)}{=} \sum_{\hat{s}\in\mathbb{I}} w_b^{(\hat{s})} \delta(s-\hat{s})$ an arbitrary particle-based belief. The discretization of $\tilde{b}(s)$ can be carried out using the sifting property of the Dirac deltas

$$\tilde{\beta}(s_d) \overset{(5.3.13)}{=} \int\limits_{s\in\mathbb{S}} \theta_{s_d}(s)\tilde{b}(s)\,\mathrm{d}s \tag{5.3.18}$$

$$= \int\limits_{s\in\mathbb{S}} \theta_{s_d}(s) \sum_{\hat{s}\in\mathbb{I}_b} w_{\tilde{b}}^{(\hat{s})} \delta(s-\hat{s})\,\mathrm{d}s \tag{5.3.19}$$

$$= \sum_{\hat{s}\in\mathbb{I}} w_{\tilde{b}}^{(\hat{s})} \int\limits_{s\in\mathbb{S}} \theta_{s_d}(s)\delta(s-\hat{s})\,\mathrm{d}s \tag{5.3.20}$$

$$= \sum_{\hat{s}\in\mathbb{I}} w_{\tilde{b}}^{(\hat{s})} \theta_{s_d}(\hat{s}) \tag{5.3.21}$$

$$\overset{(5.3.15)}{=} \sum_{\hat{s}\in\mathbb{I}} w_b^{(\hat{s})} \beta_{\hat{s}}(s_d)\,. \tag{5.3.22}$$

It is now established that $\tilde{\beta}$ can be composed as sum of vectors $\beta_{\hat{s}}$, which pose a discrete representation of the particle-based continuous belief representation $\tilde{b}$

$$\tilde{\beta} = \sum_{\hat{s}\in\mathbb{I}} w_b^{(\hat{s})} \beta_{\hat{s}}\,. \tag{5.3.23}$$

Consequently, every sampled next state $\hat{s}'$ needs to be discretized exactly once. This becomes especially beneficial in conjunction with the way our solver executes belief

Algorithm 6 Compute step $n-1$ α-functions using the discrete representation.

```
Require: samples $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$                    ▷ see Algorithm 1
 1: function COMPUTE ALPHA VALUES($b, a, \Gamma_d^{n-1}$)                    ▷ compute $\alpha(\hat{s}')$
 2:     DISCRETIZE STATES($b, a$)                                        ▷ get $\beta_{\hat{s}'}$
 3:     for all $\alpha_d \in \Gamma_d^{n-1}$ do                                ▷ iterate $\alpha$-vectors
 4:         for all $\hat{s}' \in \mathbb{J}_{b,a}$ do
 5:             store $\alpha_d[\hat{s}'] \leftarrow \langle \beta_{\hat{s}'}, \alpha_d \rangle_d$            ▷ following Equation 5.3.16
 6:         end for
 7:     end for
 8: end function
 9: function COMPUTE BELIEF VALUES($b, a, \Gamma_d^{n-1}$)
10:     COMPUTE ALPHA VALUES($b, a, \Gamma_d^{n-1}$)                        ▷ get $\alpha_d[\hat{s}']$
11:     for all $\hat{o} \in \mathbb{K}_{b,a}$ do                                ▷ iterate beliefs
12:         for all $\alpha_d \in \Gamma_d^{n-1}$ do                            ▷ iterate $\alpha$-vectors
13:             $V_{\alpha_d}[\tilde{b}'_{b,a,\hat{o}}] \leftarrow 0$
14:             for all $\hat{s}' \in \mathbb{J}_{b,a}$ do
15:                 $V_{\alpha_d}[\tilde{b}'_{b,a,\hat{o}}] + \leftarrow \alpha_d[\hat{s}']$
16:             end for
17:             store $V_{\alpha_d}[\tilde{b}'_{b,a,\hat{o}}]$
18:         end for
19:     end for
20: end function
```

prediction. All next beliefs $\tilde{b}'_{b,a,\hat{o}}$ (see Equation 5.2.6) share the sample sets $\mathbb{J}_{b,a}$ and are differentiated just by their weights. This enables us to discretize all next beliefs with minimal computational cost.

Algorithm 5 shows how to decompose discretization into two steps. First, the states samples $\hat{s}'$ are discretized independently from their belief-specific weights $w^{(\hat{s}')}_{b,a,o}$. Then, the discrete belief representations for all next beliefs $\tilde{\beta}'_{b,a,\hat{o}}$ are constructed independently from the continuous state samples $\hat{s}'$, using the discrete state representations $\beta_{\hat{s}'}$.

5.3.4 Discrete Evaluation of α-functions

To compute a point-based α-function backup for a belief b, the α-function values for all its sampled successor beliefs $b'_{b,a,\hat{o}}$ need to be computed (see the MC α-function backup in Section 5.2.4). Since the observation space $\mathbb{O}$ is continuous, the number of sampled observations $\hat{o}$ can be large and the computational effort of a naive approach immense. Our discretization approach mitigates this bottleneck by discretizing the state samples $\hat{s}' \in \mathbb{J}_{b,a}$ only a single time. The remaining computations are carried out in the discrete space.

Algorithm 6 shows how to evaluate the α-functions for the predicted states $\hat{s}'$ and the next beliefs $\tilde{b}'_{b,a,\hat{o}}$. It uses the discrete representation of the predicted states $\beta_{\hat{s}'}$ and the discrete α_d vector representation. The values $V_{\alpha_d}[\tilde{b}'_{b,a,\hat{o}}]$ and $\alpha_d[\hat{s}']$ computed in this algorithm are a prerequisite for the MC belief backup in Algorithm 3 and α-function backup in Algorithm 4.

5.4 Iterative Representation Learning

There are several ways to realize the discrete representation in order to make use of the theoretical results in Section 5.3.2 and the derived algorithms.

Two fundamentally different ways of implementing a representation can be distinguished: one way is to find several *local* representations $\langle \mathbb{S}_d, \theta \rangle_\alpha$ that are unique for every single α-function. The other way is to use a *global* representation $\langle \mathbb{S}_d, \theta \rangle$ shared by all α-functions.

We decided for the latter due to the potentially improved computational efficiency. See Appendix A.2 for a discussion of this topic with a complexity analysis.

First, we define the requirements for the representation:

- We call a discrete representation $\langle \mathbb{S}_d, \theta \rangle$ *lossless* with respect to α, if

$$\exists \alpha_d : \alpha(s) \overset{(5.3.7)}{=} \sum_{s_d \in \mathbb{S}_d} \theta_{s_d}(s) \alpha_d(s_d) \,. \tag{5.4.1}$$

For the sake of efficiency, the representation should also be as compact as possible:

- A lossless representation $\langle \mathbb{S}_d, \theta \rangle$ of α is *optimal*, if there exists no lossless representation $\langle \mathbb{S}_d, \theta \rangle'$ of α using less discrete states.

These definitions can be transferred from plans, expressed by α-functions, to policies, expressed by sets Γ of α-functions.

- $\langle \mathbb{S}_d, \theta \rangle$ is lossless with respect to Γ, if it is lossless with respect to every $\alpha \in \Gamma$.
- $\langle \mathbb{S}_d, \theta \rangle$ is optimal with respect to Γ, if there is no lossless representation $\langle \mathbb{S}_d, \theta \rangle'$ of Γ that uses less discrete states.

The ultimate goal is to find a representation for the optimal value function given by Γ^*.

- We call $\langle \mathbb{S}_d, \theta \rangle^*$ an *optimal representation* of a POMDP, if it is optimal for all $\alpha \in \Gamma^*$.

In practice, this goal can only be approximately fulfilled (similar to finding only an approximately optimal value function). Nevertheless, it is clear, that the discrete representation should be compact enough to keep the Bellman backup computations feasible but at the same time introduce as little errors as possible.

Finding a representation $\langle \mathbb{S}_d, \theta \rangle$ for the continuous α-functions is a challenging task (see Equation 5.3.7) . With the presented MC backup algorithm, continuous α-functions are only evaluated for a finite set of sampled states $\hat{s}$ (see Equation 5.2.22). This is the basis for learning the representation.

If the basis functions θ_{s_d} of the representation are fixed, finding α_d for α can be cast as a linear regression problem. The main difficulty is to find a suitable set of basis functions. If the parametric family (e.g., normal distributions) of the basis functions is fixed, parameters for the basis functions can be found by maximum-likelihood estimation (MLE).

Nevertheless, the α-functions of the optimal value are obviously not known prior to optimizing the value function. Consequently, the optimal representation cannot be found a priori. Instead, we propose to incrementally update and refine it during value iteration to be able to represent new α-functions (i.e., plans).

5.4.1 Iterative Approach

The iterative process of learning an approximately optimal representation $\langle \mathbb{S}_d, \theta \rangle$ of a POMDP requires to constantly update it. Therefore, it must satisfy to represent old ($< n-1$th step) and new (nth step) α-functions.

The index $m = 1, 2, \ldots$ indicates the version of the representation $\langle \mathbb{S}_d, \theta \rangle^m$. Every backup operation creates a new function α^n that can imply the necessity for a more expressive $\langle \mathbb{S}_d, \theta \rangle^{m+1}$.

It is important to not invalidate discrete vector representations of continuous α-functions that have been generated before the representation update.

Therefore, we choose to only add new discrete states s_d to $\mathbb{S}_d$ and according basis functions θ_{s_d} to θ. We never remove or alter existing basis functions.

Let α_d^m be a discrete vector using the representation $\langle \mathbb{S}_d, \theta \rangle^m$ that has M discrete states and basis functions. The coefficients of the vector $\alpha_d(s_d)$ of not used functions θ_{s_d} are zero. Therefore, $\alpha_d^m(s_d) = 0$ if $s_d > M$. For this reason, the continuous function $\alpha(s)$ that α_d defines through Equation 5.3.7 remains untouched. $\alpha(s)$ in this case is given by

$$\alpha(s) \overset{(5.3.7)}{=} \sum_{s_d \in \mathbb{S}_d} \theta_{s_d}(s) \alpha_d(s_d) = \sum_{s_d=1}^{M} \theta_{s_d}(s) \alpha_d(s_d) + \theta_{M+1} \cdot 0 + \theta_{M+2} \cdot 0 + \ldots \,. \tag{5.4.2}$$

This concept can be efficiently implemented using *sparse* vector representations of α_d, where every coefficient that is not explicitly set to be non-zero is assumed to be zero.

5.4.2 State Space Partitioning

The problem of finding and especially refining a representation of a policy can be simplified by not allowing two discrete states to be associated with one and the same continuous state. With this restriction, discrete regions in the continuous state space are explicitly separated. Every continuous state is uniquely mapped to a discrete state. Such a representation can be incrementally refined by dividing regions into smaller subregions. This choice has performance advantages because to evaluate a continuous α-function for a state s, only a single state in the discrete vector α_d has to be looked up.

We let the representation $\langle \mathbb{S}_d, \theta^c \rangle$ express a *partition* of the state space.

The representation $\langle \mathbb{S}_d, \theta^c \rangle$ is determined by a *mapping* c from continuous states $s \in \mathbb{S}$ to discrete states $s_d \in \mathbb{S}_d$

$$c : \mathbb{S} \rightarrow \mathbb{S}_d, \qquad \text{and} \qquad \theta^c_{s_d}(s) = \begin{cases} 1 & \text{if } c(s) = s_d \\ 0 & \text{otherwise} \end{cases} . \tag{5.4.3}$$

The function c induces a partition of the continuous space. In contrast to our general definition of θ, we have

$$\nexists s \in \mathbb{S}, s_{d,1} \neq s_{d,2} : (\theta^c_{s_{d,1}}(s) > 0) \wedge (\theta^c_{s_{d,2}}(s) > 0) . \tag{5.4.4}$$

The goal is to find a mapping c that accurately represents the continuous α-functions in Γ. This is the case, if all continuous states with different α-function values are mapped to different discrete states.

- The mapping c implicates a *lossless* representation θ^c of the α-function, if

$$\forall s_1, s_2 \in \mathbb{S} : \alpha(s_1) \neq \alpha(s_2) \Rightarrow c(s_1) \neq c(s_2) . \tag{5.4.5}$$

For computational efficiency, it is preferable to obtain a compact representation using as little discrete states as possible. An *optimal* partition θ^c realizes the intuitive idea to differentiate states only, if this differentiation is relevant for the agent's plan.

- A lossless mapping c defines an optimal representation $\langle \mathbb{S}_d, \theta^c \rangle$ with respect to the function α, if

$$\forall s_1, s_2 \in \mathbb{S} : \alpha(s_1) = \alpha(s_2) \Rightarrow c(s_1) = c(s_2) . \tag{5.4.6}$$

Both statements can be easily proven by giving counterexamples.

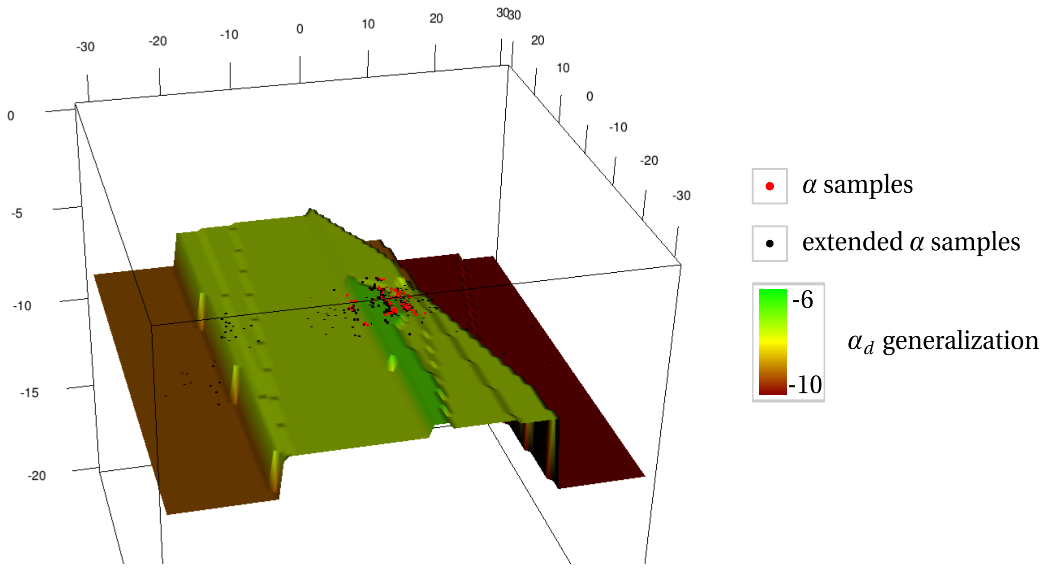


Figure 5.5: 2D α-function generalization and conflict resolution. z-axis and colors indicate the α-function value. The graph shows the generalization of the sample-based continuous α-function induced by its discrete representation α_d.

With the restriction to a partition and if $\langle \mathbb{S}_d, \theta^c \rangle$ is lossless with respect to α, we can simplify the α-function representation

$$\alpha(s) \overset{(5.3.7)}{=} \sum_{s_d \in \mathbb{S}_d} \theta^c_{s_d}(s)\alpha_d(s_d) = \alpha_d(c(s)) \,. \tag{5.4.7}$$

The α-function value of every continuous state s now only depends on a single discrete state s_d. Vice versa, the discrete representation α_d of a continuous α-function is directly determined by $\alpha_d(c(s)) = \alpha(s)$.

Figure 5.5 shows an example of a 2D continuous α-function represented by a partition. The dots indicate the continuous samples $\tilde{\alpha}(\hat{s})$ that the representation aims to represent.[5.9] In this example, the x_2 dimension does not have much influence on the value. The learned partition accounts for that: the regions are mostly aligned with the x_2-axis. The 2D problem is effectively reduced to a 1D problem.

5.4.3 Decision Tree-based Partitioning

Partitioning can be efficiently implemented using a decision tree.[5.10] Discontinuities in the α-functions (e.g., due to collisions in motion planning tasks) then can be directly modeled by *tests* in the inner nodes of the tree.

[5.9]See Section 5.5.3 for the meaning of the extended α samples.

[5.10]Here, the data mining or machine learning definition of the term *decision tree* is used: the tree describes data and not decisions. This is not to be confused with decision trees used in decision-making.

We choose to implement the mapping between discrete and continuous states using a decision tree.

The tree T consists of tree *nodes*. These are denoted by their corresponding discrete state $s_d \in \mathbb{S}_d$. Every non-leaf node s_d implies a *test* t_{s_d} that maps states s to an *outcome* c_i that points to the child node

$$t_{s_d} : \mathbb{S} \rightarrow \{c_1, c_2, \ldots\} \ . \tag{5.4.8}$$

This test can be used to divide a set of states $\hat{s} \in \mathbb{I}$ into subsets $\mathbb{I}_{c_1}, \mathbb{I}_{c_2}, \ldots$, one for every outcome. We constrain the number of outcomes to two (i.e., binary trees).[5.11]

The tree realizes the iterative representation growing approach explained in Section 5.4.1. In this context, the tree nodes can be viewed as basis functions. In order to create a discrete vector representation $\tilde{\beta}$ of a particle-based belief $\tilde{b}$, the tree needs to be traversed once for every particle of the belief. This discrete representation is valid for evaluating the values for all older α_d: when the tree grows, the previous state representations remain in the tree as inner, non-leaf nodes. The set of all leaf nodes of the tree represents a mapping c that induces a partition. When learning a discrete α-function representation α_d, only leaf nodes are assigned non-zero values.

In Figure 5.6, we give a detailed example of this decision tree-based partitioning. To determine the discrete representation β_s of a continuous state s, one has to descend from the root to the leaf according to the outcome of the tests in the nodes. The red arrows in Figure 5.6a indicate this path for an exemplary s

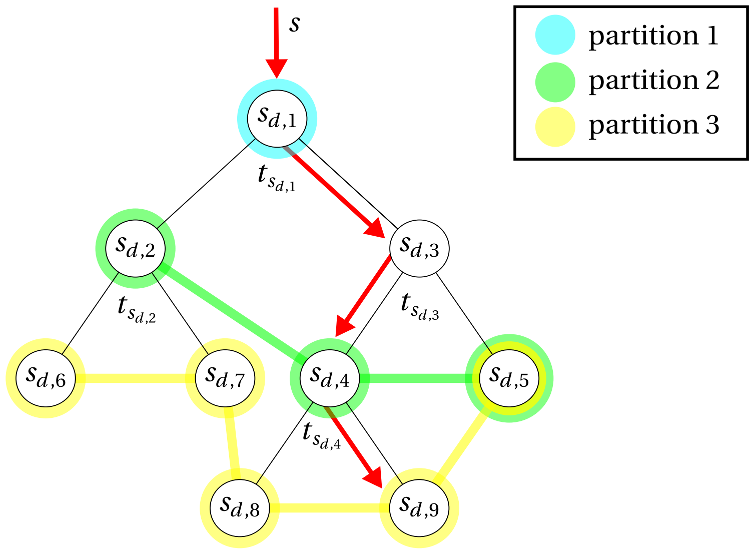
$$t_{s_{d,1}}(s) = c_2 \rightarrow \qquad t_{s_{d,3}}(s) = c_1 \rightarrow \qquad t_{s_{d,4}}(s) = c_2 \rightarrow \qquad t_{s_{d,9}} \ , \tag{5.4.9}$$

where c_1, c_2 are the possible outcomes of the tests. Three partitions are highlighted in the figure. The initial partition (blue) contains only the root node. The mapping containing $s_{d,2}$, $s_{d,4}$, and $s_{d,5}$ (green) has intermediate refinement level. The partition with the highest refinement contains all leaf nodes (yellow).

In Figure 5.6b, we demonstrate how the representation is used to compute continuous α-function values. As all vectors α_d use a partition of the continuous space, the computation always comes down to looking up a single $\alpha_d(s_d)$ value. Figure 5.7 shows how an exemplary 2D continuous state space is partitioned by a tree using linear functions for the tests. The set of all leafs poses a partition of the continuous space.

Next, we elaborate on how the decision tree can be learned with the goal to optimally represent α-functions.

[5.11] In combination with the α-function definition for partitions in Equation 5.4.7, this choice is equivalent to representing α-functions with *regression trees* [Breiman et al., 1984] (i.e., decision trees with continuous target variable) with constant basis functions.



(a) Decision tree with three different partition versions that are indicated by the connected colors. The nodes are indexed by discrete states s_d. In every node a test is performed to determine the next child node.

$$\beta_s = \begin{pmatrix} \theta^c_{s_{d,1}}(s) \\ \theta^c_{s_{d,2}}(s) \\ \theta^c_{s_{d,3}}(s) \\ \theta^c_{s_{d,4}}(s) \\ \theta^c_{s_{d,5}}(s) \\ \theta^c_{s_{d,6}}(s) \\ \theta^c_{s_{d,7}}(s) \\ \theta^c_{s_{d,8}}(s) \\ \theta^c_{s_{d,9}}(s) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \alpha_{d,1} = \begin{pmatrix} -20 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}, \quad \alpha_{d,2} = \begin{pmatrix} \cdot \\ -20 \\ \cdot \\ 5 \\ 3 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}, \quad \alpha_{d,3} = \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ 4 \\ -3 \\ -5 \\ 8 \\ 10 \end{pmatrix}$$

$$\begin{aligned} \langle \beta_s, \alpha_{d,1} \rangle_d &= 1 \cdot (-20) \\ \langle \beta_s, \alpha_{d,2} \rangle_d &= 0 \cdot (-20) + 1 \cdot 5 + 0 \cdot 3 \\ \langle \beta_s, \alpha_{d,3} \rangle_d &= 0 \cdot 4 + 0 \cdot (-3) + 0 \cdot (-5) + 0 \cdot 8 + 1 \cdot 10 \end{aligned}$$

(b) Discrete α-vectors defined by partitions. The "$\cdot$"s denote sparse zeros. $\alpha_{d,1}, \alpha_{d,2}, \alpha_{d,3}$ were created with the corresponding partitions. The values $\alpha_1(s), \alpha_2(s)$, and $\alpha_3(s)$ are computed using the discrete α-vectors and the discrete state representation β_s of the continuous state s.

Figure 5.6: Example of decision tree representation and α-function evaluation.

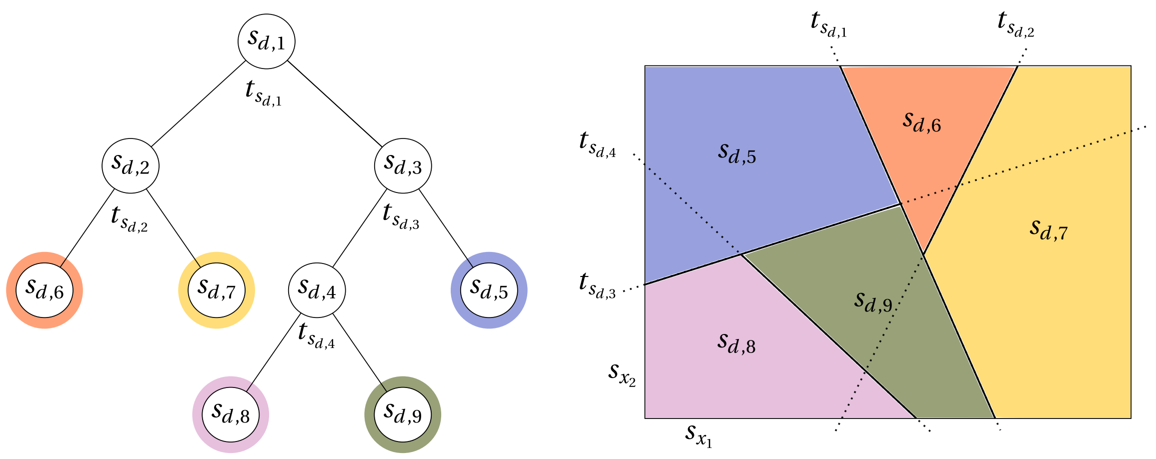


Figure 5.7: 2D example for a partition induced by a decision tree. The continuous state space $\mathbb{S}$ has dimensions s_{x_1} and s_{x_2}. The linear tests t_{s_d} recursively split and refine the continuous space representation.

5.4.4 Relationship to Decision Tree Learning for Predictive Modeling

Decision tree analysis for data prediction has a long history in data mining and machine learning. We first give a brief overview of the general problem and methods and then address the specific problem in this work and identify the differences.

The basic task is to find a *model* for a training set of data samples with *input attributes* that predicts the *target variable* with minimal error. A decision tree can be used to structure the data: starting from the root node, the tree is descended by applying *tests* to the input attributes in every node that determine the child node. The resulting leaf finally assigns a class to the data sample.[5.12] The notion of decision trees subsumes *classification trees* with discrete target variables and *regression trees* with numerical target variables.

Decision trees are usually learned for a training set of input attributes and according target variables. In *incremental learning* the complete training data is not given a priori, but arrives step by step and the tree must be updated incrementally. Top-down induction is a well known greedy strategy for learning decision trees. The underlying idea is, to recursively split the training data according to some heuristic (see [Rokach and Maimon, 2005] for an overview of existing methods). Famous examples following this top-down strategy include CART [Breiman et al., 1984], ID3 [Quinlan, 1986], and C4.5 [Quinlan, 1993]. Several improvements over these basic algorithms have been proposed. Besides modifications of the splitting heuristics, extensions to the basic top-down approach mostly aim to reduce *overfitting*. Examples are *pruning* the tree [Esposito et al., 1997], combining several trees in an *ensemble* using, for instance, *bagging* [Breiman, 1996], *boosting* [Freund et al., 1999; Schapire, 2003], and various random-

[5.12] in case of *crisp classifiers*

ization techniques [Ho, 1998; Dietterich, 2000]. These methods can also be combined. For example Random Forests [Breiman, 2001] are a well-known and very successful combination of bagging and randomization.

For the task of representation learning in this work, we use decision trees in a slightly different manner than usual. In our case, the input attributes are the continuous states $s \in \mathbb{S}$. The target attributes are the $\alpha(s)$ values. The sample-based MC approximation of the α-function Bellman backup provides the training data. In contrast to the usual formulation, we aim to find a single tree that expresses several different α-functions using different coefficients α_d. Additionally, in our case, since the training data is generated incrementally with each backup, the partition should only be refined such that the representation of previous α-functions is not invalidated. This prohibits direct application of more sophisticated decision tree learning algorithms (e.g., pruning). Note that our application is not prone to classic overfitting because the α-function Bellman backup is a deterministic function. Small errors that can induce minor overfitting are mainly introduced by the MC nature of the Bellman backup.

5.4.5 Loss Function for Representation Learning

In general, the assumption that the representation $\langle \mathbb{S}_d, \theta \rangle$ is lossless cannot be fulfilled while at the same time maintaining sensible performance. In practice, a tradeoff between compactness and accuracy has to be found.

In the optimal case, where θ^c is defined by a lossless partition, the discrete α-function representation α_d is directly determined by $\alpha_d(c(s)) \overset{(5.4.7)}{=} \alpha(s)$. However, having an approximate representation, we need to define its *loss*. We chose the variance (i.e., unbiased mean squared error) as criterion. The variance-based *exact loss e* for an approximation α_d that uses an approximate representation $\langle \mathbb{S}_d, \theta^c \rangle$ is given by

$$e = \int_{s \in \mathbb{S}} (\alpha(s) - \alpha_d(c(s)))^2 \,\mathrm{d}s \,. \tag{5.4.10}$$

We choose the values in the vector α_d to be the expected values for the corresponding partition in order to minimize the loss. The vector α_d is defined by

$$\alpha_d(s_d) = \int_{s \in \mathbb{S}} \theta^c_{s_d}(s)\, \alpha(s)\, \mathrm{d}s \overset{(5.4.3)}{=} \int_{s \in \mathbb{S} |\ c(s) = s_d} \alpha(s)\, \mathrm{d}s \,. \tag{5.4.11}$$

We recursively partition the tree structure of the representation to adapt to new α-functions. This basic approach is well-suited for incrementally refining the representation.

The data basis for the representation learning is a sampled set of states with α-function values computed using the MC backup (see Equation 5.2.23). In this context, $\bar{\alpha}_{\mathbb{I}}$ is defined using the set of tuples

$$\{\langle \hat{s}, \bar{\alpha}_{\mathbb{I}}(\hat{s})\rangle : \hat{s} \in \mathbb{I}\} \ . \tag{5.4.12}$$

Additionally, let $\theta^{c,m}$ be the current representation of version m implied by the mapping c^m. The task of representation learning is to find a vector α_d with minimal loss for the sample-based α-function representation $\bar{\alpha}_{\mathbb{I}_b}$ and, if necessary, refine c^m in the step to c^{m+1}. After finding the improved representation, we can directly create the vector α_d using only the leafs of the representation. Algorithm 7 gives an overview of the procedure.

5.4.6 Decision Tree Learning and α-function Discretization

The refinement of the tree is implemented in the style of top-down-inducers. Starting with the tree T^m that defines the partition c^m, the leafs of the tree are splitted in a greedy fashion, aiming to minimize the expected loss of a discrete representation of α for the training data $\bar{\alpha}_{\mathbb{I}}$.

Let c^m be a fixed mapping. Then, the discrete vector approximation $\bar{\alpha}_d$ is computed according to Equation 5.4.11 by the finite subset of tuples of state samples $\mathbb{I}$ and α-function values $\bar{\alpha}_{\mathbb{I}}$ by

$$\bar{\alpha}_d(s_d) = \frac{1}{|\mathbb{I}|} \sum_{\hat{s}\in\mathbb{I}|\ c(\hat{s})=s_d} \bar{\alpha}_{\mathbb{I}}(\hat{s}) \ . \tag{5.4.13}$$

The discrete vector $\bar{\alpha}_d$ represents the α-function with the *empirical loss*

$$e_{\mathbb{I}} = \frac{1}{|\mathbb{I}|} \sum_{\hat{s}\in\mathbb{I}} (\bar{\alpha}_{\mathbb{I}}(\hat{s}) - \bar{\alpha}_d(c(\hat{s})))^2 \overset{(5.4.13)}{=} \frac{1}{|\mathbb{I}|} \sum_{\hat{s}\in\mathbb{I}} \left(\bar{\alpha}_{\mathbb{I}}(\hat{s}) - \frac{1}{|\mathbb{I}|} \sum_{\hat{s}'\in\mathbb{I}|\ c(\hat{s}')=s_d} \bar{\alpha}_{\mathbb{I}}(\hat{s}') \right)^2 \ . \tag{5.4.14}$$

Using this criteria, the goodness of a binary test t can be measured using the *gain* in accuracy (i.e., reduction of loss)

$$g_{\mathbb{I}}(t) = e_{\mathbb{I}} - \left(\frac{|\mathbb{I}_{c_1}|}{|\mathbb{I}|} e_{\mathbb{I}_{c_1}} + \frac{|\mathbb{I}_{c_2}|}{|\mathbb{I}|} e_{\mathbb{I}_{c_2}} \right) , \qquad \text{where} \qquad \mathbb{I}_{c_i} = \{\hat{s} \in \mathbb{I} : t(s) = c_i\} \ . \tag{5.4.15}$$

Tree growing recursively steps down the nodes of the tree, until a leaf is reached. It then expands the leaf, if the gain excesses a threshold meaning that the split improves the representation sufficiently. Algorithm 7 explains the procedure in more detail.

Algorithm 7 α-function discretization and tree growing.

```
1: function DISCRETIZE ALPHA($\bar{\alpha}_{\mathbb{I}}, T$) ▷ discretize $\alpha$ to vector $\bar{\alpha}_d$ using decision tree $T$
2:     GROW TREE NODE($s_{d,1}, \mathbb{I}, \bar{\alpha}_{\mathbb{I}}, T$)          ▷ start in root node $s_{d,1}$ of the tree $T$
3:     store T                                  ▷ store refined representation $m \to m+1$
4:     $\bar{\alpha}_d \leftarrow$ SPARSE ZERO VECTOR()
5:     for all $s_d \in T$ do
6:         if IS LEAF($s_d$) then
7:             $\bar{\alpha}_d[s_d] \leftarrow \frac{1}{|\mathbb{I}|} \sum_{\hat{s} \in \mathbb{I} |\ c(\hat{s}) = s_d} \bar{\alpha}_{\mathbb{I}}(\hat{s})$          ▷ following (5.4.13)
8:         end if
9:     end for
10:    return $\bar{\alpha}_d$
11: end function
12: function GROW TREE NODE($s_d, \mathbb{I}, \bar{\alpha}_{\mathbb{I}}, T$)          ▷ grow node $s_d$ and modify tree $T$
13:    if IS LEAF($s_d$) then
14:        $t_{s_d} \leftarrow$ CREATE NEW SPLIT($\bar{\alpha}_{\mathbb{I}}$)          ▷ temporary new split of node $s_d$
15:        $\mathbb{I}_{c_1} \leftarrow \{\hat{s} \in \mathbb{I} : t_{s_d}(s) = c_1\}$; $\mathbb{I}_{c_2} \leftarrow \{\hat{s} \in \mathbb{I} : t_{s_d}(s) = c_2\}$
16:        if COMPUTE GAIN($\mathbb{I}, \mathbb{I}_{c_1}, \mathbb{I}_{c_2}$) $<$ THRESHOLD then          ▷ following (5.4.15)
17:            return                                      ▷ stop growing
18:        else
19:            EXPAND TREE($t_{s_d}, T$)          ▷ attach temporary split to tree
20:        end if
21:    else                                             ▷ inner node
22:        $\mathbb{I}_{c_1} \leftarrow \{\hat{s} \in \mathbb{I} : t(s) = c_1\}$; $\mathbb{I}_{c_2} \leftarrow \{\hat{s} \in \mathbb{I} : t(s) = c_2\}$
23:    end if
24:    GROW TREE(CHILD($s_d, c_1$), $\mathbb{I}_{c_1}, \bar{\alpha}_{\mathbb{I}_{c_1}}, T$)
25:    GROW TREE(CHILD($s_d, c_2$), $\mathbb{I}_{c_2}, \bar{\alpha}_{\mathbb{I}_{c_2}}, T$)
26: end function
```

5.4.7 Test Generation and Expanding the Decision Tree

The most simple *tests* only consider a single attribute by checking whether the attribute exceeds a threshold. Such tests on a single attribute are in our application equivalent to splits with axis-parallel hyperplanes in the continuous state domain $s \in \mathbb{R}^n$. Frequently used realizations of this are, for example, k-d trees [Bentley, 1975]. Due to their performance and the availability of software libraries, these are a good and simple choice for many applications. However, POMDP policies in practice usually exhibit more complex underlying patterns. A plausible example is the linear correlation between position and velocity of an object (e.g., a car). More general tests that consider the whole state space as input have to be considered in order to directly express such relations. If the tests are arbitrary hyperplanes in the attribute space, they create *oblique split functions* [Murthy et al., 1994]. It is impossible in continuous attribute spaces to perform exhaustive search for the best test function. For this reason, we perform a search based on random sampling.

We separate the algorithm for finding appropriate new tests (denoted CREATE NEW SPLIT in Algorithm 7) for a leaf into two parts: Generating a set of candidates and selecting one of them.

1. Generate n candidates by repeating the following procedure n times: Randomly draw a (randomly sized) subset of state-value pairs from $\bar{\alpha}_{\mathbb{I}}$. Create the test candidate by fitting a hyperplane to the subset using least-squares.

2. Select the candidate that yields the best gain (Equation 5.4.15) for the whole sample set $\bar{\alpha}_{\mathbb{I}}$ (not the sampled subset).

The first step involves *randomization*. As only subsets of the training data are used and the final selection is based on a criterion on the complete training data, the created splits are still focused on the task of representing the α-function.

5.5 Algorithmic Realization and Program Flow

To realize a POMDP solver with the derivations and algorithms presented in the previous sections, we have to select belief points where point-based α-function backups are performed and determine the order of the backups. We propose a heuristic exploration of the infinite-dimensional belief space. Further, due to their MC nature, the α-function Bellman backups induce various errors. We analyze this problem and present an algorithmic solution to it, which discovers and corrects errors.

5.5.1 Lower and Upper Bound

In order to realize relatively sophisticated belief space exploration heuristics and to be able to detect generalization errors, we hold a lower bound Γ_{LB} on the value as well as an upper bound Γ_{UB}. Additionally, both bounds together allow to determine the accuracy of the policy.

Let V_Γ be the value defined by a set of α-functions Γ and V^* the optimal value function. A value function approximation is called *lower bound* Γ_{LB} or *upper bound* Γ_{UB}, if

$$\forall b \in \mathbb{B} : V_{\Gamma_{\text{LB}}}(b) \leq V^*(b) \leq V_{\Gamma_{\text{UB}}}(b) \,. \tag{5.5.1}$$

Due to its PWLC property, the lower bound for the value V_{LB} can be implemented as set of α-functions that are expressed by discrete vectors $\alpha_d \in \Gamma_{\text{LB}}$. It is updated by adding new vectors α_d. These vectors are created by computing a new, sparsely represented α-function using the point-based α-function backup in Section 5.2.4 and, then, discretizing the α-function using the learning approach in Section 5.4.6. At initialization time, Γ_{LB} only holds a single vector $\alpha_{d,0}$ that must assign the worst possible value to every continuous state s. The worst possible value is yielded when repeatedly receiving the minimal reward in the system, discounted by γ. Using a discrete decision tree representation, as described in Section 5.4.3, $\alpha_{d,0}$ can be realized by

$$\alpha_{d,0}(s_d) = \begin{cases} \sum_{t=0}^{\infty} \gamma^t \min_{a\in\mathbb{A}, s\in\mathbb{S}} r(s,a) = \frac{1}{1-\gamma} \min_{a\in\mathbb{A}, s\in\mathbb{S}} r(s,a) & \text{if } s_d = s_{d,1} \\ 0 & \text{otherwise} \end{cases} . \tag{5.5.2}$$

The initialization assures Γ_{LB} to be a lower bound. Since Bellman backups are monotonous, this property is preserved.

The same is true for the *upper bound* V_{UB}, when properly initialized. In contrast to the lower bound, the V_{UB} cannot be efficiently represented by a set of α-functions because value iteration updates do not preserve convexity for upper bounds. For this reason, we use a grid-based interpolation method with arbitrary (non-regular) grid-points as proposed in [Hauskrecht, 2000]. A similar approximation has been used before by [Kurniawati et al., 2008]. Such upper bound approximations are often denoted *sawtooth* interpolation due to the look of the resulting value function approximations. For details see Appendix A.3. The upper bound is initialized by assigning the best possible value to every belief so that, initially,

$$\forall b \in \mathbb{B} : V_{\text{UB}}(b) = \frac{1}{1-\gamma} \max_{a\in\mathbb{A}, s\in\mathbb{S}} r(s,a) . \tag{5.5.3}$$

5.5.2 Belief Space Exploration

Point-based value iteration relies on a representative set of belief points. If the dimensionality of the belief space is high or even infinite, as in continuous-state POMDPs, sophisticated algorithms for searching the belief space and selecting belief points are necessary (in Section 3.5 we discussed this topic in detail). In this work, we perform greedy-guided forward simulation. Therefore, we transfer the *deep sampling* idea from HSVI, as proposed for discrete POMDPs in [Kurniawati et al., 2008; Smith and Simmons, 2004], among others, to continuous POMDPs.

Belief sampling starts in the initial belief b_0 and iteratively deep samples consecutive next beliefs until a termination criteria is met. Then, point-based Bellman backups are performed in inverse order, starting at the belief that lies the farthest in the future. The backups propagate the value results of the predicted beliefs back to the initial belief.

The next beliefs are conditioned by an action and an observation. The action and the observation that are selected for sampling are determined using a heuristic. It aims to find a tradeoff between exploration and exploitation: the search should be eventually exhaustive but at the same time stay near the (presumed) optimal policy.

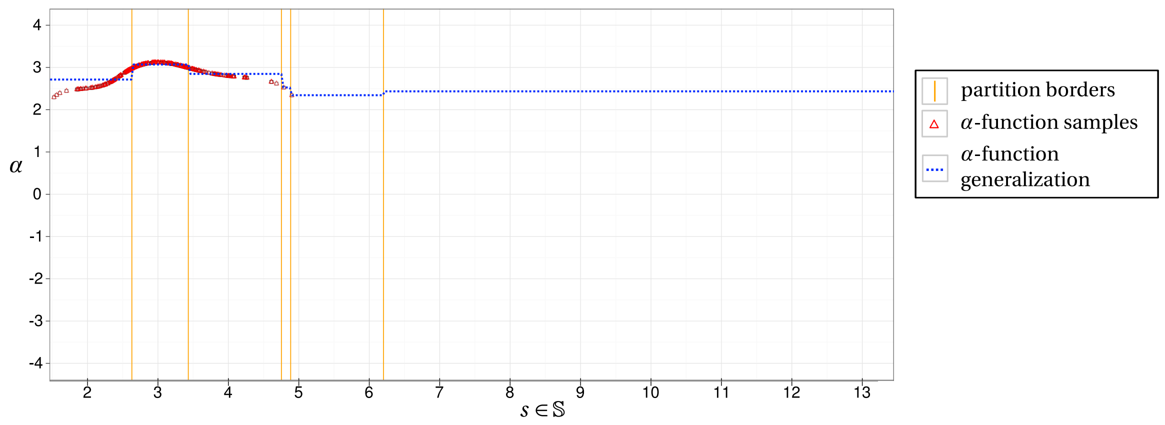


Figure 5.8: α-function generalization in a continuous 1D example.

To sample a next belief starting in b, the action a^* with the highest upper bound is chosen in accordance with the IE-MAX Heuristic from [Kaelbling, 1993] as

$$a^* = \underset{a \in \mathbb{A}}{\arg\max}\, V^n_{\text{UB,a}}(b)\,. \tag{5.5.4}$$

Eventually, this heuristic is successful: either it was the best choice or a backup reveals the opposite and the upper bound for the choice decreases. After the action is fixed, an observation o^* is selected in order to maximize expected information gain by

$$o^* = \underset{o \in \mathbb{O}}{\arg\max}\, p(o|b,a^*)\big(V_{\text{UB}}(b'_{b,a,o}) - V_{\text{LB}}(b'_{b,a,o})\big)\,. \tag{5.5.5}$$

This heuristic favors next beliefs where the uncertainty about the value (the difference between the upper and the lower bound) is high. In addition, it favors beliefs that have high probability. If a belief is unlikely, its value has less impact on the value.

5.5.3 Correction of Value Generalizations and Approximation Errors

It is impossible to perform exact Bellman backups over the (infinite) continuous state and observation space for general continuous POMDPs. The same holds for the presented algorithms: they can only approximate exact value iteration. Nevertheless, errors can be detected and corrected.

Two specific sources of errors in the backup can be identified. The first error has its origin in the MC simulation. MC backup considers a finite subset of the reachable next states and next observations (see Section 5.2.4). For an exact backup computation of $\alpha(s)$ any $s' \in \mathbb{S} : p(s'|s) > 0$ and any $o \in \mathbb{O} : p(o|s') > 0$ would have to be considered. The second error stems from the generalization of these (approximated) α-functions. We are only able to backup the α-function for a finite set of states. To obtain α-function values for the whole state space, we learn a discrete representation (see Section 5.4). The α-function values for states that were not sampled have to be predicted by the learner. When using a finite representation, even the values for the sampled points cannot be represented exactly, in general.

In Figure 5.8 we demonstrate these errors by an example. First, the MC α-function backup result for some samples is generalized over the complete state space using a discrete partition. In the example, insufficient data has been generated in the right corner of the state space. Additionally, the α-function samples are not represented well by the discrete representation. Also, the MC backup shows minor discontinuities at its left and right borders (e.g., at $s = 5$). This is typical, due to the lack of samples compared to the center. However, the potential error that is created by generalizing the high α-value of about 2.4 to the right side of the state space, is much bigger. The error due to overgeneralization, in practice, dominates the other errors.

Quantifying these errors could attenuate the problem, but it is very difficult to give tight error bounds without restricting the models of the POMDP. Imagine, for example, a reward function with a negative peak. The peak might cover only a small part of the state space, but if this peak is not sampled, it can induce large errors. Restrictions to the reward function (e.g., smoothness, maximum/minimum reward) could be imposed such that no such negative peaks are allowed. However, this is not sensible, if the peak has a reason, e.g., because the robot would collide in this area. In theory, quantifying the errors can be useful. In practice, the usefulness of these results can be limited because often restrictions to the POMDP have to be imposed that are not realistic with respect to the underlying real problem. Nevertheless, the approximation errors cannot be ignored. They can have severe consequences for the resulting policy and break convergence. In the above example, an overly optimistic policy might directly drive the agent into the negative peak and into the collision.

In contrast to quantifying the errors, we propose an approach to detect and resolve them. Therefore, we formulate a criterion to reveal violations of the monotonicity of the Bellman backup. After identification, these inconsistencies are resolved by extending the sampled α-function values.

Conflict Detection

Exact value iteration converges monotonically to the optimal value function. Hence, assuming a sufficient number of samples, the nth step continuous MC belief backup $\tilde{V}^n(b_A)$ defined in Equation 5.2.8, using a discrete lower bound $\Gamma_{\mathrm{LB}}^{n-1}$ as basis, must be superior or equal to the expected value yielded directly for a belief b_A (and its discrete representation β_A) using any α-function in $\Gamma_{\mathrm{LB}}^{n-1}$, i.e.,

$$\forall \alpha_d \in \Gamma_{\mathrm{LB}}^{n-1} : V^n(b_A) \geq \left\langle \beta_A, \alpha_d \right\rangle_d . \tag{5.5.6}$$

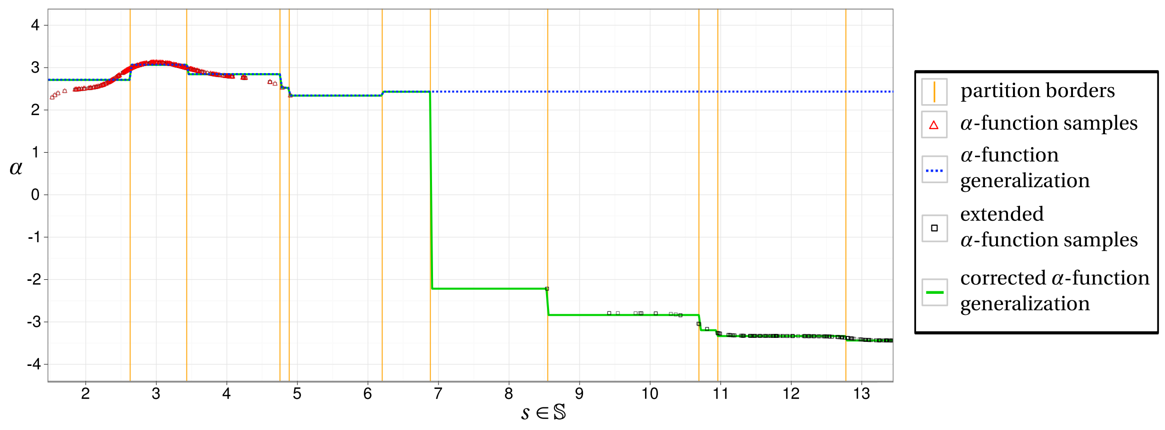


Figure 5.9: Conflict resolution in a continuous 1D example.

If this property is violated, $\Gamma^{n-1}_{\mathrm{d,LB}}$ is no lower bound. To assert consistency in the policy, any conflict generating $\alpha_d \in \Gamma^{n-1}_{\mathrm{LB}}$ must be corrected with respect to the accusing belief b_A.

Therefore, we check every new α_d vector for conflicts with any of the so far explored beliefs before adding it to the policy. Additionally, we check, if a newly explored belief reveals a conflict with any existing α_d. We resolve found conflicts as described in the next section. Algorithm 8 shows the integration into the value iteration program flow.

These checks might appear time consuming at first glance. However, using (sparse) discrete vectors, the computation of the dot-product can be carried out efficiently.

Conflict Resolution

To correct a *conflicting* vector $\alpha_{d,C}$, we simply extend the finite sampling base of the α-function approximation. Let b_A be the *accusing* belief for which $\alpha_{d,C}$ violated Equation 5.5.6. Additionally, let the vector $\alpha_{d,C}$ be the result of a point-based backup in the belief b_C following Equation 5.2.22 using the sample sets $\mathbb{I}_{b_C}$, $\mathbb{J}_{b_C,a}$, and $\mathbb{K}_{b_C,a}$. Originally, these samples were drawn from b_C, as described in Section 5.2.1. By extending these sample sets with samples representing the accusing belief b_A the reason of the error, the insufficient sample base, is eliminated. The corrected α-function $\alpha_{d,C}$ is now computed using a combination of samples for b_C and b_A: $\mathbb{I}_{b_C \cup b_A}$, $\mathbb{J}_{b_C \cup b_A,a}$, and $\mathbb{K}_{b_C \cup b_A,a}$. Note that the backup still is point-based in b_C and the choice of future α-functions in Equation 5.2.10 remains unchanged. Only the accuracy of sampling is improved.

Figure 5.9 resumes the example in Figure 5.8. The black samples on the right represent the accusing belief b_A, for which the generalization of the α-function has proven to be too optimistic. By extending the sample base of the MC backup and learning an improved representation, this error is corrected.

Finally, we have all components for the α-function backup with representation learning and conflict resolution. See Algorithm 8 for the detailed procedure.

Algorithm 8 Value iteration with conflict resolution.

```
Require: initial samples $\mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}$        ▷ see Algorithm 1
1:  function VALUE ITERATION STEP($b, a, \Gamma_{\text{LB}}, T$)
2:      for all $\alpha_{d,C} \in \Gamma_{\text{LB}}$ do        ▷ $\alpha_{d,C}$ is result of point-based backup in $b_C$
3:          $V^n(b) \leftarrow$ MC BELIEF VALUE BACKUP($b, \Gamma_{\text{LB}}$)
4:          if $V^n(b) < \langle \beta, \alpha_{d,C} \rangle_d$ then        ▷ check for conflict (5.5.6)
5:              BACKUP AND RESOLVE( $b_C, a_C, \Gamma_{\text{LB}}, \mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}, T$ )        ▷ correct $\alpha_{d,C}$
6:          end if
7:      end for
8:      BACKUP AND RESOLVE( $b, a, \Gamma_{\text{LB}}, \mathbb{I}_b, \mathbb{J}_{b,a}, \mathbb{K}_{b,a}, T$ )
9:  end function
10: function BACKUP AND RESOLVE($b, a, \Gamma_{\text{LB}}, \mathbb{I}, \mathbb{J}, \mathbb{K}, T$)
11:     $\bar{\alpha}_{\mathbb{I}} \leftarrow$ MC ALPHA BACKUP($b_C, a, \Gamma_{\text{LB}}, \mathbb{I}, \mathbb{J}, \mathbb{K}$)   ▷ continuous MC $\alpha$-function backup
12:     $\alpha_d \leftarrow$ DISCRETIZE ALPHA($\bar{\alpha}_{\mathbb{I}}, T$)        ▷ discretize $\bar{\alpha}_{\mathbb{I}}$ and refine $T$ when necessary
13:     for all $b_A \in \mathbb{B}_{\text{explored}}$ do
14:         $V^n(b_A) \leftarrow$ MC BELIEF VALUE BACKUP($b_A, \Gamma_{\text{LB}}$)
15:         if $V^n(b_A) < \langle \beta_A, \alpha_d \rangle_d$ then        ▷ check for conflict (5.5.6)
16:             $\mathbb{I} \leftarrow \mathbb{I} \cup \mathbb{I}_{b_A}$        ▷ extend samples for MC $\alpha$-backup
17:             $\mathbb{J} \leftarrow \mathbb{J} \cup \mathbb{J}_{b_A}$
18:             $\mathbb{K} \leftarrow \mathbb{K} \cup \mathbb{K}_{b_A}$
19:             goto 11        ▷ start over with extended sample sets
20:         end if
21:     end for        ▷ conflicts solved
22:     store $\Gamma_{\text{LB}} \leftarrow$ UPDATE($\Gamma_{\text{LB}}, \alpha_d$)        ▷ add or overwrite $\alpha_d$; version $(n-1) \rightarrow n$
23: end function
```

5.5.4 Solver Main Loop

To solve a POMDP, the developed components have to be executed in the right order. Algorithm 9 outlines the program flow. First, the representation, the tree of explored beliefs, as well as the upper and lower bound are initialized. Then, the following sequence is executed in a loop until predefined precision is reached: the belief space search heuristic selects a belief b. If no backups have been performed in b before, sample sets for the MC algorithms are created. Then, the discretization of the continuous sample sets is updated with the recent version of the representation. The action yielding the highest value is selected and backups of the upper bound as well as the lower bound are performed in b.

Algorithm 9 POMDP solver main loop.

```
1: function SOLVE($e_{\text{precision}}$)
2:     $T \leftarrow \{s_{d,1}\}$    ▷ initialize tree for representation $\theta^c$ with root node
3:     $T_{\mathbb{B}} \leftarrow \{b_0\}$    ▷ initialize belief tree with explored beliefs
4:     $\Gamma_{\text{LB}} \leftarrow \{\alpha_{d,0}\}$    ▷ initialize lower bound (Section 5.5.1)
5:     INIT($V_{\text{UB}}$)    ▷ initialize upper bound (Section 5.5.1)
6:     repeat
7:         $b \leftarrow$ EXPLORE($T_{\mathbb{B}}, \Gamma_{\text{LB}}, \Gamma_{\text{UB}}$)    ▷ deep sample *current* belief $b$ (Section 5.5.2)
8:         if $b \notin T_{\mathbb{B}}$ then
9:             DRAW SAMPLES($b$)    ▷ draw sample skeleton (Algorithm 1)
10:            $T_{\mathbb{B}} \leftarrow$ EXPAND BELIEF TREE($T_{\mathbb{B}}, b$)    ▷ construct next beliefs $\tilde{b}'$ (Algorithm 2)
11:        end if
12:        for all $a \in \mathbb{A}$ do
13:            DISCRETIZE STATES(b,a)    ▷ discretize $\hat{s}'$ (Algorithm 5)
14:            DISCRETIZE BELIEFS(b,a)    ▷ discretize $\tilde{b}'$ (Algorithm 5)
15:        end for    ▷ $b$ is now prepared for backups
16:        MC BELIEF VALUE BACKUP($b$)    ▷ backup $b$ see Algorithm 3
17:        $a_{\max} \leftarrow \arg\max_{a \in \mathbb{A}} \tilde{V}_a^n(b)$    ▷ create $\alpha$ only for the best action
18:        UPDATE UPPER BOUND($b, a_{\max}, V_{\text{UB}}, T$)    ▷ Appendix A.3
19:        VALUE ITERATION STEP($b, a_{\max}, \Gamma_{\text{LB}}, T$)    ▷ Algorithm 8
20:    until CONVERGENCE
21: end function
```

5.6 Summary and Conclusion

In this chapter, a method for solving continuous-state and -observation POMDPs with discrete actions is presented (Contribution 1). We found that solving the POMDP cannot be decoupled from finding a suited representation of the continuous state space. By iteratively performing Bellman backups and refining the representation, we solve the circular dependency between the value function and its representation (Contribution 1b). We combine point-based MC simulation and α-function backup with machine learning of a discrete representation. The models of the POMDP are not restricted (Contribution 1a). Further, the learned, discrete representation generalizes value functions effectively over the continuous state space and predicts values for previously unseen beliefs (Contribution 1c).

This work sets itself apart from previous approaches mainly in three aspects:

1. It integrates machine learning into value iteration so that the representation can be updated simultaneously with the value function.

2. It learns a discrete state space representation with the explicit goal to optimally represent the value function (and not the belief).

3. It learns a non-parametric representation and the family of the basis functions is, in theory, not restricted.

These three aspects are a novelty in the context of solving continuous POMDPs and significantly improve performance.

Implementation-wise, we propose a state space representation that is based on decision trees using top-down induction for incremental refinement. On the one hand, this design choice has several advantages: evaluating α-functions as well as refining the representation can be done very efficiently in decision trees. Also, when using top-down learning, the representation refinement is straight forward. On the other hand, only piecewise-constant α-function approximations are possible and overlapping basis functions prohibited. This prevents the potentially better generalization that could be achieved when applying more sophisticated machine learning techniques (e.g., improvements by pruning, boosting, or ensembles methods, or fundamentally different approaches such as deep learning).

It is important to understand, however, that the presented concept is not restricted to our choice of implementation. In Section 5.3, the type of the basis functions are not restricted. In particular, the proof of Lemma 1 in Section 5.3.2 does not require any assumptions about the family of the basis functions. The generality of our central concept leaves much and promising scope for future work.

Chapter 6

Decision-making for Autonomous Driving by Solving Continuous POMDPs

We present the components necessary for modeling safe, goal-directed, rule compliant, and efficient tactical decision-making as POMDP. By solving this POMDP, a policy is generated that maps the current situation to driving actions with the goal to optimize the expectation over future rewards.

The goal of tactical decision-making is to select the best high-level action. This action is then executed by a lower-level control. Decisions are based on preprocessed observations and background knowledge, such as maps. Recall Figure 1.3 that visualizes the architecture of the autonomous vehicle and the integration of tactical decision-making.

To provide a good basis for decisions, the POMDP model needs to realistically resemble the dynamics of the traffic environment. The central aspects are other road users, their behaviors, and if and how well they can be perceived. The immediate reward function defines the driving goal. It evaluates if the driving destination is reached and if collisions happened. In addition, it assesses how efficient and comfortable the driving is and if it complies with the traffic legislation.

The POMDP model should be as general as possible. In other words, we seek to keep the modifications that are necessary when adapting the system to different traffic scenarios to a minimum. For example, highway driving, lane merging and approaching junctions should be solved using the same underlying general model. We achieve this objective with hierarchical, continuous-space models that we structure with a dynamic Bayesian network (DBN). This way, we can also capture the behavior of the lower level control algorithm that executes the tactical decisions for the autonomous car and transforms them to continuous steering and acceleration.

For finding policies by approximating solutions to the continuous POMDP, we propose two approaches. The first approach omits partial observability and finds policies using a MDP value iteration with predefined equidistant space discretization rules and adaptively growing state space. The second approach considers the full POMDP using the continuous value iteration method presented in Chapter 5 that learns a discrete space.

Note that in this chapter, a general model and approach is presented that can be practically utilized in different ways. In the experimental evaluation in Chapter 7, exemplary realizations of highway and urban driving are shown. In Section 8.3, multiple applications of the approach in the context of driving and ways to embed it into a real autonomous car are discussed.

Chapter Overview First, in Section 6.1, the spaces and models of the (PO)MDP[6.1] are presented. We discuss which aspects are essential to obtain good policies and which can be simplified or neglected. Further, we give some details about possible efficient implementations. The second part of this chapter in Section 6.2 is concerned with the solution of this (PO)MDP in order to generate decision policies. This chapter is partly based on [Brechtel et al., 2011] and [Brechtel et al., 2014].

6.1 MDP and POMDP Spaces and Models for Driving

In this section the spaces and models of the MDP and POMDP are presented. The spaces $\mathbb{S}$, $\mathbb{O}$ and $\mathbb{A}$ define the sample space of random variables S for the state of the world, O for observations, and A for actions. The *models* define the relationship between the random variables as conditional probability distributions. Ultimately, they determine the meaning of states, actions and observations.

In order to obtain a general formulation, the state and observation spaces are continuous. The following example gives an intuitive understanding why this is more general than a symbolic representation: for instance, any position[6.2] of a car can be expressed using two continuous values x_1 and x_2, regardless of the context the car is in, or in which area of the world it is. Contrary to this, symbolic states such as *in front of intersection* or *on left lane* highly depend on the current situation.

Note that the observations space $\mathbb{O}$ and the observation model Ω are reserved to the POMDP. As MDPs have no notion of partial observability, they only consist of the state space $\mathbb{S}$, a transition model T and a reward function R.

6.1.1 State Space

The state space must suffice the requirements of decision-making. Therefore, every state must hold enough information so that the Markov property can apply and every state only depends solely on its previous state. As we directly use the continuous state

[6.1] A continuous POMDP model is presented. However, we often denote it (PO)MDP to indicate that a continuous MDP can be derived simply by omitting observations.
[6.2] The elevation can be neglected for most normal driving scenarios.

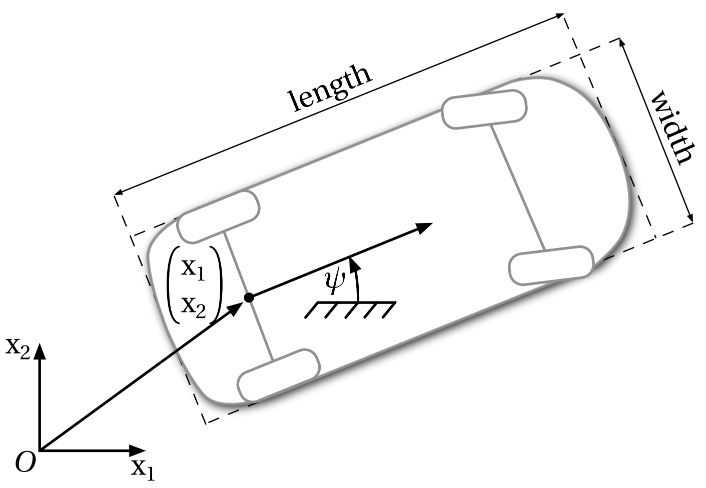


Figure 6.1: State space x_i of a single car or ego vehicle x_{ego}.

space, a representation similar to those frequently used for state estimation and prediction can be employed. The state of the world basically comprises the geometrical poses and velocities of all road users. Map information, such as the topology and geometry of the road network or information about houses or bushes at the side of the road that can block the view of the car, are assumed static. Static states are part of the background knowledge of the models and not part of the (dynamic) state space $s \in \mathbb{S}$.

The full state of the world $s \in \mathbb{S}$ either is in an environment state $s \in \mathbb{X}$, which represents normal driving situations, or in the terminal state $s = s_{\text{term}}$ so that

$$\mathbb{S} = \mathbb{X} \cup \{s_{\text{term}}\} . \qquad (6.1.1)$$

State of the Environment The state of the environment $x \in \mathbb{X}$ is a vector containing the state of the controlled (ego) vehicle $x_{\text{ego}} \in \mathbb{X}_{\text{obj}}$ and the states of all n other objects (i.e., road users) $x_i \in \mathbb{X}_{\text{obj}}$. As illustrated in Figure 6.1, the ego vehicle's state x_{ego} as well as other object's states x_i are represented equally by their global position $\left(\mathrm{x}_1, \mathrm{x}_2\right)^\mathsf{T}$, orientation ψ, and velocity v so that

$$x = \begin{pmatrix} x_{\text{ego}} \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad \text{and} \qquad x_i = \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \psi \\ \mathrm{v} \end{pmatrix} \in \mathbb{X}_{\text{obj}} \subseteq \mathbb{R}^4 . \qquad (6.1.2)$$

Terminal State In addition to the state of the environment X that represents normal driving situations, we introduce a special state that we denote *terminal state*[6.3] s_{term}. This is due to the fact that there must be some semantic which models that the planning problem is over (i.e., there is no future). The world transitions to the terminal

[6.3]In literature, the terms *terminal state*, *stopping state*, and *absorbing state* are used synonymously.

state after the ego vehicle crashes or reaches the goal. The agent cannot leave the terminal state and he receives no reward or cost being in it. This is important because it prevents the planner from drawing wrong conclusions (e.g., by assuming that the goal reward can be received multiple times or that the goal can be reached after a collision). Further, it assures that planning can be stopped, when the terminal state is reached, without affecting the resulting policy (hence the name *terminal*).

6.1.2 Observation Space

The agent is capable of observing the environment using sensors which are usually mounted to the ego vehicle. We assume that there are sensors perceiving the intrinsic state of the ego vehicle x_{ego} as well as sensors perceiving external objects x_i

$$o = \left(o_{\text{ego}}, o_1, \ldots, o_n\right)^\mathsf{T} \qquad \text{with} \qquad o_{\text{ego}}, o_1, \ldots, o_n \in \mathbb{O}_{\text{obj}}. \tag{6.1.3}$$

Further, we assume that poses and velocities in the state space $\mathbb{X}_{\text{obj}}$ are measured directly. The observation space of an object, thus, is equal to its state space, if the object is visible to the sensors. We additionally introduce a special observation o_{inv}. It models that the object is invisible to the sensors due to occlusions or other sensor limitations. The full observation space

$$\mathbb{O}_{\text{obj}} = \mathbb{X}_{\text{obj}} \cup \{o_{\text{inv}}\}\,. \tag{6.1.4}$$

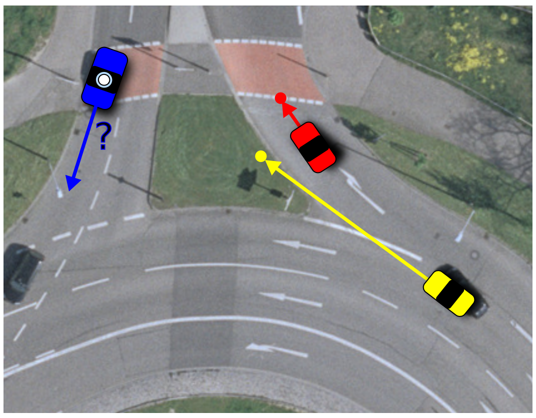
6.1.3 Action Space

In every situation, the agent aims to select the best high-level driving maneuver. Examples of maneuvers are acc-/decelerating, stopping in front of an intersection, following in a safe distance, or making a lane change. Such maneuvers have a higher level of abstraction than continuous acc-, deceleration and steering, which usually are the result of controllers or motion planning.

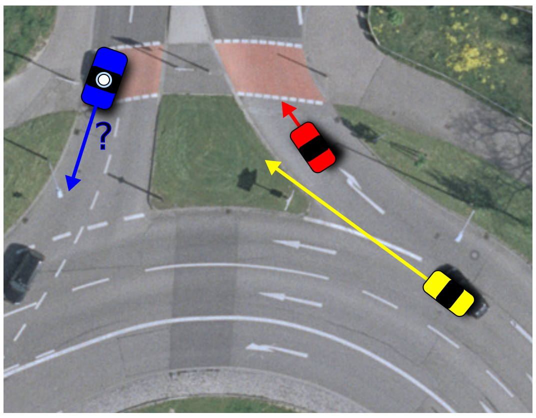
We consider a discrete set of actions $\mathbb{A}$ that encode high-level behaviors. The discrete actions parametrize a continuous control policy for the length of the time step of the (PO)MDP. This control policy finally sets the continuous control

$$a_{\text{ego}} \in \mathbb{A}_{\text{c}} \tag{6.1.5}$$

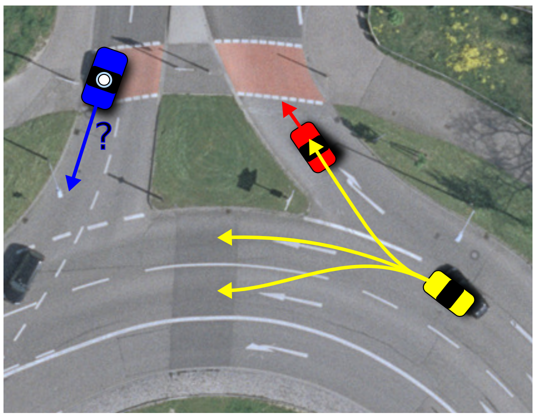
with a higher update rate of about 100 ms. It considers the high-level driving target determined by the discrete action as well as the context of the vehicle. The context information allows the control policy to follow lanes and react on other road users, if something unexpected happens in between the (PO)MDP time steps. See the transition model in Section 6.1.4 for the realization of the model.



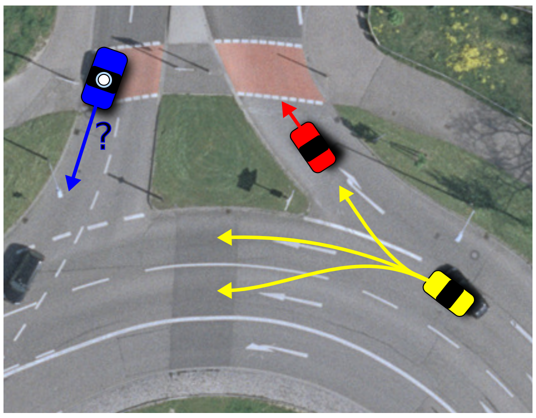
(a) Linear without uncertainty.



(b) Linear with uncertainty.



(c) Non-linear with map information.



(d) With map information and interaction.

Figure 6.2: Transition models with varying accuracy. Map Data [City of Karlsruhe].

6.1.4 Transition Model

The transition model describes the dynamics of the system and the influence of the chosen actions on the dynamics. In order to account for the uncertainty in the transition, it is modeled as conditional distribution $p(s'|s,a)$. This model is mostly about simulating the behavior of drivers and other road users. Due to human factors, it is the most complex of the presented POMDP models.

The behavior of road users depends on multiple factors, which need to be modeled in order to obtain realistic predictions. We illustrate the influence of the most important dependencies in Figure 6.2. The images show a scenario, where the blue ego vehicle has the goal to enter a traffic circle. Two other cars are involved in this scenario and need to be considered. The first image in Figure 6.2a sketches the results of a physically inspired model that assumes independent vehicle motion with constant velocities. This model has obvious flaws. The yellow car is predicted to directly drive into a green area between two roads. Further, this model is deterministic, which means that it declares the (absolutely wrong) prediction to be accurate. Despite these severe shortcomings, constant velocity models are the most frequently used basis for state estimation, today. By adding noise generously, observations can often compensate the poor predictions so that state estimation can keep track of the objects despite the poor prediction (see Figure 6.2b). Performance-wise, this simple model has the advantage that the predicted vehicle motion solely depends on its intrinsic state. Hence, for predictions with longer time horizons, the variance of the prediction is too high to suffice for tactical decision-making. Further, as can be seen in the example, the most likely prediction of the yellow car still is the green area off the road.

Accuracy can be improved by considering the static environment, such as the road geometry and topography. As illustrated in Figure 6.2c, drivers are guided by lanes. This additional information helps to reduce the uncertainty of the prediction. The complexity of the model increases (conditional distributions are non-linear and, e.g., due to forks of the road also multimodal). However, the vehicles can still be treated independently. Despite these improvements, the yellow vehicle in the example is predicted to collide with the red vehicle. For decision-making, the *interaction* between road users is essential. As can be seen in Figure 6.2d, the driver of the yellow car would, in reality, react to the red vehicle by slowing down.

In summary, it can be stated that the transition model for tactical decision-making needs to consider the full context. This includes the static context that can be part of the background knowledge (e.g., maps, traffic signs) as well as dynamic context information (e.g., other vehicles' poses) that is part of the hidden state space and must be estimated over time from measurements. Uncertainties arising in this process further complicate the problem.

We propose a hierarchical Bayesian model to realize the transition model. This approach allows to structure the various factors on different levels of abstraction and make dependencies explicit. A simplified overview of the DBN transition model is given in Figure 6.3. The DBN in this work is based on the approach to predicting traffic situations and modeling interacting driver behaviors presented in [Gindele et al., 2010, 2013, 2015]. The model is extended by adding the ego car, which can be controlled by the (PO)MDP agent by selecting actions. To get a good simulation of the road users and their behaviors, a time step shorter than 200 ms is necessary. The time step of the (PO)MDP is 1.5 s. To successfully embed the DBN, several internal prediction steps are executed when computing a single step of the (PO)MDP transition model.

It must be emphasized here that the (PO)MDP creates high-level policies with long prediction horizon (up to 20s in the examples of the evaluation in Section 7.3) and under consideration of many complex aspects and uncertainties. For this reason, reaction times of over a second suffice. For comparison: studies showed that the reaction time of unalerted human drivers to unexpected events can be commonly assumed to

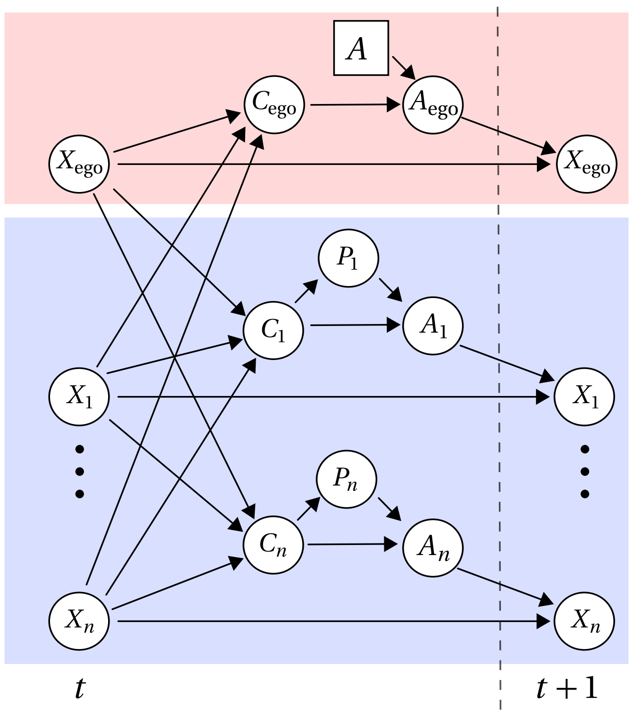


Figure 6.3: Bayesian network (BN) representation of the transition model. The area marked red models the ego vehicle and the area marked blue the other road users. The relationships between the random variables for the underlying continuous states X_i, the vehicle controls A_i, vehicle contexts C_i, driver's plan P_i, and the agents discrete action A are displayed.

be ca. 1.5 s [Taoka, 1989]. In contrast to this, e.g., a driver's steering corrections obviously have to be much quicker.

Driver's Behavior Accurate prediction of the non-controlled vehicles $i \in \{1, \dots, n\}$ is essential for decision-making. In this work, we only provide models for car-like objects. In principle, the presented approach can be applied to any kind of traffic participant with slight modifications, including pedestrians and cyclists. In [Brechtel et al., 2009], we propose a basic prediction model for pedestrians in a similar application.

A human driver's reaction has several complex dependencies from static as well as dynamic variables. Our approach is to resemble the internal decision process of every driver using a hierarchical DBN (see Figure 6.3). On the lowest abstraction level, the state of the vehicle $x_i \in \mathbb{X}_{\text{obj}}$ (see Equation 6.1.2) is the physical manifestation of his driving decisions. Despite his potentially complex thoughts, eventually, he can only control his vehicle by acc-/decelerating and steering. His control $a_i \in \mathbb{A}_C$ consists of the derivations of the velocity and orientation[6.4] $\left(\dot{\text{v}}, \dot{\psi}\right)^\top$. The conditional density function $p(x_i'|a_i, x_i)$ models the physical vehicle motion when the control a_i is applied. To

[6.4]The yaw rate $\dot{\psi}$ can be derived from the steering angle when assuming circular motion.

account for kinematic as well as dynamic constraints, we assume a one-track model [Braess and Seiffert, 2011] with restricted lateral and longitudinal acceleration.

Predicting the control a_i that a driver applies is more complex than the vehicle's reactions on the control input. First, drivers usually follow the course of the road. To model this, map information has to be given as background knowledge. Secondly, road users interact with each other and the ego car and therefore, their reactions depend on the states of other vehicles. We comprise this information in the situational *context* C_i of every road user i with the model $p(c_i|x_{\text{ego}}, x_1, \dots, x_n)$. The context contains, for example, geometric relations of the vehicle i to the road network and to the other road users. This information about the context is used to estimate the driver's *planned routes* P_i. These denote paths of road segments that he will follow. The planned route in combination with the situational context is sufficient to derive the control that the driver applies following $p(a_i|c_i, p_i)$. For more details about the prediction of other road users see [Gindele et al., 2013].

Note that the vehicles' physical state, their context, planned route and control are all latent variables that can only be indirectly inferred from measurements over time. As a consequence, even if all described static and dynamic factors are considered, road users can never be predicted (or even estimated) with absolute precision. Decision-making must be able to handle the remaining uncertainty.

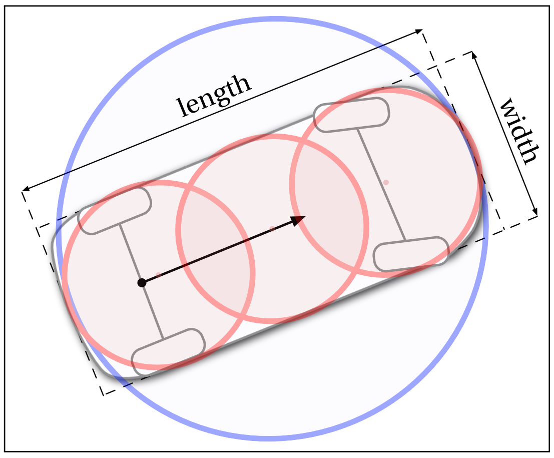
Actions and Vehicle Control The physical model assumed for the ego vehicle is basically the same as for the other road users. As can be seen in Figure 6.3, the difference is that its control $a_{\text{ego}} \in \mathbb{A}_C$ is not estimated. Instead it is set by a control policy that is parametrized by the selected discrete action $a \in \mathbb{A}$.

The model $p(a_{\text{ego}}|a, c_{\text{ego}})$ implements the continuous controller. The controller basically follows the road and adapts the velocity or changes lanes according to the selected discrete action a. Additionally, the controller has a dependency to the context of the vehicle to be able to react to other traffic participants (e.g., emergency braking, if a car driving in front suddenly brakes during the POMDP time cycle).

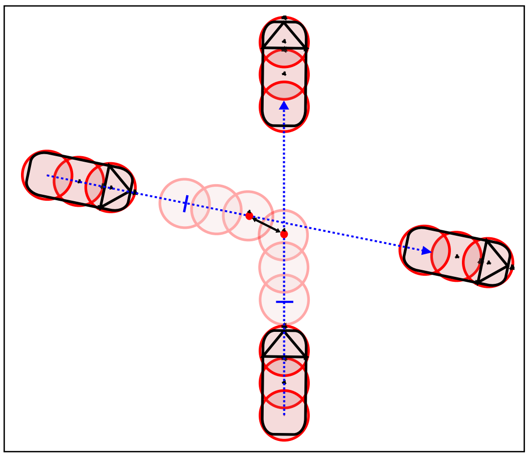
Termination The termination model is not shown in Figure 6.3, as it is an exception from the normal prediction. It has to assure that there is no way out of the terminal state, regardless of the chosen action by

$$\forall a \in \mathbb{A} : p(s'|s_{\text{term}}, a) = \begin{cases} 1 & \text{if } s' = s_{\text{term}} \\ 0 & \text{otherwise} \end{cases} \quad . \tag{6.1.6}$$

With this definition the future becomes irrelevant when being in s_{term}.



(a) Two-Level collision bounds.



(b) Continuous collision detection.

Figure 6.4: Efficient hierarchical time-continuous collision detection.

The second important property of the terminal model is that s_{term} is reached when a collision between the ego vehicle and another road user happened or when the driving goal is reached (see the reward function for more details) so that

$$\forall s \neq s_{\text{term}} : p(s' = s_{\text{term}}|s,a) = \begin{cases} 1 & \text{if collision or target area reached} \\ 0 & \text{otherwise} \end{cases} . \tag{6.1.7}$$

Collision Detection Collision detection is an important part of this termination model. It can be be computationally expensive. However, is important to detect collisions continuously. If collisions are only checked at certain points in time, vehicles with high speeds can *tunnel* each other.

We apply hierarchical two-level, time-continuous collision detection (CCD)[6.5] to avoid tunneling and assure computational efficiency. We approximate objects using circular volumes. Figure 6.4a shows the bounding circles for a car. For other objects, different object dimensions can be assumed. The first level collision check uses the larger circle surrounding the vehicle (blue). If this check detects potential collisions, all three smaller circles (red) on the second level are checked against all circles of the other object. For continuous-time collision detection, linear motion of the collision structures between the time steps is assumed [Ericson, 2005]. See Figure 6.4b for an example, where discrete collision checking would have failed due to tunneling. It sketches the assumed linear motion of the (second level) circles. When assuming linear motion between the time steps, detecting a collision between two moving objects can be simplified to detecting a collision between a static and a dynamic object. Therefore, the reference frame has to be transformed to be relative to one object (which in this reference frame is static) [Hahn, 1988]. For collision detection between a static

[6.5] In the literature, time-continuous methods are frequently denoted “a priori”.

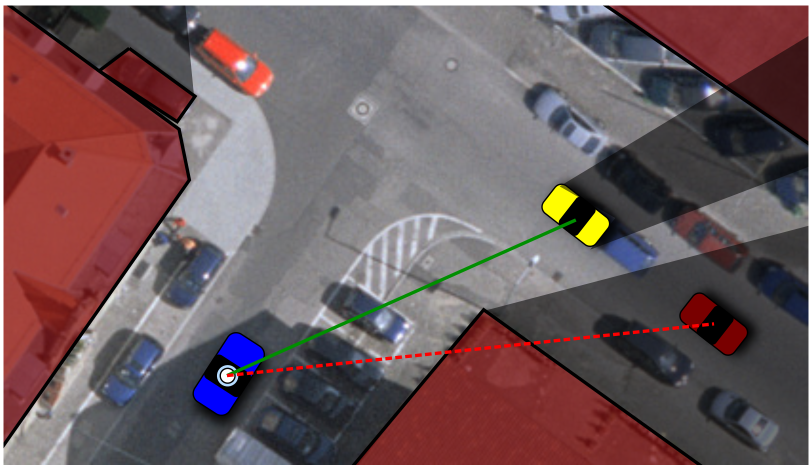


Figure 6.5: Observation model. Red polygons represent static opaque areas. The yellow car is visible and measured with normally distributed noise. It casts an area of obstructed vision. The view on the red car is blocked by a house. Map Data [City of Karlsruhe].

and a dynamic object, time can be neglected. With these simplifications, checking the collision of two dynamic circles, reduces to one vector addition and a distance check between a point and a line segment.

6.1.5 Observation Model

The task of the observation model is to represent capabilities of the autonomous car's sensors and, even more important, their limitations. Therefore, it models the probability of making an observation o for a given state of the world s. Note that it is not necessary to simulate the physical measurement process in detail. Instead, the model should cover relevant effects of sensor limitations. Two effects of sensor errors can be distinguished: noise and (in)visibility. We put special emphasis on modeling the latter, as not seeing an object usually has more severe consequences than the measurement error when it can be seen. See Figure 6.5 for an example. For decision-making, moving objects (including the ego vehicle) are particularly relevant.

We assume static parts of the environment, such as roads or houses, to be known in advance as part of the background knowledge. Recent progress in the area of simultaneous localization and mapping (SLAM) (see, e.g., [Grisetti et al., 2010]) make it possible to automatically create large scale maps of the traffic environment. Today, 3D maps already exist for many cities in the world (e.g., *Google earth* with *street view*). It can be expected that in the future more cars are equipped with sophisticated sensors and connected to an online database. If their data is fused, it can be assumed that the environment maps are sufficiently up to date and accurate for driving.

The measurements of the objects are modeled to be mostly independent from each other. For simplicity, the basic model assumes identity functions with additive normally distributed noise for all objects. Their covariance matrices Σ_{O_i} account for speed and orientation of the objects.

Self-Localization It can be assumed that pose and velocity of the ego vehicle can always be measured with small noise, especially when using state-of-the-art SLAM with additional information from odometry and Differential Global Positioning System (DGPS). The ego vehicle's state is always visible and measured with the covariance matrix $\Sigma_{O_{\text{ego}}}$ according to

$$p(o_{\text{ego}}|x'_{\text{ego}}, x'_1, \ldots, x'_n) = p(o_{\text{ego}}|x'_{\text{ego}}) = \begin{cases} 0 & \text{if } o_{\text{ego}} = o_{\text{inv}} \\ \mathcal{N}(x'_{\text{ego}}, \Sigma_{O_{\text{ego}}}) & \text{otherwise} \end{cases} . \tag{6.1.8}$$

Visibility of other Road Users For the other road users i, the conditional distribution $p(o_i|x'_{\text{ego}}, x'_1, \ldots, x'_n)$ models whether and how accurately they can be seen. In order to yield information-aware policies, the visibility of objects needs to be modeled. Multiple reasons exist, why a road user can be invisible to the sensors. The angle of view or the range of the sensors might be limited. These cases can be easily covered by determining the relative angle and distance. Also, the view can be obstructed by other road users or environment objects. We detect this by checking, if there is a direct line of sight to the road user that does not intersect with other objects (see Figure 6.5 for illustration). For static objects a map with a geometric polygon representation of objects that can block the view is part of the background knowledge. A polygon representation of the dynamic objects can be derived from their states and integrated into the polygon representation of the static environment.

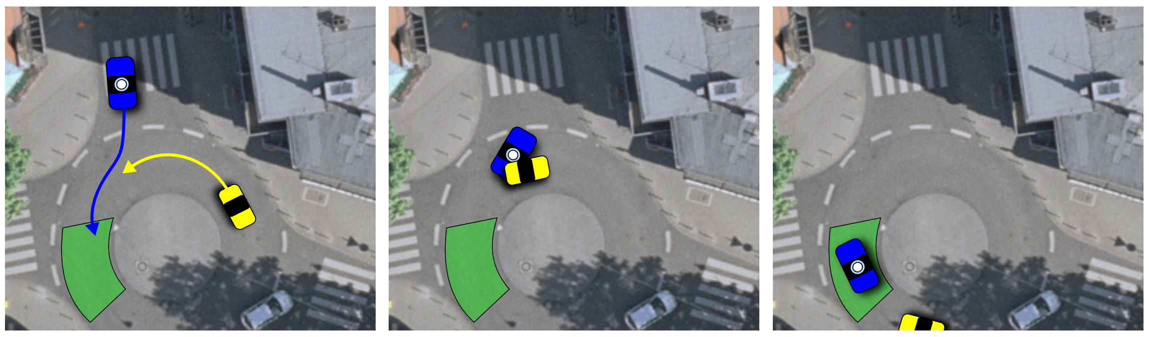
If the state x'_i of an object is invisible (i.e., the line of sight is intersected by the occlusion polygon), we have

$$p(o_i|x'_{\text{ego}}, x'_1, \ldots, x'_n) = \begin{cases} 1 & \text{if } o_i = o_{\text{inv}} \\ 0 & \text{else} \end{cases} . \tag{6.1.9}$$

If the object i is visible from x'_{ego}, the observation solely depends on x'_i by

$$p(o_i|x'_{\text{ego}}, x'_1, \ldots, x'_n) = p(o_i|x'_i) = \begin{cases} 0 & \text{if } o_i = o_{\text{inv}} \\ \mathcal{N}(x'_i, \Sigma_{O_i}) & \text{otherwise} \end{cases} . \tag{6.1.10}$$

Note that it can be sensible to adapt the covariance O_i depending on the distance and orientation to the ego vehicle. This might be particularly interesting when using stereo-based visual recognition systems, where the measurement errors increases quadratically with the distance.



(a) Initial scene. (b) Collision. (c) Goal reached.

Figure 6.6: Reward function when entering a traffic circle. Map Data [City of Karlsruhe].

6.1.6 Reward Function

The reward function combines multiple criteria to a single real value. Primary goals are to move ahead or, alternatively, reach a target destination as quickly as possible, and at the same time not to crash into other road users or leave the road. These criteria often contradict each other as illustrated in Figure 6.6. The goal of the blue car in this scenario is to enter the traffic circle and reach the green *target area*. However, it must yield to the yellow car first to avoid crashing. Secondary goals are efficiency (i.e., minimizing energy consumption), compliance with traffic rules and comfort. Engineering a well-functioning reward model is straight forward compared to the difficulty of manually modeling the value function or even policy. Only the immediate situation and not its future development has to be assessed for the immediate reward.

The real-valued, deterministic reward function $\mathring{r}(s,a,s')$ quantifies these multiple objectives. It depends on the current state s, the executed action a and the next state s'. [6.6] In practice, not all objectives can be fulfilled at the same time. It might be, for instance, necessary to cross a solid white line to avoid a collision. A good policy balances the potentially contradicting driving objectives. The reward function is composed of multiple subfunctions, each describing a single driving objective, as follows

$$\mathring{r}(s,a,s') = \begin{cases} 0 & \text{if } s = s_{\text{term}} \\ r_{\text{coll}}(s,a,s') & \text{if collision happened} \\ r_{\text{goal}}(x_{\text{ego}}, x'_{\text{ego}}) & \text{if target area reached} \\ r_{\text{rules}}(x_{\text{ego}}) + r_{\text{move}}(\text{v}) + r_{\text{eff}}(\dot{\text{v}}, \dot{\psi}) & \text{otherwise} \end{cases} . \tag{6.1.11}$$

The reward computation is separated into two parts. In the first part, it is evaluated whether the ego vehicle collides or not. In this step, information about all road users at time t as well as $t+1$ and the chosen action a is necessary to detect potential collisions (see Section 6.1.4). This is expressed by the collision reward $r_{\text{coll}}(s,a,s')$ which returns a negative reward, if the ego car crashes, and zero otherwise.

[6.6]See Appendix A.1 for the relation between this alternative reward $\mathring{r}(s,a,s')$ and $r(s,a)$.

If no collision is detected, in the second part, a reward is computed that solely depends on the current state of the ego vehicle $x_{\text{ego}} = \left(\mathrm{x}_1, \mathrm{x}_2, \psi, \mathrm{v}\right)^\mathsf{T}$ and the chosen control $a_{\text{ego}} = \left(\dot{\mathrm{v}}, \dot{\psi}\right)^\mathsf{T}$. The following driving objectives are considered in the second part: the goal model $r_{\text{goal}}(x_{\text{ego}}, x'_{\text{ego}})$ returns a positive reward, if the car reaches a target area (e.g., located behind an intersection). Target areas are not limited to positions. They can also imply goals for the other states, such as a desired speed interval (e.g., if the goal is to stop). In addition, a reward for moving ahead can be returned that depends on the velocity by $r_{\text{move}}(\mathrm{v})$. Compliance with traffic rules, such as speed limits or using the leftmost lane on a highway, are expressed in $r_{\text{rules}}(x_{\text{ego}})$. Additionally, $r_{\text{eff}}(\dot{\mathrm{v}}, \dot{\psi})$ returns costs for acc- and deceleration in order to optimize comfort and efficiency.

The numerical relation of these reward functions can be adapted to obtain a wished behavior. For example, if r_{move} is chosen to be relatively high and r_{rules} to be relatively low, more reckless driving can be the result.

Terminal Reward Again, the terminal state is an exception. The agent never receives rewards when he is in the terminal state so that

$$\forall a \in \mathbb{A}, \forall s' \in \mathbb{S} : \mathring{r}(s_{\text{term}}, a, s') = 0 \qquad \Rightarrow \qquad \forall a \in \mathbb{A} : r(s_{\text{term}}, a) = 0\,. \tag{6.1.12}$$

Intuitively, in combination with the termination model, which assures that the terminal state cannot be left, this means that planning can be stopped in the terminal state. This can be formally asserted: for POMDPs, the continuous α-function backup in Section 4.3.14 always yields 0.

Proof.

$$\alpha^n_{b,a}(s_{\text{term}}) \overset{(4.3.14)}{=} r(s,a) + \gamma \int\limits_{s' \in \mathbb{S}} p(s'|s_{\text{term}}, a) \int\limits_{o \in \mathbb{O}} \alpha^{n-1}_{b,a,o}(s') p(o|s')\, \mathrm{d}o\, \mathrm{d}s' \tag{6.1.13}$$

$$\overset{(6.1.6)}{=} r(s_{\text{term}}, a) + \gamma\, \alpha^{n-1}_{b,a,o}(s_{\text{term}}) = \sum_{t=0}^{\infty} \gamma^t\, r(s_{\text{term}}, a) \overset{(6.1.12)}{=} 0 \tag{6.1.14}$$

□

Consequently, computations for s_{term} can be skipped. The same holds for MDPs.

6.2 Generating Driving Policies by Solving the (PO)MDP

In this section, we outline how the (PO)MDP models presented in the previous section can be used to generate driving polices. Two different approaches are presented: The first approach is based on a discrete state MDP. To embed the described continuous-space models into a discrete-state decision process, we perform equidistant, lane-aligned discretization. We employ MDP value iteration to solve the resulting discrete-space problem. To be able to cope with the discrete, but still infinite state space, we propose a modification to value iteration that allows the state space to adaptively grow during planning. However, this MDP approach is limited to full observability.

In contrast to this, the second approach directly works on the continuous space and accounts for partial observability. This is realized using the continuous POMDP solver presented in Chapter 5. No modifications to the presented continuous POMDP models are necessary with this approach and they are directly used for the evaluation in the next chapter. Nevertheless, we present some ideas how the computational efficiency can be improved.

6.2.1 Discrete MDP Value Iteration with State Space Growing

The most common procedure for creating a discrete MDP representation of a (continuous) real-world problem is to, first, name all possible states and, then, precompute rewards and transition probabilities for all combinations of states. These can be stored (e.g., in matrices). Finally, the discrete MDP is solved, for example, with value iteration or policy iteration. This approach fails for state spaces with infinite or very large volumes, such as long roads. Discretizing a road of infinite length equidistantly results in an infinite number of discrete states. Solving the MDP, however, requires the state space to be finite. For this reason, we link the continuous states presented in Section 6.1 to discrete states and derive rules to translate continuous to discrete models. This way, models for any state can be computed on demand, which enables us to consider only a finite subset of the infinite state space. We grow this subset by adding states when they are reached during planning.

Discrete State Space

We link the continuous space $\mathbb{S}$, which comprises poses and velocities of road users (see Section 6.1), with a discrete state space $\mathbb{S}_{\mathrm{MDP}} \subseteq \mathbb{N}$. Formally, we define this relation by an $\mathbb{S}_{\mathrm{MDP}}$-valued random variable $I_S : \mathbb{S} \to \mathbb{S}_{\mathrm{MDP}}$. I_S maps every continuous state uniquely to a single discrete state $I_S(s)$. Vice versa, every discrete state corresponds to a set of continuous states $I_S^{-1}(s_{\mathrm{MDP}})$. The discrete states can be thought of as non-overlapping regions in the continuous space and we assume the continuous states to be uniformly distributed in their regions. Additionally, for the terminal state, a special discrete state is introduced.

The regions are aligned along lanes with equally spaced longitudinal and lateral distances. As illustrated in Figure 6.7, using the road as reference frame creates a representation that is to a certain degree invariant to the geometry and, in particular, to the

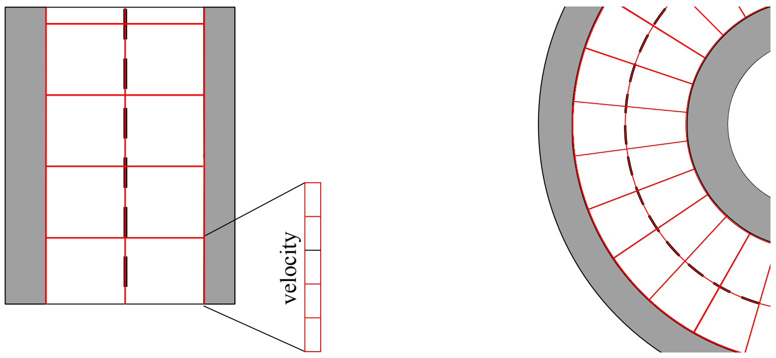


Figure 6.7: State space discretization scheme for MDP in different road curvatures.

curvature of the road. The orientation of the vehicles is not part of the discrete state because it can be assumed to be mostly aligned with the lane. In addition to this pose description, the velocities of the vehicles are discretized equidistantly.

Discrete Transition Model and Reward Function

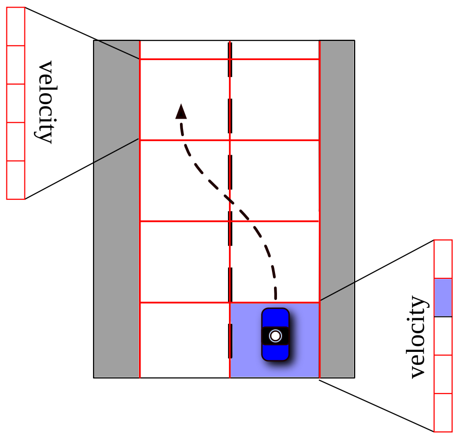
With these definitions, we are able to map continuous states to a discrete representation. However, for computing the MDP, the transition model and reward function must be defined on the discrete state space. The discrete transition model $p(s'_{\mathrm{MDP}}|s_{\mathrm{MDP}}, a)$ can be derived from the continuous transition model $p(s'|s, a)$ defined in Section 6.1 using the random variable $I_{\mathbb{S}}$. The resulting transition probability is defined as the integral of the continuous model over the current states s, corresponding to the discrete state s_{MDP}, and the next states s', corresponding to s'_{MDP} following [6.7]

$$p(s'_{\mathrm{MDP}}|s_{\mathrm{MDP}}, a) = \int\limits_{s\in\mathbb{S}|\ I_{\mathbb{S}}(s)=s_{\mathrm{MDP}}} \int\limits_{s'\in\mathbb{S}|\ I_{\mathbb{S}}(s')=s'_{\mathrm{MDP}}} p(s'|s, a)\,\mathrm{d}s'\,\mathrm{d}s\,. \tag{6.2.1}$$

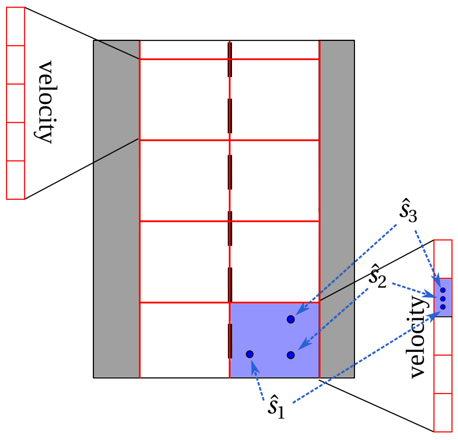
These integrals can be numerically approximated. In Figure 6.8 the approximation process is visualized for a lane change maneuver with only a single vehicle in the state space. When a discrete state is reached for the first time during planning (see Figure 6.8a), the transition probabilities are computed. To approximate the transition probabilities of a current state s_{MDP}, first, continuous states from the according region are sampled uniformly (see Figure 6.8b for three exemplary samples). These samples are predicted according to the continuous transition model $p(s'|s, a)$ (see Figure 6.8c) and mapped to the according discrete next states s'_{MDP}. Finally, the probabilities of the discrete next states s'_{MDP} given the current discrete state s_{MDP} can be determined simply by counting the samples (see Figure 6.8d).

An interesting aspect of the discrete transition model is the length of the planning time step. The requirements for the time step may vary, depending on the chosen ac-

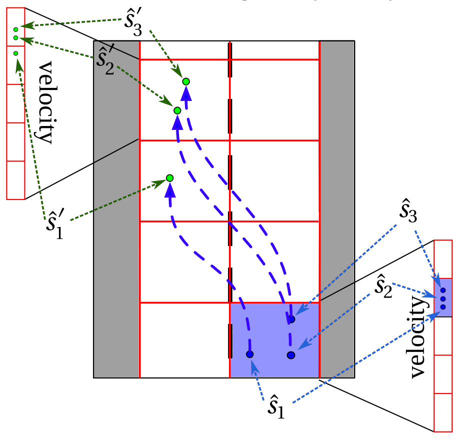
[6.7] For details on the definitions and derivations, we refer to [Brechtel et al., 2011].



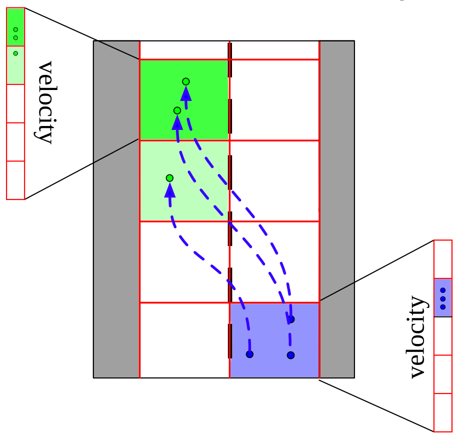
(a) Scene and target trajectory.



(b) Discretization and sampling.



(c) Transition and new follow-up states.



(d) Conditional probability for next discrete states.

Figure 6.8: State space sampling and transition.

tion. For example, the time necessary to execute a complete lane change is about 1.5 s. If the time step is too short, the resulting continuous next state can be between two lanes. This can be accounted for, by increasing the lateral resolution of the discrete grid. However, this comes at the cost of increased computational effort. Contrary to the lane change, a quick, hard braking action can endure only 0.1 s and still significantly change the state (due to the high applied forces). In this case, a shorter time step can make sense. The theory of semi-Markov decision process (SMDP) allows for actions with varying duration [Duff and Bradtke, 1995]. To compute a transition probability for the POMDP, the underlying DBN is executed several times. Therefore, it is easy to compute the transition model for different time step durations by changing the number of times that the DBN is evaluated (see Section 6.1.4). For translating the MDP to a SMDP, then, only the discount factor has to be adapted.

Applying MDP Value Iteration

At the beginning of the optimization, the set of discrete states is initialized. Typically, these are the initial states of the problem (i.e., the current situation). The following steps are repeated until convergence:

1. Compute and store conditional probabilities $p(s'_{\text{MDP}}|s_{\text{MDP}}, a)$ for all new states s_{MDP} and actions.
2. Apply MDP value iteration as explained in Section 3.3.
3. Add new s'_{MDP} that were reached in step 1 to the set of discrete states.

In every step, the discrete state space grows one step into the future until the values of the initial states converges.

The initialization of the discrete states determines whether the policy can be precomputed *offline* or is computed *online*. If the initialized states cover all possible states the agent can reach, the policy can be precomputed offline. The policy holds decisions for all states and just needs to be executed for application. In contrast to this, an online approach computes the policy when the agent reaches a state. The online computation finds only the decision for the current state. Thus, the initial state set consists only of the current state.

Online planning is faster and requires less memory because less outcomes need to be considered. In practice, however, for such a complex problem as driving that requires quick and real-time reactions, it is still too slow. Pure offline planning is infeasible because, in theory, an infinite number of states needs to be considered.

We apply a combination of both: initially, we compute a policy offline for a selection of states. This policy considers the most probable situation developments, but is by no means exhaustive. We add currently encountered states to the MDP online (during driving), analogously to the state space growing algorithm described above. With this online-offline approach, the policies for new states can be quickly computed as they benefit from the previous results through dynamic programming.

6.2.2 Continuous POMDP with Representation Learning

In the MDP approach of the previous section, high efforts are necessary to convert models from the continuous to the discrete space and vice versa. When considering partial observability with traditional discrete methods, additional modifications become necessary: despite the reward function and the transition model, the observation function would have to be discretized, too. This involves finding an adequate discrete representation of the observation space. The results of our evaluation of the discrete MDP approach in Section 7.2 suggest that such an approach does not scale to complex, urban driving problems.

Contrary to this, when using a continuous solver that directly works in the continuous spaces, the integration of the driving task is much easier. The continuous POMDP method presented in Chapter 5, for example, does most of the work automatically. It learns a suitable state space representation, directs the exploration into promising areas of the belief space and reduces the (potentially infinite) number of observations when computing the expectation. As a consequence, the spaces and models introduced in Section 6.1 can be directly plugged in.

Simplifications However, a badly chosen continuous space can lead to high computational cost, even when using an efficient continuous solver. For this reason, it can be sensible to simplify the representation of the driving problem. It is possible, to reduce the state space $\mathbb{S}$ before inserting it into POMDP value iteration, in order to reduce the complexity of the continuous POMDP.

When considering a scenario involving the ego vehicle and more than three other cars, the resulting continuous state space $\mathbb{S}$ has dimensionality greater or equal $4 \times (4+1) = 20$. Such a high dimensionality, is not only an issue for continuous solvers because of the enormous complexity (see, e.g., Section 3.2.2 and Chapter 5). Even the state estimation with sequential Monte Carlo (SMC) is pushed to its limits by this problem. Most traffic scenarios, however, can be simplified by selecting two other cars that are relevant, for example because they define a gap that the ego car can merge into. Also, the representation of positions and poses can be simplified by aligning them to lanes, similarly to Section 6.2.1. The results of the solver can be improved, not only by reducing the dimensionality. Research in assisted and autonomous driving came up with several meaningful *indicators* for driving safety, for instance, *time to collision (TTC)* and *headway* [Vogel, 2003]. Even though these indicators can be directly derived from the full state space, they make a meaningful connection between variables. If not explicitly given, a POMDP solver must first learn these relationships.

For example, TTC is the time until a collision happens under the assumption of constant speeds. It is simply defined by $\text{ttc} = \frac{\text{distance}}{\text{relative speed}}$. Nevertheless, a solver that does not have this background knowledge must find these relationships between speed and distance, first. If such indicators are directly inserted into the POMDP solver, representation learning in our approach is indirectly facilitated.

Simplifications of the state space improve the performance of a solver, if they improve generalization (e.g., by reducing the dimensionality or by creating smoother value functions so that the inductive bias applies better). As long as these state space reductions do not influence the accuracy of the value function representation, the resulting POMDP policy remains unchanged.

For special tasks that are less complex than autonomous driving, such as some driver's assistance systems, the state space dimensionality can be reduced even more. For example for the most simple variant of an *adaptive cruise control*, the state space can be reduced to the difference in velocity and distance between the ego and the lead vehicle. Other vehicles, lateral distances, etc. are not of interest for this task.

On the one hand, these simplifications can enormously speed up computation. On the other hand, most of them need to be manually applied for different tasks and traffic scenarios, which contradicts the general approach. In this work, we focus on the general solution to the driving problem. For this reason, in Chapter 7, we evaluate the policies created by solving the general model without simplifications.

6.3 Conclusion

In this section, we present a model for predicting the development of traffic situations that is suitable for planning tactical driving decisions (Contribution 2). It comprises several aspects, including the stochastic behavior of drivers, the interaction between road users, and partial observability. We propose to simulate these complex relationships in a simplified way rather than providing physically correct models, such that they suffice for tactical decision-making. The focus is on modeling decision relevant aspects such as the invisibility of hidden objects.

We formulate the tactical decision problem as a continuous POMDP by extending this model with driving actions that allow the agent to influence developments and a reward function that comprises several driving objectives. This continuous POMDP is directly based on the poses and velocities of the involved road users. Using a hierarchical DBN representation, we integrate abstract aspects like drivers routes with low-level aspects like car physics in a sound way. This way, we provide an end-to-end probabilistic formulation of the driving problem (Contribution 2b).

Based on this continuous POMDP model of driving, we present two approaches for decision-making. The first employs discrete MDP value iteration to generate decision policies. It discretizes the continuous space with predefined rules and neglects partial observability. The second approach applies the method developed in Chapter 5 to directly solve the continuous POMDP and learn a representation. No modification of the underlying model is needed when applying this method to different scenarios and situations. We realize both presented approaches and, in the next chapter, evaluate them for different tasks. Policies for highway driving as well as multiple urban driving examples, including junctions and merging into moving traffic with blocked view, are generated using the models presented in this chapter (Contribution 2a).

Chapter 7

Evaluation

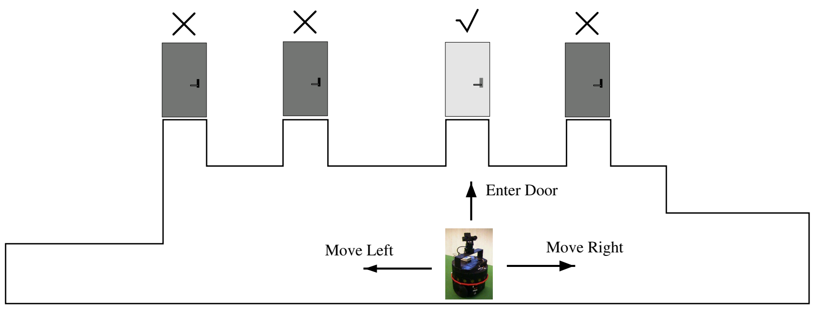
We evaluate the presented method for solving continuous POMDPs as well as the proposed approach for tactical-decision making in traffic.

The focus of the evaluation is on real-world applications from the robotics domain. Starting with synthetic continuous POMDPs, through dynamic obstacle avoidance and highway driving with full observability, we conclude with urban driving scenarios exhibiting partial observability and interactions between road users. Finally, we come back to the initial example of our introduction, where the autonomous vehicle has to merge into moving traffic while its view is partially blocked. The solution of this task is an unconventional behavior policy.

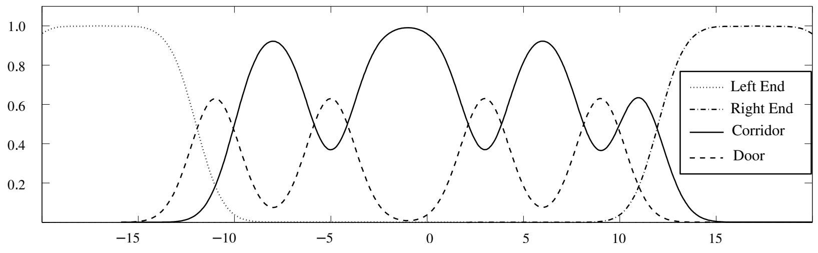
Chapter Overview First, in Section 7.1, we compare the continuous POMDP method with existing techniques. We also take a closer look at its convergence properties and how it performs for obstacle avoidance problems. Then, we move the focus from general robotics to driving in traffic. In Section 7.2, we evaluate our general approach to decision-making in traffic for highway scenarios using value iteration for discrete MDP. While it can be sensible to assume full observability for highway driving, this is not true for urban traffic. Lastly, we test the general model for driving in urban environments by simulated real-world examples exhibited in the city of Karlsruhe. The continuous POMDP solver presented in this thesis is employed to automatically solve these scenarios that are complicated due to interaction between and limited visibility of road users.

7.1 Continuous POMDP Evaluation

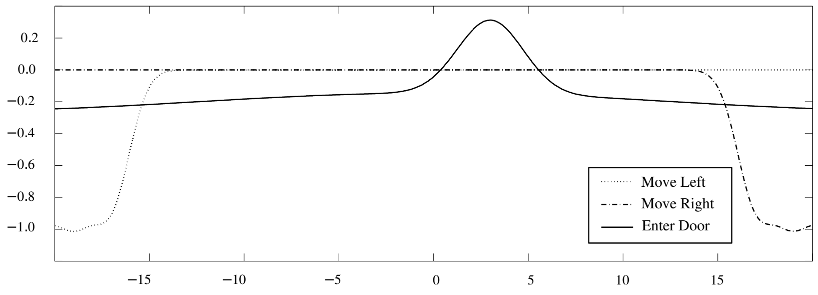
In this section, we evaluate the performance of the method presented in Chapter 5 on four continuous POMDP problems. First, we compare its performance to state-of-the-art methods for a 1D continuous corridor problem from the literature. We extend the corridor problem with a second dimension, to evaluate the convergence properties and our solvers ability to reduce the dimensionality of the problem by learning a lower-dimensional representation. The general ability to solve higher-dimensional, more



(a) Problem introduction.



(b) Discrete observation model.



(c) Reward function for the three actions.

Figure 7.1: 1D corridor problem: introduction [Porta et al., 2005]. Reprinted from Porta, J.; Spaan, M. & Vlassis, N., "Robot planning in partially observable continuous domains," in Robotics: Science and Systems I, edited by Sebastian Thrun, Gaurav S. Sukhatme, and Stefan Schaal, ©2005 Massachusetts Institute of Technology, published by The MIT Press, page 217.

realistic robotics tasks is tested by a 8D obstacle avoidance problem. Lastly, we modify this problem by assuming a circular moving obstacle in order to vividly demonstrate the approach's ability to solve non-linear models. Some of the results in this section are based on [Brechtel et al., 2013].

7.1.1 Comparison with Existing Methods—1D Corridor Problem

The first POMDP that we evaluated our solver in models a robot moving in a one-dimensional corridor limited by walls (see Figure 7.1). The aim of the robot is to open the right door, while perceiving its own position only uncertainly with a discrete number of normal distributions. See [Porta et al., 2005] for more details on the POMDP task.

Our solver was able to clearly exceed the results of state-of-the-art algorithms for this toy problem. In Figure 7.2, a selection of time steps from a run of the policy cre-

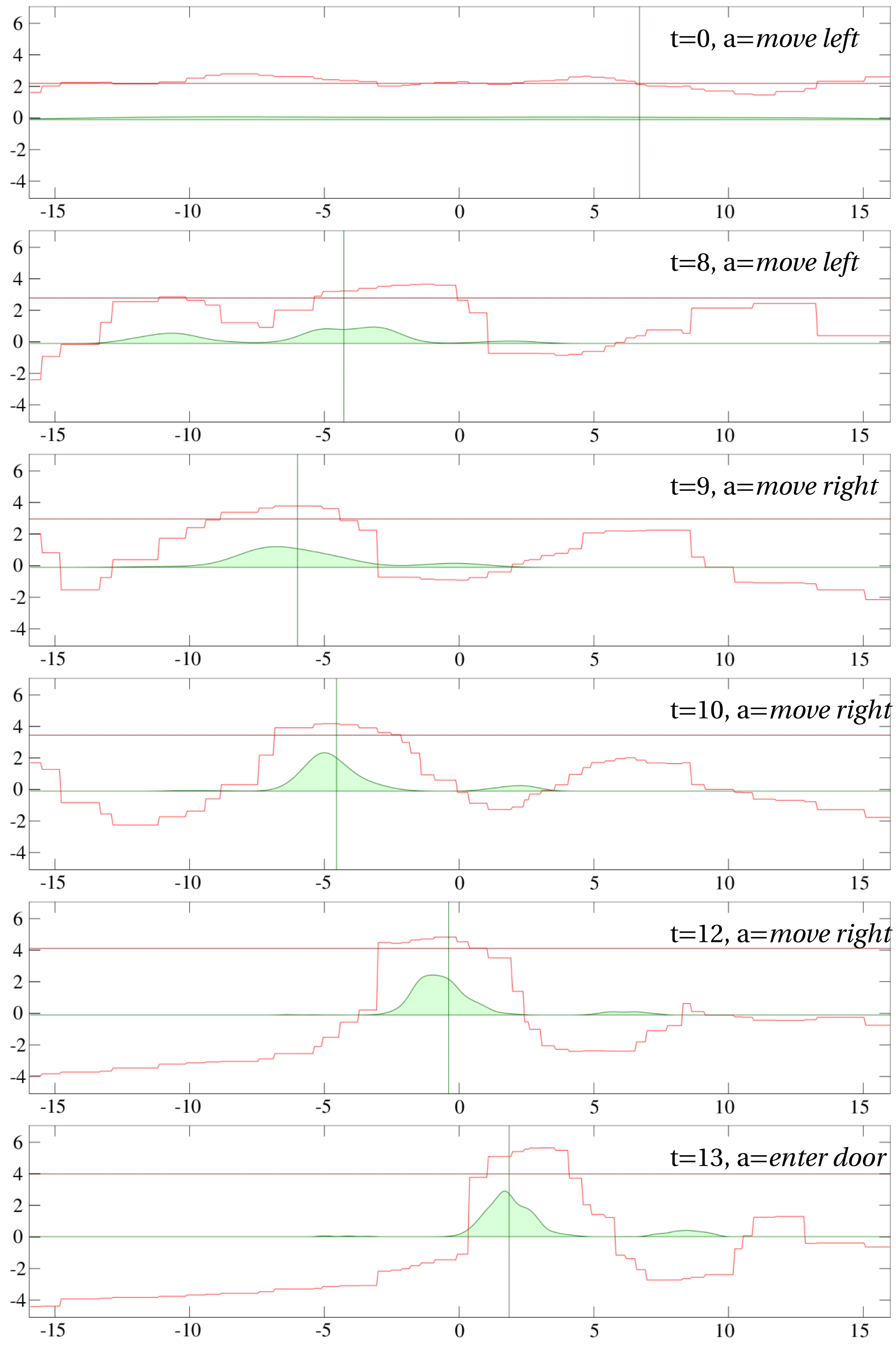


Figure 7.2: 1D corridor problem: policy simulation showing the beliefs (green), the true state (green vertical line), the best α-function (red) and the expected value for the belief (red horizontal line) at time steps $t = 0, 8, 9, 10, 12$, and 13.

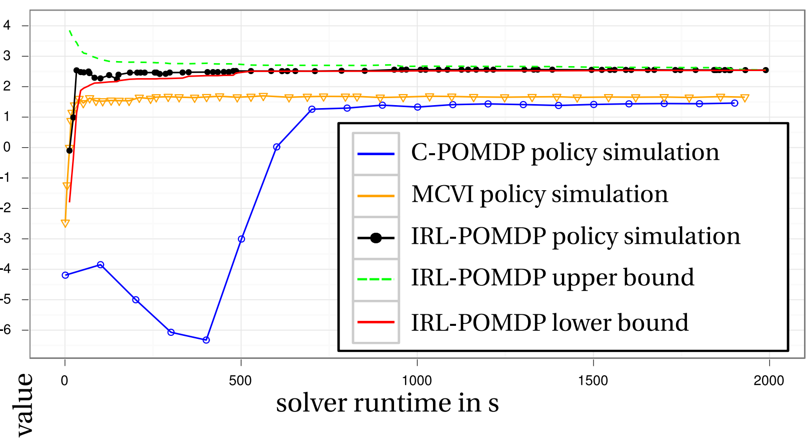


Figure 7.3: 1D corridor problem: comparison of the presented IRL (Iterative Representation Learning)-POMDP with state-of-the-art methods.

ated with the solver presented in this thesis is shown. Initially, the robot position is very uncertain. The basic policy is to move towards the corners first. Near the corridor walls, the observations are clearer than in the center. However, the full policy is more complex, as it utilizes the multimodalities that have their origin in the discrete observations. At $t = 8$, the belief is tri-modal. In $t = 9$ the leftmost mode is eliminated. This did not happen due to measuring the object position. It was discarded because *left end* was not measured. This ability to benefit from not measuring something will become even more evident in later experiments.

For assessing the overall performance of the system, in Figure 7.3, the values yielded in simulation after different solving times of the presented solver are compared to MCVI [7.1] from [Bai et al., 2010] and C-POMDP (PERSEUS) from [Porta et al., 2006] using their open source implementations and parameters. Therefore, we ran 10.000 trials simulating 100 consecutive time steps. Previous solvers only reach suboptimal average rewards after reasonable solving time. After 12 h MCVI did not exceed an average reward of 1.8. In a few seconds our algorithm finds a clearly better policy. The lower bound converges after 16 min with an average reward of about 2.55. After ca. 33 min the upper bound converges, too, and the algorithm stops. Note that the upper bound only validates the result that the lower bound achieves and has no influence on the resulting policy. This result was achieved despite the smooth nature of the normal distributions in this example, which are close to a worst case for the proposed piecewise-constant representation. In contrast to this, the representation in C-POMDP is directly based on these normal distributions. Still, it cannot achieve the performance of the presented algorithm.

[7.1]Release 0.2, 17 May 2012.

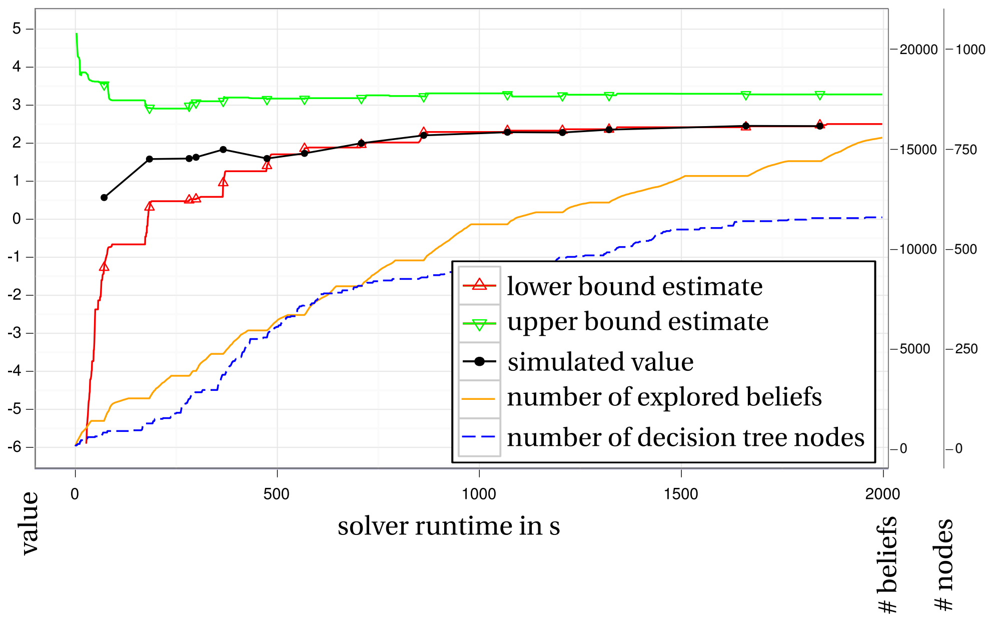


Figure 7.4: 2D corridor problem: convergence analysis.

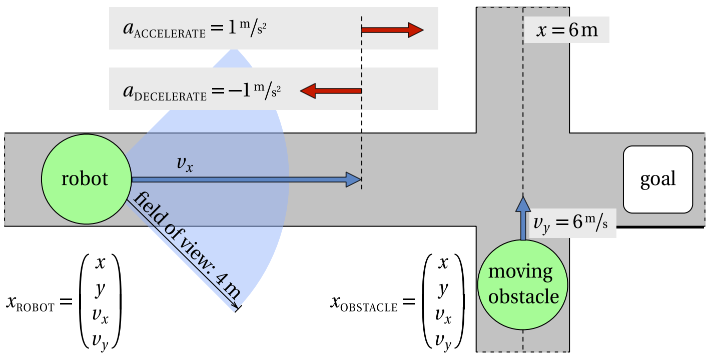
7.1.2 Convergence analysis—2D Corridor Problem

Additionally, we converted the 1D corridor problem into a 2D problem. As this makes the corridor problem more difficult, it is more suitable to analyze the convergence properties of the solver. The robot can additionally go up and down. The 12 observations are products of densities of $\mathcal{N}(-3,2)$, $\mathcal{N}(3,2)$ and $\mathcal{N}(0,100)$ in x_1 with the 4 original 1D observations in x_0. The reward only depends on x_0. Basically, x_1 only distracts the solver and, eventually, the reached value is similar and only slightly lower, due to the changed probability distribution. However, the combinatorial complexity of this problem is much higher due to the increased number of observations.

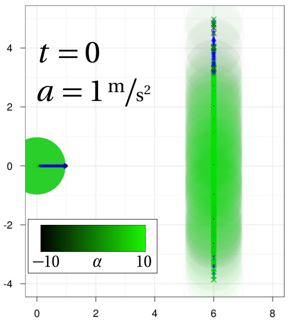
Results of the analysis are given in Figure 7.4. Interestingly, the value bounds are not monotonic. The upper bound often drops quickly at the beginning and corrects itself when a conflict detecting belief is explored. For the same reason the simulated value is at some points below the lower bound. The belief exploration slightly slows down over time because the backups slow down. This is due to the increasing complexity of the policy and its representation. The growth of the decision tree (i.e., the discrete representation of the α-functions) convergences after some time. It has reached maximum detail at about 600 nodes. Note that the number of discrete states of the representation is lower, as only leaf-nodes in the decision tree encode discrete states.

7.1.3 Higher Dimensional Problems—8D Obstacle Avoidance

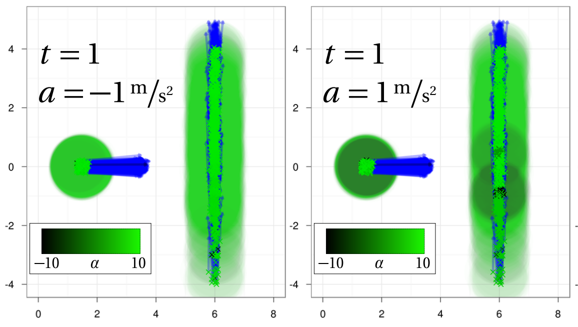
In the scenario in Figure 7.5, a robot agent and an obstacle move on a 2D plane according to a constant velocity model with small white noise. The positions and velocities of both objects put together result in an 8D joint state space. The observations are


$a_{\text{ACCELERATE}} = 1\,\mathrm{m/s^2}$
$a_{\text{DECELERATE}} = -1\,\mathrm{m/s^2}$
$x = 6\,\mathrm{m}$
robot
v_x
field of view: 4 m
goal
$v_y = 6\,\mathrm{m/s}$
moving obstacle
$x_{\text{ROBOT}} = (x, y, v_x, v_y)$
$x_{\text{OBSTACLE}} = (x, y, v_x, v_y)$

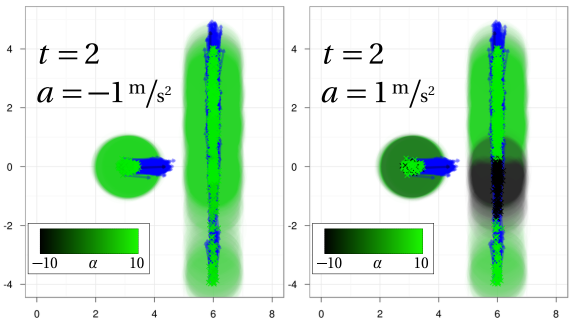
(a) Sketch of the 8D obstacle avoidance problem.



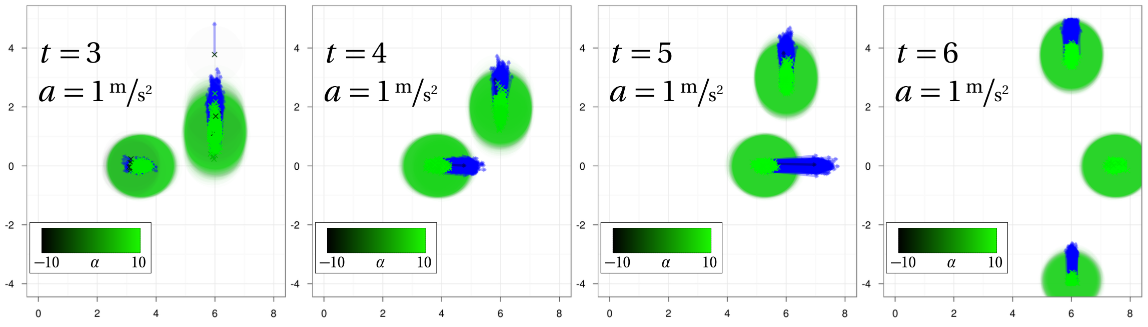
(b) Initial belief.



(c) Alternative actions in $t = 1$.



(d) Alternative actions in $t = 2$.



(e) Simulation and dominating α-functions for $t = 3$ to 6.

Figure 7.5: 8D obstacle avoidance: α-functions for alternative actions. In $t = 1$ and 2 action $a = -1\,\mathrm{m/s^2}$ is chosen. Velocities are indicated with blue arrows.

continuous 8D measurements of the state with additive white noise. For collision detection, the objects have circular shape with a radius of 1 m. Their positions are loosely bounded by *roads*. Starting at $\left(x, y, v_x, v_y\right)^\mathsf{T} = \left(0\,\mathrm{m}, 0\,\mathrm{m}, 1\,\mathrm{m/s}, 0\,\mathrm{m/s}\right)^\mathsf{T}$ the robot has to cross the intersection without hitting the moving obstacle by accelerating and decelerating. The robot receives a reward of +10 for reaching the goal and −10 in case of a collision. The discount is $\gamma = 0.95$. If the obstacle moves past $y = 8\,\mathrm{m}$, it is *respawned* at $y = -8\,\mathrm{m}$, so that it remains a constant threat.

The difficulty of the problem is that the y-position of the obstacle is not known in advance and the robot's viewing distance is limited to 4 m. If the robot can see the obstacle, he can see it with a standard deviation of $\sigma_y = 0.32$.

The policy found by the solver after 543 s achieves an average score over 5.0 in simulation using 400 particles to represent the beliefs. The final representation of the space

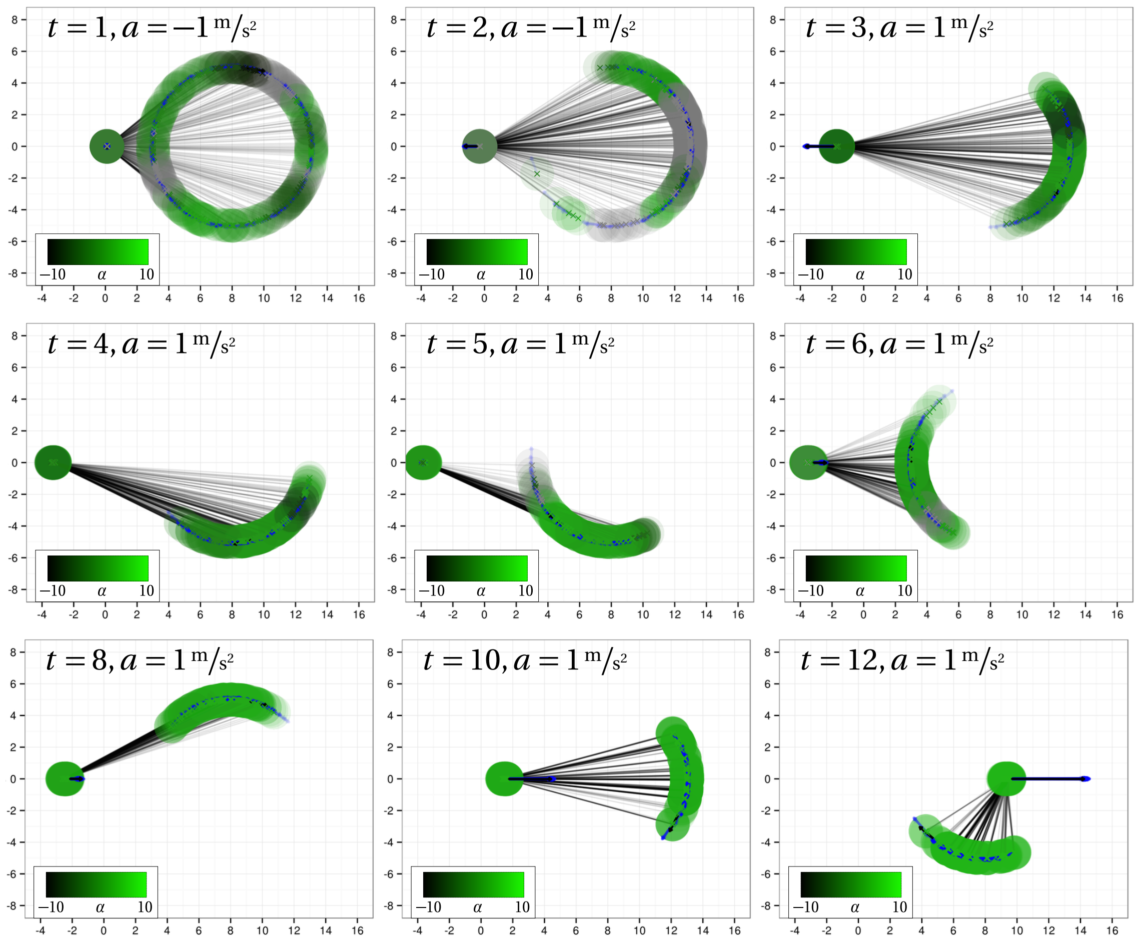


Figure 7.6: 8D obstacle avoidance with circular traffic. Velocities are indicated with blue arrows and state associations with connecting lines.

uses a total of 487 discrete states. A fixed discretization of an 8D space with just four regions per dimension would already yield 65 536 states. Apparently, the iterative refinement discovered an efficient representation. The final policy is encoded by 41 α-functions describing the behavior shown in Figure 7.5: the robot slowly approaches the intersection to be able to stop in a safe distance where he can perceive the y-position of the obstacle. Deceleration has to start several steps before the collision could happen. Otherwise the robot would not stop in time. In the example in Figure 7.5, the most important decisions are to decelerate at $t = 1$ and 2. The α-functions for the acceleration actions at these time steps show a high probability for low values (black areas). Then it accelerates instantly or moves back first to pass the obstacle.

7.1.4 Non-linear Models—8D Obstacle Avoidance with Circular Motion

To demonstrate the generality of our approach, we modified this example by assuming a circular moving obstacle (similar to a car driving in a traffic circle). Figure 7.6 shows that the obstacle moves in clockwise direction. Initially, the object could be anywhere on the circle. Note that this belief cannot not be represented well by approaches based on normal distributions. For our particle-based approach, however, there is not much difference to the previous example. Similar holds for our discrete representation learning approach that is not restricted to special families of functions.

The evaluation shows a policy that is similar to the previous example. The ability of our approach to sort out not-measured obstacle positions can be nicely seen in this example. At $t = 3$ the robot already obtained enough knowledge for planning how to pass the object. At that point in time, it has not yet seen the obstacle. Still, it could make out a gap that is large enough to fit through only by excluding obstacle states.

> The developed solver shows state-of-the-art performance and is capable of solving higher-dimensional, driving-like continuous POMDPs.

7.2 MDP-based Highway Driving

Highways are one of the few traffic environments where partial observability can be mostly ignored. For this reason, we tested and analyzed the general driving model from Chapter 6 using the MDP approach with growing state space presented in Section 6.2.1 for highway scenarios rather than urban driving. In this scenario, the ego vehicle can choose a lateral action *keep/change lane* in combination with an acc-/deceleration.

This section is based on the results of [Brechtel et al., 2011].

7.2.1 Empirical Testing

We evaluated the policy of the MDP solver for a typical highway scenario as well as driving with oncoming traffic in a 3D simulation environment (see Figure 7.7). The simulation environment allows to realistically evaluate situation developments by accurately simulating sensor data and driving physics. Vehicles other than the ego vehicle were manually driven to guarantee close to natural driving behaviors.

In Figure 7.8 a simulation run for normal highway driving is shown. At $t = 1$ the road is blocked. The ego car comes to a full stop in appropriate distance. When the yellow car drives on, it has enough space to change lane and pass it. Then, it also overtakes the second slower driving vehicle. At $t = 5$ it returns to the right lane because a cost is given for not driving in the rightmost lane.

Driving with oncoming traffic requires a different policy. However, due to the general formulation, only the driving direction has to be changed in the left lane for generating this new policy. Figure 7.9 shows an exemplary run of the policy. Until the oncoming traffic has passed the ego vehicle, it follows the slower car in an appropriate distance

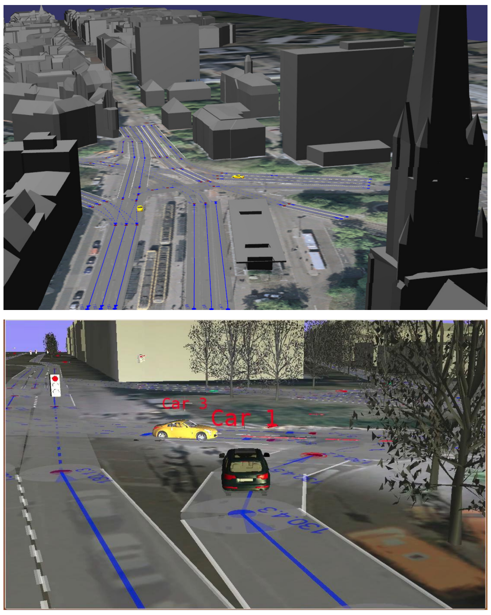


Figure 7.7: 3D simulation of scenarios in Karlsruhe.
Map Data [City of Karlsruhe].

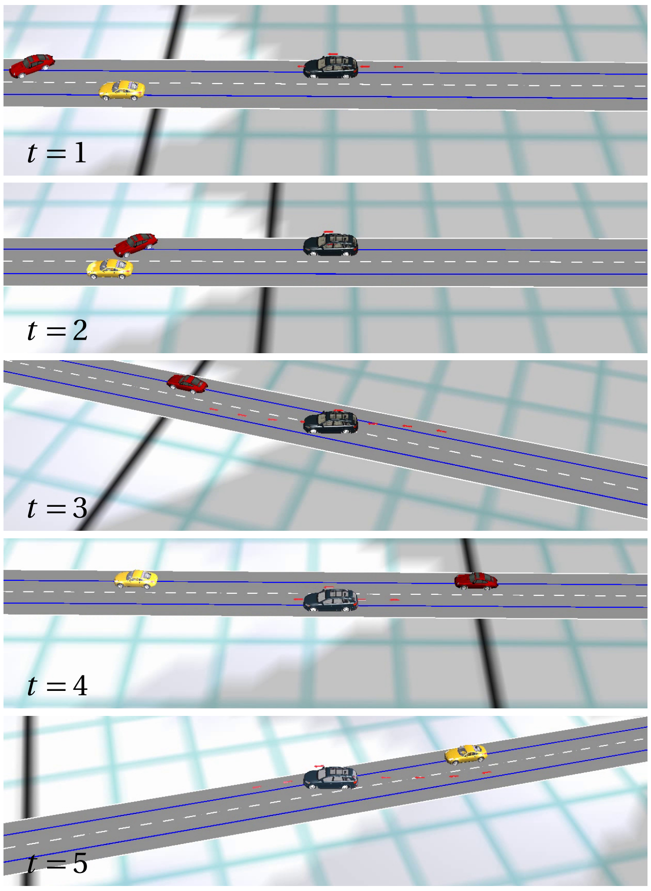


Figure 7.8: MDP highway driving: empirical policy testing with manually controlled traffic.

such that it is able to smoothly and quickly accomplish the subsequent overtaking maneuver.

7.2.2 Policy Analysis

The plots in Figure 7.10 show the MDP policy when there is no oncoming traffic and just a single vehicle stopping in the right lane. The x-axis of the plot shows the longitudinal distance between both cars. Positive distance means that the ego vehicle is behind the other vehicle. The current speed of the ego vehicle can be seen on the y-axis.

Figure 7.10a and 7.10b show the reaction of the ego vehicle, when driving on the right lane like the other vehicle. A lateral speed of 2 m/s is equivalent to a lane change to the left. The car changes lane in a safe distance, which depends on its own speed. At the same time it reduces acceleration because physical constraints limit accelerating and applying high longitudinal forces simultaneously. If too close, it makes an emergency stop without changing the lane to be able to apply more braking forces. An evasive maneuver is already too risky at that distance. Once the ego vehicle has passed the other vehicle, it accelerates. In Figure 7.10c, the predicted value (i.e., expected future

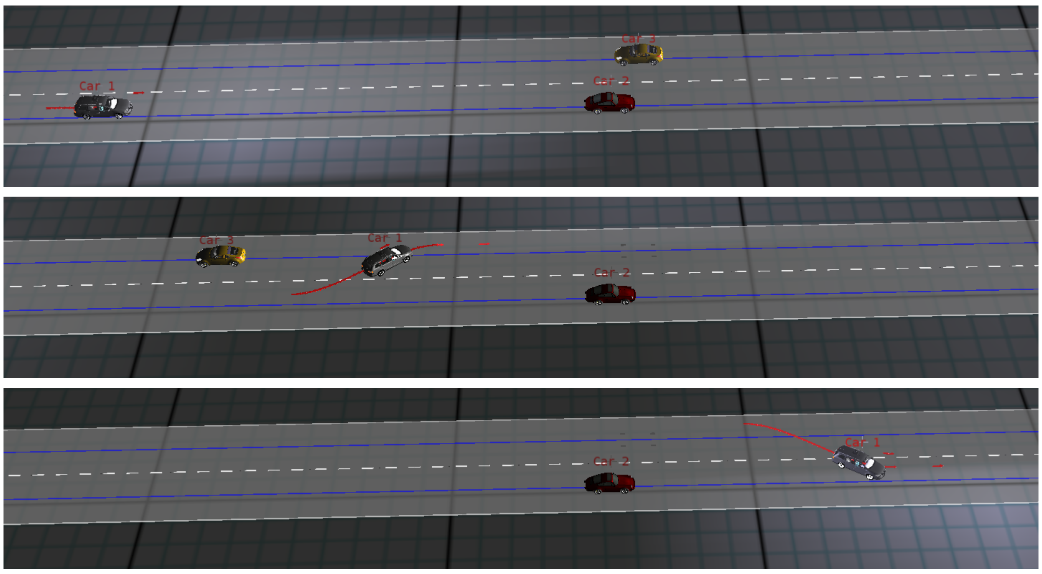


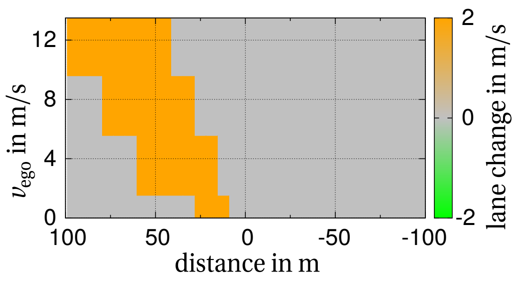
Figure 7.9: MDP highway driving: overtaking with oncoming traffic. The ego vehicle is grey. The trajectories visualizing its decisions are indicated in red [Brechtel et al., 2011]. ©2011 IEEE

reward) for these situations is shown. The closer the ego vehicle approaches the other car, the lower is the value, as there is a chance that the other car suddenly brakes or changes lane and the ego vehicle cannot react in time. In the black areas, where the value is close to −1000, accidents are unavoidable. As can be seen in Figure 7.10d, the car returns to the right lane, when it has reached a safe distance to avoid the cost of driving in the left lane.

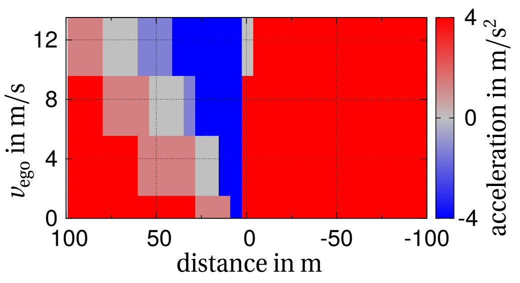
> Decisions, such as what an appropriate safe following distance is or when to overtake and when to stay behind a slower vehicle can be concluded automatically by solving the decision process.

Influence of Cooperative Interaction The simple example in Figure 7.11 already shows the necessity of sophisticated models for predicting other drivers. The situation is basically the same as in the previous example, except that the other vehicle drives with 7 m/s. The safety buffer shrinks compared to the stopping obstacle because there is more time to brake. While the result in Figure 7.11a was generated using the complex DBN behavior model, in Figure 7.11b, a simpler model was used that is frequently found in the literature. This model assumes that the other driver just holds lane and velocity.

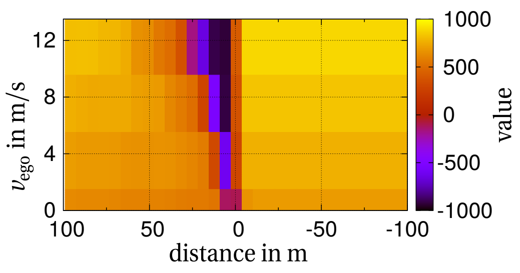
In this example, the area with negative distances (when the ego vehicle is behind the other vehicle) is of special interest. With the complex behavior model, the ego car does nothing. It expects the other car to react and overtake. Using the simple model, the ego



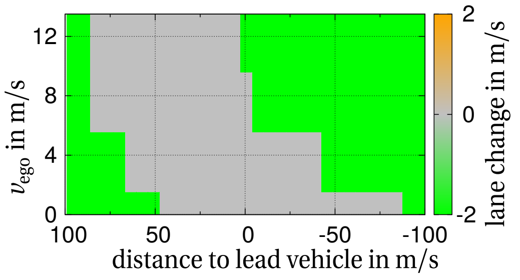
(a) Lane change action (in right lane).



(b) Acceleration action (in right lane).



(c) Expected value (in right lane).



(d) Lane change action (in left lane).

Figure 7.10: MDP highway driving: value and actions when another car stops in the right lane.

car changes to the left lane to avoid the incoming vehicle. In reality this is undoubtedly dangerous because quickly approaching cars most certainly want to overtake.

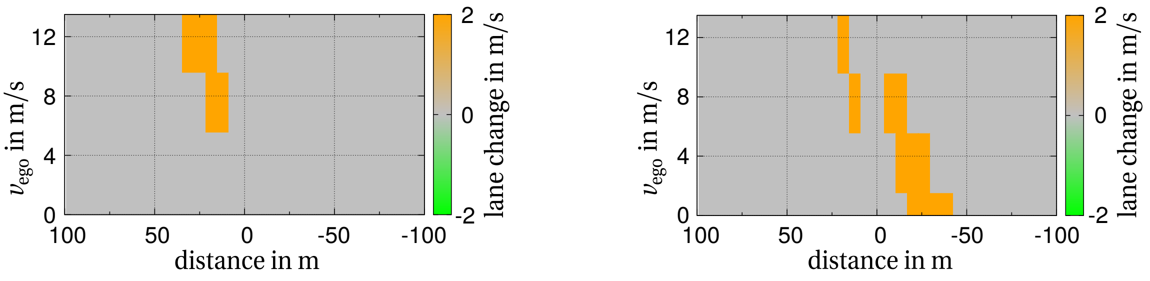
Interaction between vehicles must be considered for safe driving policies.

7.2.3 Space Representation in Decision Processes

One of the most important aspects, when modeling tasks as (PO)MDPs is the representation of the state space. The consequences of a badly chosen state space are either huge computational overhead, which renders computation infeasible, or a too coarse representation of important details, which results in malformed and unsafe policies. The example of highway driving under the assumption of full observability shows that even for simpler driving tasks, it is difficult to find a good discrete representation. Taking the above results of the MDP approach with predefined discretization rules, we illustrate this issue, the consequences and draw conclusions.

Figure 7.12 shows the value (the expected sum of discounted future rewards), when driving in the right lane together with a stopping vehicle (compare with Figure 7.10c). The latitudinal discretization in this example is 6 m. In longitudinal direction, only the two lanes with width 6 m are differentiated. $4\,{}^{\mathrm{m}}/_{\mathrm{s}}$ steps for the velocity of the vehicles are used.

On the one hand, it is evident that the results could have been improved if a more detailed discretization had been used. Especially in the area around $\boxed{\text{A}}$, where the



(a) Complex behavior model.

(b) Simple *free ride* behavior.

Figure 7.11: MDP highway driving: lane change actions with and without complex behavior model when another car is driving in the right lane with 7 m/s.

distance is small and the chance of a collision is quite high, a finer granularity would have been useful to assure a correct reaction. The aliasing effect in this area is clearly visible. On the other hand, many states were discriminated although their value and as a consequence their policy is very similar. For example in area $\boxed{\text{B}}$, where the ego vehicle is driving away from the standing vehicle, the precise distance is not important.

Due to this inefficient discretization the MDP state space contained several thousands of states. The worst case complexity for the running time of a single MDP value iteration or policy iteration step is quadratic in the cardinality of the space (recall Section 3.3). A discretized space that is better suited to the problem, for example, a linear combination of the distance and the velocity as sketched in Figure 7.12, would be able to represent more details and at the same time reduce the cardinality of the space. The idea behind this discretization is that it only distinguishes states that yield different values. The ability to represent more relevant details improves the quality of the resulting policy and consequently yields higher average rewards.

Partial observability even increases this complexity to a point where it is practically impossible to solve driving problems, if an unsuitable representation is used (see Section 3.2.2 for the reasons). What constitutes a suitable or *good* representation, however, differs very much depending on the underlying problem and situation. If, for instance, one-lane driving is considered and the other vehicle approaches quickly from behind, negative distances (area $\boxed{\text{B}}$) need to be differentiated, too. In this case, the other vehicle cannot change lane. Another example are changes in the behavior models: if the uncertainty in the acceleration that drivers apply is increased, the safe following distances also increase. Consequently, a discrete state space must represent decisions in greater distances. Any difference in the models or the initial situation of the (PO)MDP possibly entails a different optimal representation.

> An optimal representation is a property of every special decision process and situation, just like the policy and the value function are. Variations in the value function are an appropriate indicator for determining relevance.

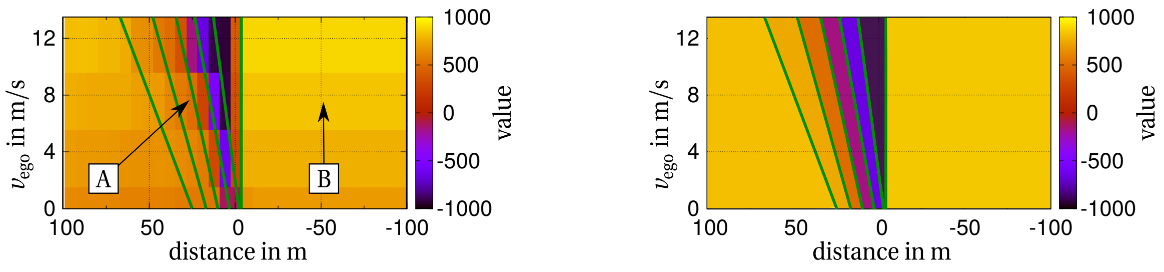


Figure 7.12: MDP value when approaching a standing vehicle (both vehicles in the right lane).

The method we develop in Chapter 5 exploits this insight by iteratively learning a suitable representation for the problem from the value functions.

7.3 POMDP Decision-Making for Urban Driving

In this section, we evaluate the POMDP driving policies in different urban scenarios. To assure the practical significance of these examples, all scenarios are taken from real-life traffic situations in Karlsruhe. We selected scenarios where simpler approaches would fail.

The driving policies are generated by solving the general continuous POMDP model for driving proposed in Chapter 6 using the novel method presented in Chapter 5. The same state, action, and observation space, transition and observation models, and reward function were used for all examples. To generate driving policies for new situations, only the initial belief b_0 (i.e., distribution encoding the knowledge about the other road users and the ego vehicle), the driving goal and static background knowledge about the current environment (i.e., map data) have to be given.

To illustrate the resulting policies, we give examples of different developments for every scenario. Additionally, we provide evaluations that throw light on interesting aspects, such as the influence of partial observability or how the low-dimensional state space representation, which is learned by the POMDP method, looks like.

Notation In order to visualize the developments over time, we plot the agent's belief at different time steps. See, for example, Figure 7.14 for an intersection scenario or Figure 7.20 for a full run from the initial belief to the time step where the agent reaches the goal. Usually, we display only a selection of beliefs at interesting time steps. The initial belief is given by the plots at time step $t = 0$. We directly plot the particles of the beliefs: arrows visualize the orientation and the speed, dots the rear-axle position, and the vehicle silhouette its extensions. The *true state* is highlighted by the *car i* labels. *car* 0 is the ego vehicle. Remember that the POMDP agent does not know the true state. It is just given for illustrative purpose. Lines of sight show whether an object is visible

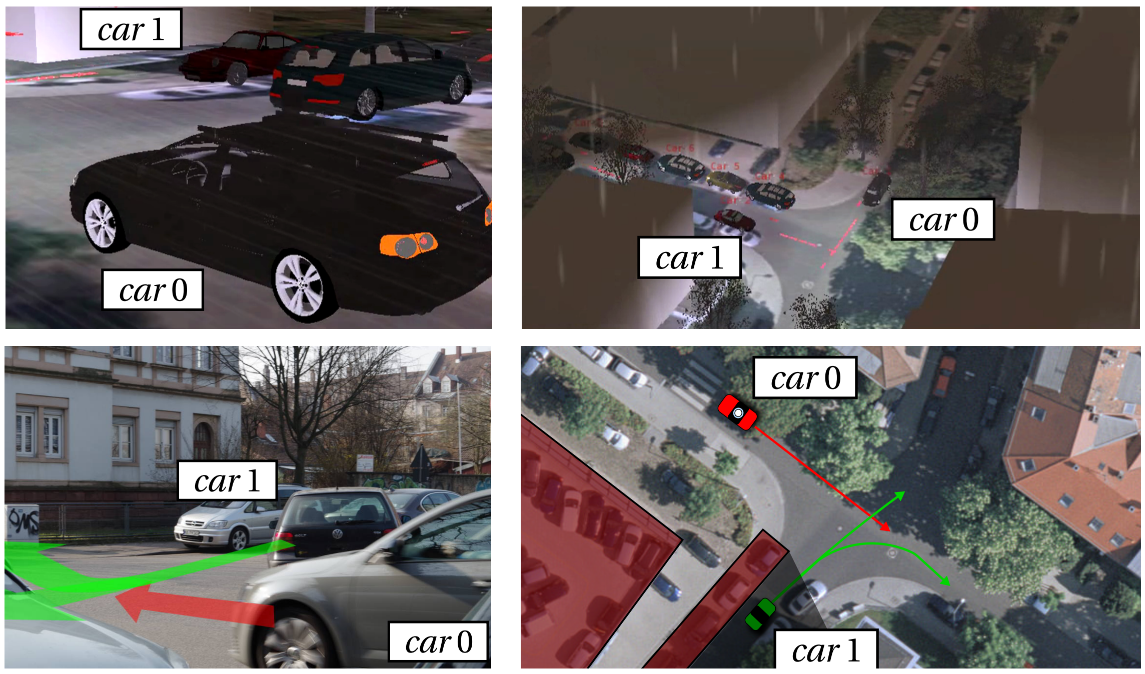


Figure 7.13: Intersection scenario with blocked view: introduction. Map Data [City of Karlsruhe].

to the ego vehicle (green) or not (red). Additionally, the association of the car's states can be seen from these connecting lines.

Model Parameters If not mentioned otherwise, the following parameters for the POMDP model are used. The timestep is 1.5 s. The initial beliefs are uniformly distributed in all dimensions. The agent can choose to accelerate or decelerate with 1 $^{\mathrm{m}}/_{\mathrm{s}^2}$ or hold speed.

In the transition model, we assume additive white noise for the acceleration applied by other drivers. Noise is also added to the position of the vehicle, lateral and longitudinal to the vehicles driving direction. Both, the noise in the velocity and the position, are scaled with the velocity and normally distributed.

If a car is visible, the variance of the position observation in both dimensions is $\Sigma_O = 1\,\mathrm{m}^2$. The velocity and the orientation are assumed to be hidden to the sensors. They have to be inferred through Bayesian filtering.

A reward of 5 is given, if *car* 0 reaches the goal area and a negative reward of -1 else. If the ego vehicle collides at any point in time, it receives a negative reward of -10 and the run is terminated so that no positive rewards can be received afterward. Additionally, to achieve efficient and comfortable driving, a small negative reward is given for braking because kinetic energy is lost.

7.3.1 Influence of Partial Observability—Intersection with Blocked View

The first example is a typical urban intersection that shows how crucial partial observability is for driving. In Figure 7.13 the situation is shown. The self-driving vehicle (red vehicle) wants to cross the intersection, but big parts of the environment are hidden behind a building and parking cars. However, the covered areas are very important because the ego vehicle must yield to cars coming from the right which can be initially hidden. We compare policies that were created considering this partial observability (Case a) and neglecting it (Case b).

In Case a (see Figure 7.14), two simulated runs are shown. Initially, at $t = 0$ of both simulation runs, the position of the other vehicle *car* 1 is not known.[7.2] The basic strategy of the POMDP is to stop in front of the intersection until the ego vehicle can see enough to safely cross it. In Simulation 1, this takes 6 time steps or $6 \times 1.5\,\mathrm{s} = 9\,\mathrm{s}$ (see Figure 7.14b). Note how the probability mass above *car* 1 has disappeared. The reason for this is that the system is not only able to directly measure *car* 1, if it can see it. It also excludes states where *car* 1 would have been visible, but has not been measured. After *car* 1 has passed, the ego vehicle crosses the intersection.

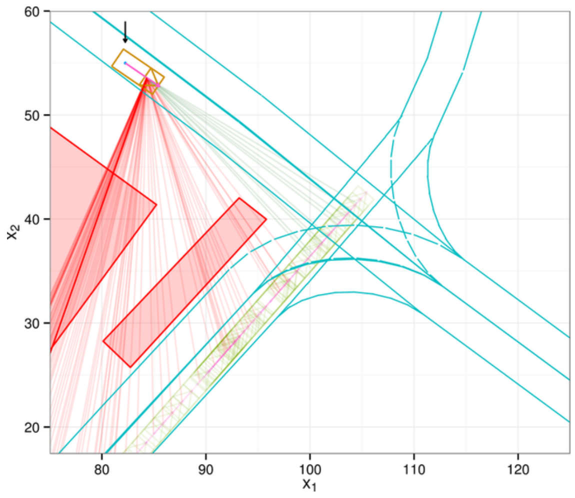
In Simulation 2, *car* 1 can be seen early at $t = 3$, so that the ego vehicle does not have to fully stop and the goal is reached earlier. Another difficulty arises from the fact that *car* 1 can either take a right turn or go straight, as Figure 7.14f shows. In this situation the consequences of this uncertainty in the transition model are minor for the decision-making. However, this is not necessarily the case for other situations so that the POMDP solver first has to find out that it can be neglected, here.

Case b in Figure 7.15 illustrates the influence of partial observability on decision-making. We assumed that the view is not blocked, but otherwise exactly the same scenario as before.[7.3] Then, the high number of possible developments of the previous case reduces mainly to two: the crossing can be passed before *car* 1 (see Simulation 1) or the ego vehicle has to slow down and wait for *car* 1 to pass (see Simulation 2). In Figure 7.15e these two developments are clearly visible. After integrating the next measurement of the vehicle, the mode in the estimate that *car* 1 takes a right turn is discarded.

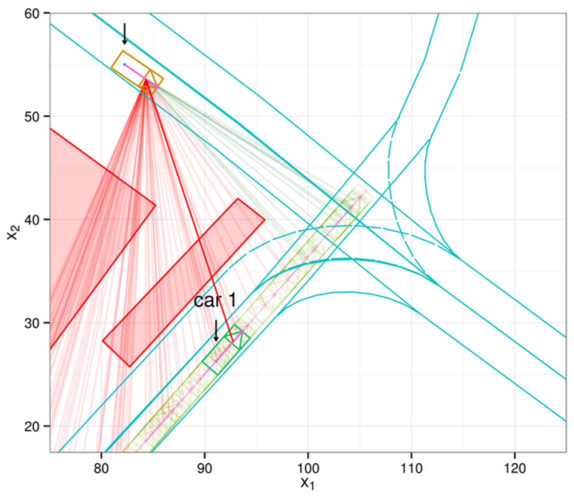
See Figure 7.16 for an illustration of the yielded values in both cases for a total of 1000 simulations. There is a much greater variety of developments due to the higher uncertainty, in Case a, where visibility is limited. In Case b, the two situation developments are clearly visible resulting in the two peaks at value 5.7 and 6.5. This complexity reduction in Case b extremely simplifies solving the problem: the POMDP solver needs

[7.2]Note that in the visualization the particles of the predicted beliefs are shown before integrating any measurements.

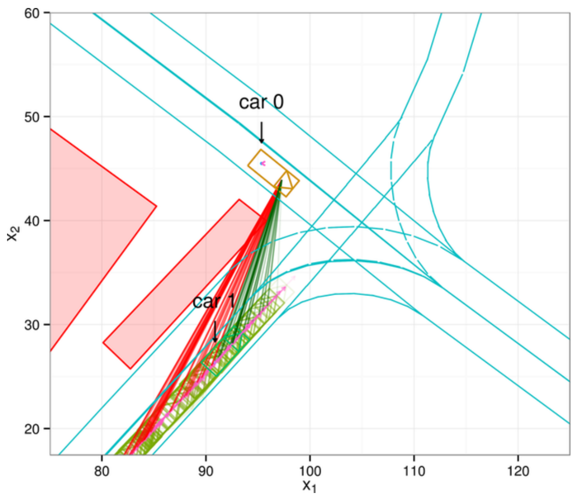
[7.3]We still assumed that the transition and observation model are uncertain.



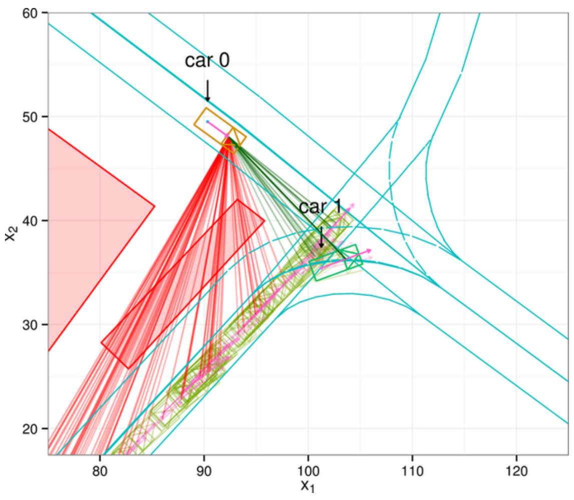
(a) Simulation 1, $t = 0$



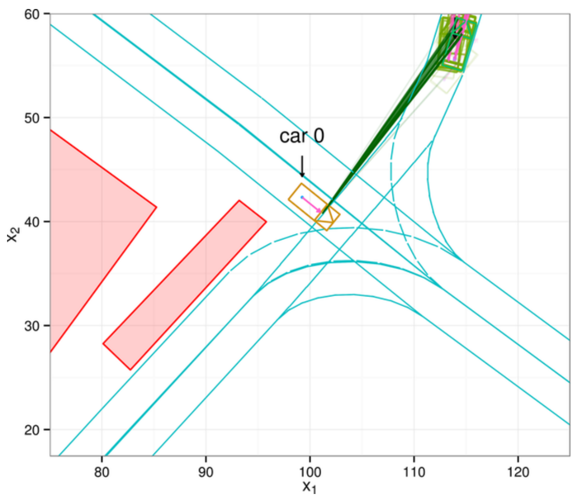
(d) Simulation 2, $t = 0$



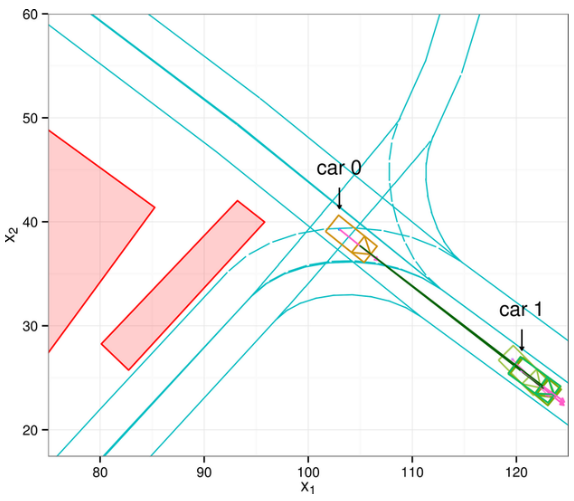
(b) Simulation 1, $t = 6$



(e) Simulation 2, $t = 3$

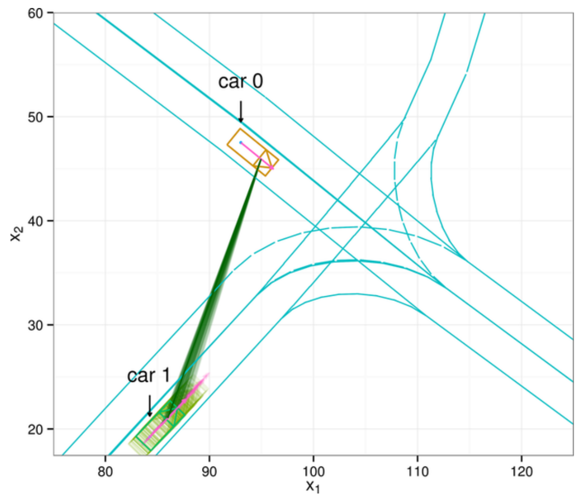


(c) Simulation 1, $t = 11$

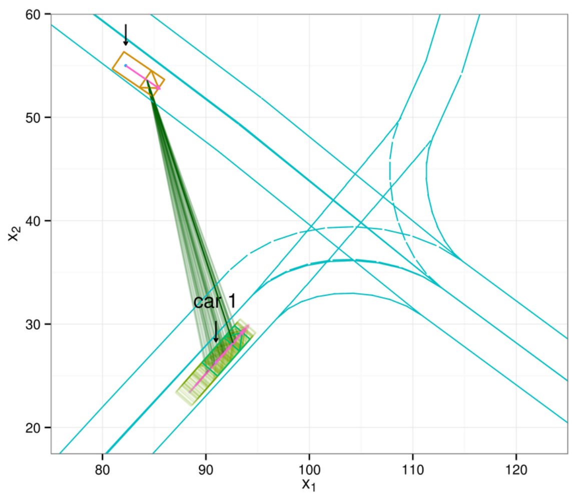


(f) Simulation 2, $t = 6$

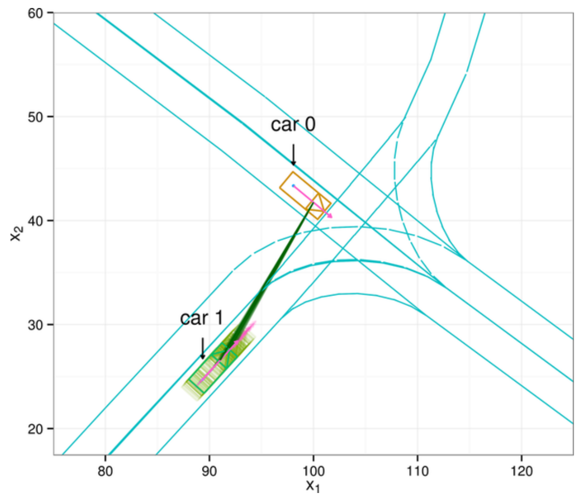
Figure 7.14: Intersection Case a: two policy simulations with parking cars and a building that blocks the view.



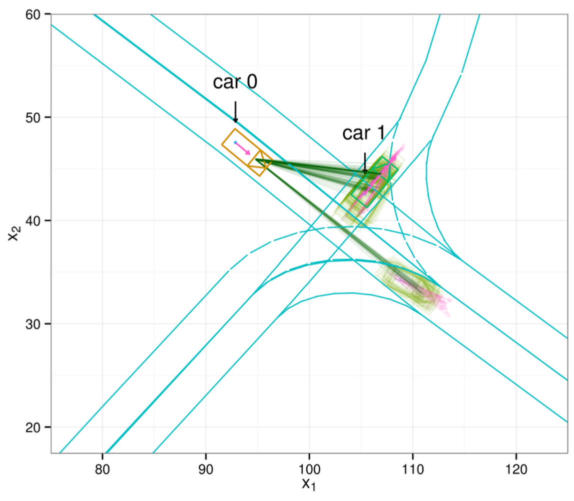
(a) Simulation 1, $t = 2$



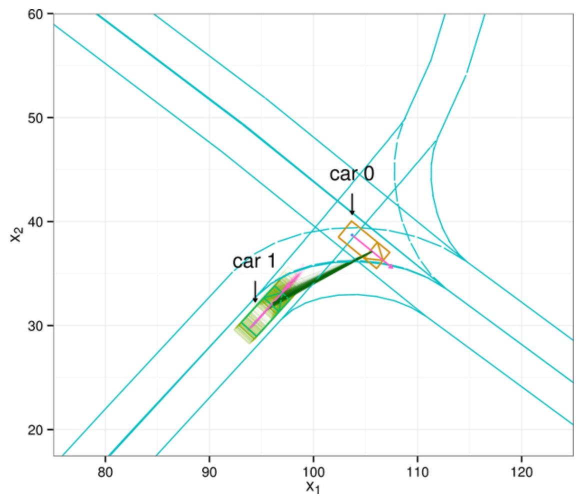
(d) Simulation 2, $t = 0$



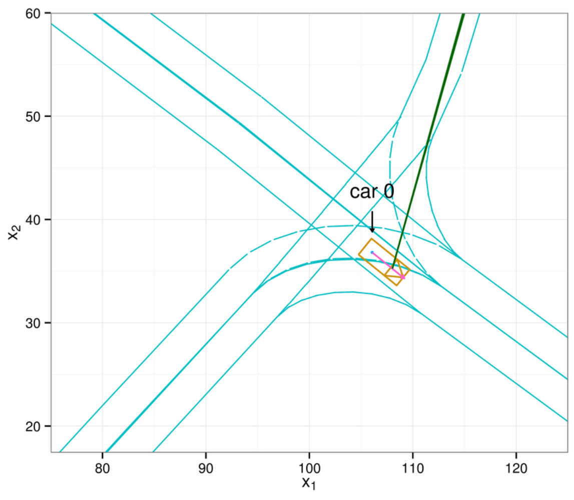
(b) Simulation 1, $t = 3$



(e) Simulation 2, $t = 3$



(c) Simulation 1, $t = 4$



(f) Simulation 2, $t = 8$

Figure 7.15: Intersection Case b: two policy simulations without blocked view with different initial beliefs.

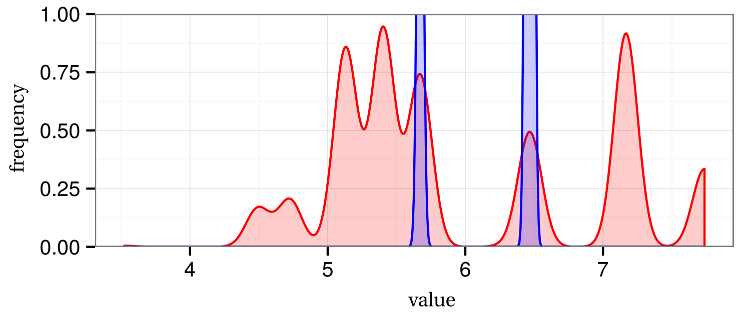


Figure 7.16: Intersection scenario: frequency of values (summed discounted rewards over the full simulation run) yielded by the POMDP policies for Case *a* (red) with and Case *b* (blue) without blocked view for 1000 runs in (see Figure 7.14 and 7.15, respectively).

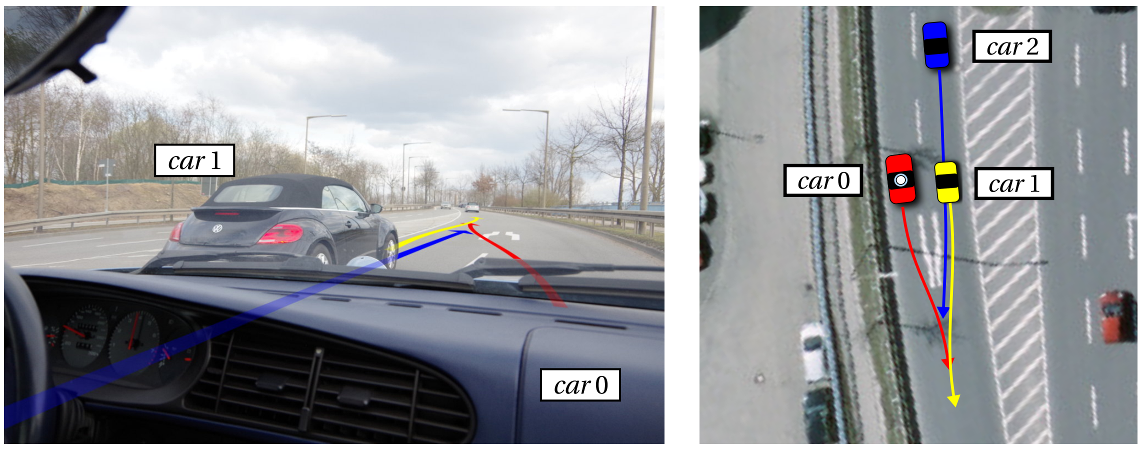


Figure 7.17: Zipper merge: scenario introduction. In-car and birds-eye view. Map Data [City of Karlsruhe].

< 10 s to fully precompute the policy compared to ca. 5 min for Case *a*. However, the policy to come to a full stop first, is not even part of the solution of Case *b* and, thus, direct application of the policy generated assuming good visibility for the case with blocked view is dangerous.

> Neglecting partial observability against better judgment results in unsafe driving.

7.3.2 Anticipation of Interaction—Zipper Merge

Road users constantly interact, be it when changing lanes, giving way, or just keeping distance to a leading vehicle without crashing into it. In Figure 7.17 a scenario is displayed where the zipper method (also called late-merge) has to be applied. Two lanes merge into one and the cars have to take alternating turns at the lane reduction point. In the specific situation, the ego vehicle (red car on the left lane) should merge between the other two cars (*car* 1 in blue and *car* 2 in yellow). However, the blue *car* 1 is

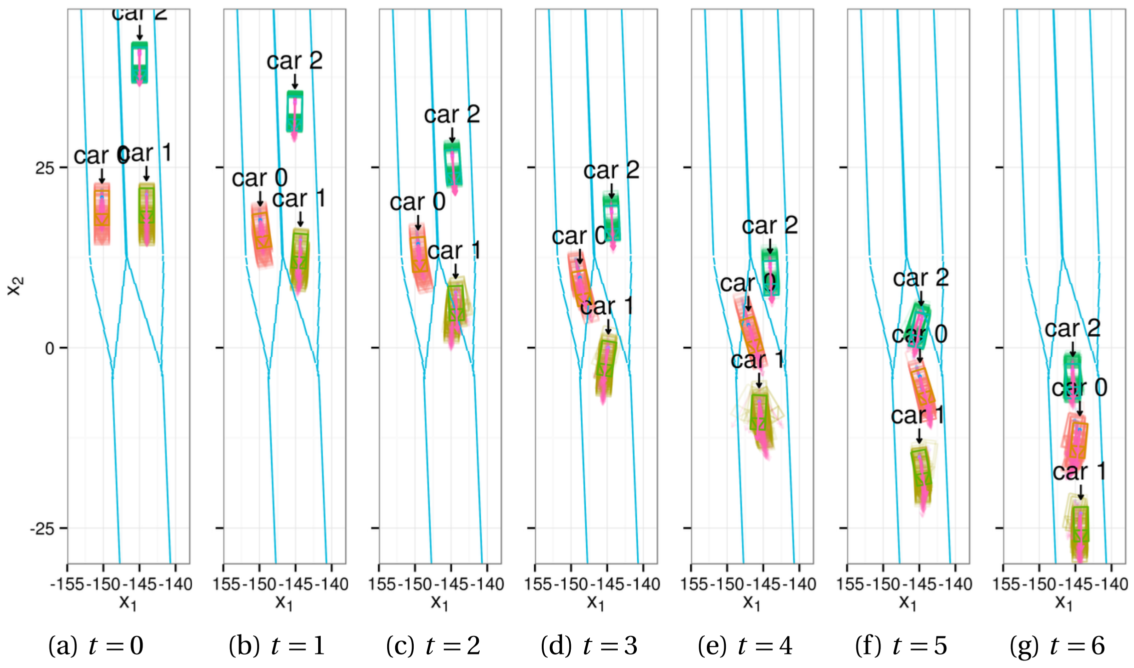


Figure 7.18: Zipper merge: policy simulation from $t = 0$ to 6. Connecting lines were omitted to improve readability.

approaching quickly and the gap would be too small when assuming constant velocities and no interaction between the cars.

Figure 7.18 shows that the POMDP policy accelerates and merges after *car* 1 and before *car* 2, despite the small gap size and the uncertainty in the behavior prediction. This can be seen more clearly in Figure 7.19, where the development of the x_2-positions and the absolute speeds of all three vehicles are depicted.[7.4] The reason for this is that *car* 2 reacts on *car* 0 and decelerates to give way.

The continuous POMDP approach is able to foresee this interaction, while at the same time balancing the risk arising through the uncertainties. If, initially, the gap between *car* 1 and *car* 2 is smaller than in the presented example, it waits because the risk that *car* 2 does not give way outweighs the benefits of not stopping.

> Interactions between road users are a natural part of driving. They must be anticipated for making good decisions.

[7.4] At $t = 0$ to 2 the positions of *car* 0 and *car* 1 overlap. However, there is no collision because they are on different lanes before they reach the lane reduction point.

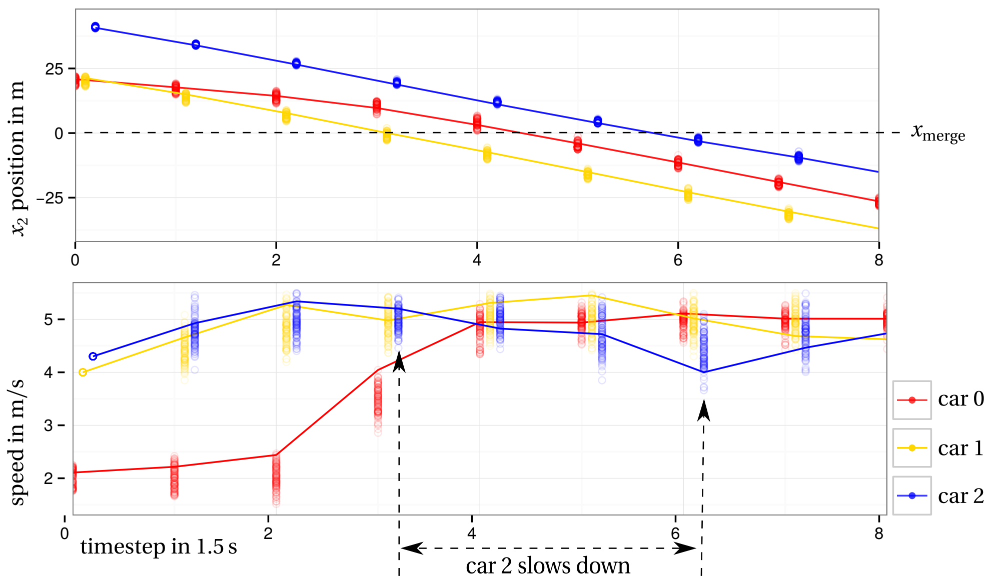


Figure 7.19: Zipper merge: position in x_2 dimension and absolute velocities. Dots indicate the particles of the beliefs and lines the true state. The lane reduction point at $x_2 = 0\,\text{m}$ is labeled with x_{merge}.

7.3.3 State Space Representation—Merging into Moving Traffic with Limited Perception

Finally, we evaluate our approach for the introductory example given in Figure 1.1 that serves as motivation for this thesis. This scenario combines all challenges of sequential-decision making under uncertainty. The self-driving car has to control its velocity to merge safely into a gap between two other cars that have priority. An unconventional policy is necessary to solve this task safely, because a truck masks the view on the scene in a non-trivial way. Note that this scenario has been presented in less detail in [Brechtel et al., 2014]. In this section, we put the focus on how our continuous POMDP method is able to solve such a complex scenario an shed some light on its internal state space representation.

Policy Analysis Figure 7.20 displays all time steps of an exemplary simulation run that demonstrates the process of observing and estimating the state of the world. The policy resembles the optimal behavior that we described in the introduction. The initial true state shows that there is a relatively big gap between *car* 1 and *car* 2. The POMDP policy makes the self-driving vehicle stop first at a position where it can see through the gap between the building and the truck, until it has gained enough information about the gap. At $t = 7$ (10.5 s) it begins to accelerate. It is not able to see *car* 1 at this point in time. Yet, it can be certain that the gap is big enough: it would have seen *car* 1, otherwise. This example shows the ability and importance of the POMDP to also utilize and plan with *evidence of absence*.

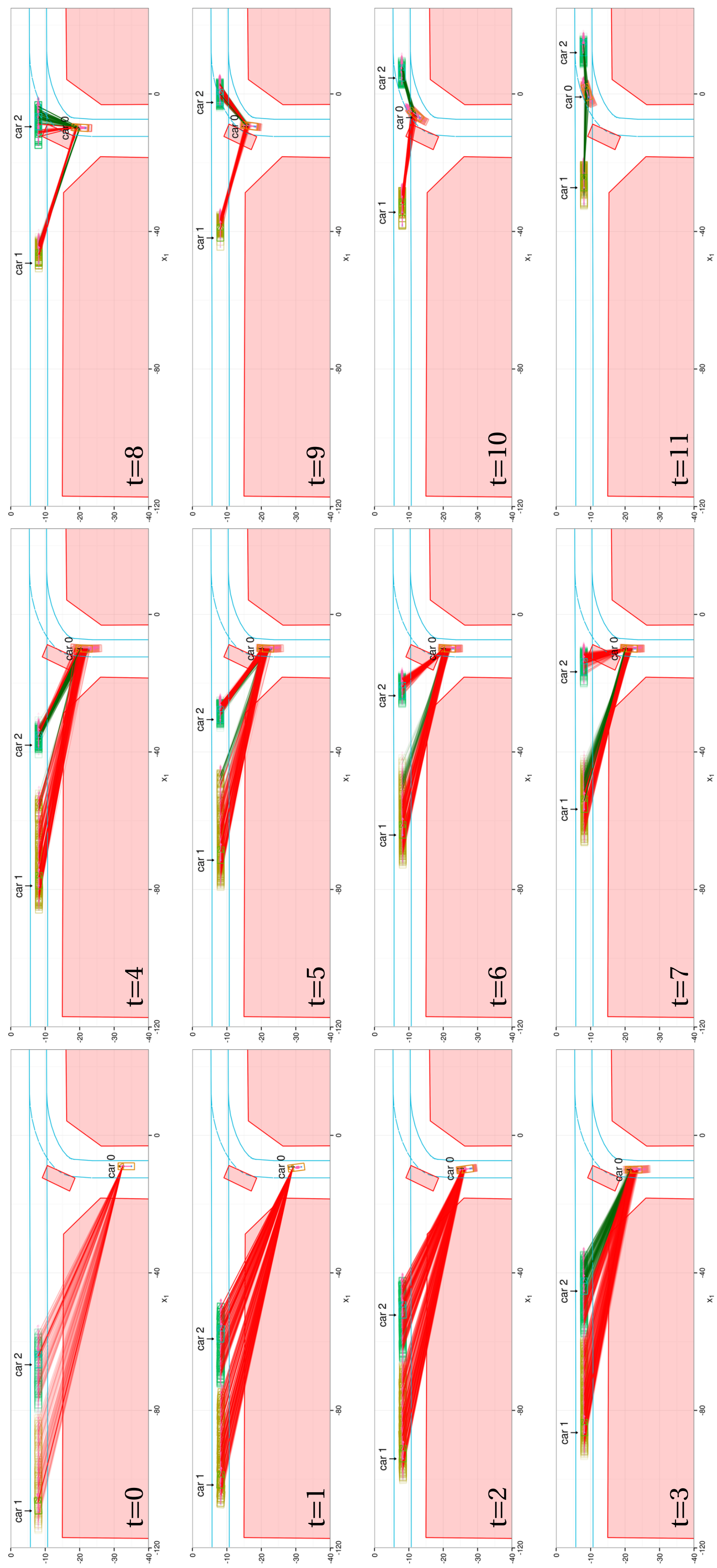


Figure 7.20: Traffic merging: policy simulation from time step $t = 0$ to 11.

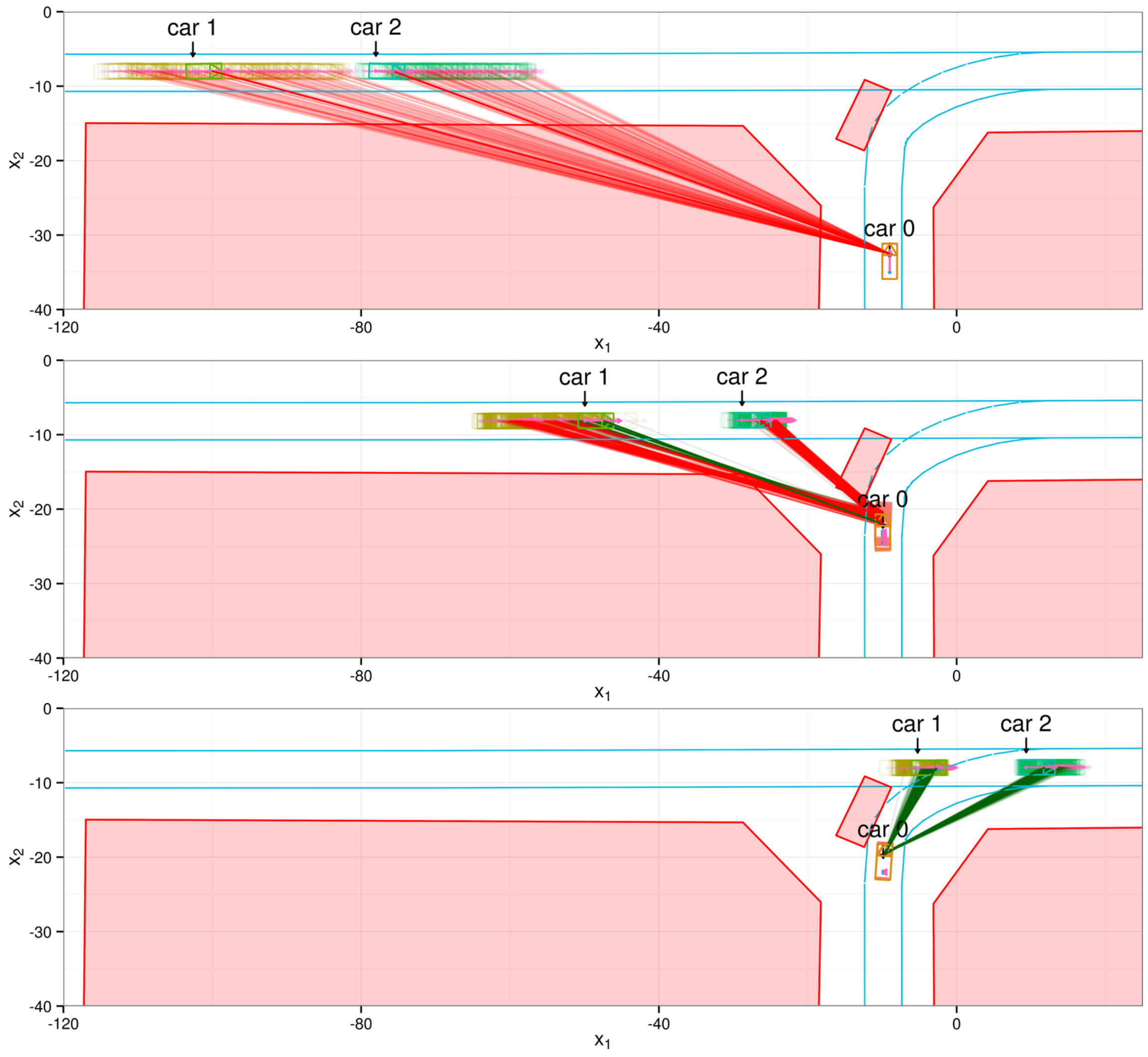


Figure 7.21: Traffic merging: policy simulation at $t = 0, 7$ and 13.

The most difficult part for the POMDP solver in this example is to assess the risk due to the uncertainty of the other cars motion. Note the increase in uncertainty in the estimate of *car* 2 at $t = 6, 7$, and 8 and of *car* 1 in $t = 8, 9$ and 10, where the cars are hidden behind the truck. In this first example the gap size is big enough to assure that it is still sufficient when the car enters the lane. However, in the next example this is not case.

In the example in Figure 7.21, the true initial state shows a gap of about 25 m, which is sufficient for the car to fit in. However, the POMDP policy decides in $t = 7$ to not merge and wait for the next gap instead. The last time step shows that this is a good decision: because of the stochastic driving of the other cars, the gap shrunk to a size of about 10 m at $t = 10$. Merging would have been quite dangerous, the other cars could change their speed unnoticed by the ego vehicle.

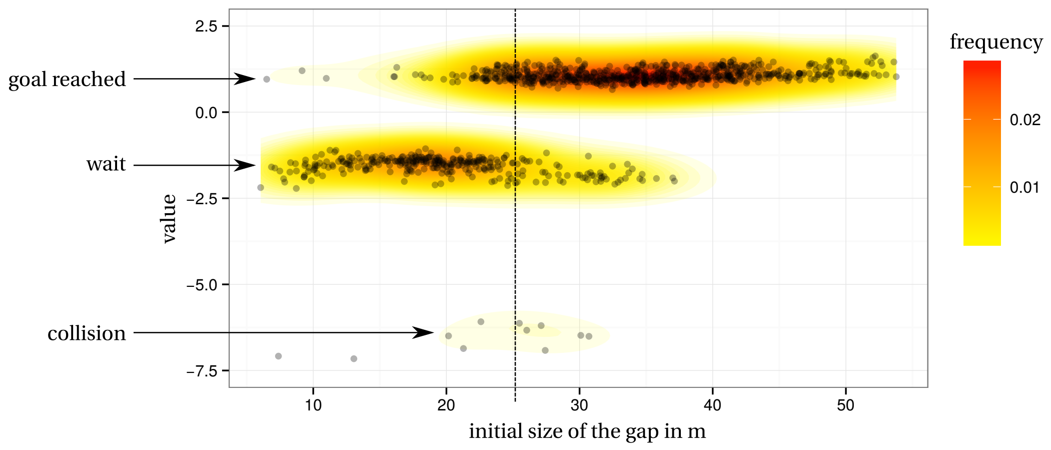


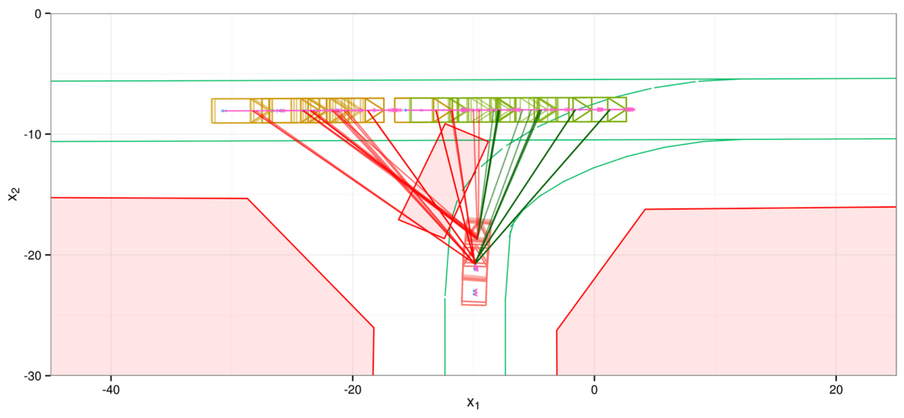
Figure 7.22: Traffic merging: relationship between the initial gap size of a simulation trial and the yielded values (summed discounted reward).

> The influences of partial observability are not trivial. Thus, they cannot be covered in a sweeping way for every situation with predefined rules.

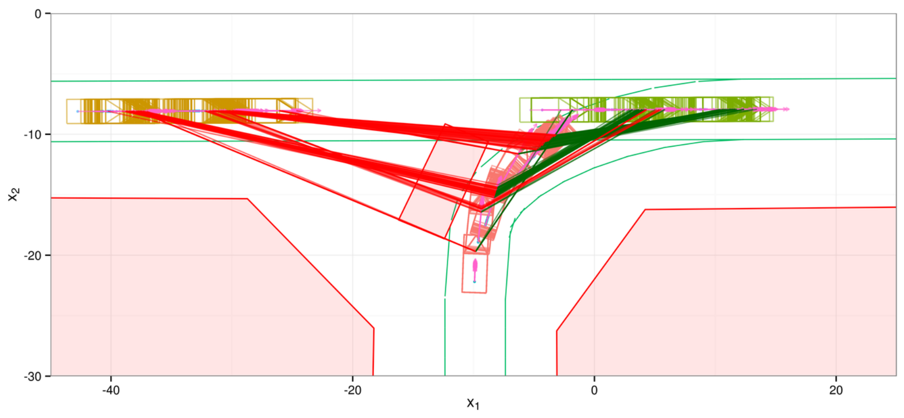
Overall Performance To assess the overall performance of the precomputed POMDP policy, we repeated 1000 simulations with different initial true states sampled from the initial belief. In Figure 7.22 the frequency of the yielded values and the true size of the gap at the beginning of the simulation are plotted. If the car collides during the simulation, it receives a negative value of about −6. The precise value depends on the time when it hit the other vehicle because of the discounting. If it reaches the goal area, it yields a value of about 1.3. If it has to wait for the next gap, it receives about −1.8 because it only has costs. As the development of the situation is uncertain, these values can be interpreted as a joint frequency distribution over the value and the initial gap size.

It can be seen that the policy chooses to merge into gaps that are bigger than ca. 25 m and waits if they are smaller. There are varying results for the gap size around 25 m. The different decisions have their reason in the stochastic development of the situation. Nuances can make the difference whether the self-driving car can accelerate quickly enough in the particular situation. Ultimately, the full joint distribution of positions and velocities of the cars must be taken into account. Nevertheless, the plot also shows a few collisions. Due to the uncertainty in the behavior of other drivers, any policy can only minimize the probability for accidents, but never prevent them.

> Simple, static rules, for example, depending on the size of the gap, are not sufficient. The full situation and its development must be considered.



(a) POMDP state: gap is too small to merge.



(b) POMDP state: gap is sufficient for merging into.

Figure 7.23: Traffic merging: POMDP state representation. Both plots show multiple continuous 12D states that are mapped to the same discrete POMDP state and thus not differentiated during planning.

Space Representation To represent the 12-dimensional combined space of all three cars efficiently for this specific problem, the POMDP solver learned a decision tree with 442 leafs. Every leaf encodes one discrete POMDP state. To reduce the complexity of the planning space in this drastic manner, multiple continuous states have to be summarized to a single discrete state. Figure 7.23 shows two discrete states of the final policy. In both graphics, several continuous samples of the 12D states are shown that are mapped to the same POMDP state. The state in Figure 7.23a subsumes several 12D states where the gap is not big enough to merge safely. From the decision-making point of view, distinguishing those is not valuable because the final outcome of those situations is the same: if the ego car does not stop, it will most likely receive a cost for a collision. The state in Figure 7.23b contains continuous states where accelerating will probably be successful. They can only be interpreted correctly with the connecting lines that show which car configurations form one joint continuous state. Note that

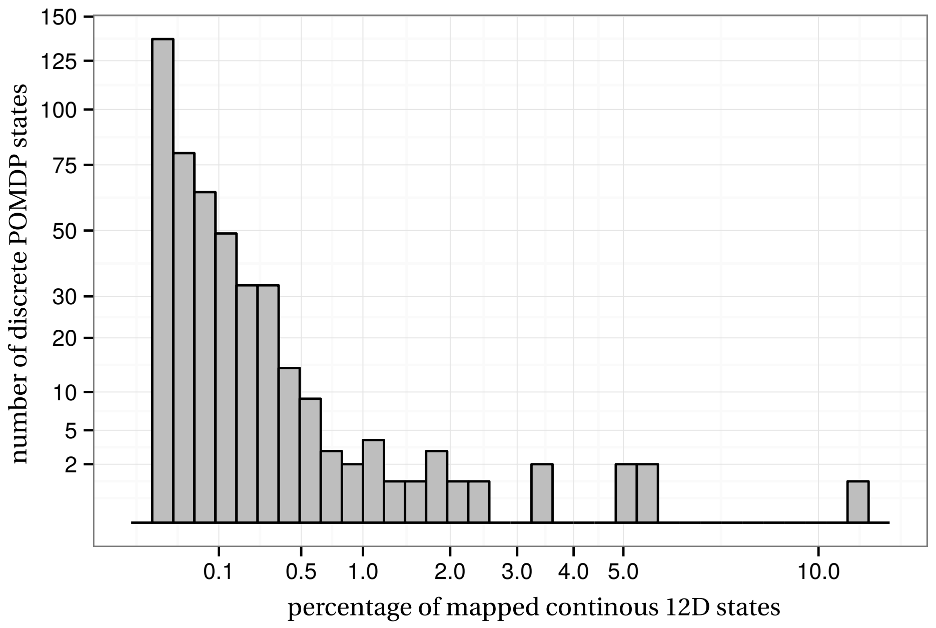


Figure 7.24: Traffic merging: histogram that visualizes the large spectrum of the size of the learned discrete POMDP states.

some continuous states that are merged to the same discrete POMDP state are quite far away from each other in the euclidean 12D space.

The histogram in Figure 7.24 shows how *big* the learned discrete states are. Therefore, we collected over 3.8 million continuous state samples from 1000 simulations. These samples cover many reachable and relevant states in the problem. Then we discretized them to the POMDP representation and plotted the percentage of these states that are mapped to the same discrete POMDP state. The histogram shows that most discrete states represent *small* areas, meaning that only a few continuous states are covered by the discrete state. These states usually model situations were the decision is a very close call and a high resolution is required. However, about 58 % of the sampled particles were mapped to only 20 discrete POMDP states. These results indicate that the discrete states are very small at some positions to represent details. At other positions, they aggregate many situations that do not have to be distinguished.

> The ability to find a suitable problem representation of the continuous space for the specific POMDP allows the presented method to abstract form the continuous state space and, ultimately, makes computation practically feasible.

7.4 Conclusion

The evaluation of the novel continuous POMDP method shows that it outperforms existing methods (Contribution 1d). It is capable of solving relatively high-dimensional obstacle avoidance tasks with dynamics and does not have problems with non-linear POMDP models. Further, it shows that our general approach to decision-making is suitable for highway as well as urban driving (Contribution 2c). Essentially, the same POMDP models were used for all evaluated driving tasks. The practical relevance of the experiments is assured by choosing real-world examples exhibited in the city of Karlsruhe.

The evaluation supports our thesis statement: in the traffic scenarios that we selected for evaluation, safe and efficient driving can only be achieved, if partial observability, interaction between road users, and uncertainties in the dynamics are taken into account. The highway as well as the zipper merge scenario demonstrate the importance of anticipating interaction. The junction and the final merging scenario show that considering partial observability is absolutely vital. We found that the resulting POMDP policies for these situations are very different. These results strongly suggest that predefined rules are not sufficient because the consequences from uncertainties can be manifold and need to be treated individually. In other words, they are an inherent part of the decision problem and must not be dealt with in a sweeping way or decomposed from the rest of the problem. By the example of the MDP policy for highway driving that was generated using an equidistant discretization, we showed the importance of a well-chosen state space representation. We illustrated the idea to use variations in the value as an indicator for relevance-based discrimination of continuous state space regions. The analyses of the proposed value iteration method indicate that learning a low-dimensional discrete state space can effectively reduce the problems (practical) complexity. In all examples, it reduces the up to 12D continuous state space to just a few hundred discrete states that are relevant for the specific task. Problem-specific, iterative representation learning allows to focus computations on relevant information and to generalize sparse planning results over the continuous state space. A surprising side-effect of the throughout Bayesian model is that our method shows the general ability to actively take into account and plan with *evidence of absence* [Copi, 2008, p. 95]. For example, in the 1D corridor problem (Figure 7.1), it dismisses the hypothesis that the robot is at the left end of the corridor, which improves the time that the robot needs to find the right door. How important this capability is, becomes even more evident in the obstacle avoidance and driving examples. For example, it makes merging in Figure 7.20 possible: seeing that there is no car is sufficient to safely merge. It could be argued that the continuous POMDP method found the concept of *free space* on its own.

Chapter 8

Discussion and Conclusion

In this work, we developed a novel, general method for solving continuous POMDPs based on learning an efficient, discrete, problem-specific representation of the continuous space. This document provides a detailed description of the motivations, the methodological background, the basic ideas, the formal derivation, and the implementation of the novel method. We also presented a general approach to tactical decision-making in traffic under uncertainty. Therefore, we modeled safe, efficient and goal-oriented driving in traffic as a continuous POMDP. We presented a hierarchical probabilistic model that anticipates developments of traffic situations as well as the acquisition of information by the self-driving car through observations .

In the evaluation, we successfully applied the developed continuous POMDP method to this model for decision-making in various realistic urban driving scenarios in the city of Karlsruhe. These scenarios were aggravated by non-negligible limitations of the perception, uncertain dynamics, and interactions between road users. The solved scenarios also included the motivating example with partially blocked visibility presented in Chapter 1. To the best of our knowledge, today, no existing general approach can make optimal decisions in this or comparable driving scenarios. A reason for this is that there is no method for decision-making in continuous-domain POMDPs that suffices for driving or tasks with similar properties.

By presenting a novel continuous POMDP method together with a general approach for decision-making for driving, this work contributes to research in artificial intelligence and machine learning as well as robotics and autonomous driving.

Chapter Overview We first review the thesis statement in the light of our results. Then, we summarize the technical and methodical advances compared to existing research. Finally, we discuss limitations and future work, and close with a conclusion.

8.1 Review of Thesis Statement

The thesis statement in Section 1.1 comprises two main messages that are discussed in the following.

Uncertainty must be considered for decision-making in real-life tasks

We presented multiple driving scenarios where considering uncertainties is imperative for safe decision-making. In the traffic domain, significant uncertainties stem from the stochastic behavior of other drivers and their constant interaction with each other and the self-driving vehicle. Even worse, the perception of the environment through sensors is very limited. Intentions of other drivers cannot be observed and relevant objects can only be measured with errors. Especially in urban driving, another factor adds to the complexity of decision-making: relevant objects are often completely hidden to the sensors because the line of sight is blocked.

Our experiments in Section 7.3 underline the crucial and complex impact of uncertainties. As expected, visibility issues such as a blocked view on other road users have the most significant influence on decisions. The most illustrative example we presented is the merging scenario from the introduction in Chapter 1. In this example, the car needs to stop several meters before the crossing. This rather unconventional behavior would not be part of any policy that does not consider uncertainty in sufficient detail.

Small variances in the measurement or the prediction often only result in slightly modified policies such as increasing the safe following distance. However, this is not the case, in general. The merging examples revealed that uncertainty can, for example, make it unsafe to merge into a gap that would be sufficiently big under the assumption of precise models and full information.

To our surprise, another aspect of information gathering showed to be extremely important for both, the synthetic problems and the driving tasks: the ability to conclude from and plan with *evidence of absence*. Especially the merging and obstacle avoidance examples make clear that observing the absence of an object is often of greater value than observing the object itself. In the merging examples, not observing vehicles is equivalent to observing a gap. The presented method is capable of drawing this conclusion because we did not impose prior restrictions to the models (e.g., by assuming linear models or normal distributions).

In the light of these results, the initial thesis statement does not only hold. It should be extended: uncertainties must be considered with sufficient detail and complexity. Existing approaches to decision-making for driving, however, oversimplify the problem. This is not necessary, as the second part of the thesis statement shows.

Decision-making can be made practically feasible by implementing the cognitive abilities to generalize thoughts and put selective attention on relevant information

The motivation behind simplifying the decision-making problem is clear: the full complexity of the problem cannot be handled by existing methods for solving continuous POMDPs (or comparable models). This problem, in theory, is even undecidable.

The second message of the thesis statement is that planning in continuous domains with uncertainties, despite their theoretical complexity, can be practically feasible. Humans are capable of planning and making safe decisions in such situations, supposedly by abstracting and focusing their attention on relevant information. This way, humans find a suitable simplification for every specific task. We argue that similar abstraction capabilities can be methodically realized to make approximately solving continuous POMDPs feasible.

A fundamental basis for inference and planning is the representation of the world. However, a generally ideal representation does not exist because it is very specific to the considered task. In this thesis, we proposed inductive learning for finding an efficient, low-dimensional discrete representation for the specific problem. Using this representation basis, computation automatically focuses on relevant features and the results are generalized over the complete continuous state space.

The policies generated with the presented continuous POMDP method showed that our approach is capable of planning the agent's complex process of information gathering through sensor measurements. In all evaluated examples, the up to 12D continuous state spaces was compressed to less than 1000 discrete states. To put that number into relation: differentiating only 4 discrete regions for every of the 12 dimensions results in a total of over 16.7 million states. A more detailed look into the learned, discrete representation showed that it clusters continuous states with similar meaning and consequences with respect to the policy. The clustered states are not necessarily close in the Euclidean space. It could be argued that this goal-oriented compression is a form of automatic conceptual abstraction.

8.2 Summary of Contributions

This work's contributions to the state-of-the-art are twofold (see Section 1.3). The first contribution is a novel, general method for solving continuous POMDPs that mainly contributes to research in machine learning and artificial intelligence. The second contribution is a general approach for tactical decision-making in traffic that is based on decision processes. This approach contributes to research in robotics, and autonomous as well as assisted driving.

Contribution 1: Novel method for efficiently solving continuous POMDPs

In Chapter 5, we present a general method for continuous-state, continuous-observation and discrete-action POMDPs. The main innovation of the method is that it finds a problem-specific, efficient, discrete representation of the continuous state space.

1a) Point-based value iteration (PBVI) without restrictions of spaces or models The requirements that we derived in Section 4.4 with the driving task in mind indicated that, at best, the POMDP's models should not be restricted, a priori. In particular, a method suited for autonomous driving must be able to consider multi-modality and non-linearity. Approximations and simplifications should be applied carefully and with respect to every particular task and situation.

We realized this by applying importance sampling for a Monte Carlo (MC) continuous belief prediction and Bellman backups (see Section 5.2). In order to generalize value functions over the continuous space for efficient dynamic programming, a non-parametric basis function representation of both, the policy and situations is learned. Previous approaches fix the representation prior to solving the POMDP or restrict the family of the basis functions (e.g., to Gaussian distributions).

1b) Bellman α-function backup that integrates representation learning and temporal inference The basic idea behind the approach is that the value is a suitable indicator for state space abstraction in decision problems. The results of our research on MDP-based highway driving in Section 7.2.3 suggested that a suitable representation can be learned automatically and showed that, in practical problems, enormous improvements over naive discretization approaches can be expected. In Section 5.3, we formulated this idea by establishing a theoretical framework for integrating discrete representations into continuous POMDPs.

To compute the optimal value function, the continuous α-functions need to be represented well. Vice versa, to find a good representation, an (approximately) optimal value function needs to be known. We resolved this circular dependency by integrating learning a representation into continuous value iteration with α-function Bellman backups. The representation is iteratively learned with the objective to optimally represent the value function (and with it the policy) well. Previous methods aimed at representing the belief distributions well, which is suboptimal for decision-making.

1c) Automatic abstraction and generalization of planning results over the continuous space Representation learning is presented in Section 5.4. It directly minimizes the loss in the representation of continuous α-function Bellman backups and thereby generalizes them over the continuous state space. For this reason, transforming continuous beliefs to the discrete representation, can be viewed as extracting policy-relevant features.

We implemented this mechanism using a partition of the continuous space that is represented by decision trees which are incrementally learned in a nonparametric way. This way, the detail and complexity of the representation adapts to the problem.

1d) Efficient implementation of the method that is able to solve high-dimensional POMDPs and shows state-of-the-art exceeding performance in experiments This implementation outperforms existing solvers on the evaluated examples, it is capable of solving higher-dimensional continuous problems (see Section 7.1), and it solved up to 12D continuous complex real-life driving tasks (see Section 7.3).

Contribution 2: Approach to tactical decision-making for driving

Our analysis of the state-of-the-art in decision-making for autonomous driving in Chapter 2 showed that most existing approaches critically simplify the task. Either they ignore uncertainty or do not consider it in sufficient detail. Alternatively, they provide solutions for restricted sub-problems, such as highway driving.

The basic idea behind our approach presented in Chapter 6 is that formulating the goals of driving is much simpler than formulating the right reactions to the huge variety of different situations which a car can encounter in traffic.

2a) General representation of driving as a decision process In Section 6.1, we presented models that represent traffic scenarios and their developments sufficiently well for tactical decision-making in traffic. Because we model the task as continuous POMDP that utilizes background knowledge (e.g., from maps), the models are very general and applicable to many different driving scenarios. By solving this POMDP with the presented method, policies can be created that are mathematically grounded through anticipating the development of the world. As the presented method for solving continuous POMDPs adaptively discretizes the state space with respect to the current situation and specifically for the current task, no potentially insufficient and inefficient a priori discretization is required.

2b) End-to-end probabilistic formulation The models are encoded in a dynamic Bayesian network (DBN) in order to structure dependencies hierarchically and to provide a throughout probabilistic approach on all levels of abstraction. This way, complex relationships, including interactions between road users, are realistically modeled. We put special emphasis on modeling partial observability arising through the limited visibility of objects. This has several advantages. For example, actions with the purpose to gain decision-relevant information are naturally integrated into planning. Additionally, solving the POMDP generates an action policy for the current and possible future situations because possible developments are already considered in the policy. Thus, in difference to deterministic planning approaches, no replanning in every step is necessary.

2c) Evaluation of autonomous intersection handling and merging with non-trivial occlusions The evaluation showed that safe driving decisions can be derived for various different traffic scenarios by solving the general continuous POMDP with the method in Chapter 5 or the simpler MDP approach in Section 6.2.1. Among other aspects, the tested scenarios differ in the road topology and geometry, the number, pose, and velocity of the involved vehicles, and the visibility conditions. Higher-speed highway driving with full observability (see Section 7.2) as well as urban driving with partial observability (see Section 7.3) are successfully tested. Despite the fundamentally different scenarios, the same general model is used for all examples.

8.3 Limitations, Applications, and Outlook

In this section, we name and assess the limitations of the developed continuous POMDP method and the approach to tactical decision-making for driving and propose potential solutions.

8.3.1 Continuous POMDPs

The general problem of solving continuous POMDPs is too complex to be solved efficiently for all possible tasks by the same algorithm. Complexity makes approximations indispensable. However, it depends on the specific problem which kind and which extent of approximation can be tolerated. By iteratively learning a suited representation, we made a step towards problem-adaptive approximation. Other than assuming a discrete action space, we did not make limiting prior assumptions. However, in order to solve more complex problems (e.g., with higher dimensions, or a longer planning horizon) the performance has to be further improved. We identified multiple starting points for improvements.

Our implementation, using partitions represented by decision trees, restricts the representation and holds back the potential of learning. We would be interested to see the potential of generalization when applying more sophisticated algorithms to our general approach of iterative value-directed representation learning. Promising developments in this context can be found, for instance, in the area of deep learning [Bengio, 2009].

The MC approach that we apply to approximate the continuous belief prediction and α-function backup has disadvantages for some types of distributions (e.g., very noisy distributions) and, typically, yields only approximate results. Rao–Blackwellization might pose a solution to this problem by exploiting the structure of the DBN. It has shown significant improvements for the related tasks of sequential Monte Carlo (SMC) filtering [Doucet et al., 2000] and simultaneous localization and mapping (SLAM) [Grisetti et al., 2007]). Another promising research direction is to extend our approach to hierarchical POMDP planning instead of solving the flat POMDP (see, e.g., [Pineau et al., 2002; Theocharous, 2002; Foka and Trahanias, 2007]).

We also think that the efficient and sufficient exploration of the infinite-dimensional belief space still is an open question. Exploration determines the belief points where backups are performed. In this work, we transferred heuristics from discrete point-based POMDP methods to the continuous space, but there is room for improvements in this important question. A related question is how the number of α-functions can be kept small. Solutions for discrete spaces exist (e.g., by pruning the set), but directly applying them to the continuous problem does not promise success.

For some of the open questions, solutions can be found in other areas of research. Optimal control, for instance, is concerned with continuous action spaces and could be a source of inspiration for approaching this problem in POMDPs. Making the step from discrete to continuous POMDPs, the borders between decision-making, motion planning and optimal control diminish. We think that in the future, research in these areas will further benefit from each other.

Although we evaluated the method only in the context of robotics and driving, it is generally applicable to any kind of continuous POMDP. As continuous POMDPs are among the most general models for dynamic decision making under uncertainties and we did not assume further restrictions, many more applications, e.g., for operations research, medical diagnostics and treatment, health-care, and machine maintenance, are thinkable.

8.3.2 Tactical Decision-making for Driving

Decision-making is one of the most important challenges that has to be solved for autonomous driving to become a reality. In this work, a novel approach to this task is presented and evaluated that relies on optimization and learning rather than manual programming. For this reasons, our approach can master a greater variety of driving situations than previous approaches.

However, for integrating this approach into self-driving cars, further research is needed. For the experiments, we solved the scenarios offline and then executed the precomputed policy. A precomputed policy provides decisions for all different developments of the situation and also generalizes to variations of the initial situation. Contrary to plans obtained by deterministic planning, the policy does not have to be recalculated in every step because it anticipates uncertainty. For realizing continuous driving, our approach can be applied in different ways. The most general way is to find a general driving policy applicable to all situations. It could be improved during driving, using an approach similar to the presented offline-online learning for MDPs or reinforcement learning. However, it is unclear how well the method scales to such a huge problem and probably further hierarchization or simplification is needed. For example, only the scenarios exhibited on a planned navigational route could be precomputed. Alternatively, special driving tasks could be completely covered using relative state representations (similar to the presented MDP highway driving approach).

Further, the presented implementation of the model should be viewed as a prototype to demonstrate the potential of the approach. It is not comprehensive. For example, the provided models are limited to car-like vehicles and do not yet integrate other road users, such as pedestrians and cyclists. Further improvements through more realistic models can be achieved, when integrating behavior models learned from observations as proposed in [Gindele et al., 2015].

Additionally, developments move towards vehicles which are connected with the infrastructure and each other. Communication can be added as another measurement channel into the presented model to reduce uncertainties and improve results. Especially interesting results can be expected, if autonomous cars communicate their intentions in addition to their current state. This information can be integrated into our model by adding a measurement for the road users's *planned route*. Cooperation between road users is not restricted to sharing information. Cooperative planning for decision-making could be realized with a centralized POMDP solver with combined actions and rewards for all controllable road users.

Besides fully autonomous driving, the model can be easily modified for advanced driver assistance systems. Therefore, behaviors and intentions of the ego vehicle's driver can be modeled with conditional distributions. Assistance actions, like showing warning messages, would indirectly influence the ego vehicle through the driver.

Due to the very general representation of the driving problem as hierarchical DBN and because the continuous POMDP models are not restricted, these aspects can be easily integrated into the proposed model.

8.4 Conclusion

Due to the exceeding complexity, especially for continuous-state real-life tasks, it is still a common opinion among researchers that decision-making under uncertainty is practically infeasible. However, it is neither necessary nor sensible to categorically restrain to simplifications, such as omitting or simplifying partial observability. Uncertainties are just too important for most applications to ignore them. Making robust decisions despite not knowing the exact state of the world or its development over time is a key ability for all kinds of autonomous systems, be it physical systems such as robots or virtual systems such as dialog systems. For example, planning actions with the purpose of gaining relevant information is a natural part of problem solving. Humans are so used to act under the severe lack of information that they often forget about it.

Indeed, planning under uncertainty without approximations, is not only computationally infeasible: the problem is even provably unsolvable. However, restricting the models or spaces, for example, by discretizing the continuous space prior to solving the problem is the wrong answer to this problem. This line of action can impair the resulting decisions to a non-tolerable extent with potentially dangerous consequences. We showed in this thesis that continuous POMDPs can be efficiently approximately solved when implementing the ability to make abstractions in order to focus computations on relevant aspects.

We did not stop at developing a theoretical method but developed a concept for utilizing it in practice for one of today's most important research topics: autonomous driving. Self-driving vehicles promise to revolutionize personal transport and logistics. Great improvements in all areas of autonomous driving, such as environment perception and interpretation, self-localization, mapping, navigation, and control, are made by researchers all over the world. In our opinion, decision-making more and more emerges as one of the main remaining technical challenges.

We showed that applying the developed continuous POMDP solver to a general traffic model can offer a solution to this problem. The application of continuous POMDPs, however, is not restricted to this particular task. On the contrary, the general approach and method developed in this thesis can be used for various applications from all conceivable domains.

Appendix A

A.1 Alternative Reward Function Definition $\mathring{r}(s,a,s')$

For some POMDP applications it is sensible to add a dependency to the next state s' to the reward (e.g., to account for collisions between time steps). To obtain the reward notation $r(s,a)$ used throughout this thesis from the alternative reward function $\mathring{r}(s,a,s')$, the dependency to the next state s' can be ruled out by

$$r(s,a) = \int\limits_{s'\in\mathbb{S}} \mathring{r}(s,a,s')p(s'|s,a)\,\mathrm{d}s' \,. \tag{A.1.1}$$

The α-function backup in Equation 4.3.14 remains untouched by the reward definition in Equation A.1.1. It can be rewritten as follows

$$\alpha^{n}_{b,a}(s) \overset{(4.3.14)}{=} r(s,a) + \gamma \int\limits_{s'\in\mathbb{S}} p(s'|s,a) \int\limits_{o\in\mathbb{O}} \alpha^{n-1}_{b,a,o}(s')p(o|s')\,\mathrm{d}o\;\mathrm{d}s' \tag{A.1.2}$$

$$\overset{(A.1.1)}{=} \int\limits_{s'\in\mathbb{S}} \mathring{r}(s,a,s')p(s'|s,a)\,\mathrm{d}s' + \gamma \int\limits_{s'\in\mathbb{S}} p(s'|s,a) \int\limits_{o\in\mathbb{O}} \alpha^{n-1}_{b,a,o}(s')p(o|s')\,\mathrm{d}o\;\mathrm{d}s' \tag{A.1.3}$$

$$= \int\limits_{s'\in\mathbb{S}} p(s'|s,a)\left(\mathring{r}(s,a,s') + \gamma \int\limits_{o\in\mathbb{O}} \alpha^{n-1}_{b,a,o}(s')p(o|s')\,\mathrm{d}o\right)\mathrm{d}s' \,. \tag{A.1.4}$$

For the MC approximations in Section 5.2, $\mathring{r}(\hat{s},a,\hat{s}')$ needs to be computed for every combination of $\hat{s}\in\mathbb{I}_b$, $\hat{s}'\in\mathbb{J}_{b,a}$ and actions $a\in\mathbb{A}$. Due to the biasing of $\mathbb{J}_{b,a}$ towards $p(s'|b,a)$, the reward for the belief can be simply approximated by the sum

$$\tilde{r}_b(\tilde{b},a) = \sum_{\hat{s}\in\mathbb{I}_b}\sum_{\hat{s}'\in\mathbb{J}_{b,a}} \mathring{r}(\hat{s},a,\hat{s}') \,. \tag{A.1.5}$$

To obtain the approximated results of $r(\hat{s},a)$ for the α-function backup, the bias must be compensated. This can be accomplished with the weights $u^{(\hat{s}'\leftarrow\hat{s})}_{b,a}$ from Equation 5.2.17 so that

$$\tilde{r}(\hat{s},a) = \sum_{\hat{s}'\in\mathbb{J}_{b,a}} \mathring{r}(\hat{s},a,\hat{s}')u^{(\hat{s}'\leftarrow\hat{s})}_{b,a} \,. \tag{A.1.6}$$

The reward computation can be integrated into the MC α-function backup from Equation 5.2.22, analogously to Equation A.1.2. The backup with the alternative reward results to

$$\tilde{\alpha}^{n}_{b,a}(s) = \sum_{\hat{s}' \in \mathbb{J}_{b,a}} u^{(\hat{s}' \leftarrow \hat{s})}_{b,a} \left(\mathring{r}(\hat{s}, a, \hat{s}') + \gamma \sum_{\hat{o} \in \mathbb{K}_{b,a}} v^{(\hat{o} \leftarrow \hat{s}')}_{b,a} \, \alpha^{n-1}_{b,a,\hat{o}}(\hat{s}') \right) . \tag{A.1.7}$$

A.2 Global α-function Representation

To evaluate all α-functions in a set Γ for a belief b when using local representations $\langle \mathbb{S}_d, \theta \rangle_\alpha$, a discrete representation of b for every α-function must be obtained. Therefore, the integral of the discretization in Equation 5.3.13 must be computed for every basis function of every α-function representation. For particle-based beliefs with Q samples, $\theta_{s_d}(s)$ has to be computed

$$Q \, |\Gamma| \, |\mathbb{S}_{d,\alpha}| \tag{A.2.1}$$

times. Especially if the continuous state space is high-dimensional and the α-functions are quite complex, this is expensive. Compared to this, the computational effort of the discrete dot products that have to be evaluated after the discretization can be neglected.

We noticed that many α-functions share an underlying structure. The reason for this simply is that they all have the same domain: the continuous state space. Additionally, they depend on the same models for reward and prediction. For example, states where the agent has a collision with an obstacle yield low values for any α-function. States where the agent reaches the goal of the POMDP yield high values for any α-function. Such a high/low value area can spread over the state space with every Bellman backup.

We utilize this finding by applying a single global representation for every α-function. Computation of the value yielded by all α in a belief b then only requires the belief b to be transformed to the (global) discrete representation β_b once. θ has to be computed

$$Q \, |\mathbb{S}_d| \tag{A.2.2}$$

times.

Redundant or recurring patterns in the α-functions can be detected and exploited early, at the time when the representation is created. In other words: we allow the learning algorithm to share components but this is not mandatory. In the worst case, if there is absolutely no redundancy in the α-functions and no components are shared, the number of components in the global state set is

$$|\mathbb{S}_d| = |\Gamma|\,|\mathbb{S}_{d,\alpha}| \tag{A.2.3}$$

and the performance is on a par with the local representation.

An overhead for the discrete dot-product computation is created, however. The computational cost of the dot-product is the cost of $|\mathbb{S}_d|$ multiplications. It is negligible compared to the θ computations and most of the multiplications can even be prevented, when using a sparse representation for the vector β_b. In practice, when components can be shared between α-functions, the local representation is much smaller and significant performance gains can be expected.

A.3 Upper Bound Approximation

V_{UB} is represented by a set of belief-value pairs (BVP) and their values $\left\langle b^{(p)}, V^{(p)}\right\rangle$, accompanied by values for the extreme points $s \in \mathbb{S}$ of the belief simplex $V_{\mathbb{S}} : \mathbb{S} \to \mathbb{R}$. The upper bound approximation for a belief b is obtained by finding the BVP that yields the minimal interpolation value $V_{\mathrm{UB}}(b) = \min_p V_p(b)$ with

$$V_p(b) = \langle V_{\mathbb{S}}, b\rangle - \left(\left\langle V_{\mathbb{S}}, b^{(p)}\right\rangle - V^{(p)}\right) \min_{s\in\mathbb{S}|\ b^{(p)}(s)>0} \frac{b(s)}{b^{(p)}(s)}\,. \tag{A.3.1}$$

As the interpolation only considers the dimension with minimal overlap with the interpolation belief, it is a conservative upper bound approximation. This computation can be effectively carried out in the discrete space $\mathbb{S}_d$ by

$$V_p(b) = \left\langle V_{\mathbb{S}_d}, \beta_b\right\rangle_d - \left(\left\langle V_{\mathbb{S}_d}, \beta_{b^{(p)}}\right\rangle_d - V^{(p)}\right) \min_{s_d\in\mathbb{S}_d|\ \beta_{b^{(p)}}(s_d)>0} \frac{\beta_b(s_d)}{\beta_{b^{(p)}}(s_d)}\,. \tag{A.3.2}$$

The precondition for this computation in order to actually resemble the continuous interpolation is that θ accurately represents beliefs. Note that this dissents from the loss in the α-function representations. Ultimate accuracy of the upper bound, however, is not crucial because it is used only for belief space exploration. In most cases, it suffices for this task. The remaining cases can be covered by adding randomness to the exploration.

To update the upper bound, BVPs are added, when a belief is explored or traversed. BVPs are created through MC belief state Bellman backup in Equation 5.2.8 with the next beliefs' values determined by the upper bound interpolation

$$\tilde{V}^{n}_{\mathrm{UB}}(b) = \max_{a \in \mathbb{A}} \tilde{V}^{n}_{a,\mathrm{UB}}(b) \tag{A.3.3}$$

$$\tilde{V}^{n}_{a,\mathrm{UB}}(b) = \left(\sum_{\hat{s} \in \mathbb{I}} r(\hat{s}, a) \right) + \gamma \sum_{\hat{o} \in \mathbb{K}_{b,a}} \tilde{V}^{n-1}_{\mathrm{UB}}(b'_{b,a,\hat{o}}) \,. \tag{A.3.4}$$

The extreme or corner points $V_{\mathbb{S}_d}$ can be efficiently updated by a fast informed bounds (FIB) update (see [Hauskrecht, 2000]).

Acronyms

BN Bayesian network.

BOF Bayesian occupancy filter.

DBN dynamic Bayesian network.

DDN dynamic decision network.

DN decision network.

DP dynamic programming.

GMM Gaussian mixture model.

HSVI heuristic search value iteration.

IS importance sampling.

KL divergence Kullback–Leibler divergence.

LQG linear-quadratic Gaussian.

MC Monte Carlo.

MDP Markov decision process.

MOMDP mixed observable Markov decision process.

MPC model predictive control.

PBVI point-based value iteration.

POMDP partially observable Markov decision process.

PWLC piecewise-linear and convex.

QMDP POMDP approximation with Q-functions.

RL	reinforcement learning.
SIR	sequential importance resampling.
SLAM	simultaneous localization and mapping.
SMC	sequential Monte Carlo.
UMDP	unobservable Markov decision process.

Glossary

belief Probability distribution over the state space. Representation of situations.

Bellman backup Propagating the *value* (and gradient in case of α-backups) back in time in order to assess future consequences of actions.

decision-making Cognitive process of selecting one of several actions in order to optimize criteria or fulfill a goal.

discrete representation Discrete set of parameters that expresses a (potentially continuous) function. In this work, the term is mostly used for state space representations.

ego vehicle The vehicle that is under control of the agent.

inductive learning Learning a generalization from specific examples.

planning The cognitive process of finding the *optimal policy* by predicting developments of *situations* in order to assess consequences of actions.

policy Determines the action for every situation. The *optimal* policy yields the best possible value.

rational agent Autonomous entity which perceives the state of the world through *observations* and influences its development over time through *actions*.

reward Immediate return for the agent when reaching a good state. Opposite of *cost*.

situation Information that an agent has about the state of the world to base his decisions on.

state Condition of variables at a certain point in time describing aspects of the agent's environment.

value Expectation of sum over future rewards. Maximizing the value is the goal of decision-making.

List of Figures

List of Algorithms

Bibliography

Alin, A., Fritsch, J., and Butz, M. V. Improved Tracking and Behavior Anticipation by Combining Street Map Information with Bayesian-Filtering. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 2235–2242, 2013.

Althoff, M. and Dolan, J. Online Verification of Automated Road Vehicles Using Reachability Analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

Ardelt, M., Coester, C., and Kaempchen, N. Highly Automated Driving on Freeways in Real Traffic Using a Probabilistic Framework. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1576–1585, 2012.

Astrom, K. Optimal Control of Markov Processes with Incomplete State Information II. The Convexity of the Lossfunction. *Journal of Mathematical Analysis and Applications*, 26(2):403–406, 1969.

Ayer, T., Alagoz, O., and Stout, N. K. OR Forum-A POMPD Approach to Personalize Mammography Screening Decisions. *Operations Research*, 60(5):1019–1034, 2012.

Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., , Cacciola, S., Currier, P., , Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Covern, D. V., and Webster, M. Odin: Team VictorTango's Entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25(8): 467–492, 2008.

Bahram, M., Wolf, A., Aeberhard, M., and Wollherr, D. A Prediction-based Reactive Driving Strategy for Highly Automated Driving Function on Freeways. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 400–406, 2014.

Bai, H., Hsu, D., Lee, W., and V.A., N. Monte Carlo Value Iteration for Continuous-state POMDPs. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pages 175–191, 2010.

Bai, H. *Continuous POMDPs for Robotic Tasks.* PhD thesis, National University of Singapore, 2014.

Bai, H., Hsu, D., Kochenderfer, M. J., and Lee, W. S. Unmanned Aircraft Collision Avoidance Using Continuous-state POMDPs. In *Proceedings of the Robotics: Science and Systems Conference*, 2011.

Bai, H., Hsu, D., and Lee, W. S. Integrated Perception and Planning in the Continuous Space: A POMDP Approach. *The International Journal of Robotics Research*, 33(9): 1288–1302, 2014.

Baker, C. R. and Dolan, J. M. Traffic Interaction in the Urban Challenge: Putting Boss on its Best Behavior. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1752–1758, 2008.

Bandyopadhyay, T., Jie, C. Z., Hsu, D., Ang Jr, M. H., Rus, D., and Frazzoli, E. Intention-aware Pedestrian Avoidance. In *Proceedings of the International Symposium on Experimental Robotics*, pages 963–977. Springer, 2013a.

Bandyopadhyay, T., Won, K. S., Frazzoli, E., Hsu, D., Lee, W. S., and Rus, D. Intention-aware Motion Planning. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pages 475–491. Springer, 2013b.

Bar-Shalom, Y. Multitarget-multisensor Tracking: Applications and Advances. Volume III. *Norwood, MA: Artech House*, 2000.

Bartlett, P. L. and Baxter, J. Infinite-horizon Policy-gradient Estimation. *Journal of Artificial Intelligence Research 15*, pages 319–350, 2001.

Bellman, R. *Dynamic Programming*. Princeton University Press, 1957a.

Bellman, R. A Markovian Decision Process. *Journal of Applied Mathematics and Mechanics*, 6:679–684, 1957b.

Bengio, Y. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

Bentley, J. L. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

Bertsekas, D. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Englewood Cliffs, NJ., 1987.

Bertsekas, D. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2007.

Boger, J., Poupart, P., Hoey, J., Boutilier, C., Fernie, G., and Mihailidis, A. A Decision-theoretic Approach to Task Assistance for Persons with Dementia. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1293–1299, 2005.

Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J., and Satterfield, B. Little Ben: The Ben Franklin Racing Team's Entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

Bonet, B. An ϵ-optimal Grid-based Algorithm for Partially Observable Markov Decision Processes. In *Proceedings of the International Conference on Machine Learning*, pages 51–58, 2002.

Botvinick, M. and Toussaint, M. Planning as Inference. *Trends in Cognitive Sciences*, 16(10):485–488, 2012.

Braess, H.-H. and Seiffert, U. *Vieweg Handbuch Kraftfahrzeugtechnik*. Springer-Verlag, 2011.

Brafman, R. I. A Heuristic Variable Grid Solution Method for POMDPs. In *Proceedings of the AAAI Conference on Innovative Applications of Artificial Intelligence*, pages 727–733, 1997.

Brechtel, S., Gindele, T., Vogelgesang, J., and Dillmann, R. Probabilistisches Belegtheitsfilter zur Schätzung dynamischer Umgebungen unter Verwendung multipler Bewegungsmodelle. In *Proceedings of the Fachgespräch Autonome Mobile Systeme*, pages 49–56, Karlsruhe, 2009.

Brechtel, S., Gindele, T., and Dillmann, R. Recursive Importance Sampling for Efficient Grid-based Occupancy Filtering in Dynamic Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3932–3938, Anchorage, AL, USA, 2010.

Brechtel, S., Gindele, T., and Dillmann, R. Probabilistic MDP-behavior Planning for Cars. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1537–1542, Washington DC, USA, 2011.

Brechtel, S., Gindele, T., et al. Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation. In *Proceedings of the International Conference on Machine Learning*, pages 370–378, 2013.

Brechtel, S., Gindele, T., and Dillmann, R. Probabilistic Decision-making under Uncertainty for Autonomous Driving Using Continuous POMDPs. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 392–399, Qingdao, China, 2014.

Breiman, L. Bagging Predictors. *Machine learning*, 24(2):123–140, 1996.

Breiman, L. Random Forests. *Machine learning*, 45(1):5–32, 2001.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and Regression Trees*. CRC Press, 1984.

Brooks, A., Makarenko, A., Williams, S., and Durrant-Whyte, H. Parametric POMDPs for Planning in Continuous State Spaces. *Robotics and Autonomous Systems*, 54(11): 887–897, 2006.

Bry, A. and Roy, N. Rapidly-exploring Random Belief Trees for Motion Planning under Uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 723–730, 2011.

Buehler, M., Iagnemma, K., and Singh, S. The 2005 DARPA Grand Challenge. *Springer Tracts in Advanced Robotics*, 36(5):1–43, 2007.

Buehler, M., Iagnemma, K., and Singh, S. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, volume 56. Springer, 2009.

Bui, H. H. A General Model for Online Probabilistic Plan Recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 18, pages 1309–1318, 2003.

Cassandra, A., Littman, M. L., and Zhang, N. L. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 54–61. Morgan Kaufmann Publishers Inc., 1997.

Cassandra, A. R. A Survey of POMDP Applications. In *Proceedings of the Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24, 1998.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. Acting Optimally in Partially Observable Stochastic Domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 94, pages 1023–1028, 1994.

Cheng, H.-T. Algorithms for Partially Observable Markov Decision Processes. 1988.

City of Karlsruhe. Basis of Map Data for Illustrations: @ Stadt Karlsruhe.

Copi, I. M. *Introduction to Logic*. Number 49. Pearson/Prentice Hall, 2008.

Coue, C., Pradalier, C., Laugier, C., Fraichard, T., and Bessiere, P. Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application. *The International Journal of Robotics Research*, 25(1):19, 2006.

Dagli, I., Brost, M., and Breuel, G. Action Recognition and Prediction for Driver Assistance Systems Using Dynamic Belief Networks. *Lecture Notes in Computer Science*, pages 179–194, 2003.

Dickmanns, E., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., Thomanek, F., and Schiehlen, J. The Seeing Passenger Car 'VaMoRs-P'. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 68–73, 1994.

Dietterich, T. G. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40(2):139–157, 2000.

Doucet, A., De Freitas, N., and Gordon, N. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

Doucet, A., De Freitas, N., Murphy, K., and Russell, S. Rao-blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000.

Drake, A. W. *Observation of a Markov Process through a Noisy Channel*. PhD thesis, Massachusetts Institute of Technology, 1962.

Duff, S. J. and Bradtke, M. O. Reinforcement Learning Methods for Continuous-time Markov Decision Problems. In *Proceedings of the Neural Information Processing Systems Conference*, volume 7, page 393, 1995.

Elfes, A. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 22(6):46–57, 1989.

Erez, T. and Smart, W. D. A Scalable Method for Solving High-dimensional Continuous POMDPs Using Local Approximation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2010.

Ericson, C. *Real-time Collision Detection*. Morgan Kaufmann Series in Interactive 3D Technology. Elsevier, Amsterdam, 2005.

Esposito, F., Malerba, D., Semeraro, G., and Kay, J. A Comparative Analysis of Methods for Pruning Decision Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

Feng, Z. and Hansen, E. An Approach to State Aggregation for POMDPs. In *Proceedings of the Workshop on Learning and Planning in Markov Processes*, pages 7–12, 2004.

Foka, A. and Trahanias, P. Real-time Hierarchical POMDPs for Autonomous Robot Navigation. *Robotics and Autonomous Systems*, 55(7):561–571, 2007.

Folsom-Kovarik, J. T., Sukthankar, G., Schatz, S. L., and Nicholson, D. M. Scalable POMDPs for Diagnosis and Planning in Intelligent Tutoring Systems. In *Proceedings of the AAAI Fall Symposium: Proactive Assistant Agents*, 2010.

Forbes, J., Huang, T., Kanazawa, K., and Russell, S. The Batmobile: Towards a Bayesian Automated Taxi. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 14, pages 1878–1885, Montreal, Quebec Canada, 1995.

Frese, C. and Beyerer, J. A Comparison of Motion Planning Algorithms for Cooperative Collision Avoidance of Multiple Cognitive Automobiles. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1156–1162, 2011.

Freund, Y., Schapire, R., and Abe, N. A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(771–780):1612, 1999.

Furda, A. and Vlacic, L. Enabling Safe Autonomous Driving in Real-world City Traffic Using Multiple Criteria Decision Making. *IEEE Intelligent Transportation Systems Magazine*, 3(1):4–17, 2011.

Furda, A. *Real-time Decision Making by Driverless City Vehicles: A Discrete Event Driven Approach*. PhD thesis, Griffith University, 2010.

FZI Forschungszentrum Informatik (Research Center for Information Technology). CoCar—The Instrumented Cognitive Car. URL `www.fzi.de/en/research/projekt-details/cocar/`. Last accessed April 14, 2015.

Geiger, A. *Probabilistic Models for 3D Urban Scene Understanding from Movable Platforms*. PhD thesis, Karlsruher Institute of Technology (KIT), Karlsruhe, 2013.

Gindele, T., Jagszent, D., Pitzer, B., and Dillmann, R. Design of the Planner of Team AnnieWAY's Autonomous Vehicle Used in the DARPA Urban Challenge 2007. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1131–1136, 2008.

Gindele, T. *Learning Behavior Models for Interpreting and Predicting Traffic Situations*. PhD thesis, Karlsruher Institute of Technology (KIT), 2014.

Gindele, T., Brechtel, S., Schröder, J., and Dillmann, R. Bayesian Occupancy Grid Filter for Dynamic Environments Using Prior Map Knowledge. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 669–676, Xi'an, China, 2009.

Gindele, T., Brechtel, S., and Dillmann, R. A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1625–1631, Madeira, Portugal, 2010.

Gindele, T., Brechtel, S., and Dillmann, R. Learning Context Sensitive Behavior Models from Observations for Predicting Traffic Situations. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1764–1771, 2013.

Gindele, T., Brechtel, S., and Dillmann, R. Learning Driver Behavior Models from Traffic Observations for Decision Making and Planning. *Intelligent Transporation Systems Magazine Special Issue on ITSC 2013*, 7(1):69–79, 2015.

Girault, A. A Hybrid Controller for Autonomous Vehicles Driving on Automated Highways. *Transportation Research Part C: Emerging Technologies*, 12(6):421–452, 2004.

Google. Self-Driving Car Project. URL `plus.google.com/+GoogleSelfDrivingCars`. Last accessed April 14, 2015.

Grisetti, G., Stachniss, C., and Burgard, W. Improved Techniques for Grid Mapping with Rao-blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

Grisetti, G., Kummerle, R., Stachniss, C., and Burgard, W. A Tutorial on Graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

Hahn, J. K. Realistic Animation of Rigid Bodies. In *Proceedings of the ACM SIGGRAPH Computer Graphics*, volume 22, pages 299–308, 1988.

Hansen, E. A. Solving POMDPs by Searching in Policy Space. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 211–219, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

Hauskrecht, M. Value-function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13(1):33–94, 2000.

Hauskrecht, M. and Fraser, H. Modeling Treatment of Ischemic Heart Disease with Partially Observable Markov Decision Processes. In *Proceedings of the American Medical Informatics Association Symposium*, page 538, 1998a.

Hauskrecht, M. and Fraser, H. Planning Medical Therapy Using Partially Observable Markov Decision Processes. In *Proceedings of the International Workshop on Principles of Diagnosis*, pages 182–189, 1998b.

Ho, T. K. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

Hoey, J. and Poupart, P. Solving POMDPs with Continuous or Large Discrete Observation Spaces. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 1332, 2005.

Hoey, J., von Bertoldi, A., Poupart, P., and Mihailidis, A. Assisting Persons with Dementia during Handwashing Using a Partially Observable Markov Decision Process. In *Proceedings of the International Conference on Vision Systems*, Bielefeld, Germany, 2007.

Hogarth, R. M. Cognitive Processes and the Assessment of Subjective Probability Distributions. *Journal of the American Statistical Association*, 70(350):271–289, 1975.

Howard, R. *Dynamic Programming and Markov Processes.* The MIT Press, 1960.

Hsiao, K., Kaelbling, L. P., and Lozano-Perez, T. Grasping POMDPs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4685–4692, 2007.

Hsu, D., Lee, W. S., and Rong, N. What Makes some POMDP Problems Easy to Approximate? In *Proceedings of the Neural Information Processing Systems Conference*, pages 689–696, 2007.

Indelman, V., Carlone, L., and Dellaert, F. Planning under Uncertainty in the Continuous Domain: A Generalized Belief Space Approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6763–6770, 2014.

Kaelbling, L. P. *Learning in Embedded Systems.* MIT Press, 1993.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1):99–134, 1998.

Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., Schröder, J., Thuy, M., Goebl, M., von Hundelshausen, F., Pink, O., Frese, C., and Stiller, C. Team AnnieWAY's Autonomous System for the DARPA Urban Challenge 2007. *International Journal of Field Robotics Research*, 2008.

Kasper, D., Weidl, G., Dang, T., Breuel, G., Tamke, A., and Rosenstiel, W. Object-oriented Bayesian Networks for Detection of Lane Change Maneuvers. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 673–678, 2011.

Kohlhaas, R., Bittner, T., Schamm, T., and Zollner, J. Semantic State Space for High-level Maneuver Planning in Structured Traffic Scenes. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1060–1065, 2014.

Kretzschmar, H., Kuderer, M., and Burgard, W. Learning to Predict Trajectories of Cooperatively Navigating Agents. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4015–4020, 2014.

Kuderer, M., Kretzschmar, H., Sprunk, C., and Burgard, W. Feature-based Prediction of Trajectories for Socially Compliant Navigation. In *Proceedings of the Robotics: Science and Systems Conference*, 2012.

Kuderer, M., Gulati, S., and Burgard, W. Learning Driving Styles for Autonomous Vehicles from Demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Seattle, USA, 2015.

Kurniawati, H., Hsu, D., and Lee, W. SARSOP: Efficient Point-based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Proceedings of the Robotics: Science and Systems Conference*, Zürich, Switzerland, 2008.

Lee, T. and Kim, Y. J. GPU-based Motion Planning under Uncertainties Using POMDP. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4576–4581, 2013.

Lefèvre, S., Ibanez-Guzman, J., and Laugier, C. Context-based Estimation of Driver Intent at Road Intersections. In *Proceedings of the IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems*, 2011.

Littman, M. L., Dean, T. L., and Kaelbling, L. P. On the Complexity of Solving Markov Decision Problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 394–402. Morgan Kaufmann Publishers Inc., 1995.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. Efficient Dynamic-programming Updates in Partially Observable Markov Decision Processes. Technical report, 1996.

Littman, M. L. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

Lovejoy, W. S. Computationally Feasible Bounds for Partially Observed Markov Decision Processes. *Operations Research*, 39(1):162–175, 1991a.

Lovejoy, W. S. A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes. *Annals of Operations Research*, 28(1):47–65, 1991b.

MacKay, D. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

Madani, O., Hanks, S., and Condon, A. On the Undecidability of Probabilistic Planning and Infinite-horizon Partially Observable Markov Decision Problems. In *Proceedings of the AAAI Conference on Innovative Applications of Artificial Intelligence*, pages 541–548, 1999.

McNaughton, M., Urmson, C., Dolan, J. M., and Lee, J.-W. Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4889–4895, 2011.

Mercedes-Benz. Mercedes-Benz, S-Class S 500 INTELLIGENT DRIVE. URL `http://media.daimler.com`. Photo number: 13C857_10, Last accessed April 14, 2015.

Meuleau, N., Kim, K.-E., Kaelbling, L. P., and Cassandra, A. R. Solving POMDPs by Searching the Space of Finite Policies. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 417–426. Morgan Kaufmann Publishers Inc., 1999.

Miller, I., Campbell, M., Huttenlocher, D., Kline, F.-R., Nathan, A., Lupashin, S., Catlin, J., Schimpf, B., Moran, P., Zych, N., Garcia, E., Kurdziel, M., and Fujishima, H. Team Cornell's Skynet: Robust Perception and Planning in an Urban Environment. *Journal of Field Robotics*, 25(8):493–527, 2008.

Monahan, G. E. State of the Art—A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms. *Management Science*, 28(1):1–16, 1982.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al. Junior: The Stanford Entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

Moosmann, F. *Interlacing Self-Localization, Moving Object Tracking and Mapping for 3D Range Sensors*. PhD thesis, Karlsruher Institute of Technology (KIT), 2013.

Munos, R. and Moore, A. Variable Resolution Discretization in Optimal Control. *Machine Learning*, 49(2):291–323, 2002.

Munos, R. and Moore, A. W. Variable Resolution Discretization for High-accuracy Solutions of Optimal Control Problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, page 256, 1999.

Murthy, S. K., Kasif, S., and Salzberg, S. A System for Induction of Oblique Decision Trees. *Journal of Artificial Intelligence Research*, 1994.

Ng, A. Y. and Jordan, M. PEGASUS: A Policy Search Method for Large MDPs and POMDPs. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 406–415. Morgan Kaufmann Publishers Inc., 2000.

Ngai, D. C. K. and Yung, N. H. C. A Multiple-goal Reinforcement Learning Method for Complex Vehicle Overtaking Maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):509–522, 2011.

Ngai, D. C. and Yung, N. H. Automated Vehicle Overtaking Based on a Multiple-goal Reinforcement Learning Framework. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 818–823, 2007.

Nienhüser, D. *Kontextsensitive Erkennung und Interpretation fahrrelevanter statischer Verkehrselemente*. PhD thesis, Karlsruher Institute of Technology (KIT), 2014.

Nilsson, J. and Sjoberg, J. Strategic Decision Making for Automated Driving on Two-lane, One Way Roads Using Model Predictive Control. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 1253–1258, 2013.

Ong, S. C., Hsu, D., Lee, W. S., and Kurniawati, H. Partially Observable Markov Decision Process (POMDP) Technologies for Sign Language Based Human-computer Interaction. *Universal Access in Human-Computer Interaction. Applications and Services*, 5616(4):577–586, 2009.

Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, 29 (8):1053–1068, 2010.

Papadimitriou, C. H. and Tsitsiklis, J. N. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Petrich, D., Dang, T., Kasper, D., Breuel, G., and Stiller, C. Map-based Long Term Motion Prediction for Vehicles in Traffic Environments. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 2166–2172, 2013.

Pineau, J., Gordon, G., and Thrun, S. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 18, pages 1025–1032, Acapulco, Mexico, 2003.

Pineau, J., Gordon, G., and Thrun, S. Policy-contingent Abstraction for Robust Robot Control. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 477–484. Morgan Kaufmann Publishers Inc., 2002.

Pineau, J., Gordon, G. J., and Thrun, S. Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

Pitt, M. K. and Shephard, N. Filtering via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

Platt Jr, R., Tedrake, R., Kaelbling, L., and Lozano-Perez, T. Belief Space Planning Assuming Maximum Likelihood Observations. In *Proceedings of the Robotics: Science and Systems Conference*, Zaragoza, Spain, 2010.

Pomerleau, D. Defense and Civilian Applications of the ALVINN Robot Driving System. In *Proceedings of the Government Microcircuit Applications Conference*, pages 358–362, 1994.

Pomerleau, D. A. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation*, 3(1):88–97, 1991.

Porta, J., Spaan, M., and Vlassis, N. Robot Planning in Partially Observable Continuous Domains. In *Proceedings of the Robotics: Science and Systems Conference*, volume 1, page 217. The MIT Press, 2005.

Porta, J., Vlassis, N., Spaan, M., and Poupart, P. Point-based Value Iteration for Continuous POMDPs. *The Journal of Machine Learning Research*, 7:2329–2367, 2006.

Poupart, P. and Boutilier, C. Value-directed Compression of POMDPs. In *Proceedings of the Neural Information Processing Systems Conference*, volume 15, pages 1547–1554, 2002.

Poupart, P., Kim, K., and Kim, D. Closing the Gap: Improved Bounds on Optimal POMDP Solutions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2011.

Poupart, P. and Boutilier, C. Bounded Finite State Controllers. In *Proceedings of the Neural Information Processing Systems Conference*, pages 823–830, 2003.

Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.

Quinlan, J. R. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

Quinlan, J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Rafferty, A. N., Brunskill, E., Griffiths, T. L., and Shafto, P. Faster Teaching by POMDP Planning. In *Proceedings of the Artificial Intelligence in Education*, pages 280–287. Springer, 2011.

Raimondi, F. M. and Melluso, M. Fuzzy Motion Control Strategy for Cooperation of Multiple Automated Vehicles with Passengers Comfort. *Automatica*, 44(11):2804–2816, 2008.

Rauskolb, F. W., Berger, K., Lipski, C., Magnor, M., Cornelsen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., et al. Caroline: An Autonomously Driving Vehicle for Urban Environments. *Journal of Field Robotics*, 25(9):674–724, 2008.

Riedmiller, M., Montemerlo, M., and Dahlkamp, H. Learning to Drive a Real Car in 20 Minutes. In *Proceedings of the IEEE Frontiers in the Convergence of Bioscience and Information Technologies*, pages 645–650, 2007.

Rokach, L. and Maimon, O. Top-down Induction of Decision Trees Classifiers—A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):476–487, 2005.

Rosenblatt, J. K. DAMN: A Distributed Architecture for Mobile Navigation. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2–3):339–360, 1997.

Roy, N. and Gordon, G. Exponential Family PCA for Belief Compression in POMDPs. In *Proceedings of the Neural Information Processing Systems Conference*, volume 15, pages 1635–1642, 2002.

Roy, N., Gordon, G., and Thrun, S. Finding Approximate POMDP Solutions through Belief Compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

Roy, N., Gordon, G., and Thrun, S. Planning under Uncertainty for Reliable Health Care Robotics. In *Proceedings of the Field and Service Robotics*, pages 417–426. Springer, 2006.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*, volume 25. Prentice-Hall, Englewood Cliffs, NJ., 1995.

Sawaki, K. Optimal Control for Partially Observable Markov Decision Processes over an Infinite Horizon. *Journal of the Operations Research Society of Japan*, 21:1–16, 1978.

Schamm, T. *Modellbasierter Ansatz zur probabilistischen Interpretation von Fahrsituationen*. PhD thesis, Karlsruher Institute of Technology (KIT), 2014.

Schapire, R. E. The Boosting Approach to Machine Learning: An Overview. In *Proceedings of the Nonlinear Estimation and Classification*, pages 149–171. Springer, 2003.

Schröder, J. *Adaptive Verhaltensentscheidung und Bahnplanung für kognitive Automobile*. PhD thesis, Karlsruher Institute of Technology (KIT), Karlsruhe, 2009.

Schubert, R. Evaluating the Utility of Driving: Toward Automated Decision Making under Uncertainty. *IEEE Transactions on Intelligent Transportation Systems*, 13(1): 354–364, 2012.

Schwarting, W. and Pascheka, P. Recursive Conflict Resolution for Cooperative Motion Planning in Dynamic Highway Traffic. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 1039–1044, 2014.

Shani, G., Brafman, R. I., and Shimony, S. E. Forward Search Value Iteration for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2619–2624, 2007.

Shani, G., Pineau, J., and Kaplow, R. A Survey of Point-based POMDP Solvers. *Autonomous Agents and Multi-agent Systems*, 27(1):1–51, 2013.

Smallwood, R. and Sondik, E. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research*, pages 1071–1088, 1973.

Smith, T. and Simmons, R. Heuristic Search Value Iteration for POMDPs. In *Proceedings of the Uncertainty in Artificial Intelligence*, pages 520–527, 2004.

Smith, T. and Simmons, R. Point-based POMDP Algorithms: Improved Analysis and Implementation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2012.

Smith, T., Thompson, D. R., and Wettergreen, D. Generating Exponentially Smaller POMDP Models Using Conditionally Irrelevant Variable Abstraction. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 304–311, 2007.

Sondik, E. J. The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Operations Research*, 26(2):282–304, 1978.

Sondik, E. The Optimal Control of Partially Observable Markov Decision Processes. *PhD thesis, Stanford University*, 1971.

Spaan, M. and Vlassis, N. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195–220, 2005.

Sutton, R. and Barto, A. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Taoka, G. T. Brake Reaction Times of Unalerted Drivers. *Journal of Transportation of the Institute of Transportation Engineers*, 59(3):19–21, 1989.

Tay, M., Mekhnacha, K., Chen, C., and Yguel, M. An Efficient Formulation of the Bayesian Occupation Filter for Target Tracking in Dynamic Environments. *International Journal of Vehicle Autonomous Systems*, 6(1):155–171, 2008.

Temizer, S., Kochenderfer, M. J., Kaelbling, L. P., Lozano-Pérez, T., and Kuchar, J. K. Collision Avoidance for Unmanned Aircraft Using Markov Decision Processes. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Toronto, Canada*, 2010.

Theocharous, G. *Hierarchical Learning and Planning in Partially Observable Markov Decision Processes.* PhD thesis, Michigan State University, 2002.

Thrun, S. Monte Carlo POMDPs. In *Proceedings of the Neural Information Processing Systems Conference*, pages 1064–1070, 2000.

Thrun, S. Robotic Mapping: A Survey. *Exploring Artificial Intelligence in the New Millennium*, 2002.

Thrun, S., Burgard, W., and Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* MIT Press, Cambridge, Massachusetts, USA, 2005.

Todorov, E. and Li, W. A Generalized Iterative LQG Method for Locally-optimal Feedback Control of Constrained Nonlinear Stochastic Systems. In *Proceedings of the American Control Conference*, pages 300–306, 2005.

Ulbrich, S. and Maurer, M. Probabilistic Online POMDP Decision Making for Lane Changes in Fully Automated Driving. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 2063–2070, 2013.

Ulbrich, S., Nothdurft, T., Maurer, M., and Hecker, P. Graph-based Context Representation, Environment Modeling and Information Aggregation for Automated Driving. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 541–547, 2014.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C., et al. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Van Den Berg, J., Patil, S., and Alterovitz, R. Motion Planning under Uncertainty Using Iterative Local Optimization in Belief Space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

Vogel, K. A Comparison of Headway and Time to Collision as Safety Indicators. *Accident Analysis & Prevention*, 35(3):427–433, 2003.

Waharte, S. and Trigoni, N. Supporting Search and Rescue Operations with UAVs. In *Proceedings of the International Conference on Emerging Security Technologies*, pages 142–147, 2010.

Wei, J., Dolan, J. M., Snider, J. M., and Litkouhi, B. A Point-based MDP for Robust Single-lane Autonomous Driving Behavior under Uncertainties. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2586–2592, 2011.

Werling, M., Ziegler, J., Kammel, S., and Thrun, S. Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 987–993, 2010.

Werling, M., Kammel, S., Ziegler, J., and Gröll, L. Optimal Trajectories for Time-critical Street Scenarios Using Discretized Terminal Manifolds. *The International Journal of Robotics Research*, pages 346–359, 2011.

White III, C. C. A Survey of Solution Techniques for the Partially Observed Markov Decision Process. *Annals of Operations Research*, 32(1):215–230, 1991.

Williams, J. D. and Young, S. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech & Language*, 21(2):393–422, 2007.

Williams, R. J. and Baird, L. C. Tight Performance Bounds on Greedy Policies Based on Imperfect Value Functions. Technical report, 1993.

Xu, W., Pan, J., Wei, J., and Dolan, J. M. Motion Planning under Uncertainty for on-road Autonomous Driving. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2507–2512, 2014.

Young, S. Using POMDPs for Dialog Management. In *Proceedings of the IEEE Spoken Language Technology Workshop*, pages 8–13, 2006.

Young, S., Gasic, M., Thomson, B., and Williams, J. D. POMPD-based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

Zhang, J. and Rössler, B. Situation Analysis and Adaptive Risk Assessment for Intersection Safety Systems in Advanced Assisted Driving. In *Proceedings of the Fachgespräch Autonome Mobile Systeme*, pages 249–258, 2009.

Zhang, N. L. and Liu, W. Planning in Stochastic Domains: Problem Characteristics and Approximation. Technical report, The Hong Kong University of Science and Technology, 1996.

Zhang, N. L. and Zhang, W. Speeding Up the Convergence of Value Iteration in Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 14:29–51, 2001.

Zhang, N. L. and Liu, W. A Model Approximation Scheme for Planning in Partially Observable Stochastic Domains. *Journal of Artificial Intelligence Research*, 7:199–230, 1997.

Zhang, Z., Hsu, D., and Lee, W. S. Covering Number for Efficient Heuristic-based POMDP Planning. In *Proceedings of the International Conference on Machine Learning*, pages 28–36, 2014.

Zhou, E., Fu, M., and Marcus, S. Solving Continuous-state POMDPs via Density Projection. *Transactions on Automatic Control*, 55(5):1101–1116, 2010.

Ziegler, Bender, P., Dang, T., and Stiller, C. Trajectory Planning for Bertha — A Local, Continuous Method. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 450–457, 2014a.

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C., et al. Making Bertha Drive? An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, 6(2): 8–20, 2014b.