Towards Human-Like Prediction and Decision-Making for Automated Vehicles in Highway Scenarios

David Sierra Gonzalez

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

David SIERRA GONZÁLEZ

Thèse dirigée par **Emmanuel MAZER**
codirigée par **Christian LAUGIER**
et coencadrée par **Jilles Steeve DIBANGOYE**

préparée au sein de **Inria Grenoble Rhône-Alpes**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Towards Human-Like Prediction and Decision-Making for Automated Vehicles in Highway Scenarios

Thèse soutenue publiquement le **01 Avril 2019**,
devant le jury composé de :

M. James L. CROWLEY
Professeur, Grenoble INP et Inria Grenoble, France, Président
M. Philippe MARTINET
Directeur de Recherche, Inria Sophia Antipolis, France, Rapporteur
M. Luis MERINO
Professeur, Universidad Pablo de Olavide, Espagne, Rapporteur
Mme. Anne VERROUST-BLONDET
Chargée de Recherche, Inria Paris, France, Examinatrice
M. Nicolas VIGNARD
Ingénieur R&D, Toyota Motor Europe, Belgique, Invité
M. Jilles Steeve DIBANGOYE
Maître de conférences, INSA Lyon et Inria Grenoble, France, Co-Encadrant
M. Christian LAUGIER
Directeur de Recherche, Inria Grenoble, France, Co-Directeur de thèse

Abstract

During the past few decades automakers have consistently introduced technological innovations aimed to make road vehicles safer. The level of sophistication of these advanced driver assistance systems has increased parallel to developments in sensor technology and embedded computing power. More recently, a lot of the research made both by industry and institutions has concentrated on achieving fully automated driving. The potential societal benefits of this technology are numerous, including safer roads, improved traffic flows, increased mobility for the elderly and the disabled, and optimized human productivity. However, before autonomous vehicles can be commercialized they should be able to safely share the road with human drivers. In other words, they should be capable of inferring the state and intentions of surrounding traffic from the raw data provided by a variety of onboard sensors, and to use this information to make safe navigation decisions. Moreover, in order to truly navigate safely they should also consider potential obstacles not observed by the sensors (such as occluded vehicles or pedestrians). Despite the apparent complexity of the task, humans are extremely good at predicting the development of traffic situations. After all, the actions of any traffic participant are constrained by the road network, by the traffic rules, and by a risk-aversive common sense. The lack of this ability to naturally understand a traffic scene constitutes perhaps the major challenge holding back the large-scale deployment of truly autonomous vehicles in the roads.

This thesis addresses the full pipeline from driver behavior modeling and inference to decision-making for navigation. In the first place, we model the behavior of a generic driver automatically from demonstrated driving data, avoiding thus the traditional hand-tuning of the model parameters. This model encodes the preferences of a driver with respect to the road network (e.g. preferred lane or speed) and also with respect to other road users (e.g. preferred distance to the leading vehicle). Secondly, we describe a method that exploits the learned model to predict the most likely future sequence of actions of any driver in a traffic scene up to the distant future. This model-based prediction method assumes that all traffic participants behave in a risk-aware manner and can therefore fail to predict dangerous maneuvers or accidents. To be able to handle such cases, we propose a more sophisticated probabilistic model that estimates the states and intentions of surrounding traffic by combining the model-based prediction with the dynamic evidence provided by the sensors. In a way, the proposed model mimics the reasoning process of human drivers: we know what a given vehicle is likely to do given the situation (this is given by the model), but we closely monitor its dynamics to detect deviations from the expected behavior. In this work, we demonstrate how combining both sources of information results in an increased robustness of the intention estimates in comparison with

approaches relying only on dynamic evidence. Finally, the learned driver behavioral model and the prediction models are integrated within a probabilistic decision-making framework. The proposed methods are validated with real-world data collected with an instrumented vehicle. Although focused on highway environments, this work could be adapted to handle alternative traffic scenarios.

Keywords: Autonomous driving, driver behavior modeling, motion prediction, decision-making under uncertainty

Résumé

Au cours des dernières décennies, les constructeurs automobiles ont constamment introduit des innovations technologiques visant à rendre les véhicules plus sûrs. Le niveau de sophistication de ces systèmes avancés d'aide à la conduite s'est accru parallèlement aux progrès de la technologie des capteurs et de la puissance informatique intégrée. Plus récemment, une grande partie de la recherche effectuée par l'industrie et les institutions s'est concentrée sur l'obtention d'une conduite entièrement automatisée. Les avantages sociétaux potentiels de cette technologie sont nombreux, notamment des routes plus sûres, des flux de trafic améliorés et une mobilité accrue pour les personnes âgées et les handicapés. Toutefois, avant que les véhicules autonomes puissent être commercialisés, ils doivent pouvoir partager la route en toute sécurité avec d'autres véhicules conduits par des conducteurs humains. En d'autres termes, ils doivent pouvoir déduire l'état et les intentions du trafic environnant à partir des données brutes fournies par divers capteurs embarqués, et les utiliser afin de pouvoir prendre les bonnes décisions de conduite sécurisée. Malgré la complexité apparente de cette tâche, les conducteurs humains ont la capacité de prédire correctement l'évolution du trafic environnant dans la plupart des situations. Cette capacité de prédiction est rendu plus simple grâce aux règles imposées par le code de la route qui limitent le nombre d'hypothèses ; elle repose aussi sur l'expérience du conducteur en matière d'évaluation et de réduction du risque. L'absence de cette capacité à comprendre naturellement une scène de trafic constitue peut-être, le principal défi qui freine le déploiement à grande échelle de véhicules véritablement autonomes sur les routes.

Dans cette thèse, nous abordons les problèmes de modélisation du comportement du conducteur, d'inférence sur le comportement des autres véhicules, et de la prise de décision pour la navigation sûre. En premier lieu, nous modélisons automatiquement le comportement d'un conducteur générique à partir de données de conduite démontrées, évitant ainsi le réglage manuel traditionnel des paramètres du modèle. Ce modèle codant les préférences d'un conducteur par rapport au réseau routier (par exemple, voie ou vitesse préférées) et aux autres usagers de la route (par exemple, distance préférée au véhicule de devant). Deuxièmement, nous décrivons une méthode qui utilise le modèle appris pour prédire la séquence des actions à long terme de tout conducteur dans une scène de trafic. Cette méthode de prédiction suppose que tous les acteurs du trafic se comportent de manière aversive au risque, et donc ne peut pas prévoir les manœuvres dangereux ou les accidents. Pour pouvoir traiter de tels cas, nous proposons un modèle probabiliste plus sophistiqué, qui estime l'état et les intentions du trafic environnant en combinant la prédiction basée sur le modèle avec les preuves dynamiques fournies par les capteurs. Le modèle proposé imite ainsi en quelque sorte le processus de raisonnement des

humains. Nous humains, savons ce qu'un véhicule est susceptible de faire compte tenu de la situation (ceci est donné par le modèle), mais nous surveillerons sa dynamique pour en détecter les écarts par rapport au comportement attendu. En pratique, la combinaison de ces deux sources d'informations se traduit par une robustesse accrue des estimations de l'intention par rapport aux approches reposant uniquement sur des preuves dynamiques. En dernière partie, les deux modèles présentés (comportemental et prédictif) sont intégrés dans le cadre d´une approche décisionnel probabiliste. Les méthodes proposées se sont vues évalués avec des données réelles collectées avec un véhicule instrumenté, attestant de leur efficacité dans le cadre de la conduite autonome sur autoroute. Bien que centré sur les autoroutes, ce travail pourrait être adapté pour gérer des scénarios de trafic alternatifs.

Mots-clés : Conduite autonome, modélisation et apprentissage de comportements, prédiction de mouvement, prise de décision dans l'incertain, navigation en milieu humain

Acknowledgments

Completing a PhD is a long and very demanding journey, one that would be hard to navigate alone. I would like to thank my supervisors, Christian Laugier, Jilles Dibangoye and Emmanuel Mazer for their continuous support and their encouragement at the tougher moments of the journey. Thank you Christian for your positive attitude and your infectious energy and passion. Thank you Jilles for making yourself always available and for teaching me so much about how to do research. Thank you Emmanuel for your advices and honesty and for keeping me on track.

Although not present in the official supervisors list, Dizan Vasquez and Víctor Romero-Cano also had a tremendous impact on my work. Thank you Dizan for your guidance during the first year (and for preparing my first-ever guacamole!). Thank you Víctor for the many discussions on robotics, one of which led to a contribution of this thesis. Your glass-half-full mindset changed everything.

I extend a special thank you to Toyota Motor Europe for financially supporting this thesis. More particularly, I thank Nicolas Vignard and Gabriel Othmezouri for the numerous technical discussions and for their constructive feedback and insightful suggestions.

I would also like to thank the members of my jury for thoroughly reading the manuscript, their invaluable feedback and the genuine interest they showed on my work. Special thanks to Luis Merino and Anne Verroust-Blondet who traveled very long distances to attend my defense.

Despite my great supervisors I probably would have not reached the end of the journey without the support of my friends and colleagues from Inria Grenoble and from the CHROMA team. Thank you Jose for all the laughs and for the great *raclette* that never happened. Thank you Pavan for being there in the *long* gym sessions. Thanks to Mathieu, Anshul, Ozgur, Niranjan, Mario and many others for the fun times over lunch and coffee breaks, the hikes, trips and adventures. Thank you also to the team's engineers for their help with the experimental platform, especially to Jean-Alix and Jérôme.

Gracias infinitas a mi familia por su enorme apoyo y amor incondicional durante estos años. En especial, gracias a mi madre, Dulce, por todos los sacrificios que hizo para darme la oportunidad de estudiar.

Finally, I would like to thank Nadia, my partner, who helped me through the lows and celebrated with me the highs; you made it all worthwhile.

Financial acknowledgment The work presented in this thesis was supported by Toyota Motor Europe.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ADAS	Advanced Driver-Assistance System
aSLDS	Augmented Switching Linear Dynamical System
aSSSM	Augmented Switching State Space Model
AV	Autonomous Vehicle
CNN	Convolutional Neural Network
CV	Constant Velocity
CA	Constant Acceleration
DBN	Dynamic Bayesian Network
DNN	Deep Neural Network
DPW	Double Progressive Widening
EM	Expectation-Maximization
FSC	Finite State Controller
FSM	Finite State Machine
GP	Gaussian Process
HMM	Hidden Markov Model
IDM	Intelligent Driver Model
IMM	Interacting Multiple Model
IV	Intelligent Vehicle
IRL	Inverse Reinforcement Learning
JPD	Joint Probability Distribution
KFM	Kalman Filter Model
LDS	Linear Dynamical System
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MOBIL	Minimizing Overall Braking Induced by Lane Changes
MOMDP	Mixed Observability MDP
PDF	Probability Density Function
PMF	Probability Mass Function

POMDP Partially Observable Markov Decision Process

RL Reinforcement Learning

RNN Recurrent Neural Network

ROS Robot Operating System

sMDP Semi-Markov Decision Process

SSM State Space Model

SSSM Switching State Space Model

SUMO Simulation of Urban MObility

SVM Support Vector Machine

TH Time Headway

TTC Time To Collision

UCT Upper Confidence Tree

Mathematical Notation

$S = \{x_1, x_2, \ldots, x_n\}$	Finite set
$S = \{x_1, x_2, \ldots\}$	Infinite set
$\mathbb{R}$	Set of real numbers
X	Random variable
$P(X)$	Probability distribution of the random variable X
$X \sim P$	X is distributed according to P
$P(X = x_1) \equiv P(x_1)$	Probability of the realization x_1 of X
$P(X, Y)$	Joint probability distribution of X and Y
$P(X_{1:T})$	Joint probability distribution of the random variables $X_1, X_2, \ldots, X_T$
$P(X\|Y)$	Conditional probability distribution
$\mathbb{E}_{x\sim P(X)}[f(X)]$	Expected value of $f(X)$ under distribution $P(X)$
$\mathbb{E}[f(X)]$	Expected value of $f(X)$ under distribution deduced from context
Var$(f(X))$	Variance of $f(X)$ under $P(X)$
Cov$(f(X), g(X))$	Covariance of $f(X)$ and $g(X)$ under $P(X)$
$\mathbf{x}$	Column vector
$\mathbf{x}^T$	Row vector
$\boldsymbol{A}$	Matrix
$\boldsymbol{I}$	Identity matrix
c, β	Scalar
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\mathbf{x}$ with parameters $\boldsymbol{\mu}$ (mean) and $\boldsymbol{\Sigma}$ (covariance)
$T(n) = \mathcal{O}(f(n))$	The function T is bounded above by f (see appendix A)

In a (partially observable) Markov decision process:

$\mathcal{S}$	Set of possible states
$\mathcal{A}$	Set of possible actions
$\mathcal{T}$	Transition function
$\mathcal{R}$	Reward function
$\mathcal{C}$	Cost function
γ	Discount factor
$\mathcal{Z}$	Set of possible obervations
$\mathcal{O}$	Observation function
s, s'	States
a	Action
s_t	State at time step t
a_t	Action at time step t
h_t	History at time step t
$b_t(s)$	Belief of state s at time step t
π	Policy
$\pi(s)$	Action at state s under deterministic policy π
$\pi(a\|s)$	Probability of taking action a at state s under stochastic policy π
$V_\pi(s)$	Value function of policy π at state s
$V^*(s)$	Optimal value function at state s
$Q_\pi(s, a)$	State-action value function at state s given action a
$Q^*(s, a)$	Optimal state-action value function at state s given action a

1 Introduction

Recent advances in sensor technology and processing power have made the dream of fully automated vehicles technically feasible. This technology has the potential to radically transform the transportation industry and make roads much safer, taking human error out of the equation. However, before autonomous vehicles can be commercialized and deployed at large scale they should be able to safely share the road with human drivers. This constitutes a major challenge because human behavior is determined by a complex set of interdependent factors, which are very hard to model (e.g. intentions, perception, emotions). Still, realistic human behavior models are crucial to understand and predict the development of traffic situations. This thesis studies, for the particular case of highway scenarios, the interrelated problems of modeling human driver behavior, predicting the short- and long-term maneuvers of human drivers, and exploiting these predictions in order to make safe driving decisions.

This chapter starts by presenting the context around highly and fully automated vehicles. Next, in section 1.2, we formalize the problem addressed in this thesis and describe the associated challenges. To clarify the goals of this work, we discuss several illustrative examples of desired outcomes. Section 1.3 describes the proposed approach. In section 1.4, we enumerate the contributions of the thesis and, finally, present the outline of the document in section 1.5.

1.1 General context

Road safety is a major global issue. Each year, 1.25 million people are killed on the world's roads (World Health Organization, 2015). In 2009, more than 35,000 people died on the roads of the European Union alone, and no fewer than 1.5 million persons were injured. Numerous initiatives are in motion to reduce the number of road accidents. For example, the resolution 64/255 of the United Nations General Assembly aims to halve the global number of deaths from road accidents by 2020 (United Nations Economic Commission for Europe, 2015). Pushed both by government regulations and by customer demand, automakers have been consistently introducing technological innovations to improve the security of their vehicles: in 1978, the Anti-lock Braking System (ABS) was introduced onto the market; in 1995, the Electronic Stability Program (ESP); and in 1998 the Adaptive Cruise Control (ACC). More recent developments focus on getting rid of the human error, which accounts for an estimated 94% of all traffic accidents (National Highway Traffic Safety Administration, 2015). Safety systems such as the *lane departure warning*, *driver drowsiness detection*, *blind spot monitoring*, or *automatic emergency braking* are available on many commercial vehicles nowadays (Verband der Automobilindustrie, 2015). These systems are typically known as ADAS, which stands for Advanced Driver-Assistance Systems. They make use of a wide range of sensors to monitor the driver and the environment, including cameras, ultrasound, radar, and lidar (Light Detection And Ranging). Figure 1.1 shows two examples of highly instrumented vehicles, including our own experimental platform.

(a) Inria-TME Lexus development platform.

(b) Boss, from the Tartan Racing Team, winner of the 2007 DARPA Urban Challenge [Image credit: (Urmson et al., 2009)].

Fig. 1.1.: Two examples of instrumented vehicles.

Some ADAS systems can take partial control of the vehicle temporarily. For instance, the automatic emergency braking system monitors the road ahead of the vehicle and can apply an emergency brake to avoid or lessen the consequences of an impending crash (Grover et al., 2008). A 5-level classification of automated vehicles depending on their degree of autonomy has been established (SAE International, 2016), and it is shown in Table 1.1.

In automated vehicles of level 0, the driver has full control. The onboard systems focus on warning the driver of potentially dangerous situations. Among these systems we find the *lane departure warning*, *blind spot monitoring*, and *driver drowsiness detection*. Level 1 automated functions can temporarily take over the longitudinal or lateral control of the vehicle. This can be done for comfort or security reasons. Examples include the *adaptive cruise control*, *lane keeping assistant*, and *automatic emergency braking*. In level 2 automated vehicles, the system can already assume full control of the vehicle in particular use cases. However, the driver is required to monitor the vehicle at all times and be ready to take back control. Examples of level 2 include *parking assistance systems* and *highway driving assistants*. The Autopilot technology commercialized by Tesla is a level 2 automated function (*Tesla Autopilot*). Level 2 systems are considered dangerous in the sense that human drivers can become complacent and fail at their monitoring task (Naser et al., 2017). A proposed alternative is that of parallel autonomy, where the human is still in control of the vehicle, but a full autonomy system is continuously running in the background (Schwarting et al., 2017; Naser et al., 2017). The system can then decide to take over when potentially dangerous errors are detected in the driving of the human. Automated vehicles of level 3 can drive fully autonomously without continuous monitoring of the human driver, who is only required to be able to resume control if requested. In this level we find the *traffic jam/queue autonomous driving systems* and the *highway chauffeur system*. Finally, in automated levels 4 and 5 no intervention is required from the human driver during the complete journey. Level 4 describes systems capable of full autonomy in specific uses cases (such as driving in a particular city or neighborhood, or under favorable weather conditions), whereas level 5 represents unconditional full autonomy. Companies such as Waymo, Tesla and GM Cruise, among many others, are currently developing level 4 automated vehicles and testing them on public roads (Waymo, 2017; Tesla, 2016; GM Cruise, 2017). The performance

Level of Autonomy	Name	Characteristics
0	No automation	The driver has full control.
1	Driver Assistance	The system can assume either longitudinal *or* lateral control of the vehicle.
2	Partial Automation	The system can assume both longitudinal *and* lateral control of the vehicle in specific use cases. The driver must monitor the vehicle and be able to resume control at all times.
3	Conditional Automation	The system can assume full control of the vehicle in specific use cases. The driver is not required to monitor the vehicle, but should be able to retake control when requested by the system.
4	High Automation	The system can assume full control of the vehicle in specific use cases. No action is required from the driver.
5	Full Automation	The system has unconditional full control of the vehicle. No action is required from the driver.

Tab. 1.1.: Levels of driving automation (SAE International, 2016).

of these level 4 vehicles is evaluated based on the number of disengagements of the system, that is, the number of times that the autonomous navigation mode needs to be deactivated, either automatically or by the test driver. As an example, Waymo reports a total of 63 disengagements in 352,544.6 miles driven on public roads of the state of California (USA) for the period December 2016 - November 2017, which gives an idea of the maturity of the technology (Waymo, 2017).

The benefits of achieving a reliable, fully automated driving technology go beyond road safety. The potential societal benefits of fully autonomous vehicles include as well improved traffic flows, increased mobility for the elderly and the disabled, optimized human productivity, and economy-wide positive effects. However, despite the impressive disengagement figures reported by the companies developing fully automated vehicles, the technology does not seem ready yet for mass deployment, as evidenced by a long trail of incidents in recent years involving fully automated vehicles (California Department of Motor Vehicles, 2016; National Transportation Safety Board, 2017; National Transportation Safety Board, 2018). We can identify two main challenges:

1. **The need for reliable and robust perception systems**. Automated vehicles are equipped with a wide range of technologically advanced sensors that complement each other, i.e. some sensors might be more reliable than others under low-light conditions, others better in rainy or sunny weather, etc. Intelligent sensor fusion approaches that can produce accurate observations of the world regardless of the environmental conditions are crucial for the success of the technology. A measure of confidence is likewise needed for the observations, so that the automated driving system can decide upon more cautious maneuvers or even bring the car to a stop if the perception of the surroundings is not certain.
2. **Sharing the road with humans, be it pedestrians or drivers**. Predicting human behavior is hard for machines, as they only have access to raw sensor data. The state and intentions of other road users need to be estimated. For that, the automated system needs first to understand the traffic scene, which can be arbitrarily complex, and then use the

existing evidence to predict the intentions of the humans. Furthermore, the automated vehicle also needs to plan its own motion by taking into account the uncertainty about the predictions, just like human drivers do. This includes as well the uncertainty from what it cannot perceive, such as pedestrians behind a stopped bus, or occluded vehicles in an intersection.

This thesis addresses the second challenge for the particular case of highway scenarios.

1.2 Problem description

We consider an automated vehicle navigating a highway scene. The goal of this thesis is to develop the systems necessary to:

1. Predict with high accuracy the short-term behavior of surrounding vehicles, so as to pass this information to a collision avoidance or crash warning system. In the highway, a short-term prediction can be understood as a prediction with an horizon of up to 3-5 seconds.
2. Predict the long-term development of highway traffic scenes, so as to use this information to plan the future maneuvers of the ego-vehicle. A long-term prediction can be understood as a prediction reaching up to 10-15 seconds into the future.
3. Exploit the aforementioned short- and long-term predictions to determine the most convenient sequence of maneuvers for the ego-vehicle up to the long-term future. A metric that defines the convenience of a sequence of maneuvers needs to be specified.

In practice, predicting the behavior of a dynamic obstacle means that, for any particular future time step, the system provides an estimation of the state of the obstacle and of the maneuver it is executing. In this work, a particular focus will be given to the prediction of lane change maneuvers, which constitute 9% of all police-reported motor vehicle crashes (National Highway Traffic Safety Administration, 2013). An important characteristic of lane change prediction systems for automated vehicle applications is the false positive rate (also known as false alarm rate). If the lane change predictions are used within a *crash warning system* or even an *automatic emergency braking system*, a reduced number of false alerts is essential for efficacy and user acceptance (Bliss and Acton, 2003). Additionally, the driving datasets for the evaluation of lane change prediction systems are typically skewed, containing many more examples of lane keeping behavior than actual lane changes (U.S. Department of Transportation, 2006). In these cases, metrics like the true positive rate (also known as recall or sensitivity) and the false positive rate become extremely relevant (Nilsson et al., 2015). Hence, one of the goals of this thesis is to obtain a lane change prediction system with a high recall and a low false positive rate.

To plan a sequence of maneuvers for the ego-vehicle, also known as tactical or behavior planning, the short- and long-term predictions of behavior for the surrounding traffic are required. However, predicting and planning are two problems that are tightly coupled. Each future sequence of maneuvers of the ego-vehicle has an associated prediction of behavior for the surrounding vehicles. The proposed system should take into account this interaction between prediction and planning in order to find a suitable long-term plan for the ego-vehicle. A metric

to evaluate the suitability of a given plan will also need to be defined. In general, as discussed by Ulbrich and Maurer, the decision-making system of a fully automated vehicle navigating between humans should have the following characteristics (Ulbrich and Maurer, 2013):

- Rapidity: the update frequency of the ego-vehicle tactical decisions should be fast enough. For highway environments, it is reasonable to expect new plans every 3 to 5 seconds (assuming that emergency maneuvers and lateral/longitudinal controls are taken care of by a different system).
- Coherency: the decision-making system should be consistent, avoiding unnecessary changes in the plan unless the environment dynamics justify it.
- Providentness: the future consequences of following a given plan should be considered during planning. Adequate prediction models are a necessary requirement for this.
- Predictability: the maneuvers performed by the ego-vehicle are required to be predictable for the human drivers. In other words, the automated vehicle should drive in a human-like manner.

The proposed decision-making system should have all the characteristics above. Except for rapidity, all the desired characteristics directly depend on finding an appropriate metric to discriminate between different alternative plans.

To clarify the scope of the thesis, the inputs and outputs of the proposed systems are specified in the following subsection. Next, in subsection 1.2.2, we discuss some example target scenarios and their desired outcomes.

1.2.1 Inputs and outputs

The block diagram in Figure 1.2 shows a simplified version of the architecture of an autonomous vehicle. The software system of the vehicle, named here *Automated Vehicle Processing Unit*, interacts with the world through a variety of sensors and outputs the control commands and/or the information to be transmitted to the human driver.

The Automated Vehicle Processing Unit is divided into several layers, namely, perception, strategic, tactical, and operational layers. The *perception layer* processes the data from the sensors to produce higher-level abstractions, such as the state of the surrounding vehicles, the position of the lane markings relative to the ego-vehicle or the current location within a road map. The *strategic layer* takes care of finding the optimal route from the current location to the goal destination. The *tactical layer* constitutes the focus of this thesis. This layer takes as inputs the route provided by the strategic layer and the abstractions from the perception layer, and performs the scene reasoning and the maneuver planning for the ego-vehicle. To guide the behavior of the ego-vehicle and to reason about the behavior of others, this layer has access to a database of driver models. The resulting high-level plan containing a sequence of maneuvers is then passed to the *operational layer*, which takes care of translating the maneuvers into trajectories and control commands for the vehicle.

1.2.2 Examples of target scenarios

To further identify the desired outcomes of this work, several target scenarios are discussed here and illustrated in Figure 1.3. In the first place, Figure 1.3a shows a highway traffic scene

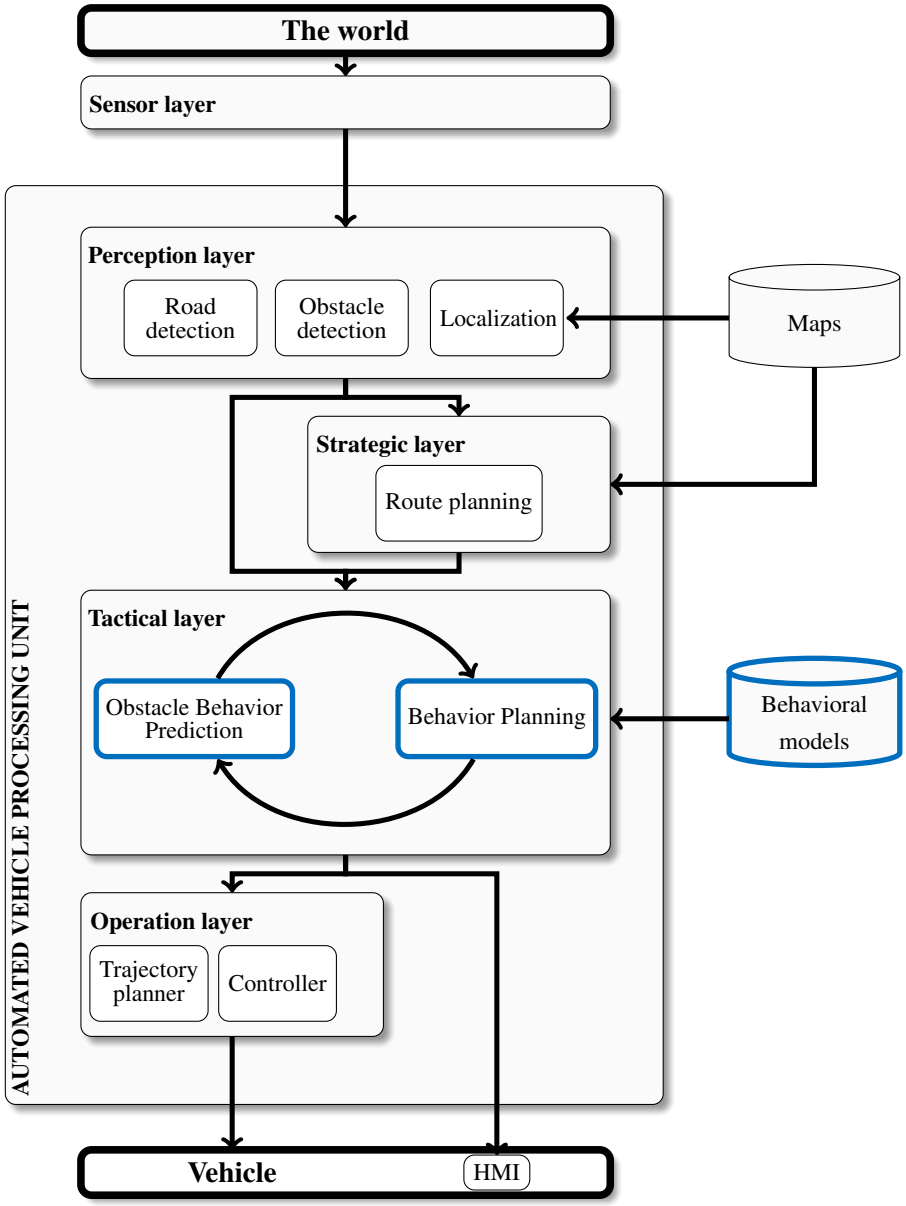


Fig. 1.2.: Simplified block-diagram of the architecture of a fully or highly automated vehicle. The blocks with thicker blue edges are the focus of this thesis.

in which a vehicle is performing a lane change to overtake. In the instants leading up to the moment shown in the figure, the target vehicle begins to approach the central lane marking and to head towards the neighboring lane. This dynamic information seems to indicate a potential lane change and, in principle, should be sufficient to perform short-term predictions of lane changes. However, human drivers in the highway do not drive always exactly in the center of their lanes; instead, they only follow the center of the lanes approximately and can, at times, approach the lane boundaries without the intention of traversing them. For this reason, lane change prediction approaches based only on this kind of dynamic evidence are prone to false alarms, as shown in Figure 1.3b. In contrast, human drivers do not fail often at predicting lane changes. This can be explained by their capability to understand traffic scenes and to interpret the motion cues of other drivers. Understanding the traffic scene shown in Figure 1.3a requires:

1) Knowing that drivers in highways typically prefer to drive in the right-most lane and that they perform lane changes to overtake slower vehicles; and 2) Considering the interaction between the target and its leading vehicle. In this scene, the target is approaching the leading vehicle at a high relative speed, while the left lane is clear. This intuitively points towards a likely lane change, which is confirmed by the observations of the dynamics of the target. A goal for this thesis will be to combine high-level scene understanding with low-level dynamic evidence for the short-term prediction of lane change maneuvers.

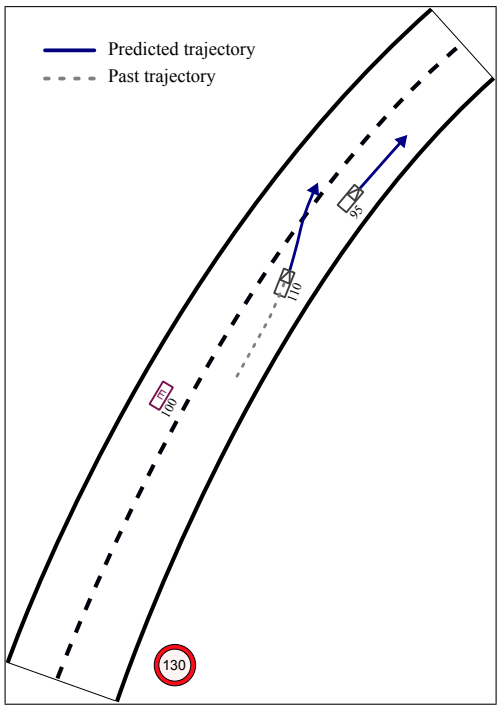
Understanding traffic situations and how human drivers interact with each other can also be exploited for the long-term prediction of traffic scenes. This idea is illustrated in Figure 1.3c, where it is reasonable to assume that the vehicle in the right-most lane will facilitate, potentially with a lane change to the left-most lane, the entrance to the vehicle merging from the entry ramp. Being able to predict the long-term behavior of other drivers opens the door to the long-term planning of maneuvers for the automated vehicle, as shown in Figure 1.3d. In this scene, the ego-vehicle is approaching a very slow vehicle on the right lane. The left lane is free and with sufficient gap for a lane change, but some vehicles are quickly approaching from behind. A decision-making system with sufficient foresight could decide to perform an early lane change, avoiding the possibility of being forced to decelerate and getting stuck behind the slow vehicle. Plans like this are enabled by the interaction-aware prediction models, which in this scenario would predict a deceleration for the trailing vehicles as a response to the lane change of the ego-vehicle.

1.2.3 Challenges and classic approaches

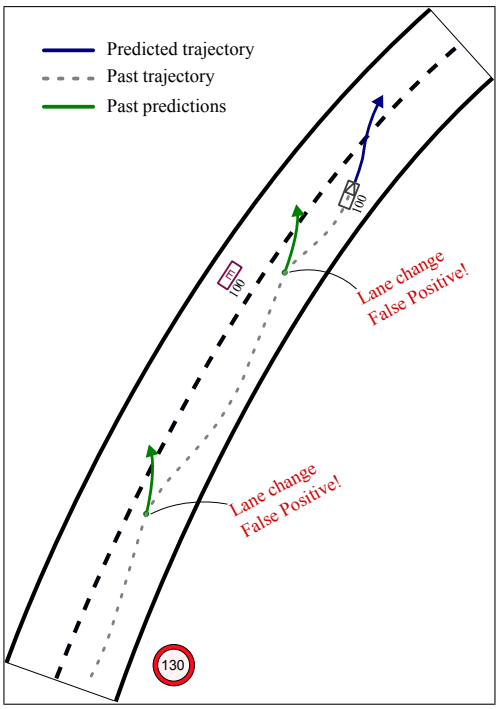
This thesis deals with three interconnected problems, namely, predicting the short-term behavior of drivers (short-term motion prediction), predicting the long-term development of traffic scenes (long-term motion prediction), and planning the long-term tactical behavior of the ego-vehicle (decision-making). Each of these problems comes with a number of associated challenges, some of them shared among them.

Noisy observations of the world. Sensor readings are noisy by nature. As a consequence, the high-level abstractions provided by the perception layer are also inaccurate. Ignoring this issue can lead, in the presence of outliers, to incorrect predictions and dangerous plans.
Existing prediction and planning approaches in the literature, especially those working with synthetic data, often assume perfect perception (Lawitzky et al., 2013; Schwarting and Pascheka, 2014; Hubmann et al., 2016).

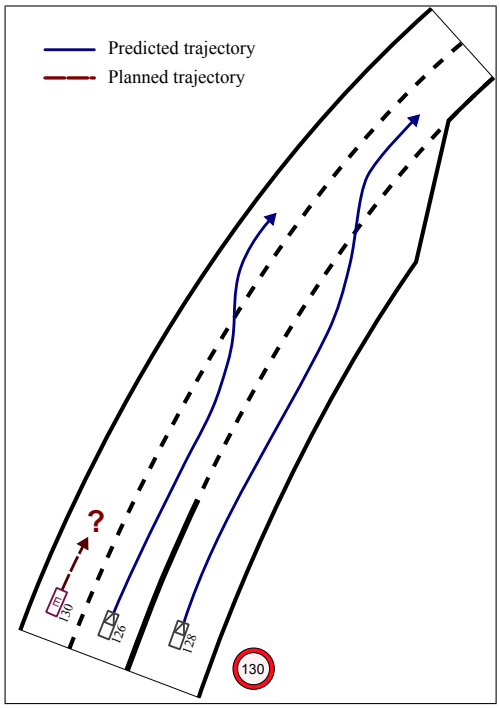
Understanding traffic scenes. As presented in the examples from Figure 1.3, the situational understanding of traffic scenes constitutes a key component for both the short- and long-term predictions of driver behavior.
Traditional prediction approaches often rely exclusively on the observed dynamics of the targets, completely ignoring the structure of the traffic scene and the interactions between vehicles (Kumar et al., 2013; Tay, 2009). More sophisticated approaches, in line with the approach proposed in this thesis, model the interactions between the traffic participants using dynamic Bayesian networks (Gindele et al., 2013; Agamennoni et al., 2012).



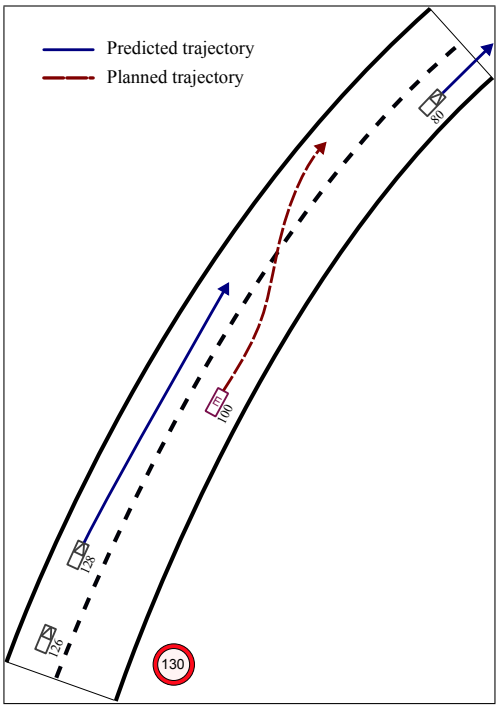
(a) Short-term prediction of behavior leveraging dynamic evidence and situation understanding. Both the dynamics and the relative distance and velocity of the target with respect to the leading vehicle indicate a potential lane change.



(b) Vehicle trajectories in the real world can produce a high number of false positives in prediction approaches that rely exclusively on dynamic evidence.



(c) Predicting the long-term development of highway traffic scenes requires a degree of situation understanding and an appropriate model of the interactions between traffic participants.



(d) The long-term planning of maneuvers for the EV is enabled by the ability to predict the long-term consequences of each plan.

Fig. 1.3.: Motivating scenarios for this work. These include short-term predictions of behavior (a, b), as well as long-term prediction (c) and planning (d).

Minimizing the number of false positive predictions A low false positive rate is a desired characteristic for a lane change prediction system. Prediction approaches based on simple kinematic models (Lefèvre et al., 2014), or on identifying the maneuver of the target from the dynamics alone (Kumar et al., 2013; Tay, 2009) are prone to false positives due to the noise in the observations and to the irregular driving style of human drivers.

Making long-term plans, under uncertainty and with real-time constraints. Just like human drivers, the automated driving system needs to take decisions without being absolutely sure of the intentions of surrounding traffic. The prediction models will provide stochastic behavior predictions and it is up to the decision-making system to determine the best sequence of maneuvers for the ego-vehicle. A scoring function to evaluate the different alternative plans is therefore also necessary. The planning needs to be done online and, ideally, with a sufficiently high replanning frequency. The resulting plans should be human-like, meaning that they should be easily interpretable by the surrounding human drivers.
The first approaches for decision-making in the domain of Intelligent Vehicles (IV) were based on hierarchical finite state machines, where a set of rules tuned with expert knowledge determined the transitions between states (Montemerlo et al., 2009; Urmson et al., 2009; Ziegler et al., 2014a; Ziegler et al., 2014b). However, as the complexity of the traffic situations increases the robustness and transparency of these approaches decline. Furthermore, they lack a systematic methodology to handle uncertainty. Existing approaches tackle this last issue by modeling the problem as a partially observable Markov decision process, a principled mathematical framework for modeling sequential decision problems under uncertainty. However, the computational complexity of these approaches is high, so they typically simplify the problem by neglecting uncertainties (e.g. the uncertainty about the intentions of surrounding traffic) (Ulbrich and Maurer, 2013). Additionally, the reward function of the partially observable Markov decision process framework, which determines in great part the final behavior of the ego-vehicle, is commonly hand-tuned and overly simple (e.g. containing a term to reward reaching the goal, and another term to penalize collisions) (Bouton et al., 2017).

1.3 Proposed approach

The proposed approach for prediction and decision-making in highway scenarios is composed of different interconnected steps, which are described below:

Modeling the behavior of drivers A good way to understand traffic situations is through the use of appropriate models of human behavior. While these models often appear in the intelligent vehicles literature (for example, to guide the search of a trajectory motion planner), they are typically hand-tuned, with no systematic approach to obtain them. This hand-tuning task becomes increasingly difficult with the complexity of the desired model. Furthermore, multiple models are necessary to address different kind of scenarios, e.g. models for the highway, urban roads, residential roads, parking lots..., making it a repetitive task. This highlights the need for a systematic and mathematically consistent approach to obtain the models. A convenient avenue to solve this problem is to learn the models automatically from human-demonstrated driving data using Inverse Reinforcement Learning (Russell, 1998; Ng

and Russell, 2000). Approaches that learn simple driver models in synthetic domains exist in the literature (Abbeel and Ng, 2004; Levine et al., 2011). However, these approaches do not scale well to problems with continuous state spaces and realistic dynamic environments. In this thesis, we propose an approximate algorithm that combines Inverse Reinforcement Learning with an spatiotemporal lattice-based trajectory planner in order to learn driver behavioral models from real-world data. The resulting driver model is essentially a cost function that maps states to cost. The model captures the driving preferences of the human that demonstrated the task, and balances multiple elements, some of them risk-related (e.g. the time-to-collision to the vehicle in front) and some of them related to the driving style (e.g. lane or speed preference).

Predicting short- and long-term driver behavior The human-like model learned from demonstrated driving data can be used in the prediction of driver behavior. For the **long-term prediction** of highway scenes, we propose an algorithm in which it is assumed that each driver will follow the behavior encoded in the model. This is not different from the way humans predict the future. If a driver is asked to predict the long-term development (e.g. 10 seconds into the future) of a given traffic scene, he or she will probably take into account: 1) The interactions between vehicles (vehicles do not jump over each other, and a safe distance is usually respected); and 2) That drivers prefer to maintain their speed, changing lanes if necessary. This is exactly the kind of behavior encoded in the driver model. In practice, the algorithm iterates over the vehicles in the scene, from the front to the back. For each of them, the probability of a given maneuver is predicted as a function of its associated cost. By means of the risk-related components of the model, the interactions between traffic participants are explicitly considered.

The model-based prediction method described above assumes that all traffic participants behave in a risk-aware manner and can therefore fail to predict dangerous maneuvers or accidents. This is particularly relevant for the **short-term predictions** of behavior. To be able to handle such cases, it is necessary to consider the dynamics of the targets. This thesis proposes a probabilistic model that estimates the state and intentions of surrounding traffic by combining the model-based prediction with the noisy dynamic evidence provided by the perception layer. The model-based prediction represents the logical expected behavior of the target and brings therefore a degree of scene understanding into the predictions. This translates into an improved prediction robustness with respect to approaches relying on dynamics alone. In a way, this model mimics the reasoning process of human drivers: they can guess what a given vehicle is likely to do given the situation (the model-based prediction), but they closely monitor its dynamics to detect deviations from the expected behavior (the observations from the perception layer).

The proposed approach to combine both sources of information builds upon the Augmented Switching State Space Model (Lerner, 2012; Barber, 2012), which has already been applied in the IV domain (Agamennoni et al., 2012). Our approach modifies the work from Agamennoni et al. in order to: 1) Handle highway scenarios; 2) Use a more complex behavioral model, learned offline using Inverse Reinforcement Learning; and 3) Consider the long-term consequences of any action in the intention estimation step.

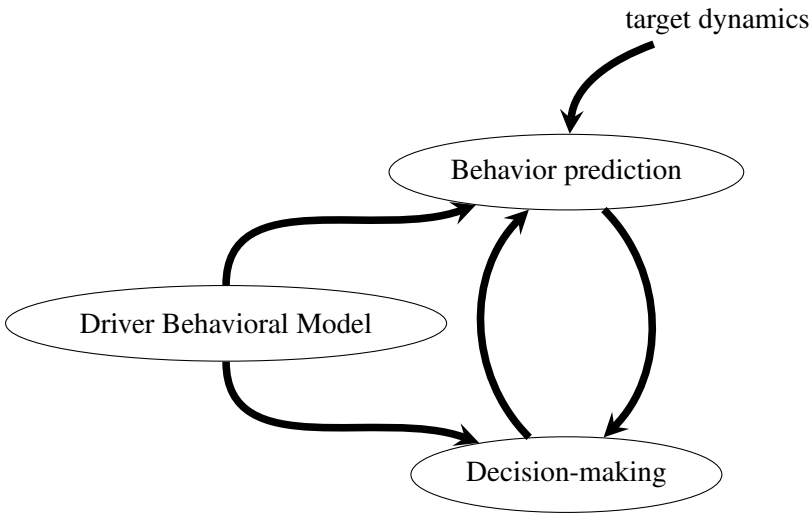


Fig. 1.4.: Synergies between driver model, behavior prediction, and decision-making.

Planning under uncertainty Finally, the learned driver behavioral model and the short- and long-term prediction models are integrated within a probabilistic decision-making framework that accounts for the uncertainty in the states and behaviors of the traffic participants. This method modifies the POMCP planning approach in order to use parametric belief representations (Silver and Veness, 2010). The resulting online algorithm is divided into a planning stage and an execution stage, which are applied alternately at each time step. In the planning stage, the algorithm starts from the current state and intention estimates of all traffic participants. From there, it explores several possible sequences of actions for the ego-vehicle, and scores each of them based on their associated scene predictions (obtained with the proposed prediction models), and on the human-like driver behavioral model. Once the planning stage ends, the best maneuver found is executed and the tree and estimates are updated.

The synergies between the three different subproblems are shown in Figure 1.4. The driver model can be exploited, together with the dynamic evidence provided by the perception layer, to improve the quality of the short-term predictions. Furthermore, it constitutes the key component to predict the long-term development of traffic scenes. The same behavioral model can be used to guide the tactical planning of the ego-vehicle. During the search and evaluation of different alternative plans, the interaction-aware predictions can change as a consequence of the actions of the ego-vehicle.

1.3.1 Applications

The technology proposed in this thesis can be applied as an ADAS or as part of the navigation stack of a fully automated vehicle (Figure 1.2). Regarding the ADAS applications, the behavior prediction approaches proposed in this thesis could be directly integrated as part of: a) A collision warning system; or b) An adaptive cruise control system. The early prediction of cut-in lane change maneuvers can lead to earlier collision detections (for the collision warning system) and smoother acceleration controls (for the adaptive cruise control system). Additionally, the

proposed behavior modeling approach is suitable for learning driver preferences from real-world driving data. This opens the door to the learning of customized behavioral models, which can increase the level of security and comfort experienced by the occupants of a fully automated vehicle (Shariff et al., 2017).

1.4 Contributions

The main contributions of this thesis are the following:

- **An approximate algorithm to learn driver behavioral models from real-world demonstrated driving data**. The proposed algorithm combines Inverse Reinforcement Learning with an spatiotemporal lattice-based trajectory planner in order to handle continuous state spaces and dynamic environments. In practice, the driver behavioral model is a cost function that maps states to costs. The function is assumed linear on a combination of static and dynamic features. The static features capture the preferences of the driver while the dynamic features, which change over time depending on the actions of the other traffic participants, capture the driver's risk-aversive behavior.
- **A model-based approach for the long-term prediction of highway traffic scenes**. Each traffic participant is modeled as a Markov Decision Process with a cost function learned from demonstrated driving data. The interactions between drivers are explicitly considered through the dynamic features of the driver model. The computational complexity of the approach grows quadratically with the number of vehicles in the scene.
- **A probabilistic model to estimate the state and intentions of traffic participants**. The proposed model integrates two different sources of information to produce accurate human-like estimations. The first source of information is a model-based prediction, based on the behavioral model learned from demonstrations, that identifies the most likely, risk-aversive, anticipatory maneuver for each driver in the scene. The second source of information is the noisy observation of the dynamics of the targets obtained from the perception layer. Integrating the anticipatory model-based prediction into the intention inference engine brings a degree of scene understanding into the estimates, and leads to faster and more robust detections of lane change maneuvers compared to approaches relying on dynamics alone.
- **An online, human-like, decision-making approach for automated vehicles in highway scenarios**. The proposed algorithm integrates the behavioral model learned from demonstrated driving data to guide the search for the optimal sequence of maneuvers. Since the model is learned from human driving demonstrations, the resulting behavior of the host vehicle resembles that of human drivers, meeting thus the predictability requirement. Furthermore, the planning approach also integrates the short- and long-term predictive models to determine the consequences of alternative sequences of actions, therefore fulfilling the providentness requirement. In practice, the proposed model is an online partially observable Markov Decision Process based on Monte-Carlo tree search. It builds upon the POMCP algorithm but, instead of using particles, it maintains the beliefs using the proposed short-term predictive model. To keep the approach tractable,

the long-term consequences of any plan are estimated using a lightweight model-based prediction.

1.5 Thesis outline

Part One: Background and State of the Art

Chapter 2: Background. This chapter presents the basic mathematical background on probability theory, state-space models and sequential decision-making in uncertain, dynamic environments.

Chapter 3: State of the Art. This chapter contains a review of the existing literature on driver behavior modeling, motion prediction and tactical decision making for intelligent vehicles.

Part Two: Proposed Models and Algorithms

Chapter 4: Modeling Driver Behavior From Demonstrations. In this chapter, an approximate algorithm to model the behavior of highway drivers from driving demonstrations is presented. The proposed method is validated using a custom dataset of demonstrated suboptimal highway driving data gathered with an instrumented vehicle.

Chapter 5: Model-Based Driver Behavior Prediction. Once an appropriate model of driver behavior is available, it can be used to predict the development of traffic scenes. This chapter proposes an algorithm for the prediction of highway traffic scenes that takes into account the interactions between traffic participants through the use of a feature-based driver model.

Chapter 6: Human-Like Driver Behavior Estimation. Exclusively relying on a driver model to predict the behavior of surrounding traffic might fail to predict dangerous maneuvers. In order to predict behaviors that the model cannot explain, the dynamics of the targets need to be considered. In this chapter, a probabilistic filtering framework to estimate the state and intentions of surrounding traffic is presented. The proposed framework combines dynamics-based estimates with an interaction-aware, model-based prediction. The framework is applied to the detection of lane-change maneuvers in highway environments and validated on a custom dataset from real highway data.

Chapter 7: Towards Human-Like Tactical Planning. This chapter presents a tactical decision-making approach that integrates all the previously discussed elements. The realistic driver model learned from demonstrations is used to guide the search for the best anticipatory maneuver of the ego-vehicle. The driver behavior estimation framework is used to determine the consequences of any maneuver in the short-term future. A variation of the model-based prediction approach provides guidance about the possible consequences in the long-term future. The approach is validated using a highway driving simulator.

Part Three: Conclusions

Chapter 8: Conclusions and Future Work. Finally, in this chapter we present the conclusions of the thesis, summarize its main contributions, and discuss some potential avenues for future work.

Part I

Background and State of the Art

Background

2

2.1 Introduction

Making an autonomous vehicle navigate environments shared with humans involves the following challenges:

- Firstly, human behavior is extremely difficult to model, as it is determined by a wide range of factors such as intentions, perception and emotions. Without a perfect model, any prediction of future behavior is inherently uncertain.
- Secondly, the sensors of the vehicle only provide noisy observations of the surrounding environment. The software of the autonomous vehicle is required to reason about the true state of the world based on the noisy observations.
- And thirdly, given the uncertain present and future states of the world, the vehicle needs to select appropriate actions in order to make progress towards its goal in a safe way.

In this chapter, we present the basic theoretical concepts that will let us formalize and address the above-mentioned problems in the following chapters. The uncertainty about the future can be represented and reasoned about in a consistent way using probability theory. The state of the world can be recursively estimated from noisy sensor observations using probabilistic models such as the Bayes filter. Then, we can reason about what action to take in uncertain environments using a mathematical framework known as Markov Decision Processes. Finally, at the end of the chapter, some common terminology in the context of motion prediction and motion planning for intelligent vehicles is presented.

This chapter is highly based on materials from (Sutton and Barto, 1998; Kaelbling et al., 1998; Bertsekas and Tsitsiklis, 2002; Thrun et al., 2005; Bishop, 2006; Koller and Friedman, 2009; Barber, 2012; Prince, 2012; Goodfellow et al., 2016). The reader is referred to consult these references for further details on the specific topics.

2.2 Probabilistic models

This section covers the background in probability theory, probabilistic models and probabilistic inference that underlie the methods presented throughout this document.

2.2.1 Basic probability theory

As introduced in this chapter, probability theory provides a mathematical framework to reason in the presence of uncertainty.

From sets to random variables

A set S is a collection of elements. It can contain a finite number of elements $S = \{x_1, x_2, \ldots, x_n\}$, or an infinite number of elements $S = \{x_1, x_2, \ldots\}$. The complement of a set S with respect

to the universe Ω (that contains all possible elements) is the set of all elements not present in S, that is, $S^C = \{x \in \Omega | x \notin S\}$. The union of two sets S_1 and S_2 is the set of all their elements, $S_1 \cup S_2 = \{x | x \in S_1 \text{ or } x \in S_2\}$. The intersection of two sets S_1 and S_2 is the set of all elements contained in both of them, $S_1 \cap S_2 = \{x | x \in S_1 \text{ and } x \in S_2\}$.

The sample space Ω of an experiment is the set of all its possible outcomes. Any subset of the sample space is called an event. A typical example of an experiment is rolling a 6-sided dice. The sample space for this experiment is $\Omega = \{1, 2, 3, 4, 5, 6\}$. Example of events could be $E_1 = \{1\}$ or $E_2 = \{1, 2, 3\}$.

A random variable is a function of the outcome of an experiment, i.e., it associates a value to each element in the sample space. For the experiment of rolling two 6-sided dices, an example of random variable could be $X = \{\textit{sum of the two rolls}\}$. We represent random variables with uppercase letters and the values that they can take with lowercase letters. The domain of a random variable $dom(X) = \{x_1, x_2, \ldots, x_n\}$ is the set of values that it can take. Depending on their domain, random variables can be classified as discrete or continuous.

Probability distributions

A probability distribution of a random variable indicates the likelihood of each of the values in its domain. For example, $P(X = x_1)$ indicates the probability that the random variable X takes on the value x_1. The notation can be abbreviated as $P(x_1)$. Oftentimes, the notation $P(X)$ is used to refer to the complete probability distribution. It is also possible to define a probability distribution over multiple random variables. Let us consider the variables X and Y. Then, $P(X, Y)$ represents their Joint Probability Distribution (JPD). An example instantiation of this JPD could be $P(x_1, y_1)$, which represents the probability that $X = x_1$ and $Y = y_1$.

Discrete probability distributions A discrete probability distribution is known as a Probability Mass Function (PMF). A PMF P over the random variable X satisfies the following properties (Goodfellow et al., 2016):

- The domain of P must be equal to the domain of X
- $0 \leq P(x) \leq 1, \ \forall x \in X$
- $\sum_{x \in X} P(x) = 1$

Continuous probability distributions A continuous probability distribution is known as a Probability Density Function (PDF). A PDF P over the random variable X satisfies the following properties (Goodfellow et al., 2016):

- The domain of P must be equal to the domain of X
- $P(x) \geq 0, \ \forall x \in X$
- $\int P(x)dx = 1$

The probability that X takes on a value on the interval $[x_1, x_2]$ is given by $\int_{x_1}^{x_2} P(x)dx$.

Useful probability properties

We list here a number of important and useful probability properties. For continuous variables it will suffice to substitute the summations for integrals to obtain the corresponding expressions.

Marginalization Given the JPD $P(X, Y)$ of two random variables X and Y, we can find the marginal probability $P(X)$ as:

$$P(X) = \sum_{y \in Y} P(X, y) \tag{2.1}$$

Conditional probability The conditional probability $P(X|y)$ denotes a probability distribution over X, given an observation of $Y = y$. The more general notation $P(X|Y)$ indicates a set of probability distributions (one for each possible value of Y). The conditional probability can be expressed as:

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \tag{2.2}$$

If $P(y) = 0$, the conditional $P(X|y)$ is not defined.

Independence Two random variables X and Y are independent if and only if:

$$P(X, Y) = P(X)P(Y) \tag{2.3}$$

If X and Y are independent, we write $X \perp\!\!\!\perp Y$. Otherwise, they are dependent: $X \top\!\!\!\top Y$.

Conditional independence Two random variables X and Y are independent given Z, if and only if:

$$P(X, Y|Z) = P(X|Z)P(Y|Z) \tag{2.4}$$

If X and Y are conditionally independent given Z, we write $X \perp\!\!\!\perp Y|Z$. Otherwise, they are conditionally dependent given Z: $X \top\!\!\!\top Y|Z$.

The chain rule By rearranging the terms in equation 2.2, we can obtain the chain rule (also know as the product rule):

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{2.5}$$

This rule can be generalized to n random variables:

$$\begin{aligned} P(X_1, X_2, \ldots, X_n) &= P(X_n|X_{n-1}, \ldots, X_1)P(X_{n-1}|X_{n-2}, \ldots, X_1) \ldots P(X_2|X_1)P(X_1) \\ &= P(X_1) \prod_{i=2}^{n} P(X_i|X_1, \ldots, X_{i-1}) \end{aligned} \tag{2.6}$$

The factorization shown above is just one of the many admissible factorizations for the JPD. In practice, factorizations that enable further simplifications due to conditional independence assumptions are selected, as they lead to computational and storage savings.

Bayes' rule Through Bayes' theorem a relationship is established between $P(X|Y)$ and $P(Y|X)$. Following equations 2.2 and 2.5:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} = \frac{P(Y|X)P(X)}{\sum_{x \in X} P(Y|X)P(X)} \tag{2.7}$$

Each of the terms in this equation has a name. The term $P(X|Y)$ is known as the posterior, the term $P(X)$ is the prior, the term $P(Y)$ is the evidence, and the term $P(Y|X)$ is the data likelihood.

Expectation, variance and covariance

The expectation of a function $f(X)$ under distribution $P(X)$ is defined as:

$$\mathbb{E}_{x \sim P(X)}[f(X)] = \sum_{x \in X} P(x)f(x) \tag{2.8}$$

As usual, the continuous case can be obtained by switching the summation with an integral.

The variance of a function of a random variable gives a measure of how the values of the function deviate from its mean as we sample from its PMF/PDF. It is represented by σ^2 or $\text{Var}(f(X))$ and calculated as:

$$\text{Var}(f(X)) = \mathbb{E}\left[(f(X) - \mathbb{E}[f(X)])^2\right] = \mathbb{E}[f(X)^2] - \mathbb{E}[f(X)]^2 \tag{2.9}$$

For two functions $f(X)$ and $g(Y)$ of the random variables X and Y, the covariance provides a measure of how much their values are correlated with each other. It is calculated as:

$$\text{Cov}(f(X), g(Y)) = \mathbb{E}\left[(f(X) - \mathbb{E}[f(X)])(g(Y) - \mathbb{E}[g(Y)])\right] \tag{2.10}$$

2.2.2 Common probability distributions

We present here some common probability distributions that will be used in later chapters. These distributions are known as parametric forms because their behavior is characterized by a given set of parameters.

Bernoulli distribution

The Bernoulli distribution is a probability distribution over a single binary random variable $X \in \{0, 1\}$. It has one parameter $\lambda \in [0, 1]$, which defines the probability that the random variable is equal to 1:

$$\begin{aligned} P(X = 0; \lambda) &= 1 - \lambda \\ P(X = 1; \lambda) &= \lambda \\ P(X = x; \lambda) &= \lambda^x (1 - \lambda)^{1-x} \end{aligned} \tag{2.11}$$

Categorical distribution

The categorical distribution is a probability distribution over a single discrete random variable which can take on one of K mutually exclusive different values. It has as parameter a K-dimensional vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_K]^T$ that contains the probabilities for each of the outcomes, with $\lambda_k \in [0, 1]$ and $\sum_{k=1}^{K} \lambda_k = 1$. Let vector $\boldsymbol{x}$ be a 1-of-K representation of the discrete random variable X. Then:

$$P(\mathbf{x}; \boldsymbol{\lambda}) = \prod_{k=1}^{K} \lambda_k^{x_k} \tag{2.12}$$

Univariate Gaussian distribution

The univariate Gaussian distribution is defined over a continuous random variable $X \in (-\infty, +\infty)$. The parameters of the distribution are the mean μ and the variance σ^2. It is defined as:

$$P(X; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right) \tag{2.13}$$

The parameter μ controls the location of the peak of the distribution, whereas the parameter σ^2 controls its width.

Multivariate Gaussian distribution

The generalization of the univariate Gaussian distribution is the multivariate Gaussian distribution, which is defined over a D-dimensional vector $\mathbf{x} = [X_1, X_2, \ldots, X_D]^T$ of continuous random variables. The parameters of the distribution are the D-dimensional mean vector $\boldsymbol{\mu}$ and the $D \times D$ positive definite covariance matrix $\boldsymbol{\Sigma}$. The distribution is defined as:

$$P(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \tag{2.14}$$

The parameter $\boldsymbol{\mu}$ controls the location of the mean of the distribution (the center of the ellipsoid), whereas the parameter $\boldsymbol{\Sigma}$ controls its shape. The univariate and multivariate Gaussian distributions have some relevant properties that are described in detail in Appendix B.

Mixture of Gaussians

The mixture of Gaussians, also known as Gaussian mixture model, is a probability distribution that is described by a weighted sum of a discrete number K of Gaussian distributions:

$$P(X) = \sum_{k=1}^{K} P(C = k) P(X; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.15}$$

where $P(C)$ is a categorical distribution and $P(X; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ represents the k*th* Gaussian component. The parameters of the mixture of Gaussians are the mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$ of the

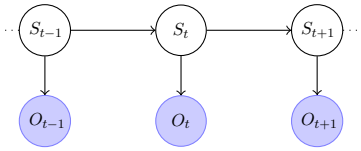


Fig. 2.1.: Graphical model representation of a state-space model. Shaded nodes indicate observed variables.

individual Gaussian components, and the vector of probabilities $\boldsymbol{\lambda}$ of the categorical distribution. A mixture of Gaussians is known as a universal approximator of densities, i.e., given a sufficient number of components with properly adjusted means, covariances and weights, almost any continuous density can be approximated to arbitrary accuracy (Bishop, 2006).

2.2.3 Probabilistic graphical models and sequential data

Probabilistic graphical models enable the visual representation of dependence/independence relations of probability distributions. Although they have limited expressibility (Barber, 2012), they constitute a useful tool to clarify modeling assumptions. In this thesis, we are particularly concerned with modeling sequential data, in which some *hidden* variable (the state of the world) evolves over time and generates noisy observations. Such models are often referred to as state-space models (SSM) (Murphy, 2002). A directed graphical model representing an exemplary state-space model is shown in Figure 2.1.

In this model, S_t and O_t represent the state and the observation at the discrete timestep t, respectively. The modeling assumptions encoded in the graphical model are $S_{t+1} \perp\!\!\!\perp S_{t-1} \,|S_t$ and $O_t \perp\!\!\!\perp S_{t'}, O_{t'} \,|S_t$ for $t \neq t'$. The former is known as the Markov property. The JPD of the model factorizes as follows:

$$P(S_{1:T}, O_{1:T}) = P(O_1|S_1)P(S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(O_t|S_t) \tag{2.16}$$

The model is defined by the state prior $P(S_1)$, the transition distribution $P(S_t|S_{t-1})$ and the emission distribution $P(O_t|S_t)$. In robotic applications such as tracking, the transition distribution is related to the dynamics of the target. Depending on the nature of the state and observation variables and of the transition and emission models, this model corresponds to some well-studied problems:

- If the state variable is discrete, the model is a Hidden Markov Model (HMM).
- If the state and observation are vectors of continuous random variables and the transition and emission models are linear Gaussian, we refer to this model as a Kalman Filter Model (KFM) (Kalman, 1960).

The most common inference problems with state-space models are:

Filtering Filtering is the problem of estimating the current state given all observations up to now, that is, the probability $P(S_t|O_{1:t})$. This probability is known as the belief of state S_t (Thrun et al., 2005).

Smoothing Smoothing refers to estimating the past: $P(S_t|O_{1:t'})$ with $t < t'$.

Prediction Predicting refers to estimating the future: $P(S_t|O_{1:t'})$ with $t > t'$.

Viterbi decoding Viterbi decoding consists in finding the most likely sequence of states given the observations: $\arg\max_{S_{1:t}} P(S_{1:t}|O_{1:t})$.

In chapter 6, filtering is performed to estimate the state and intentions of surrounding vehicles from noisy observations provided by an obstacle tracker. In the next subsection, we explain the general method to perform recursive Bayesian estimation.

2.2.4 Bayes filter

The Bayes filter is a general method for the recursive estimation of the hidden state of an SSM using incoming observations at each time step. Each time an observation O_t is received, the marginal posterior $P(S_t|O_{1:t})$ is computed. Starting off the state prior $P(S_1)$, we can compute the posterior $P(S_1|O_1)$ after the first observation O_1 is received using Bayes' rule:

$$P(S_1|O_1) = \frac{1}{Z}P(O_1|S_1)P(S_1) \tag{2.17}$$

where Z is a normalization factor that guarantees that the posterior is a proper probability distribution, and $P(O_1|S_1)$ is given by the emission model. Modifying the probability of a state after receiving new information is known as the *observation incorporation step*.

At the following time step, and after the second observation S_2 is received, we compute:

$$P(S_2|O_1, O_2) = \frac{1}{Z}P(O_2|S_2, O_1)P(S_2|O_1) = \frac{1}{Z}P(O_2|S_2)P(S_2|O_1) \tag{2.18}$$

The generalization to time step t is given by:

$$P(S_t|O_{1:t}) = \frac{1}{Z}P(O_t|S_t)P(S_t|O_{1:t-1}) \tag{2.19}$$

To obtain the term $P(S_t|O_{1:t-1})$ it is necessary to apply the transition model to the previous marginal posterior $P(S_{t-1}|O_{1:t-1})$. This is known as the *prediction step* and computed recursively with the *Chapman-Kolmogorov* relation:

$$P(S_t|O_{1:t-1}) = \sum_{S_{t-1}} P(S_t|S_{t-1})P(S_{t-1}|O_{1:t-1}) \tag{2.20}$$

The Bayes filter works by alternating the two explained steps. The prediction step propagates the state probability forward one time step by applying the Chapman-Kolmogorov relation, returning a state prior probability. Then, in the observation incorporation step, the information received is used to update the state estimate using equation 2.19. At each recursion step, only the previous posterior and the new observation are needed; no additional information is required.

The Kalman filters are similarly based on this two-step procedure. Inference in these models is based on the fact that the transition and emission models are linear functions, and that

all the probabilities involved are Gaussian distributions. This enables an efficient, closed-form procedure for state estimation. More details about the Kalman filter and the extended Kalman filter are provided in Appendix C.

2.3 Making decisions

In this section, we present two mathematical frameworks for sequential decision-making under uncertainty. Two main sources of uncertainty can be distinguished: the uncertainty due to non-deterministic decision effects, and the uncertainty in perception (we cannot observe the true state of the world) (Thrun et al., 2005). When the former is the only source of uncertainty (if any), the Markov Decision Process framework provides a viable option to formalize and solve a decision-making problem (Bellman, 1957). If in addition the problem involves perception uncertainty, then the partially observable Markov Decision Process framework can be applied (Åström, 1965).

2.3.1 Markov Decision Processes

In a Markov Decision Process (MDP) framework, an agent interacts with a given environment by taking actions at discrete time steps. Upon taking an action, the state of the world changes and the agent receives a reward signal. The state of the world represents the information available to the agent. The environment represents everything that cannot be changed arbitrarily by the agent (Sutton and Barto, 1998). In this setting, the goal of the agent is to take actions so as to maximize the long-term expected cumulative amount of reward it receives. Formally, a finite-state and action MDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, where:

- $\mathcal{S}$ is a finite set of n states of the world. The state at time t is denoted s_t, where t is a discrete time index.
- $\mathcal{A} = \{a_0, a_1, \dots a_k\}$ is a finite set of k actions.
- The state transition function is defined by $\mathcal{T}(s, a, s')$ such that:

$$\begin{aligned} \mathcal{T}(s, a, s') &= P(s_t = s' | s_0, a_0, s_1, a_1, \dots, s_{t-1} = s, a_{t-1} = a) \\ &= P(s_t = s' | s_{t-1} = s, a_{t-1} = a) \quad \forall t \end{aligned} \tag{2.21}$$

  where the Markov property has been applied (subsection 2.2.3).

- $\mathcal{R} : \mathcal{S} \mapsto \mathbb{R}$ is the reward function, which assigns to each state a numeric value representing the immediate reward. A representation $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ that maps each state-action pair to a reward is also commonly used.

A policy is defined as any map $\pi : \mathcal{S} \mapsto \mathcal{A}$[1]. Solving an MDP implies finding the policy π that maximizes the amount of reward accumulated in the long-term. The value associated to a

[1] In this review of MDPs and POMDPs we focus mainly on infinite-horizon problems, with deterministic and stationary policies.

given policy at any arbitrary state $s \in \mathcal{S}$ is given by the value function $V_\pi(s)$, which is equal to the expected sum of discounted future rewards starting from state s and following policy π:

$$V_\pi(s) \doteq \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, \pi(s), s') V_\pi(s'), \quad \forall s \in \mathcal{S} \tag{2.22}$$

where the discount factor $\gamma \in [0, 1)$ makes future rewards less valuable. The equation above is known as the Bellman equation. It defines a set of n linear equations, with n unknowns, and it has a unique solution (Bellman, 1954).

In a similar fashion, we can define the state-action value function $Q_\pi(s, a)$ as the expected discounted sum of future reward starting in state s, taking action a, and then following policy π:

$$Q_\pi(s, a) \doteq \mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V_\pi(s'), \quad \forall s \in \mathcal{S}, a \in \mathcal{A} \tag{2.23}$$

The optimal value function $V^*(s)$ is the maximum value function that can be achieved by any policy:

$$V^*(s) = \max_\pi V_\pi(s) \tag{2.24}$$

$$= \max_a \underbrace{\left[\mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V^*(s') \right]}_{Q^*(s,a) = \max_\pi Q_\pi(s,a)} \quad \forall s \in \mathcal{S} \tag{2.25}$$

which is known as the Bellman optimality equation and has a unique solution (Bellman, 1954). The term $Q^*(s, a)$ is the optimal state-action value function. Given the complete model of the environment, a solution can be found using dynamic programming approaches such as value iteration or policy iteration, or also linear programming (Bellman, 1957; Puterman, 1994). However, these solutions are not tractable when the number of states is large. In this case, one can resort to function approximation methods to find approximate solutions (Powell, 2007; Bertsekas, 2000).

A greedy policy can be extracted from any given value function using a one timestep lookahead, or directly from the state-action value. A greedy policy extracted from the optimal value function is optimal:

$$\pi^*(s) = \arg\max_a \left[\mathcal{R}(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') V^*(s') \right] \tag{2.26}$$

Despite the fact that a single time step lookahead is applied, the policy obtained is optimal in the long-term due to the recursive nature of Equation 2.26. Quoting Bellman: "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" (Bellman, 1954).

2.3.2 Partially Observable Markov Decision Processes

The Partially Observable Markov Decision Processes (POMDP) framework extends MDPs to environments that cannot be observed directly (Åström, 1965). Instead, noisy or incomplete observations of the state are available and the agent takes actions based on an estimation of what the true state of the world is. The goal of the agent is still to maximize the long-term expected cumulative reward it receives. Formally, a POMDP is described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \mathcal{O} \rangle$, where:

- The tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ defines a (fully observable) MDP.
- $\mathcal{Z}$ is a finite set of m possible observations. The observation at time t is denoted o_t , where t is a discrete time index.
- $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \mathscr{P}(\mathcal{Z})$ is the observation function. For each action a and resulting state s', the observation function defines a probability distribution over the possible observations:

$$O(o, a, s') = P(o_{t+1} = o | s_{t+1} = s', a_t = a) \tag{2.27}$$

Since an agent acting on a POMDP cannot observe the state of the world, it might keep track of the history of actions taken and observations received, and take actions based on this information. The history at time t is defined as:

$$h_t \doteq \{a_0, o_1, \ldots, a_{t-1}, o_t\} \tag{2.28}$$

Similarly, we can define a history $h_t a_t \doteq \{a_0, o_1, \ldots, a_{t-1}, o_t, a_t\}$. Then, a policy could be described as a mapping from histories to actions. However, a common alternative is to maintain an estimate of the state of world, known as the belief distribution, and map instead beliefs to actions. The belief is a sufficient statistic for the history and it is defined as:

$$b_t(s) \doteq P(s_t = s | o_t, a_{t-1}, o_{t-1}, \ldots, a_0, b_0) \tag{2.29}$$

where b_0 is the belief at $t = 0$. The belief b_t at any arbitrary time step t can be recursively computed given the belief b_{t-1} and action taken a_{t-1} at the previous time step, and the observation o_t received:

$$\begin{aligned}
\tau(b_{t-1}, a_{t-1}, o_t) = b_t(s') &= P(s' | o_t, a_{t-1}, b_{t-1}) \\
&= \frac{P(o_t | s', a_{t-1}, b_{t-1}) P(s' | a_{t-1}, b_{t-1})}{P(o_t | a_{t-1}, b_{t-1})} \\
&= \frac{P(o_t | s', a_{t-1}) \sum_s P(s', s | a_{t-1}, b_{t-1})}{P(o_t | a_{t-1}, b_{t-1})} \\
&= \frac{P(o_t | s', a_{t-1}) \sum_s P(s' | s, a_{t-1}, \cancel{b_{t-1}}) P(s | \cancel{a_{t-1}}, b_{t-1})}{P(o_t | a_{t-1}, b_{t-1})} \\
&= \frac{\mathcal{O}(o_t, a_{t-1}, s') \sum_s \mathcal{T}(s, a_{t-1}, s') b_{t-1}(s)}{P(o_t | a_{t-1}, b_{t-1})}
\end{aligned} \tag{2.30}$$

A POMDP policy $\pi : \mathcal{B} \mapsto \mathcal{A}$ is defined as the mapping from beliefs $b \in \mathcal{B}$ to actions $a \in \mathcal{A}$. Here, $\mathcal{B}$ represents the belief space, which is continuous with dimension $\dim(\mathcal{B}) = |S| - 1$[2]. In POMDPs, the value function gives the expected discounted reward of following policy π from any given belief $b \in \mathcal{B}$:

$$V_\pi(b) \doteq \mathcal{R}(b) + \gamma \sum_{o \in \mathcal{Z}} P(o|b, \pi(b)) V_\pi(\tau(b, \pi(b), o)) \tag{2.31}$$

where $\mathcal{R}(b) = \sum_{s \in \mathcal{S}} b(s)\mathcal{R}(s)$. Similarly, the state-action value function for POMDPs is defined as:

$$Q_\pi(b, a) \doteq \mathcal{R}(b) + \gamma \sum_{o \in \mathcal{Z}} P(o|b, a) V_\pi(\tau(b, a, o)) \tag{2.32}$$

The optimal value function for all $b \in \mathcal{B}$ is then given by:

$$V^*(b) = \max_{a \in \mathcal{A}} \underbrace{\left[\mathcal{R}(b) + \gamma \sum_{o \in \mathcal{Z}} P(o|b, a) V^*(\tau(b, a, o)) \right]}_{Q^*(b,a)} \tag{2.33}$$

from where the optimal policy could be extracted using a one timestep lookahead. However, the optimal value function cannot any longer be obtained using the iterative updates of dynamic programming algorithms since $\mathcal{B}$ is continuous. Luckily, a key result by Sondik shows that the value function has a special structure: it is piecewise-linear and convex (PWLC)[3] and this property is maintained after each iterative update (Sondik, 1971). The value function can then be represented by the upper surface determined by a set of hyperplanes:

$$V(b) = \max_{\alpha \in \Gamma} b \cdot \alpha \tag{2.34}$$

where each α is called an α-vector and denotes a single hyperplane, and Γ represents the set of hyperplanes. Intuitively, each α-vector represents the value of following a particular policy tree (Kaelbling et al., 1998). Given a PWLC value function, the optimal action for any belief can be extracted by considering the policy tree associated to the dominant α-vector at that particular belief.

By exploiting the PWLC property, the value function can be found through iterative complete backups. Unfortunately, the number of α-vectors grows exponentially in the number of observations at each iteration, which is known as the curse of history (Pineau et al., 2006). To reduce computation, most exact approaches prune the set of hyperplanes at each iteration by discarding the dominated α-vectors (Smallwood and Sondik, 1973; Cassandra et al., 1997; Kaelbling et al., 1998). Despite this, exactly solving a POMDP remains limited to problems with small state, action, and observation spaces. In the next subsection, we review some of

[2] Any POMDP can be cast as a continuous-state MDP, where the states represent the beliefs of the POMDP. The solution of this *belief-state* MDP, is also a solution for the original POMDP.

[3] This is true both for finite-horizon and infinite-horizon POMDPs. However, representing the optimal value function in infinite-horizon POMDPs may require infinitely many hyperplanes. Luckily, they can be approximated arbitrarily close by a finite-horizon value function with a sufficiently long horizon (Sondik, 1978; Kaelbling et al., 1998).

the existing approximate approaches for solving POMDPs that are capable of handling larger problems.

Approximate approaches to solve POMDPs

As we have discussed above, POMDPs are affected by the curse of history; their computational complexity grows exponentially with the planning horizon. A second challenge, shared with MDPs, is the curse of dimensionality. The size of the state space grows exponentially with the number of state variables. POMDPs, in particular, need to plan on a $(|S| - 1)$-dimensional continuous belief space. If we were to discretize this belief space, the number of possible beliefs would grow exponentially with the number of states. In this subsection, we present some of the existing literature to overcome these challenges and to solve POMDPs approximately. We focus here on offline approaches, where the optimal action is approximately calculated for all possible beliefs before any execution takes place. The next subsection will cover online approaches, which interleave execution and planning. The main advantage of offline methods is a faster policy execution, since no further planning is required.

Approximate solution methods for POMDPs can be classified into value-based and policy-based.

Value-based solutions Approximate value-based methods avoid the exponential increase in the number of $\boldsymbol{\alpha}$-vectors at each value function backup by applying it only at specific belief states instead of over the complete belief space. This technique is generally known as point-based value iteration and it addresses the curse of history. We discuss here some of more representative point-based approaches. The main difference between them lies in how to select the belief states, and on how to order the point-based backups.

- **Grid-based approaches**. A straight-forward approach to select the set of belief states is to do so in a grid-pattern, covering the complete belief space (Lovejoy, 1991). Only the α-vectors that are optimal at each of the selected belief states are maintained. The main drawback of this approach is that many of the selected belief states might be located in an unreachable part of the belief space.
- **Point-Based Value Iteration (PBVI)**. An alternative to grid-like patterns is to select a set of belief states that are reachable from the starting belief state b_0. The PBVI algorithm alternates a step of belief set expansion, where only reachable beliefs from those already in the set are considered, with a step of value iteration (Pineau et al., 2003). Hence, the algorithm works in an anytime fashion, trading off computation time and solution quality. For the belief set expansion, forward simulation is performed at each of the beliefs in the set. The resulting beliefs that are the furthest away from the existing set according to a distance metric are added to the set, so as to spread the set of belief states as much as possible in the reachable belief space.
- **Perseus: Randomized Point-based Value Iteration for POMDPs**. The Perseus algorithm (Spaan and Vlassis, 2005) also starts from an initial belief b_0, from where it randomly samples trajectories to select a large set of belief states. The set of beliefs will

remain constant throughout the complete algorithm. Similarly to PBVI, Perseus performs point-based backups. However, at each value iteration epoch, the backup is performed only at a few randomly sampled belief points. The procedure to select and backup the belief points guarantees that the value is improved for every point in the belief set. Unlike PBVI, Perseus does not require expensive computations to iteratively expand the set of belief states; however, randomly choosing the order of the backups can lead to a slow convergence of the value function (Shani et al., 2013).

- **Heuristic Search Value Iteration (HSVI)**. The HSVI algorithm (Smith and Simmons, 2004; Smith and Simmons, 2005) constructs a belief search tree starting from an initial belief state b_0. The nodes in the tree correspond to explored belief states, and an upper and lower bound on the optimal value function are maintained for each node. During the exploration of the tree, the algorithm chooses the action with the greatest upper bound and then it chooses the observation that leads to the greatest information gain. Once a sufficient depth is reached, the tree is traversed in the opposite direction up to the initial belief state, updating the value at each of the traversed belief states. The backup order from successors to predecessors can lead to faster convergence of the value function compared to randomized algorithms like Perseus. The algorithm works in an anytime fashion. A path in the tree is no longer explored once a sufficiently small gap between the bounds is found. Performance-wise, the maintenance of the bounds is an expensive operation but it balances out with the reduced number of updates.

Policy-based solutions Alternatively to value-based methods, it is possible to search for the optimal plan directly in the space of policies. Policies can be represented as plan graphs, which are essentially finite-state controllers (FSC) (Kaelbling et al., 1998). The graph nodes represent a particular partition of the belief space, and hence an explicit representation of the belief space is no longer required. Based on this representation, Hansen proposes a policy iteration algorithm for POMDPs (Hansen, 1998a). The algorithm starts by representing the policy as an initial FSC, and continues by alternating policy evaluation and policy improvement steps. During the policy evaluation step, the value of the policy is calculated by retrieving the α-vector representation from the FSC, which can be done in a straightforward manner. During the policy improvement step, a full dynamic-programming backup is performed to obtain a new set of α-vectors, which are in turn used to update the FSC. This update can require changing, adding or pruning machine states. In the case of an optimal infinite horizon value function that has a finite representation, this procedure continues until convergence to an optimal FSC. Otherwise, policy iteration will converge to an ϵ-optimal FSC after a finite number of iterations (Hansen, 1998b).

Exactly solving a POMDP using policy iteration remains tractable only for small POMDPs. The reason for this is that the number of nodes grows exponentially with the planning horizon. Approximate policy-based methods render this approach tractable by constraining the policy space. This is achieved in practice by bounding the size of the FSC, which typically involves the pruning of less dominating nodes or performing selective backups (Poupart and Boutilier, 2004; Ji et al., 2007).

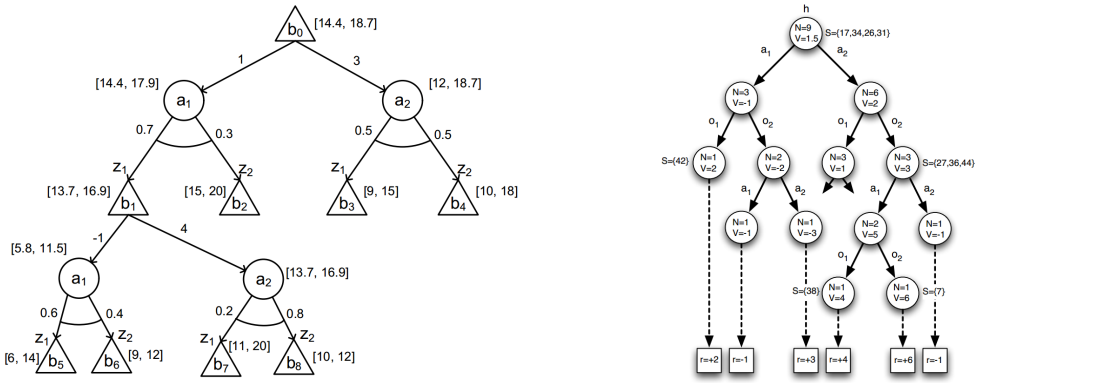
Online planning approaches

Offline POMDP approaches aim to find, prior to execution, an optimal policy under consideration of all possible future situations. This restricts their applicability to small and mid-size domains. In contrast, online approaches focus on finding a good local policy for the current belief state of the agent. Generally, a search tree rooted at the current belief is constructed by considering all possible actions and observations; the tree alternates layers with belief nodes and with action nodes (Figure 2.2a). The goal is to "smartly" explore the tree so that the value at the current belief is quickly and correctly estimated. Online POMDP algorithms interleave search steps with execution steps. As a consequence, the time available for planning is usually shorter than for offline approaches. Because of this, some online approaches leverage policies computed offline in order to speed up the search. After a search step is completed, the agent executes the best action from the current belief and receives an observation from the environment; the search procedure continues then from the new belief. Depending on the method used to explore the search tree, online POMDP algorithms are classified into three main groups: branch-and-bound pruning, heuristic search, and Monte Carlo sampling (Ross et al., 2008).

Branch-and-bound pruning The algorithms based on branch-and-bound pruning maintain upper and lower bounds on the value $Q^*(b, a)$ at each action node from the tree. If the upper bound on the value of taking an action from a given belief is lower than the lower bound of the value associated to taking a different action, then that branch is pruned out from the tree. The level of pruning that can be done depends on the quality of the bounds computed offline to evaluate the leaves of the tree. An example of an algorithm that relies on branch-and-bound pruning is the Real-Time Belief Space Search (RTBSS) algorithm (Paquet et al., 2005; Paquet et al., 2006).

Heuristic search Similar in spirit to HSVI, online algorithms based on heuristic search maintain upper and lower bounds on the value of all nodes of the search tree, and use them to direct the search towards promising parts of the tree (Satia and Lave., 1973; Washington, 1997; Ross and Chaib-draa, 2007).

Monte Carlo sampling Exploring the complete search tree that considers all actions and observations is only tractable for very shallow depths, as the number of possible histories grows exponentially with the planning horizon. To address this problem, Monte Carlo random simulations can be used to explore deep in the search tree. By sampling only specific state transitions and observations, this technique addresses both the curse of dimensionality and the curse of history. An example of this approach that has been able to solve large POMDP problems that remained intractable for traditional state-of-the-art full-width online planners is the Partially Observable Monte Carlo Planning (POMCP) algorithm (Silver and Veness, 2010). In POMCP, a search tree of histories is constructed (Figure 2.2b). For each node in the tree, an estimate of its value is maintained based on the return of the simulated trajectories that passed through it. The exploration and expansion of the tree is guided with the Upper Confidence Bounds 1 (UCB1) algorithm to balance exploration and exploitation (Auer et al.,



(a) AND-OR search tree for a POMDP with 2 actions and 2 observations. The belief states correspond to OR nodes and the actions to AND nodes. The values between brackets are lower and upper bounds on the value [Image credit: (Ross et al., 2008)].

(b) POMCP search tree for a POMDP with 2 actions, 2 observations, 50 states and no intermediate rewards. The value (V) and number of visits (N) is maintained for each history [Image credit: (Silver and Veness, 2010)].

Fig. 2.2.: Examples of search trees for online POMDPs.

2002). The key idea of POMCP is that the same simulations used to explore the tree are used for the belief state updates. Successor algorithms of POMCP include the Determinized Sparse Partially Observable Tree (DESPOT) (Somani et al., 2013) and the Adaptive Belief Tree (ABT) (Kurniawati and Yadav, 2016).

Solving continuous POMDPs

The POMDP solvers discussed so far assume that the POMDP model has discrete state, observation and action spaces. However, this is hardly the case in the robotics field. A straightforward solution can be to apply discretization and then use one of the discrete POMDP solvers. Unfortunately, in many cases it is complicated to find a compact discrete representation of the problem that remains tractable for existing solvers. Alternative approaches to handle continuous POMDPs include parametric approaches and Monte Carlo sampling.

In continuous POMDPs, the summations shown in equations 2.30 and 2.33 become integrals. If the POMDP model is restricted to have a parametric form, these integrals can be computed efficiently in closed-form. Following this idea, Porta et al. perform point-based value iteration by representing the observation and transition models and the value function as mixtures of Gaussian distributions (Porta et al., 2006). The downside of this approach is the exponential growth in the number of Gaussian components during value iteration. To avoid this, Brooks et al. propose to represent only the belief as a Gaussian distribution and to apply fitted value iteration (Brooks et al., 2006). The main problem shared by these approaches is the loss in the expressiveness of the model, which leads to poor performance in scenarios where the value function has sharp discontinuities (Brooks et al., 2006). Furthermore, they only consider continuous-state POMDPs. To handle continuous state or observation spaces sampling strategies are suggested.

Instead of using parametric forms, beliefs in continuous POMDPs can be represented with sets of particles (Thrun, 1999; Bai et al., 2011). Thrun proposes to use a particle filter for

the belief updates and fitted value iteration for solving the POMDP. The proposed function approximator is K-nearest neighbors with the Kullback-Leibler divergence between kernel-based belief density estimates as distance metric. The repeated distance computations, which are expensive in high-dimensional spaces, represent a computational burden for this approach. Additionally, tuning is required for the parameters of the kernels. To escape these issues, Bai et al. combine a particle-based belief representation with the use of α-functions encoded as policy graphs to represent the value function (Bai et al., 2011). Instead of using policy graphs, which pose some limitations for long planning horizons, Brechtel et al. propose to represent the α-functions using regression trees (Brechtel et al., 2013; Brechtel, 2015). Finally, Sunberg and Kochenderfer propose an extension of POMCP that can handle fully continuous POMDPs by applying progressive widening to the action and observation spaces (Sunberg and Kochenderfer, 2018).

2.3.3 Inverse reinforcement learning

In reinforcement learning, the goal is to find a policy that maximizes the expected cumulative long-term rewards. Consequently, the behavior of an agent for a given task is implicitly encoded by the reward function, i.e., given the reward function and the agent's dynamics, the optimal policy is determined. Reward shaping techniques focus on the design of reward functions that can guide and scale up the search process of an exploring agent (Randløv and Alstrøm, 1998; Ng et al., 1999). However, specifying an appropriate reward function for complex, high-dimensional robotic problems can be extremely challenging, and reward shaping techniques can lead to bugs in which the reinforcement learning algorithm exploits the reward function in ways not anticipated by the designer (Kober et al., 2013). Instead of relying on hand-engineered reward functions, the desired behavior of an agent in a given task can be specified by providing expert demonstrations of the task (Argall et al., 2009). While it is possible to extract policies directly from the demonstrations using imitation learning techniques, the resulting policies tend to perform poorly when the agent diverges too much from the demonstrations (Pomerleau, 1991). It is commonly accepted that in reinforcement learning "the reward function, rather than the policy or the value function, is the most succinct, robust, and transferable definition of a task" (Abbeel and Ng, 2004; Ng and Russell, 2000). Because of this reason, a lot of research efforts in recent years have been devoted to the development of algorithms that can extract a reward function from demonstrations, a task that is known as Inverse Reinforcement Learning (IRL) (Russell, 1998; Ng and Russell, 2000).

Traditionally, IRL is formulated in the context of an MDP. Let M be the MDP that models the behavior of an agent in a given task. In the IRL problem, the reward function $\mathcal{R}$ is unknown. Instead, a set of trajectories $\mathcal{D} = \{\xi^{[1]}, \xi^{[2]}, \ldots, \xi^{[m]}\}$ (ideally) sampled from the optimal policy is available. A trajectory is defined as a sequence of states of the agent $\xi = \{s_1, \ldots, s_T\}$. In robotics, the demonstrated trajectories often come from an expert who demonstrates the task. The goal is to find the hidden reward function $\mathcal{R}$ that best explains the available data $\mathcal{D}$. To

simplify the problem, the majority of existing IRL algorithms assume that the reward function can be represented as a linear combination of features:

$$\mathcal{R}(s) = \mathbf{w}^T \mathbf{f}(s) \tag{2.35}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ is the unknown weight vector and $\mathbf{f}(s) = (f_1(s), \ldots, f_K(s))$ is the feature vector that parameterizes state s, both of dimension K. The goal of IRL is then to fit the weight parameters $\mathbf{w}$ so that a reward is obtained that makes the trajectories in $\mathcal{D}$ optimal. However, as noted by Ng and Russell, restricting the reward to a linear function approximator might preclude finding one such solution. The goal is then transformed into finding the reward parameters that make the value function of the policy implicitly described by $\mathcal{D}$ better than the value of any other policy. Based on this idea, Ng and Russell propose a linear programming solution to the problem (Ng and Russell, 2000). Following the same lines, Abbeel and Ng propose the *margin-based* formulation of IRL (Abbeel and Ng, 2004). First, they highlight the importance of the empirical sum of features $\mathbf{f}_\mathcal{D} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} \gamma^t \mathbf{f}(s_t^{(i)})$ to characterize the demonstrated behavior. Any policy whose feature expectations match $\mathbf{f}_\mathcal{D}$ will achieve the same performance as the expert who demonstrated the task:

$$\begin{aligned} V^{\pi}_{s_0 \sim S_0}(s_0) &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t) | \pi] \\ &= \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbf{w}^T \mathbf{f}(s_t) | \pi] \\ &= \mathbf{w}^T \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbf{f}(s_t) | \pi] \end{aligned} \tag{2.36}$$

where S_0 is the initial state distribution. Based on this fact, they proposed an iterative algorithm composed of two main steps: 1) finding the optimal policy of the MDP under the current reward model; and 2) recomputing the reward model so that the value of the expert policy is larger by a margin than any other policy. The algorithm terminates once a policy is found with feature expectations close to those of the demonstrations. However, they noted that matching feature expectations does not imply that the true reward function has been recovered. This is so because IRL is an ill-posed problem: many policies can lead to the same feature expectations and a policy can be optimal under multiple reward functions.

To resolve the ambiguity, Ziebart et al. propose to apply the principle of maximum entropy to find the policy that does not exhibit any preferences other than matching the feature expectations (Ziebart et al., 2008). This is known as *Maximum Entropy IRL*. Solving the corresponding constrained optimization problem results for deterministic MDPs in a distribution of the form:

$$P(\xi | \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp \left(\sum_{s \in \xi} \mathbf{w}^T \mathbf{f}(s) \right) \tag{2.37}$$

Intuitively, trajectories that achieve the same rewards are equally likely, and those with greater rewards are exponentially preferred (Ziebart et al., 2008). The reward parameters can then be found by maximizing the likelihood of the demonstrations under this model:

$$\begin{aligned}\mathbf{w}^* &= \arg\max_{\mathbf{w}} \prod_{\xi_i \in \mathcal{D}} P(\xi_i|\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_{\xi_i \in \mathcal{D}} \log \frac{1}{Z(\mathbf{w})} e^{\mathcal{R}_{\xi_i}} \\ &= \arg\max_{\mathbf{w}} \underbrace{\frac{1}{m} \sum_{\xi_i \in \mathcal{D}} \left(\mathcal{R}_{\xi_i} - \log \sum_{\xi} e^{\mathcal{R}_\xi} \right)}_{\mathcal{L}(\mathbf{w})}\end{aligned} \tag{2.38}$$

where we have used the shorthand notation $\mathcal{R}_\xi = \sum_{s \in \xi} \mathcal{R}(s)$, and the term $1/m$ is introduced to simplify the expression of the gradient (it does not alter the solution of the optimization problem). The function $\mathcal{L}(\mathbf{w})$ is concave and differentiable, and its maximum can be found using gradient-based optimization. The gradient can be expressed as the difference between the (undiscounted) sum of features of the demonstrations $\tilde{\mathbf{f}}_\mathcal{D}$ and the expected feature counts:

$$\begin{aligned}\frac{dL(\mathbf{w})}{d\mathbf{w}} &= \frac{1}{m} \sum_{\xi_i \in \mathcal{D}} \left(\frac{d\mathcal{R}_{\xi_i}}{d\mathbf{w}} - \frac{1}{\sum_\xi e^{\mathcal{R}_\xi}} \Big(\sum_\xi e^{\mathcal{R}_\xi} \frac{d\mathcal{R}_\xi}{d\mathbf{w}} \Big) \right) \\ &= \frac{1}{m} \sum_{\xi_i \in \mathcal{D}} \Big(\mathbf{f}_{\xi_i} - \sum_\xi \frac{e^{\mathcal{R}_\xi}}{\sum_\xi e^{\mathcal{R}_\xi}} \mathbf{f}_\xi \Big) \\ &= \frac{1}{m} \sum_{\xi_i \in \mathcal{D}} \mathbf{f}_{\xi_i} - \sum_\xi P(\xi|\mathbf{w}) \mathbf{f}_\xi \\ &= \tilde{\mathbf{f}}_\mathcal{D} - \mathbb{E}_{P(\xi|\mathbf{w})}[\mathbf{f}_\xi]\end{aligned} \tag{2.39}$$

where the shorthand $\mathbf{f}_\xi = \sum_{s \in \xi} \mathbf{f}(s)$ was used. To calculate the expectation in Equation 2.39 a sum over all possible trajectories is required. To avoid the complexity of the naïve solution, which scales exponentially with the planning horizon, Ziebart et al. propose a dynamic-programming algorithm that calculates the expected state visitation frequencies in polynomial time (Ziebart et al., 2008). The maximum entropy IRL was subsequently extended to handle cases where only noisy trajectories of the agent are available (Kitani et al., 2012).

Learning non-linear reward models

In contrast to the IRL approaches discussed so far, which assume that a linear reward model can be used to encode the behavior of the agent, several works in the literature focus on learning non-linear models (Ratliff et al., 2006b; Levine et al., 2011; Wulfmeier et al., 2015). Ratliff et al. propose a structured max-margin formulation of IRL (Ratliff et al., 2006a). Their optimization-based solution looks for a reward that makes the demonstrated trajectories better by a margin than those of any other policy, where the margin should be larger for policies very different from

the expert's. This approach is extended to learn non-linear rewards through feature construction using classifiers; new features are iteratively constructed as non-linear functions of the initial set of features until the planned trajectories resemble the demonstrations (Ratliff et al., 2006b). Another alternative to model the reward as a non-linear function is by using Gaussian Processes (GP) as the mapping from features to rewards (Levine et al., 2011). In this approach, known as GPIRL, the structure of the reward is determined by the kernel function of the GP. An advantage of this approach is that it can provide a measure of uncertainty of the predicted reward. However, its computational complexity does not scale well to large state-spaces. To avoid this high computational cost, as well as the manual design of the structure of the reward model, Deep Neural Networks (DNNs) have also been explored as the function approximator in IRL problems (Wulfmeier et al., 2015; Wulfmeier et al., 2017). In their DeepIRL approach, Wulfmeier et al. propose to use Fully Convolutional Neural Networks (FCNN) to directly learn the mapping from states to rewards (Wulfmeier et al., 2015; Wulfmeier et al., 2017). A FCNN is a Convolutional Neural Network (CNN) without the final fully-connected layer. For IRL, the structure of the network is configured such that the dimensions of the input and the output are the same, effectively mapping input states to rewards in the output. To train the network, this approach relies on the maximum entropy IRL paradigm. The difference between the state visitation frequencies from the demonstrations and the ones induced by the current configuration of the network (calculated using the dynamic programming solution introduced by Ziebart et al.) is backpropagated through the network to update its parameters. Results on synthetic domains show how DeepIRL outperforms GPIRL at capturing complex non-linear reward structures. However, DeepIRL needs in general more samples to achieve the same performance as GPIRL, and it is also prone to overfitting when not regularized.

Learning reward models in high-dimensional or continuous spaces

A general operational pattern can be recognized in most IRL approaches. Starting off a random reward model, the optimal MDP policy is obtained and used to calculate a similarity metric with the demonstrations, which is subsequently used to update the reward model. This iterative process continues until the demonstrations and the optimal policy are close enough according to the similarity metric. In the robotics domain, problems often have high-dimensional or continuous state spaces, and the exact dynamics of the system might be unknown. In such cases, the computation of the full policy becomes intractable and the discussed approaches cannot be applied. To overcome these issues, several approximations can be applied.

Firstly, solving the forward problem under unknown dynamics could be achieved through model-free reinforcement learning methods. However, these methods are generally too slow to be called repetitively during the optimization process. Instead, Boularias et al. propose to find a policy that minimizes the relative entropy with respect to the empirical distribution of demonstrated trajectories, subject to matching feature counts (Boularias et al., 2011). To cope with the unknown dynamics, the gradient is approximated by sampling trajectories from an arbitrary policy and then applying importance sampling. Finn et al. propose a similar method based on sampling (Finn et al., 2016). However, instead of uniformly sampling trajectories,

they iteratively refine the sampling distribution using guided policy search (Levine and Abbeel, 2014) to focus the sampling on regions with high rewards. This has the drawback of producing a reward model that only explains the demonstrations locally and fails to generalize to unexplored regions of the state space (Finn et al., 2016).

Secondly, in large or continuous state spaces approximations can be used instead of calculating the full policy. Levine and Koltun consider a local approximation of the reward function likelihood ($\mathcal{L}(\mathbf{w})$ in Equation 2.38) and directly maximize it for the demonstrated trajectories using gradient-based optimization (Levine and Koltun, 2012). In this setting, only locally optimal demonstrated trajectories are required. However, if globally optimal data is available, the approach will fail to exploit this information. Another option to maximize the reward likelihood without calculating the full policy is to use problem-dependent, trajectory planners to solve the forward problem. This way, the expected value of the features under the current reward model from Equation 2.39 is calculated over a set of trajectories produced by the planner (Lee and Seo, 2017; Pérez-Higueras et al., 2018), or approximated with the features of the best trajectory (Kuderer et al., 2015). Alternatively, sampling methods such as Markov chain Monte Carlo (MCMC) can be used efficiently to approximate the expectation by exploiting prior knowledge on the structure of the trajectory distributions (Kretzschmar et al., 2016). The prior knowledge on the structure of the problem can also be exploited to construct meaningful, graph-based representations of the underlying continuous domain, in which discrete IRL algorithms remain tractable (Byravan et al., 2015; Okal and Arras, 2016).

2.4 Nomenclature in the context of automated driving

In this section, we introduce some common terminology used in the context of motion prediction and motion planning for automated vehicles. We adhere to the nomenclature established in the literature (Fraichard and Howard, 2012; Katrakazas et al., 2015; Paden et al., 2016). As we will see, there is a big overlap between the robotics nomenclature and that of MDPs. For convenience and clarity, we restate here several of the concepts introduced in section 2.3.

In the field of robotics, the *configuration of a robot* is the set of parameters that uniquely specifies the position and orientation of all its components (Fraichard and Howard, 2012). For mobile car-like robots, we can simplify this definition to say that the configuration is the specification of the position and orientation of the robot relative to some global coordinate system. This is also commonly known as the *pose* of the robot. The set of all possible robot configurations defines the *configuration space*, or $\mathcal{C}$-space. In a single-agent setting, the robot navigates in a given environment where there could be static and dynamic obstacles. A *state* $\mathbf{x}_t$ represents a characterization of the environment at a given point in time t. The state contains all information that could be relevant to predict the future; that naturally includes the configurations of the robot and of any potential obstacle, their velocities, intentions, etc. The *state space* is the set of all possible states. An autonomous robot navigating in an environment can take actions, that is, control commands $\mathbf{u}_t$ that alter the state of the environment. The way the state changes after a control command is given by a *motion model*, normally expressed as a set of differential equations $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$. The state change will therefore depend on the current state, control

command and on the amount of time past. When not all the degrees of freedom of the robot are controlled, the motion is called *non-holonomic*, which is the case with car-like robots.

A common problem with mobile robots is that of finding a continuous, collision-free path between an initial and a goal configuration of the robot. This problem is known as *path planning* and it involves only geometric information, without any consideration of time. A path planner is *complete* if, in finite time, it can produce a solution or find out that there is none. A plan is *feasible* if it satisfies the constraints of the problem, and it is *optimal* if it optimizes a given quality metric. In the context of the architecture of an automated vehicle (Figure 1.2), the path planner is often used for route planning in the strategic layer.

Once a global route is found, the robot proceeds to follow it. During the navigation along its global path, an automated vehicle will need to adhere to the traffic rules and driving conventions, while potentially interacting with other traffic participants. The handling of such tasks is generally known as *decision-making*, and taken care of by a *behavioral planner* (also called maneuver planner or tactical planner). The behavioral planner dictates the maneuver to be performed by the vehicle, e.g. yielding at an unsignaled intersection or doing a lane change in the highway. Once selected, the maneuver is passed down to the operational layer of the vehicle to be transformed into a trajectory and, ultimately, into control commands for the vehicle.

In contrast to a path, which is a continuous sequence of configurations of the robot connecting two different endpoints, a *trajectory* is a continuous sequence of states parameterized by time. If the velocity of the robot is part of the state, then a trajectory defines both a geometric path and a velocity profile. Generally, the *trajectory planner* of an automated vehicle takes as inputs the maneuver to be performed and the kinematic and dynamic constraints of the vehicle. The trajectory planning problem is often formulated as finding the sequence of control commands that make the vehicle perform the maneuver while respecting its kinematic and dynamic constraints, and optionally also optimizing a given cost function. The cost function can include terms to promote passenger comfort and minimize the risk of collisions (McNaughton et al., 2011). In the literature, trajectory planning is often referred to as *motion planning*. In the end, the calculated trajectory is sent to a controller to be transformed into commands for the vehicle's actuators.

State of the Art

3

In this chapter, we review the state of the art on three of the major building blocks of the architecture of an autonomous vehicle. First, we review existing methods to construct models that encode the driving preferences of a human driver. Such models are used, for example, when an autonomous vehicle needs to choose among alternative, equally safe, trajectories. Secondly, the state of the art on vehicle motion prediction is described. This concerns the construction of motion models to predict the evolution of traffic scenes and constitutes the key element behind collision risk assessment methods. Finally, we present the state of the art on tactical decision-making for intelligent vehicles. The goal here is to construct plans of maneuvers for the automated vehicle that result in safe progress towards its destination.

3.1 Modeling driver preferences

Human drivers instinctively plan trajectories that are safe, comfortable, adhere to the traffic rules and are congruent with the global path that will take them to their destination. Most autonomous driving motion planners attempt to imitate this driving behavior by optimizing the trajectories with respect to a behavioral model, typically formulated as a cost function. In consequence, the cost function usually contains terms that discourage getting close to obstacles, penalize passenger discomfort (e.g. high lateral accelerations), encode the traffic rules (e.g. maximum speed allowed), or reflect the driver's personal driving preferences (McNaughton et al., 2011). As a clear example of this idea, Wolf and Burdick propose a number of potential functions whose superposition results in a potential field that induces human-like motion in multi-lane highways (Wolf and Burdick, 2008). The functions that compose this potential field, shown in Figure 3.1, encourage not traversing the highway borders, navigating in the center of the lanes, maintaining a safe distance with surrounding traffic and driving at a desired speed. However, both McNaughton et al. and Wolf and Burdick remark the importance of correctly tuning the model, with the former relying on manual hand-tuning and the latter on automatic parameter exploration. Unfortunately, as the complexity of the model increases these tuning techniques become less effective, usually involving a high-degree of human effort. Despite this, manual hand-tuning is the most often used technique in the prediction (Sorstedt et al., 2011; Bahram et al., 2016) and planning (Thrun et al., 2006; Montemerlo et al., 2009; Wei and Dolan, 2009; Ardelt et al., 2012) literature.

To avoid the expensive hand-tuning of the cost function, several approaches presented in recent years rely on IRL to automatically learn the cost function's parameters from demonstrated driving data. Most of such approaches assume a cost function that is linear on a set of relevant features selected using background knowledge. Given a dataset of demonstrated trajectories, IRL can provide the weight vector that balances the relevance of the features according to the demonstrated behavior. This technique has been applied for example to learn cost functions for path planning in parking lots (Abbeel et al., 2008) and for tactical decision-making on

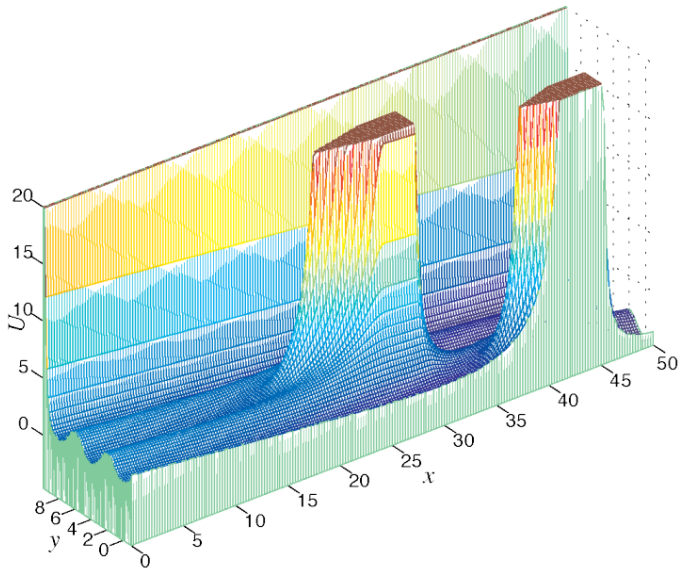


Fig. 3.1.: Example of potential field for a 3-lane highway with 2 dynamic obstacles [Image credit: (Wolf and Burdick, 2008)]

residential roads (Shimosaka et al., 2014). Furthermore, the feature construction step can also be automatized through the use of DNN function approximators (Wulfmeier et al., 2017). Wulfmeier et al. parameterize the cost function as a fully convolutional DNN that maps statistics extracted from LiDAR point-clouds directly to a grid-based cost map. By optimizing the parameters of the NN using maximum entropy IRL on a dataset of human-driven trajectories, a non-linear cost function that represents drivable areas is obtained.

A common characteristic of the IRL-based approaches above is that the cost function is learned for static environments, where the feature representation of a state does not change over time. However, this is not the case for environments with dynamic obstacles, unless time is considered part of the state representation. Unfortunately, augmenting the state with time significantly increases the complexity of the forward problem, whose solution is necessary to calculate the expectation from Equation 2.39. Henry et al. and Vasquez et al. propose a workaround that consists in replanning at each time step assuming a constant environment (Henry et al., 2010; Vasquez et al., 2014). Due to this assumption, a partial path is used at each time step to calculate the gradient and update the model. This computationally expensive procedure has yielded good results in simulated mobile robot navigation experiments. Nonetheless, assuming a static environment even for a small time window seems a poor fit for highly dynamic tasks like highway driving. Kuderer et al. propose an alternative solution based on calculating the expectation from Equation 2.39 using a single trajectory instead of the full policy (Kuderer et al., 2015). The trajectory is obtained using spline-based trajectory optimization with Resilient Backpropagation (RPROP) (Riedmiller and Braun, 1993). The benefit of this approach is that the resulting trajectory is optimized with respect to the trajectories of surrounding vehicles, constituting thus an adequate approximation to the optimal policy induced by the model. On the other hand, this technique is susceptible to local minima. The approach is applied to learning different driving styles from real data of highway driving. Lee and Seo apply IRL to learn a

cost function that models overtaking in urban environments (Lee and Seo, 2017). Similarly to the approach from Kuderer et al., a single trajectory is used to calculate the gradient at each optimization step. In this approach, the trajectory is obtained by sampling multiple candidate trajectories from a GP and selecting the one with the lowest cost under the current model hypothesis. The drawback of this approach is that it requires learning additional GP models in order to learn cost functions for different road topologies.

In conclusion, IRL constitutes an adequate technique to automatically learn models that capture the driving preferences of human drivers. However, the potential trajectories of an automated vehicle lie in a high dimensional continuous space, making approximations necessary to solve the IRL problem, especially in dynamic environments. Despite this, promising driver modeling results have been obtained in urban and highway environments, confirming IRL as a reliable option to avoid the manual tuning of the parameters of the model.

3.2 Motion prediction

In order to take safe driving decisions, an intelligent vehicle needs to be able to accurately predict the development of traffic situations. This involves inferring the intentions of the surrounding drivers and predicting what their states[1] will be in future timesteps. This problem is generally known as motion prediction. Existing motion prediction approaches are typically classified into three categories (Lefèvre et al., 2014): physics-based, maneuver-based and interaction-aware.

3.2.1 Physics-based approaches

In physics-based approaches, the future states of any vehicle are predicted by propagating forward its current trajectory using simple dynamic or kinematic models (Rong Li and Jilkov, 2000; Schubert et al., 2008). Existing approaches differ mainly in the motion model used to propagate the trajectory, and on how they handle uncertainties.

One of the most often used motion models in the context of motion prediction, despite their extreme simplicity, are the linear motion models. These models predict the motion of the target vehicle assuming it moves with constant velocity (CV) or with constant acceleration (CA). Ammoun and Nashashibi apply a CV motion model in the context of collision risk estimation between vehicles (Ammoun and Nashashibi, 2009). The uncertainties about the states of the vehicles are modeled as multivariate Gaussian distributions and propagated forward in time using the prediction step of a Kalman filter. The time to collision (TTC) metric and the degree of overlap between the covariance matrices of the different vehicles are proposed as collision risk indicators. Rummelhard et al. also propose to estimate the collision risk based on the TTC metric obtained by applying a CV model (Rummelhard et al., 2014). In this case, the risk estimation is performed in the context of a dynamic occupancy grid framework, where each cell has associated estimations of its occupancy and dynamics (Nègre et al., 2014). By considering the occupancy of each cell and projecting forward its dynamics a risk grid can be constructed. This approach is highly parallelizable and allows for a quick and efficient

[1] In scenarios with N vehicles, we often talk about the state $\mathbf{x}_t^i$ of each vehicle i, with $i = 1, \ldots, N$. The state $\mathbf{x}$, as discussed in section 2.4, can be constructed as the concatenation of the states of each individual vehicle.

estimation of collision risk. However, the assumption about the linearity of the vehicles' motion limits these approaches to predicting only the most immediate future.

Instead of relying on a single motion model for prediction, which is counterintuitive for maneuvering targets such as road vehicles, several approaches propose instead to consider a group of motion models and continuously estimate which of them captures more accurately the dynamics exhibited by the target (Kaempchen et al., 2004; Jo et al., 2016). Such an approach is known as an Interacting Multiple Model (IMM). In practice, it consists of a bank of Kalman filters, each with different dynamics (e.g. lane keeping with constant velocity or lane changing with constant acceleration). A probability distribution over the filters in the bank is maintained and updated with each observation; this distribution intuitively represents the current motion model of the target. The final predictions are produced as a weighted combination of the individual predictions of each filter. The uncertainties in the predictions continue therefore to be represented as Gaussian distributions.

Monte Carlo sampling constitutes an alternative technique to model the uncertainty in the predictions (Broadhurst et al., 2005; Kaempchen et al., 2009). Broadhurst et al. propose to sample in the control space of the motion model to generate a probability distribution over the possible future motions of every obstacle in a traffic scene (Broadhurst et al., 2005). Sampled trajectories that lead to collisions or to non-drivable areas are disregarded, and the rest are ranked based on criteria such as the level of risk and comfort. An approach in the same lines is proposed for the assessment of emergency traffic situations (Kaempchen et al., 2009). The input control space of all traffic participants (including the host vehicle) is searched to find a combination of trajectories that does not lead to a collision. If such a combination is not found, a collision is unavoidable and the emergency brake of the host vehicle is engaged.

Overall, the main advantages of the physics-based approaches are their simplicity and low computational cost, their consideration of the kinematic and/or dynamic constraints of the vehicles, and their lack of assumptions regarding the road layout and traffic situation. However, in these approaches no reasoning is done about the intent of the vehicles in the scene and the interactions between traffic participants are completely neglected. Consequently, the predicted trajectories are extremely conservative and only valuable in the short-term future.

3.2.2 Maneuver-based approaches

Maneuver-based motion prediction approaches consist in a two-step procedure. First, the maneuver currently being executed by each vehicle is estimated. Based on that estimation, the maneuver execution is predicted. Existing approaches differ significantly in how they identify the maneuvers and in how the maneuver-dependent trajectories are predicted.

Maneuver recognition

Road vehicles are maneuvering entities whose dynamics change significantly depending on the maneuver in execution. Several techniques have been proposed to identify the maneuver being executed by a vehicle. These techniques include dynamics matching, supervised learning classification models, state-space models and trajectory matching.

The IMM approach discussed in the context of physics-based approaches maintains a probability distribution over the set of motion models in the filter. The distribution is updated with each incoming observation, increasing the probability of the models under which the observation is more likely. While originally designed for enhanced state tracking, the IMM can be used for maneuver inference by making each of the motion models correspond to a different maneuver (Weiss et al., 2004; Toledo-Moreo and Zamora-Izquierdo, 2009). We refer to this approach as dynamics matching. Weiss et al. use an IMM filter to detect lane change maneuvers by considering two CV motion models with different process noise in the longitudinal and lateral velocity (Weiss et al., 2004). Similarly, Toledo-Moreo and Zamora-Izquierdo propose an IMM with two CA and constant yaw-rate motion models to detect lane changes in highways (Toledo-Moreo and Zamora-Izquierdo, 2009). In this case, the curvature of the road is estimated from GPS data and leveraged to improve the accuracy of the state estimations in curved roads. While these approaches based on dynamics matching have proved effective at identifying simple maneuvers like lane changes, they are yet to be applied to capture more complex behaviors, which is a consequence of their inability to model complex control inputs. Furthermore, the performance of the IMM filter depends largely on the correct tuning of the noise parameters and of the Markovian transition matrix, which encodes the prior probability of the target changing its dynamics.

Another straightforward approach to infer the current maneuver of a target consists in applying supervised learning classification models such as Support Vector Machines (SVMs) (Kumar et al., 2013), Logistic Regression (Klingelschmitt et al., 2014) or Recurrent Neural Networks (RNNs) (Zyner et al., 2018). Kumar et al. propose to use an SVM with Gaussian kernel to predict the lane change intentions of the host vehicle (Kumar et al., 2013). The input features to the classifier are collected with a sliding time window and consist of the lateral position in the lane, the steering angle with respect to the road curvature, and their corresponding derivatives. A Bayesian filter is used on top of the SVM to smooth the predictions and mitigate the number of false positives. This approach is able to predict the lane changes of the host vehicle 1.3s before it crosses the lane marking on average. However, the number of false positive predictions remains relatively high even after filtering. Other approaches focus on predicting the behavior of drivers approaching intersections (Klingelschmitt et al., 2014; Zyner et al., 2018). Klingelschmitt et al. apply logistic regression to identify the maneuver intention of the driver from three possibilities, namely, going straight, turning right and stopping at the traffic light (Klingelschmitt et al., 2014). As relevant features for the classification task they consider the velocity profile and the anticipated velocity at the stop line. Zyner et al. focus instead on predicting the exit destination of drivers on a single-lane roundabout by means of an RNN (Zyner et al., 2018). The features considered are the position, heading and speed of the vehicle. The proposed system achieves a high accuracy at predicting the drivers' destinations and manages to do so 1.3s on average before the target vehicle reaches a conflict point (part of the roundabout in front of the entrances). Overall, applying supervised learning algorithms for the prediction of driver behavior constitutes a clear and straightforward solution. The downside of using such techniques is that their performance normally suffers when the movement of the

target vehicle is constrained by the surrounding traffic. For this reason, some authors propose extensions that aim to infer when the target vehicle is being influenced by others (Klingelschmitt et al., 2014).

State-space models, and more particularly Hidden Markov Models, have also been widely applied to model and infer the behavior of drivers (Oliver and Pentland, 2000; Tay, 2009; Vasquez et al., 2009). In an HMM, the motion patterns of the target vehicle are modeled as Markov chains, where the transition and emission models are learned offline from driving data. Thus, after observing a partial trajectory it is possible to infer the most-likely motion pattern and use it to predict the future. Oliver and Pentland use an HMM to identify the maneuvers of the host vehicle, modeling up to seven driver maneuvers, including lane changes, turns, starting and stopping (Oliver and Pentland, 2000). The input data includes most signals from the host vehicle (steering, throttle, brake, speed, acceleration, driver's gaze) as well as contextual information about the surrounding traffic. The proposed system is able to predict maneuvers one second on average before any significant change in the lane position takes place. Other authors propose a Hierarchical HMM with two layers to model driver behavior (Tay, 2009). The upper layer is an HMM where the hidden states represent high-level maneuvers such as overtaking, turning or going straight. For each of these high-level maneuvers, there is a corresponding HMM in the lower layer representing the low-level vehicle actions that characterize them. For example, a turn maneuver can be decomposed into a deceleration before the turn, the turn execution, and the following acceleration to resume normal speed. Inference about the high-level maneuvers is done by using the observation likelihoods of the lower-level HMMs as observations in the upper layer. The system is trained and evaluated on a simulator, where it achieves an average accuracy of 0.8 at recognizing high-level behaviors. In general, HMMs are learned completely offline and therefore the number of behaviors they can recognize during prediction is fixed. To overcome this limitation, an extension known as Growing HMMs (GHMMs) has been proposed (Vasquez et al., 2009). In a GHMM, the structure of the model and the parameters of the different motion patterns are learned incrementally at prediction time. The main difficulty faced by these discrete state-space models is finding the right state discretization; an excessively fine discretization will lead to overfitting and require a large amount of training data, whereas a course one could fail to properly capture the motion patterns.

A closely related technique to discrete SSMs for prediction is that of learning trajectory patterns through clustering (Tay, 2009; Atev et al., 2010; Joseph et al., 2010; Joseph et al., 2011). They key idea is that previously observed trajectories can be clustered together forming a set of prototype trajectories, each of them representing a different motion pattern. Once a partial trajectory is observed, it can be matched to a particular prototype trajectory, which can then be used to predict future motion. Most trajectory clustering approaches are based on selecting a similarity metric and a clustering algorithm. For example, Atev et al. use a similarity metric based on the Hausdorff distance together with agglomerative clustering (Atev et al., 2010). Another option to model distributions over trajectories is by using Gaussian Processes (GPs) (Tay, 2009; Joseph et al., 2010; Joseph et al., 2011). GPs are Gaussian distributions over functions (Rasmussen and Williams, 2006). They are often used to do non-linear regression,

where at test time a conditional Gaussian distribution over the output is obtained. To extract the different trajectory patterns from collected training data of a traffic scene, Tay proposes to learn a Gaussian process mixture model using the Expectation-Maximization (EM) algorithm (Tay, 2009). Each of the possible trajectory patterns is represented by two Gaussian processes in the mixture model, which map time to the x and y trajectory coordinates. Joseph et al. propose a similar approach based on learning GP-mixture models, but where the mappings learned are from positions to velocities (Joseph et al., 2010; Joseph et al., 2011). The advantage of this representation over the former is that it avoids dealing with the lengths and the discretization of the trajectories. After observing the partial trajectory of a target, its likelihood under all the learned GPs can be easily obtained, identifying thus the most likely trajectory pattern. This approach overcomes the discretization problems of discrete state-space models. Unfortunately, inference on a GP grows cubically on the number of training samples, which can limit its applicability.

Predicting maneuver-dependent trajectories

The second step of maneuver-based motion prediction approaches after the identification of the target's maneuver is the prediction of the future trajectory. When the maneuver is identified using a physics-based approach such as an IMM, the future states of the target are predicted by propagating the state estimate forward with the corresponding motion model. Typically, this involves propagating the Gaussian distribution that models the state uncertainty. As the prediction horizon increases, so does the state uncertainty, which can be exploited for collision risk estimation (Ammoun and Nashashibi, 2009).

In HMMs, if a model is learned for each different maneuver, then it can be used to stochastically predict the future state of the target (Meyer-Delius et al., 2009). However, when a single HMM is used to model high-level behavior, there is no straightforward way to predict the future states of the target. For this reason Tay proposes to combine HMM-based maneuver recognition with GP-based maneuver realizations (Tay, 2009). Each maneuver modeled with the HMM is associated to a trajectory pattern represented with a GP. After inferring the maneuver of a target using the HMM, its future trajectory is stochastically predicted by sampling from the predictive distribution of the GP. More precisely, a path is sampled using the GP and transformed into a trajectory by assuming a CA motion. This technique has been applied to collision risk estimation (Tay, 2009) and trajectory planning in dynamic environments (Fulgenzi et al., 2008). An obvious drawback of predicting trajectories by sampling from the predictive distribution of a GP is their inability to consider the vehicle dynamical constraints or the feasibility of the paths (no prior knowledge of road boundaries or static obstacles). To overcome this issue, Aoude et al. use the GP patterns to bias the growth of dynamically feasible trajectories for the target (Aoude et al., 2011).

In general, the predictions obtained from maneuver-based approaches are relevant for the duration of the identified maneuver, and thus longer than physics-based approaches. Unfortunately, these approaches fail to completely consider the interactions between vehicles. This

contrasts with the foresighted way in which human drivers make driving decisions and can lead to erroneous predictions of traffic situations.

3.2.3 Interaction-aware approaches

The motion of any vehicle in a traffic scene is always going to be influenced by the surrounding traffic. The physics- and maneuver-based approaches for motion prediction discussed so far do not completely model these inter-vehicle dependencies. In the best case, the unidirectional influence of the surrounding context is exploited to infer the most likely maneuver of the target vehicle (Oliver and Pentland, 2000; Klingelschmitt et al., 2014). However, this kind of asymmetrical reasoning completely disregards the complex interactions that explain human drivers' actions. Furthermore, it is often followed by a trajectory prediction that is agnostic to the surrounding traffic (Tay, 2009). These shortcomings have a direct impact in the robustness of the predictions, which are limited to short-term horizons. To reliably predict the long-term development of traffic situations the interaction-aware behavior of drivers needs to be modeled. Existing interaction-aware approaches can be classified into those based on leveraging context-dependent behavioral driver models, and those that model the interactions between traffic participants with dynamic Bayesian networks.

Model-based motion prediction

Model-based motion prediction approaches are based on the assumption that drivers behave in a risk-averse manner and will always choose maneuvers that keep them away from danger. This risk-averse behavior can be encoded in a driver model, typically formulated as a cost function that maps states of the vehicle to cost. As discussed in subsection 3.1, the cost function can contain risk-related terms (e.g. the distance to the closest vehicle in front), as well as terms related to the comfort or driving preferences of the driver (e.g. preferred speed). The cost function can therefore be used to score the alternative maneuvers or trajectories of the target vehicle, whose motion will be predicted assuming a preference for the lowest cost options. Since the cost value depends on the state of the surrounding traffic through the risk-related terms, the interactions between vehicles are implicitly taken into account. Existing approaches differ mainly in how to use the cost function to produce the maneuver or trajectory prediction for each vehicle, and on the assumptions made to keep the approach tractable.

Sorstedt et al. propose a vehicle motion model for single-lane, car-following scenarios in which the driver's control commands are taken into account (Sorstedt et al., 2011). To anticipate the control commands of the driver an optimal control problem is solved, where the optimality criterion is defined by a hand-tuned cost function. The cost function contains terms such as desired speed, preferred lane, and time-headway and distance to the leading vehicle. The proposed motion model is used to track the state of the host vehicle on a real car-following experiment, where it outperforms a CA motion model. It is not straightforward to adapt this approach for the motion prediction of all interacting vehicles in a more complex traffic scene. In contrast, Lawitzky et al. propose an approach to predict the maneuvers of all traffic participants in highway scenes (Lawitzky et al., 2013). It assumes perfect perception and

a discrete number of available maneuvers for the vehicles, such as switching lanes or changing the speed. The number of possible maneuver combinations at each prediction timestep grows thus exponentially in the number of vehicles. For each vehicle, a prior distribution over the possible maneuvers is calculated by considering their driving preferences and ignoring the presence of surrounding traffic. Using these priors, the probability of a collision is calculated for each possible combination of maneuvers. Finally, the posterior distribution over maneuvers for each individual vehicle is obtained by penalizing the maneuvers that led to collisions. The final posterior over maneuvers takes therefore into account the driving preferences of the drivers and their risk-averse behavior. The drawback of this approach is its time complexity, which grows exponentially in the number of vehicles. To address this issue, Schwarting and Pascheka propose to perform exhaustive maneuver search only when the egoistic maneuvers of two vehicles lead to a collision (Schwarting and Pascheka, 2014). As a result, the time complexity of this approach grows only quadratically with the number of vehicles. Among all the possible maneuver combinations between the two vehicles involved in a conflict, the one that minimizes a hand-tuned cost function is chosen. The cost function contains terms that discourage driving too close to each other or deviating from the drivers' desired speeds. This approach assumes thus a cooperative behavior between drivers. To avoid the manual hand-tuning of the cost function, Shimosaka et al. propose to use IRL to automatically learn the weights from driving data (Shimosaka et al., 2014). Using this approach, a driver model for residential areas is automatically learned and then used to predict the velocity changes of the host vehicle. However, the learned cost function contains only terms that depend on static parts of the environment, such as traffic signs or blind corners. Likewise, the prediction is focused on static environments.

Overall, the major advantage of model-based prediction approaches is that they can produce sensible long-term predictions of traffic situations that take into account the interactions between traffic participants. However, these approaches assume that all vehicles in the scene will behave so as to avoid collisions. As a consequence, they can fail to predict dangerous situations that arise when the behavior of a traffic participant deviates from the norm encoded in the model. This is particularly undesirable in the short-term future, where even the physics- and maneuver-based approaches could detect a dangerous situation from the dynamics of the target.

To circumvent the above-mentioned problem inherent to model-based approaches, Bahram et al. propose to predict the future sequence of maneuvers of any driver with a combination of model-based prediction and supervised learning classification (Bahram et al., 2016). As first step, a hand-tuned cost-function is used to iteratively predict the greedy longitudinal and lateral accelerations of all traffic participants up to a given prediction horizon. From the resulting sequence of states, the lane change intention of all drivers is calculated. The final lateral motion prediction is then produced using a Bayesian Network classifier. This classifier models the dependencies between the predicted model-based lane change intentions and recently observed dynamic features of the targets such as lateral position and lateral velocity. As a consequence, this approach infers the intention of surrounding vehicles by considering their

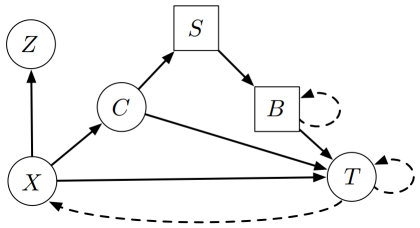
dynamics and their expected risk-aware behavior. This idea is in line with the interaction-aware driver behavior estimation approach that we propose in chapter 6. However, in contrast to this approach, we propose to explicitly model the interactions between traffic participants using a dynamic Bayesian network. Prediction approaches that rely on this technique are discussed next.

Modeling interactions with dynamic Bayesian networks

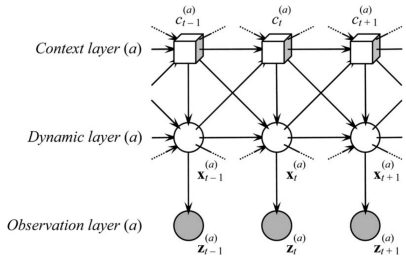
A dynamic Bayesian network (DBN) is a graphical model very similar to an HMM (subsection 2.2.3). The main difference between them is that in a DBN the hidden state is represented by a set of random variables (Murphy, 2002). HMMs and KFMs can be seen as particular instantiations of a DBN. DBNs offer a great flexibility to model the conditional dependencies between the different random variables representing the hidden state.

The modeling flexibility of DBNs has been taken advantage of in the IV literature to model the complex interactions between traffic participants. Gindele et al. propose a DBN to estimate the behaviors and trajectories of traffic participants in highway scenes (Gindele et al., 2010). A temporal slice of the proposed model is shown in Figure 3.2a. In this model, the local context of each vehicle (C) is modeled as a function of the state of all traffic participants (X). Examples of context information include the longitudinal distance and relative velocity to the vehicle ahead, and the lateral deviation in the lane. The context distribution is mapped into a discrete number of high-level situations (S), which are used to infer the expected behavior of all vehicles (B). They consider behaviors such as free riding, car-following or changing lanes to overtake. From the behaviors, the trajectory realizations (T) are produced. Inference in this probabilistic model is performed using particle filtering. To keep the approach tractable, several assumptions are made, namely that the behavioral decisions and trajectories of the involved vehicles are selected independent of each other. While in this work the decision rule for each vehicle is hand-engineered, a later work by the authors proposes to learn it from data using the EM algorithm (Gindele et al., 2013).

Agamennoni et al. propose a different probabilistic model to estimate the state and intentions of all traffic participants based on the same idea of extracting contextual information from the states (Agamennoni et al., 2012). In this model, whose corresponding graphical model is shown in Figure 3.2b, the discrete distribution over maneuvers of a vehicle (denoted as c_t in the figure) is determined based on the value of several hand-engineered feature functions extracted from the estimated states and maneuvers of all vehicles in the scene. Hence, in contrast to the model from Gindele et al., the motion of an agent is fully conditioned by the motion and intentions of surrounding traffic. The authors perform variational inference on this model to track interacting trucks in an opencast mine from GPS data. The features used for this task are fairly straightforward, comprising the TTC and the right-of-way. A DBN has also been proposed to model the motion of vehicles negotiating intersections (Lefevre et al., 2012). In contrast to the previous approaches, this work focuses on the early detection of dangerous traffic situations. Three hidden states are considered representing the physical state of the vehicles, their navigational intentions and their expected behavior. The expected behavior for each agent



(a) Temporal slice of the DBN proposed by Gindele et al. Dashed lines: temporal dependencies. Solid lines: causal dependencies. Square nodes: discrete. [Image credit: (Gindele et al., 2010)]



(b) Slice of the DBN proposed by Agamennoni et al. corresponding to a single traffic participant. Dotted lines connect to slices of surrounding vehicles. [Image credit: (Agamennoni et al., 2012)].

Fig. 3.2.: Examples of DBN models used to characterize the interactions between traffic participants.

is calculated using a gap acceptance model under consideration of the right-of-way traffic rules. The intended maneuvers of the vehicles are inferred using particle filtering, and compared to their expected behavior. If the intended and expected maneuver of a vehicle do not match, the situation is classified as dangerous and an alert is triggered.

The models based on DBNs constitute the most advanced motion prediction technique in the literature. Their flexibility to model the complex interactions of traffic participants with each other and with respect to the road network results in motion estimations of increased accuracy compared to those of physics- or maneuver-based prediction approaches. This is particularly true for long-term prediction horizons, where the consideration of vehicle interactions plays a major role. Furthermore, unlike model-based prediction approaches, they are also capable of predicting dangerous situations. However, all these properties come at a cost. Inference on these models is in general computationally expensive, which leads to a trade-off between prediction accuracy and real-time applicability, especially in complex traffic situations with many interacting vehicles.

3.3 Decision-making for intelligent vehicles

In this section, we review the state of the art on tactical decision-making for autonomous vehicles. The goal of the decision-making system is to find a sequence of maneuvers that enables the ego-vehicle to follow the global route provided by the strategic layer (see Figure 1.2). This sequence of maneuvers needs to be optimized with respect to the desired driving preferences encoded in the driver model. To carry out that optimization, the predictions of the intended motion of surrounding traffic are necessary. Hence, tactical planners are usually presented together with a model to predict the development of traffic scenes. The techniques used for selecting the maneuvers range from manual decision programming to complex probabilistic models that account for the uncertainties in the state and in the intentions of surrounding traffic.

3.3.1 Manual decision programming

Different systems for the behavior guidance of autonomous vehicles were presented already in the 1990's (Pomerleau, 1991; Dickmanns et al., 1994). The ALVINN system proposed by

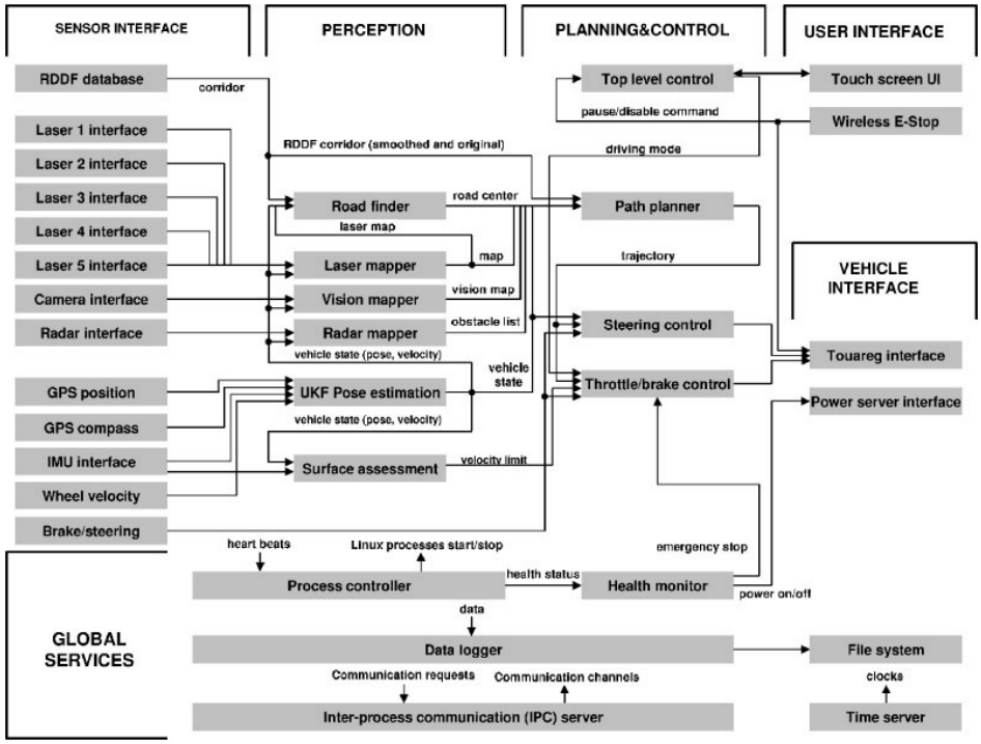


Fig. 3.3.: Software architecture of *Stanley*, the 2005 DARPA Grand Challenge winner [Image credit: (Thrun et al., 2006)]

Pomerleau was a reactive approach that would steer the autonomous vehicle to keep it on the road using an artificial neural network (Pomerleau, 1991). A more complex architecture was presented by Dickmanns et al., where a rule-based system would enable alternative maneuvers for the ego-vehicle (Dickmanns et al., 1994). However, the focus of these early approaches was not on tactical decision-making but on the development of the different components that integrate an autonomous vehicle architecture: computer vision for road and obstacle detection, obstacle detection and tracking, longitudinal and lateral controllers, computing architectures, etc.

The 2004 and 2005 DARPA Grand Challenges gave a new impulse to the research in the autonomous driving domain (Buehler et al., 2007). The goal of these contests was to develop an autonomous vehicle capable of navigating a desert off-road course with a length of 175 miles in less than 10 hours. Their main challenge lied on the perception systems for road finding and obstacle detection, as well as high-speed obstacle avoidance. In consequence, there was no need for complex tactical decision-making. Figure 3.3 shows the flowchart of the software system of Stanley, the winner vehicle of the 2005 DARPA Grand Challenge developed by Stanford University (in the 2004 edition no vehicle was capable of completing the course and there was no winner) (Thrun et al., 2006). In the flowchart, the estimated state of the vehicle, the detected obstacles and the characteristics of the road are directly passed to the path planner module, which in turn generates a feasible trajectory for the vehicle. No tactical reasoning about the most convenient maneuver for the ego-vehicle needs to be performed given the characteristics of the race.

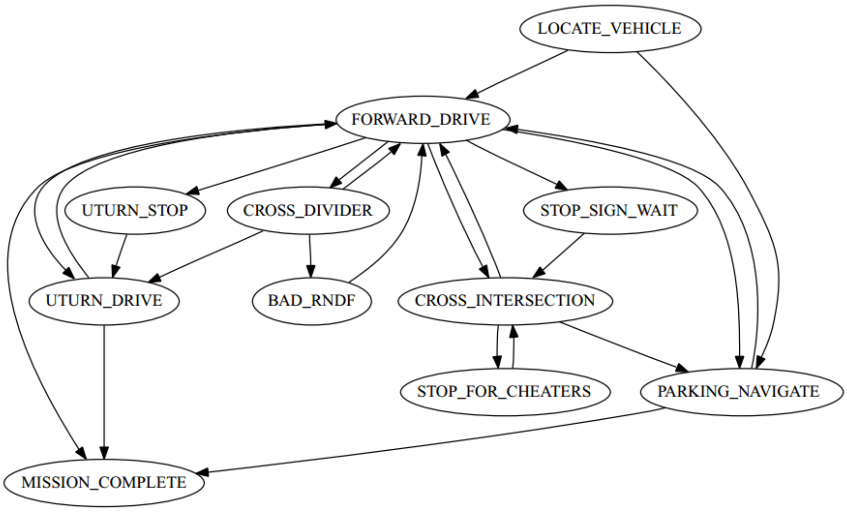


Fig. 3.4.: Finite State Machine for the tactical decision-making of Junior, second-ranked vehicle in the 2007 DARPA Urban Challenge [Image credit: (Montemerlo et al., 2009)].

The 2007 DARPA Urban Challenge changed radically the rules of the contest with respect to the preceding Grand Challenges (Buehler et al., 2009). The goal now was to have the autonomous vehicles navigate a mock urban environment, while respecting the traffic rules and interacting with other participants. The vehicles had to handle situations such as overtaking parked or slow vehicles, navigating intersections, merging into traffic, and doing U-turns when the road was blocked. This made it necessary for the participating teams to include tactical decision-making mechanisms to guide the behavior of the vehicle in such scenarios. The most common approach in this competition was to use deterministic Finite State Machines (FSM) (Gill, 1962). Informally, a deterministic FSM is given by a finite set of states in which an agent can be, and by the transitions between the states as a response to external stimuli. As an example, the states and transitions of the FSM that governed the behavior of *Junior*, Stanford University's entry to the contest, are shown in Figure 3.4 (Montemerlo et al., 2009). The initial state of the FSM is *locate-vehicle*, which involves the localization of the vehicle in the map. Once localized, the vehicle can start moving and the FSM can transition to the states *forward-drive* or *parking-navigate*, depending on its location. The state *forward-drive* corresponds to normal forward driving and lane keeping. If an obstacle is blocking the current lane, the FSM can transition to the *cross-divider* state, so that the vehicle crosses the yellow line to circumvent the obstacle. If all lanes are blocked, the FSM can instead transition to the *uturn-stop* state, and a U-turn maneuver will be initiated. Similarly, when reaching an intersection the FSM transitions to the *cross-intersection* state, in which the vehicle waits until it is safe to cross the intersection. Most of these transitions are based on a set of hand-coded rules and timeout values. For this reason, this kind of approach is also referred to as manual decision programming (Brechtel, 2015). Several decision-making approaches presented years after the DARPA Urban Challenge have continued to rely on FSMs (Ardelt et al., 2012; Ziegler et al., 2014b). Ardelt et al. propose a network of concurrent FSMs in which the states correspond to the lateral and longitudinal behaviors of an autonomous vehicle navigating highway scenarios (Ardelt et al., 2012). The transitions between states follow a two-step procedure. First, a

hand-tuned utility function is used to determine the probability that a particular transition is advantageous for the ego-vehicle. This involves considering the short-term consequences of the transition by predicting the development of the traffic situation using a constant velocity model. If a transition is found to be advantageous, a ruled-based system considering the worst-case scenario of the traffic scene is used to assess the feasibility of the transition. The proposed system can perform unsupervised automated lane changes and it is tested on an experimental platform under real traffic conditions.

Along the same lines, Ziegler et al. propose a hierarchical, concurrent FSM to generate the high-level behaviors of an autonomous vehicle (Ziegler et al., 2014b). One of the key components of the system is a detailed digital road map that contains lane segments, i.e. drivable sections of a lane. Each lane segment is annotated with information about itself (e.g. the presence of stop lines or signs), as well as information about its relation with other segments (e.g. priority rules). The map semantic information and the dynamic obstacle detections provided by the sensor layers are then used to determine the transitions in the FSM. The system was validated by having an experimental vehicle navigate autonomously the Bertha Benz Memorial Route, which has a length of 103 km and comprises city and rural environments.

The main advantages of manual decision programming using deterministic FSMs are their simplicity and transparency. However, this approach does not scale well when the complexity and number of possible traffic scenarios increases. In the DARPA Urban Challenge, the conditions were still far from those of real-life scenarios. For example, there were no occlusions in the intersections, the organizers frequently stopped vehicles to clear up traffic jams, and there were no other traffic participants like cyclists or pedestrians. Despite all of this, several collisions and multiple near-misses were reported (Fletcher et al., 2008). An example of one of the near-misses is shown in Figure 3.5. In this incident, the vehicle *Knight Rider* was stopped at the entrance of an intersection making no apparent progress. Immediately behind it, there were two human-driven vehicles from DARPA, and the vehicle *Skynet* from Cornell's University (Miller et al., 2008). Given the lack of progress, the rule-based system from *Skynet* decided to overtake the three stopped vehicles in order to proceed to the intersection... only to find itself facing the vehicle *XAV-250*, which had just turned right into Utah St. Both vehicles had to be stopped by the organizers to avoid a collision. The decision to overtake an apparently blocked vehicle might have been correct at any other part of the road network; however, it constitutes an unreasonable risk at the entrance of an intersection. This shows how, in practice, slight variations of the same problem might require different tailored solutions on a rule-based decision system. The decision diagrams become then increasingly complex and their transparency is reduced. An additional drawback of the presented approaches is that they typically handle the obstacle motion uncertainty by considering worst-case scenarios, which leads to extremely defensive driving behaviors of the robot.

3.3.2 Handling uncertainty

Ideally, the tactical planner should be able to cope with different sources of uncertainty. In the first place, existing ego-localization techniques for autonomous vehicles only provide approximate estimates of the state of the ego-vehicle in an environment. Moreover, the resulting

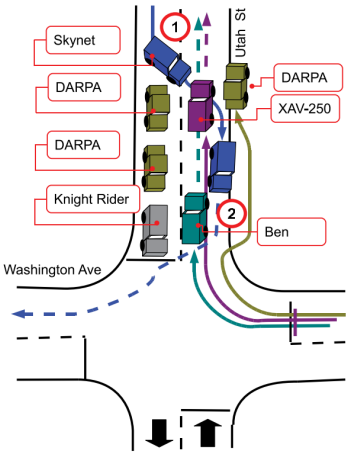


Fig. 3.5.: Skynet near-miss in the DARPA Urban challenge [Image credit: (Fletcher et al., 2008)].

state of a vehicle after performing a maneuver cannot be predicted with certainty since the dynamic model and the state space representation do not exactly represent the real-world. Secondly, the vehicle's sensors only provide noisy measurements of the surrounding traffic, leading to uncertain obstacle state estimations. Furthermore, several aspects of their state, like the intentions, cannot even be observed. As a consequence, the future development of traffic situations can only be predicted stochastically, which complicates the tactical planner's job. Existing tactical decision-making approaches in the literature differ significantly in the assumptions that they make concerning uncertainty. They range from completely ignoring uncertainty, like the FSM approaches from the previous section, to POMDP models that consider uncertainty in the controls, states, and even the intentions of the traffic participants. Because reasoning in the presence of uncertainty is generally computationally expensive, the following approaches need to rely on simplifying modeling assumptions to remain tractable. For IV applications, the simplifications are usually applied in the technique used to model the interactions between agents.

In contrast to rule-based decision-making approaches, Schubert proposes to use decision networks in the context of a lane change assistant system under partial observability of the state of surrounding traffic (Schubert, 2012). Decision networks (also known as influence diagrams) are an extension of Bayesian networks that, in addition to random variable nodes, also contain utility and decision nodes (Barber, 2012). They can be used for decision-making by selecting the decisions in the decision nodes that lead to the highest expected utility. Schubert proposes a single decision node with lane keeping and lane change maneuvers, and multiple discrete random variable nodes describing the state of the environment (lane marking types, occupancy of the lanes and feasibility of lane change maneuvers). The main drawback of this approach is that it requires tuning the value of the utility node for each possible combination of decisions and random variable states. Furthermore, the approach only plans a single decision (as opposed to a sequence of decisions) and it does not consider any long-term implications.

To plan foresighted sequences of maneuvers under uncertainty several approaches in the literature rely on MDP formulations. Brechtel et al. propose to use a semi MDP (sMDP) to plan sequences of lane change maneuvers in simulated two-lane highway scenarios (Brechtel

et al., 2011). A sMDP generalizes MDPs to allow actions that take varying amount of times to complete. The transition probabilities of the sMDP are modeled using a DBN that accounts for the interactions between vehicles (Gindele et al., 2010). A hard-coded reward function that, over all things, penalizes crashes is used to guide the planning. Each time the DBN-based transitions lead to new states, the discrete state space is grown and the policy recalculated using value iteration. In consequence, each time a new traffic situation is encountered (for instance, changing number of vehicles), the policy is recalculated. This prevents the applicability of this approach in an online setting. Another complication of this approach is finding an appropriate discretization of the state space.

To cut down the complexity of the highway navigation task, Sonu et al. propose a hierarchical MDP approach with two levels (Sonu et al., 2018). The upper-level MDP has a roughly discretized state space, with very limited maneuvers (lane changes and lane keeping) and no obstacles. This upper-level MDP is solved offline using value iteration. In contrast, the lower-level MDP is solved online using Monte Carlo Tree Search (MCTS) with Double Progressive Widening (DPW). Its continuous state space is constructed as a function of the first action of the upper-level policy. Only the traffic that will directly interact with the ego-vehicle during the execution of that action will be represented in the state space, significantly reducing thus the complexity. The action space of the lower-level MDP is richer, containing discrete longitudinal accelerations and lateral velocities. The actions of surrounding vehicles are deterministically predicted using the Intelligent Driver Model (IDM) (Kesting et al., 2010) and the Minimizing Overall Braking Induced by Lane Changes (MOBIL) model (Treiber and Kesting, 2009). A possible complication of this approach is that the obstacle-agnostic, upper-level MDP might lead the solution towards low-reward regions from where the lower-level plan could not escape.

Despite the assumptions from the MDP-based approaches, the state of surrounding traffic cannot always be exactly observed and this should be taken into account during planning. Ulbrich and Maurer propose to model the highway lane-changing decision task using a small POMDP with 8 possible states (Ulbrich and Maurer, 2013). The states correspond to binary random variables representing whether a lane change is possible, beneficial, or currently in progress. The observation of these states is modeled with a series of complicated heuristics and mathematical operations. The action space contains only three actions, namely, lane keeping, starting a lane change to the left, and aborting a lane change. The POMDP is solved online using a branch-and-bound POMDP solver—the RTBSS algorithm (Paquet et al., 2005). Everything in this approach is hand-engineered, from the complex observation model to the reward and transition models. In consequence, it is not clear how this approach could be adapted to plan more complex sequences of maneuvers.

Instead of discretizing the state using heuristics, Brechtel et al. propose to automatically and iteratively learn a low-level, discrete state-space representation that minimizes the error of the value function representation (Brechtel et al., 2014; Brechtel et al., 2013). This approach is applied to planning in intersection merging scenarios, for which a policy is learned offline using value iteration. The interaction-aware dynamics of all vehicles are modeled using a DBN

(Gindele et al., 2010). However, the uncertainty in the intentions of surrounding vehicles is not explicitly considered.

In contrast, Bandyopadhyay et al. propose to consider the uncertainty only in the intentions of surrounding traffic participants (Bandyopadhyay et al., 2013). In particular, they address an scenario in which the ego-vehicle circulates in the presence of pedestrians. The intentions of the pedestrians are modeled as target destinations. The pedestrians' motion is therefore dependent on their intended destination and on the state of the ego-vehicle, and predicted using a pedestrian behavioral model (Helbing et al., 2005). The planning task is modeled as a Mixed Observability MDP (MOMDP) (Ong et al., 2010), a framework in-between MDPs and POMDPs in which only part of the state is partially observable—the pedestrians intentions in this case. The MOMDP is solved using SARSOP, a point-based POMDP solver (Kurniawati et al., 2008). In this approach, the interactions between pedestrians are not considered. The authors suggest solving a different MOMDP for each pedestrian in the scene, and choose the most conservative action for the ego-vehicle. In consequence, this approach cannot be directly applicable to highway driving where the behavior of traffic participants is highly correlated.

Liu et al. propose instead to infer the motion of traffic participants by exploiting the road context (Liu et al., 2015). The road is discretized into a set of mutually exclusive regions, each of them with associated transition probabilities to other regions and with associated velocity profiles. This information is exploited to classify the motion intentions of surrounding obstacles into stopping, hesitating, normal and aggressive. The situation-aware decision making problem is modeled as a POMDP to handle the uncertainties in the motion intention and the perception noise. The action space for the ego-vehicle contains a discrete set of longitudinal accelerations. The POMDP is solved in an online fashion using the DESPOT solver and evaluated on simulated T-junction and roundabout scenarios. A drawback of this approach is that the interactions between vehicles are only considered in the motion predictions through a series of hard-coded rules specific for intersection scenarios.

Similarly, Bouton et al. also propose a POMDP planner for the task of navigating unsignaled intersections (Bouton et al., 2017). In this approach, the path of the ego-vehicle is assumed given by a high-level path planner and the goal is to compute the acceleration profile along this path. The state is factored into the physical state of the ego-vehicle (pose, velocity, acceleration), the physical state of all other vehicles, and their maneuver intentions. The POMDP is solved online using the Monte-Carlo-based algorithm POMCP with double progressive widening to control the branching factor. At each execution step, a Gaussian belief over the physical states of the obstacles and over their intentions is maintained using an IMM with two modes, CA and CV. This is the same model that is used to simulate the dynamics of the system and constitutes the main drawback of this approach. As discussed in subsection 3.2.1, the IMM completely disregards the interactions between vehicles. In consequence, the search tree is built with observation samples that do not capture the essence of human driving, which might explain some of the collision rates this approach achieved in simulated scenarios. POMCP with double progressive widening has also been applied to the task of highway navigation (Sunberg et al., 2017). In this case, the behavior of surrounding traffic is modeled using the IDM (for lane

Reference	Uncertainty			Discretization			Offline/ Online	Bi-directional interactions	Approach
	C	S	I	S	A	O			
(Schubert, 2012)	✗	✓	✗	D	D	-	Offline	✗	Decision network
(Brechtel et al., 2011)	✓	✗	✗	D	D	-	Online	✓ DBN	sMDP, VI
(Sonu et al., 2018)	✓	✗	✗	C	D	-	Online	✓ IDM + MOBIL	Hierarchical MDP, MCTS
(Ulbrich and Maurer, 2013)	✗	✓	✗	D	D	D	Online	✗	POMDP, RTBSS
(Brechtel et al., 2014)	✓	✓	✗	C	D	C	Offline	✓ DBN	POMDP, MCVI
(Liu et al., 2015)	✓	✓	✓	D	D	D	Online	✓ hard-coded	POMDP, DESPOT
(Bandyopadhyay et al., 2013)	✓	✗	✓	D	D	D	Offline	✓ pedestrian model	MOMDP, SARSOP
(Sunberg et al., 2017)	✓	✗	✓	C	D	C	Online	✓ IDM + MOBIL	POMDP, POMCP-DPW
(Bouton et al., 2017)	✓	✓	✓	C	D	C	Online	✗	POMDP, IMM + POMCP
Proposed (chap. 7)	✓	✓	✓	C	D	C	Online	✓ DBN + driver mod.	POMDP, DBN + POMCP

Tab. 3.1.: Tactical decision-making approaches in the literature that account for uncertainties. The discretization column shows whether the state (S), action (A) and observation (O) spaces are discretized (D), or not (C). The uncertainty can be considered in the controls (C), states (S) or intentions (I). An approach considers bi-directional interactions if the ego-vehicle actions affect surrounding vehicles and vice-versa.

keeping) and MOBIL (for lane changes) models. The only uncertainty considered is in the parameters of the models, which have a direct incidence in the aggressiveness of the drivers. The performance of the approach is compared to those of an omniscient planner that has access to the true obstacle models, and an MDP planner that assumes a single normal model for all vehicles. Although based on a simple syntethic domain, this approach shows the benefit of considering the uncertainty in the behavior of surrounding traffic.

Table 3.1 summarizes the main characteristics of the discussed approaches, along with those of the decision-making approach that is presented in chapter 7. The first difference among them resides in the uncertainties considered. The most complete models consider the uncertainty in the dynamics, state, and intentions of surrounding traffic. Taking into account the uncertainty in the future behavior of surrounding vehicles is of particular importance in the design of safe planning systems for IVs. Another aspect in which they differ is in their offline or online nature. An offline planner learns the sequences of maneuvers that should executed for all possible traffic situations during an offline training phase. Given the large number of possible traffic situations, this task is normally intractable unless some domain knowledge is used to reduce the size of the state space. In contrast, online approaches try to find the best possible maneuver to be executed in the current driving situation. However, they have only a limited time to reason about the future consequences of each maneuver. The discretization of the state space is also an important differentiating characteristic. In general, it is very complicated to find an adequate state discretization for the driving task. Besides that, discretizing the state space usually induces unwanted problems in the planner such as aliasing. Brechtel et al. deals with this problem by automatically learning a discrete state-space representation using machine learning (Brechtel et al., 2014). Bouton et al. and Sonu et al. opt instead to sample the state and observation spaces while constructing the search tree (Bouton et al., 2017; Sonu et al., 2018). We adopt a similar technique in our decision-making approach. Finally, only by rigorously considering the interactions between traffic participants can we accurately assess the viability and safety of a tactical plan. In subsection 3.2.3, we showed how DBNs are among the most advanced techniques to model the interdependent motion of traffic participants. However,

inference in these models in generally expensive. That is why some of the discussed approaches rely on simpler interaction models or even task-specific, hard-coded interaction rules. In our tactical planner, the interactions up to the mid-term future are accurately predicted using a DBN. However, to ease the computation burden, the long-term developments of traffic scenes are estimated using a lightweight model-based prediction.

Part II

Proposed Models and Algorithms

4 Modeling Driver Behavior From Demonstrations

4.1 Introduction and related work

The design of cost functions to model the behavior of drivers is a common topic in the Intelligent Vehicles research domain. Traditionally, the process involves the design of each component of the model using background knowledge (Wolf and Burdick, 2008), as well as finding a balance between components that appropriately captures the desired behavior. This last step is typically addressed by hand-tuning the model parameters (Wolf and Burdick, 2008; Montemerlo et al., 2009; Wei and Dolan, 2009), which becomes an increasingly difficult task as the complexity of the model grows. Moreover, different scenarios usually require different behavioral models, making the hand-tuning a recurring operation. To avoid these problems, recent efforts learn the model automatically from demonstrated driving data using IRL for applications as diverse as outdoor autonomous navigation (Silver et al., 2008), risk anticipation on residential roads (Shimosaka et al., 2014), highway trajectory planning (Kuderer et al., 2015), parking lot navigation (Abbeel et al., 2008) and communicative human-aware autonomous driving (Sadigh et al., 2016).

Traditional IRL approaches require solving the forward control problem to obtain a policy in the inner loop of an iterative optimization algorithm. A similarity metric between the demonstrations and the policy is then computed and used to determine if an appropriate model has been found. In continuous domains, such as motion planning for autonomous vehicles, finding the optimal policy is intractable. Existing approximations propose to solve instead a local control problem (Levine and Koltun, 2012), construct a graph-based representation of the state-space (Byravan et al., 2015; Okal and Arras, 2016) or find instead a set of representative trajectories of the model using path-planning techniques such as A* (Ratliff et al., 2006a), spline optimization (Kuderer et al., 2015), Rapidly-exploring Random Trees (Shiarlis et al., 2017; Pérez-Higueras et al., 2018), or Gaussian process sampling (Lee and Seo, 2017). In dynamic environments, even finding a single trajectory can become a challenging task. As a consequence, in the particular case of driver behavior modeling most existing approaches focus on static environments (Shimosaka et al., 2014; Shimosaka et al., 2017; Shiarlis et al., 2017).

In this chapter, we propose to integrate the IRL paradigm with a motion planner based on conformal spatiotemporal state lattices (McNaughton et al., 2011). State lattices provide a search space of dynamically feasible actions that can additionally be conformed to the structured environment of public roadways (Howard et al., 2008). This has the immediate advantages of limiting the size of the search space and producing trajectories that satisfy the kinematic constraints of the vehicle. By using the spatiotemporal variant, which includes time in the state-space, we sacrifice the optimality of the solution in exchange for increased awareness about the surrounding traffic. We hypothesize that this planning variant resembles more closely

the behavior of human drivers and that in consequence, the resulting trajectories constitute a better evaluation metric for the model. The proposed approach is experimentally validated by successfully modeling the highway driving task from suboptimal driving demonstrations gathered with an instrumented vehicle.

4.2 Preliminaries

In this section, we first state the Markov decision process formulation of IRL for the specific task of learning driver behavioral models. In particular, we focus on learning a cost function that encodes the behavior of the demonstrated driving data. While in the reinforcement learning domain it is common to work in terms of rewards, the reformulation to learn a cost function is straightforward since costs can be seen as negative rewards. After that, we present the trajectory planning approach based on spatiotemporal state lattices that we use to efficiently apply IRL in continuous, dynamic environments.

4.2.1 Inverse reinforcement learning

Generally speaking, the goal of IRL is to undercover the preferences behind exhibited behaviors (Ng and Russell, 2000). The task is typically framed in the MDP framework. For the case of highway driving, an MDP can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \{P_{sa}\}, \mathcal{C}, \gamma \rangle$, where:

- A state $s \in \mathcal{S}$ is given by the physical states of all vehicles involved, that is, $s_t = [\mathbf{x}_t^e, \mathbf{x}_t^{1:N}]$ where the superscript e indexes the ego-vehicle and we have assumed the presence of N obstacles in the scene. A physical state $\mathbf{x} = [x, y, \psi, v]^T \in \mathbb{R}^4$ contains the longitudinal (x) and lateral (y) positions of the vehicle's center of mass in road coordinates, its heading (ψ), and its velocity (v). We denote by $\mathcal{X}$ the joint physical states of the vehicles in the scene. Additionally, for each vehicle in the scene, we assume that its desired velocity $v_\bullet$ is known. This value can be set for the ego-vehicle to the speed limit, and for the obstacles to their highest velocity recorded since they are in sensor range.
- An action a contains the ego-vehicle's steering wheel angular velocity and its acceleration command, that is, $a \in \mathbb{R}^2$.
- $P_{sa}(\cdot)$ is the state transition probability upon taking action a in state s.
- $\mathcal{C} : \mathcal{S} \mapsto \mathbb{R}$ is the cost function.
- $\gamma \in [0, 1)$ is the discount factor.

In this chapter, we will compactly represent the described continuous state and action spaces using a state-lattice, which exploits the structure of the environment and the non-holonomic motion constraints of the vehicle.

A policy is defined as any map $\pi : \mathcal{S} \to \mathcal{A}$. The value function for a policy π, can be evaluated at any arbitrary state $s \in \mathcal{S}$ as follows:

$$V_\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \mathcal{C}(s_t) \mid \pi, s_0 = s] \tag{4.1}$$

Solving an MDP implies finding the policy π that minimizes V_π, that is, for any arbitrary state $s \in \mathcal{S}$ we have that:

$$\begin{aligned} V^*(s) &= \min_{\pi} \; V_\pi(s) \\ &= \mathcal{C}(s) + \min_{a \in \mathcal{A}} \; E_{s' \sim P_{sa}(\cdot)}[V^*(s')] \end{aligned} \tag{4.2}$$

where notation $s' \sim P_{sa}(\cdot)$ means the expectation is with respect to s' distributed according to $P_{sa}(\cdot)$.

IRL addresses the case where the cost function $\mathcal{C}$ is not known, but instead we have access to a set of expert, possibly suboptimal, demonstrated trajectories $\mathcal{D} = \{\xi^{[1]}, \xi^{[2]}, \ldots, \xi^{[m]}\}$. A trajectory is defined as a sequence of states of a vehicle $\xi = \{s_1, \ldots, s_T\}$. We assume that the cost function can be fully specified as a linear combination of features:

$$\mathcal{C}(s) = \mathbf{w}^T \mathbf{f}(s) \tag{4.3}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_K)^T$ is the unknown weight vector and $\mathbf{f}(s) = (f_1(s), \ldots, f_K(s))^T$ is the feature vector that parameterizes state s, both of dimension K. We shall use short-hand notation $\mathbf{f}_\xi$ to denote the sum of features along any arbitrary trajectory ξ:

$$\mathbf{f}_\xi \doteq \left[\sum_{t=1}^{T} \mathbf{f}(s_t) | \xi = (s_1, \ldots, s_T) \right] \tag{4.4}$$

The goal of IRL is then to find the weight parameters $\mathbf{w}$ for which the optimal policy, obtained by solving the corresponding planning problem, would achieve similar trajectories to those demonstrated according to a given statistic. Abbeel and Ng propose to use the feature expectations as the similarity metric (Abbeel and Ng, 2004). Unfortunately, this is an ill-posed problem, as many different weights can make the demonstrated behavior optimal. An additional complication is that, in real-world applications, the demonstrations are typically suboptimal.

The Maximum Entropy IRL framework addresses both problems by modeling expert behavior as a distribution over trajectories and applying the principle of Maximum Entropy to select the one that does not exhibit any additional preferences beyond matching feature expectations (Ziebart et al., 2008). Under this model, the probability of a trajectory is proportional to the negative exponential of the costs encountered along the trajectory:

$$P_{\mathbf{w}}(\xi) = \frac{1}{Z(\mathbf{w})} \exp(-\mathbf{w}^T \mathbf{f}_\xi) \tag{4.5}$$

Let $L(\mathbf{w})$ be the likelihood function of $\mathbf{w}$ given observed trajectories. To find the parameter vector $\mathbf{w}^*$ that optimizes $L(\mathbf{w})$, one can rely on its gradient:

$$\nabla L(\mathbf{w}) = \tilde{\mathbf{f}} - \mathbb{E}_{\xi \sim P_{\mathbf{w}}(\cdot)}[\mathbf{f}_\xi] = \tilde{\mathbf{f}} - \sum_{s \in \mathcal{S}} \rho_{\mathbf{w}}(s) \mathbf{f}(s) \tag{4.6}$$

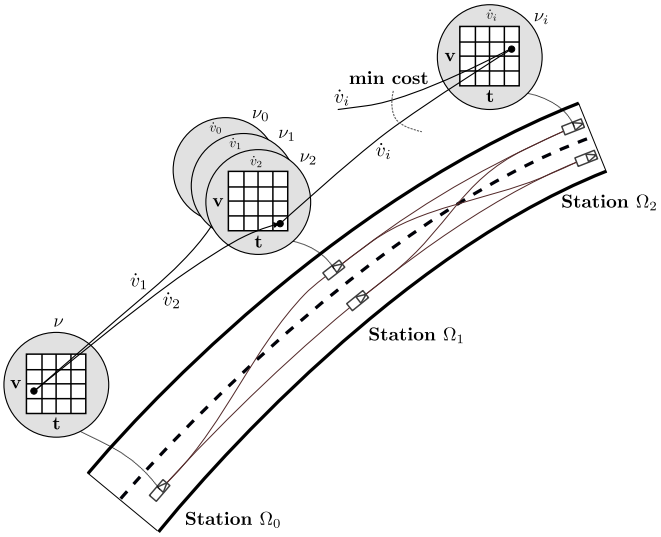


Fig. 4.1.: Structure of the spatiotemporal state lattice. Road-aligned states are sampled at regular longitudinal intervals on each lane and connected with feasible paths using parametric optimal control. The spatiotemporal planner searches for the best trajectories in the lattice and keeps the problem tractable by pruning the search space (McNaughton et al., 2011).

where $\tilde{\mathbf{f}} \doteq \frac{1}{M}\sum_{i=1}^{M} \mathbf{f}_{\xi^{[i]}}$ represents the average sum of features of the demonstrations and $\rho_{\mathbf{w}}$ denotes the expected state visitation frequencies under parameter vector $\mathbf{w}$.

To obtain the expectation in equation 4.6, the calculation of the full policy is required, which is not tractable in continuous or large domains. A common workaround is to compute, at each optimization step, a set of model-dependent trajectories and use them to approximate the expectation. This can be done using a variety of planners such as RRT* (Perez-Higueras et al., 2018), spline-based trajectory optimization (Kuderer et al., 2015) or Gaussian process sampling (Lee and Seo, 2017). In dynamic environments, finding a trajectory that minimizes the cost can be particularly challenging. In this chapter, we propose to use a conformal spatiotemporal lattice, which is described next.

4.2.2 Conformal spatiotemporal state lattices

State lattices provide a concise representation of continuous state-spaces using a discrete search graph of dynamically attainable states (Pivtoraiko and Kelly, 2005). They are a very convenient approach for path planning in structured environments as they can be conformed to the environment by applying a state-based sampling strategy (Howard et al., 2008). Sampled states can then be connected using an inverse path generator (Kelly and Nagy, 2003). A path, denoted τ, is defined here as a continuous curve through a 4-dimensional state-space $[x, y, \psi, \kappa]$ connecting two endpoint states; x represents the longitudinal position along the road, y the lateral position, ψ the heading, and κ the curvature or rate of change of the heading. In Figure 4.1, we show the paths that connect sampled feasible vehicle states along the road.

The state lattice graph is built over the states of the MDP, which is an idea already explored in the RL literature (Guestrin and Ormoneit, 2001; Neumann et al., 2007; Okal and Arras, 2016). The state space of the underlying MDP is 7-dimensional. Each state s is a tuple $(x, y, \psi, \kappa, t, v, \dot{v})$, where t denotes time, v velocity, and $\dot{v}$ acceleration. The nodes of the graph

Algorithm 1: Spatiotemporal trajectory planning.

input : $\mathbf{w}$: weight vector
T: duration of demonstrated trajectory
$\dot{\mathbf{v}}$: available discrete acceleration controls
s_0: initial state of demonstrated trajectory
N_T: number of desired trajectories

output : Best N_T trajectories found $\xi_1^*, \ldots, \xi_{N_T}^*$

```
SpatioTempPlan
    foreach station Ω_i do
        foreach vertex ν_j do
            foreach node s_k do
                if ĉ(s_k) ≠ ∞ then
                    foreach v̇ ∈ v̇ do
                        Construct edge e
                        Determine edge end node s_end
                        Evaluate c(e) using (4.7)
                        Get node s_old at cell of arrival
                        if ĉ(s_k) + c(e) < ĉ(s_old) then
                            s_old ← s_end
    Find final nodes s*_1, ..., s*_{N_T} such that:
        ĉ(s*_1) ≤ ··· ≤ ĉ(s*_{N_T}) ≤ ĉ(s)  ∀s ∉ (s*_1, ..., s*_{N_T}) and s[t] ≥ T
    return Backtrack trajectories ξ*_1, ..., ξ*_{N_T}
```

are samples from the state space. The edges correspond to the actions. An edge (or local trajectory) represents the traversal of the path between two nodes with a given acceleration value, i.e. $e_j : [t_j^s, t_j^e] \mapsto \mathbb{R}^6$ with $j \in \{1, 2, \ldots, |E|\}$. The number of edges $|E|$ is determined by the number of paths in the lattice and the number of discrete acceleration actions available. Within this lattice framework, a full trajectory ξ is a sequence of edges from a start to a goal state. Due to the different lengths and durations of the edges, the lattice graph represents a deterministic semi-Markov Decision Process. We will refer to this deterministic sMDP as a graph-MDP. The cost of traversing an edge e_j is then given by

$$c(e_j) \doteq \int_{t_j^s}^{t_j^e} \gamma^{t-t_j^s} \mathcal{C}(s_t = e_j(t)) dt \tag{4.7}$$

In practice, this integral can be approximated by sampling states along the edge.

To handle highly-dynamic environments, we have considered time as part of the state-space. This renders the exact solution of the graph-MDP using traditional dynamic programming methods intractable. Instead, we solve the trajectory planning problem using the approach presented by McNaughton et al., which is based on exhaustive search combined with a pruning strategy that keeps the size of the search space within feasible limits (McNaughton et al., 2011).

The key idea of this approach in order to attain tractability is to specify the time-dependent components of the graph (that is, the velocity and time components of the nodes) online as the search proceeds. In other words, the search graph is dynamically built during the search. Let us

define a vertex ν as the set of all nodes with the same $[x, y, \theta, \kappa, \dot{v}]$ components. Additionally, a station Ω is defined as the set of all vertices whose states' x component is equal. For all vertices, a discrete range of threshold times and velocities is specified, determining a grid-like structure (see Figure 4.1). For each vertex, only a single node is allowed to exist per cell, which efficiently prunes the search space.

Typically, trajectory planners search only for the best trajectory. In contrast, our goal here is to approximate the expectation in equation 4.6 and therefore we are interested in finding the best N_T trajectories induced by the model parameters. The search for the best trajectories is based on maintaining the minimum cost $\hat{c} : \mathcal{S} \mapsto \mathbb{R}$ that requires traveling from the start node s_0 to any other node in the graph. Initially, this value is set to infinity for all nodes other than the start node. The search begins from the start node and proceeds recursively along the stations, as specified in Algorithm 1. For each node in a given station, the outgoing edges (which are determined by the paths of the lattice and the allowed accelerations $\dot{\mathbf{v}}$) are created and evaluated. The acceleration followed during each edge determines the vertex to which the end node is associated. The time and velocity components of the end node determine the cell within the vertex to which it is assigned. If another node (s_{old} in Algorithm 1) had already been assigned to the same cell, the one with the lowest accumulated cost between the two is kept. The search continues until the final station is reached. In general, one of the challenges of searching with this spatiotemporal lattice is selecting the final nodes. For our IRL learning purpose this becomes a trivial task: we select the nodes with the lowest accumulated costs and with an associated time beyond the threshold time T, which is given by the duration of the corresponding demonstrated trajectory. We detail this in the next section. The best trajectories $\xi_1^*, \ldots, \xi_{N_T}^*$ are found by backtracking until the initial node.

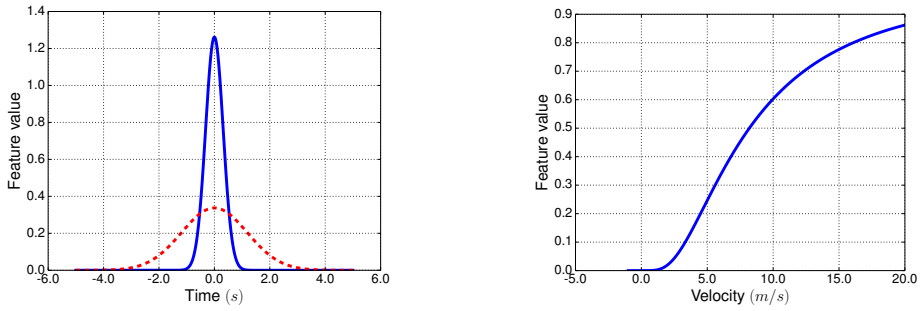
4.3 Modeling driver behavior from demonstrations

In this section, we detail how to put together the MaxEnt IRL framework with the spatiotemporal lattice planner in order to learn driver behaviors from demonstrations in dynamic environments. In the first place, we will list the features that we have selected to compose the cost function. Then, we describe the proposed algorithm.

4.3.1 Feature selection

In this chapter, we focus on learning highway driver behavioral models from demonstrated driving data. These models will be exploited in later chapters to predict the expected, risk-averse behavior of highway drivers and to guide the tactical planner of the ego-vehicle. The capacity of the models to capture any behavior will always be constrained by the composing features and the shape of the function. The following features have been selected using background knowledge:

Lane: this is a binary mutually-exclusive feature aimed to model the preference of the driver to stay on a given lane. Typically, a driver will remain on the right-most lane unless an overtake needs to be performed.



(a) Gaussians used to parameterize the time-to-collision and time-headway features

(b) Function used to parameterize the feature associated with the deviation from desired velocity.

Fig. 4.2.: Continuous features selected for the time-to-collision, time-headway, and desired velocity deviation signals.

Time-to-collision (TTC): this is an indicator of highly dangerous states. It is defined as the remaining time until a collision occurs if two vehicles continue on the same course and at the same speed. We model it with two Gaussian distributions: a narrow one that aims to capture the high cost associated to low TTC values, and a wider one that represents mildly dangerous states. This feature is considered both for the front and the back in the lane of the corresponding state of the vehicle, accounting for a total of four features. The one-dimensional Gaussians used are shown in Figure 4.2.

Time-headway (TH): the time-headway indicates potentially dangerous situations. It is defined as the time elapsed between the back of the lead vehicle passing a point and the front of the following vehicle passing the same point. It can be seen as the equivalent of distance when driving at high speeds. We model it similarly as the TTC.

Deviation from desired speed: this feature models the cost that is paid when the vehicle is forced to deviate from the desired speed, which we set to the speed limit. We model it with the cumulative distribution function of the inverse-gamma (Figure 4.2b). For low deviations, the feature is not activated and no cost is paid. Beyond a given threshold, the value of the feature rises sharply and with it the associated cost. This should force the vehicle to overtake if it is safe.

Acceleration: we consider the absolute value of the acceleration as a feature to discourage strong accelerations or decelerations.

Distance: the distance traveled along the road is considered as a feature with a negative weight value in order to encourage the vehicle to make progress.

4.3.2 Inverse reinforcement learning using spatiotemporal state lattices

As we have introduced in subsection 4.2.1, we approximate the feature expectations in Equation 4.6 with the average of the features associated to the best N_T trajectories found using the spatiotemporal planner. This enables us to calculate the approximated gradient and perform gradient-based optimization.

Algorithm 2: IRL algorithm for driver behavior modeling in dynamic environments using spatiotemporal state lattices.

input : $\mathcal{D}$: set of M demonstrated trajectories
$\mathbf{w}_0$: initial weight parameters
δ: optimization step-size
$\dot{\mathbf{v}}$: available discrete acceleration controls
N_T: number of trajectories for approximation

output : optimized weights $\mathbf{w}^*$

IRLdriverModeling
- $\mathbf{w} \leftarrow \mathbf{w}_0$
- $\Xi_{\tilde{\mathbf{f}}} \leftarrow \frac{1}{M} \sum_{\xi \in \mathcal{D}} \mathbf{f}_\xi$ (avg. demonstrated sum of features)
- **while** *not converged* **do**
  - $\Xi_{\mathbf{f}_{\xi^*}} \leftarrow 0$
  - **foreach** $\xi_i \in \mathcal{D}$ **do**
    - **Determine road waypoints**
    - **Construct state lattice** (Fig. 4.1)
    - $\xi_1^*, \ldots, \xi_{N_T}^* \leftarrow \text{SpatioTempPlan}(\mathbf{w}, T(\xi_i), \dot{\mathbf{v}}, s_0(\xi_i), N_T)$
    - $\Xi_{\mathbf{f}_{\xi^*}} \leftarrow \Xi_{\mathbf{f}_{\xi^*}} + \frac{1}{N_T} \left(\mathbf{f}_{\xi_1^*} + \cdots + \mathbf{f}_{\xi_{N_T}^*} \right)$
  - $\nabla_{\mathbf{w}} \leftarrow \frac{1}{M} (\Xi_{\tilde{\mathbf{f}}} - \Xi_{\mathbf{f}_{\xi^*}})$
  - $\mathbf{w} \leftarrow \mathbf{w} - \delta \nabla_{\mathbf{w}}$
- **return** $\mathbf{w}$

The whole process is described in Algorithm 2. We set the initial weights $\mathbf{w}_0$ as an input to the algorithm, as they can be initialized using background knowledge. Additionally, a set of discrete valid acceleration actions $\dot{\mathbf{v}}$, the optimization step-size δ, and the number of trajectories that will be used to approximate the expected features need to be specified. First, the average sum of features of the demonstrations $\Xi_{\tilde{\mathbf{f}}}$ is calculated. This feature vector is assumed to encode the general behavior encoded in the demonstrations. Then, and until convergence, the algorithm repeatedly iterates over the demonstrated trajectories $\mathcal{D}$ to optimize the weight vector $\mathbf{w}$. For each demonstration, a state lattice is deployed adapted to the shape of the road. Starting from the initial state of the demonstrated trajectory, and with the threshold time set to the last time step in the demonstration, we obtain the best N_T trajectories $\xi_1^*, \ldots, \xi_{N_T}^*$ with Algorithm 1. These trajectories are obtained in an obstacle-aware manner and, in consequence, they can be used in the evaluation of the current estimate of the model. The average sum of features of these N_T trajectories approximate the expected sum of features in Equation 4.6. However, we do not update the weight parameters for each individual demonstrated trajectory. Instead, we accumulate the approximated expected sum of features in the container variable $\Xi_{\mathbf{f}_{\xi^*}}$ and calculate the gradient only after all demonstrated trajectories have been processed. In our experience, this leads to smoother optimizations and also follows previous work (Kuderer et al., 2015; Pérez-Higueras et al., 2018). Finally, once the value of the weight parameters has converged, the algorithm terminates.

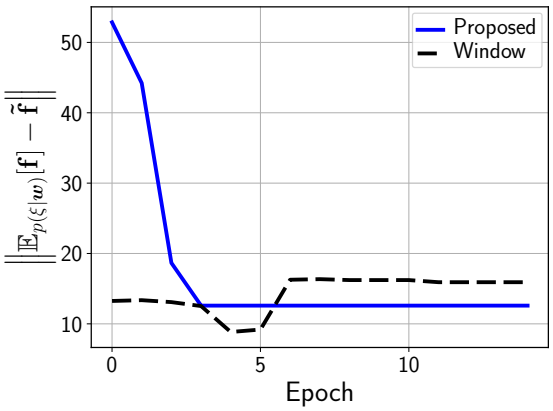


Fig. 4.3.: Convergence of the different learning algorithms considered.

4.4 Experimental evaluation

We evaluate our driver modeling approach based on IRL with spatiotemporal state lattices using demonstrated driving data gathered on a French two-lane highway. We first present the dataset used and provide details about the tested lattice configuration; then we discuss the evaluation metrics and the competing approach against we compare; finally, we present the qualitative and quantitative results obtained.

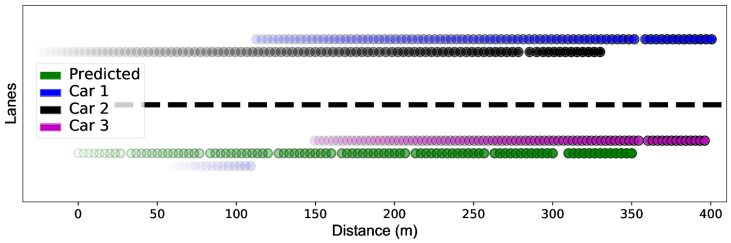
4.4.1 Dataset and lattice configuration

As training data, we use $170\,\mathrm{s}$ of annotated highway driving data. It consists of a variety of traffic scenes with an average duration of $10\,\mathrm{s}$, including lane changes in both directions, getting stuck behind slow traffic and free driving. The tracking of the obstacles was performed with a grid-based tracker (Rummelhard et al., 2015), but only the lane in which they circulate was annotated, i.e. there is no lateral in-lane position information. This is also true for the ego-vehicle performing the demonstrations. This motivated us to setup the lattice as seen in Figure 4.1, with the nodes of the lattice always at the center of the lanes. The longitudinal separation between stations was set to $50\,\mathrm{m}$, the allowed accelerations were 9 equidistant discrete values in the range $[-2, 2]\ \mathrm{m/s^2}$. The threshold values that determine the composition of the grid of subnodes were set to 6 equidistant values in the interval $[25, 38]\ \mathrm{m/s}$ for the velocity, and the range between $0.5\,\mathrm{s}$ and the maximum time step in the demonstration sampled every $0.5\,\mathrm{s}$ for the time dimension. This defines a much finer grid than the one proposed in the original paper (McNaughton et al., 2011).

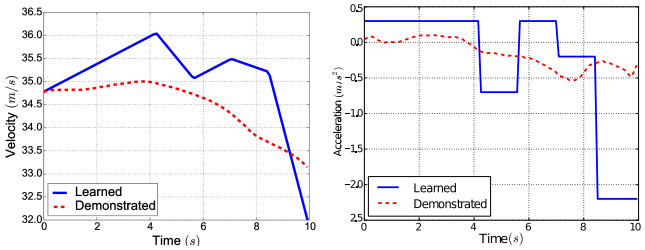
4.4.2 Metrics and competing approaches

The metric that we use is the modified Hausdorff distance (MHD), which is a generalized distance metric between point sequences and typically used for this kind of evaluation (Kitani et al., 2012; Wulfmeier et al., 2016; Shimosaka et al., 2017). Given the lack of a ground truth cost function, we follow previous work and additionally provide the values of task-specific metrics obtained by solving the planning problem with the optimized model. We consider average speed, average acceleration and the total time spent on each lane. Regarding the competing approach, we run the IRL optimization procedure using a sliding time window over

(a) Front view at $t = 7.0$ **(b)** Back view at $t = 7.0$



(c) Predicted behavior of the ego-vehicle surrounded by obstacle traces. The stronger the color gradient, the higher the time.



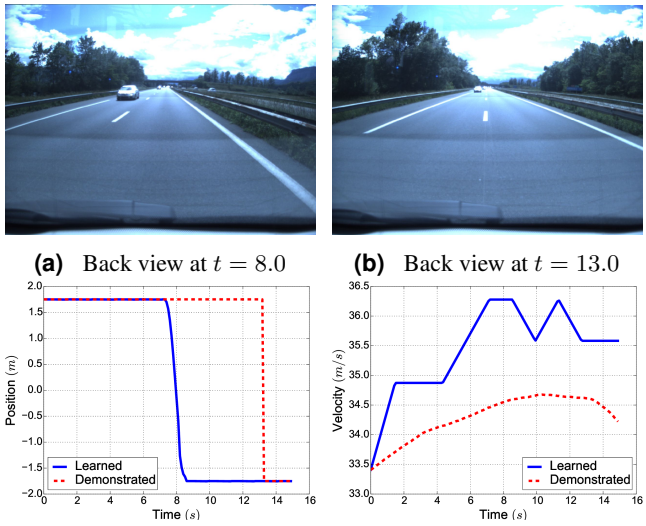
(d) Velocity prediction **(e)** Acceleration prediction

Fig. 4.4.: Prediction of behavior based on the optimized model.

each demonstration. For each setting of the time window, we run the optimization for the fixed duration of the window under the assumption that the environment remains static. To compensate for this assumption, the optimization is performed on overlapping time windows. This approach has been used in the context of social robotic navigation (Vasquez et al., 2014; Henry et al., 2010). We set the length of the time window to $5\,\mathrm{s}$, with an overlapping of $2.5\,\mathrm{s}$ between optimization cycles.

4.4.3 Qualitative results

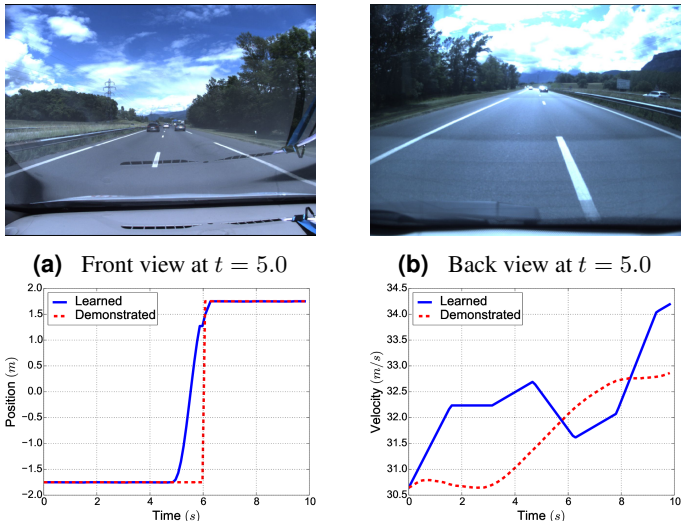
The results presented here have been obtained by approximating the feature expectations in the gradient equation with the features associated to the best trajectory found by the planner, that is, N_T is set to one. In the first place, we show the convergence of the algorithms in Figure 4.3. The proposed approach converges smoothly from an initial high deviation between the empirical features and the approximated expected sum of features. The initial low value for the window approach is probably due to the shorter length of the "windowed" demonstrations. As it can be seen, the norm of the proposed approach does not converge to zero. This is due to the high suboptimality of the demonstrations and also to their long distances. One of the features considered in our model is the deviation from the desired velocity, which we assumed to be the speed limit. We can see for instance in Figure 4.5d, that the demonstrated velocity does not



(a) Back view at $t = 8.0$ **(b)** Back view at $t = 13.0$

(c) Position prediction **(d)** Velocity prediction

Fig. 4.5.: Prediction of behavior for a right-merge scene.



(a) Front view at $t = 5.0$ **(b)** Back view at $t = 5.0$

(c) Position prediction **(d)** Velocity prediction

Fig. 4.6.: Prediction of overtaking behavior.

attempt to reach the speed limit of $36.1\,{}^{\mathrm{m}}/_{\mathrm{s}}$, despite not having any obstacles in front (not seen in the figure).

The results shown in Figures 4.4-4.6 correspond to roughly $150\,s$ of demonstrated driving data that were used for testing the resulting optimized model. We used the spatiotemporal state lattice planner to produce a trajectory based on the optimized model and we compare it against the demonstrated data. Figure 4.4 shows the results for a scene in which the ego-vehicle approaches slow traffic ahead and it is not safe to perform a lane change due to the presence of a vehicle on lane 2. The predicted behavior is a deceleration in order to reduce the relative velocity with the traffic ahead, which matches with the demonstrated behavior. Figure 4.5 consists of a merge to the right lane after an overtake. The model correctly captures that, in the

METHOD	MHD	Avg speed (m/s)	Avg accel. (m/s^2)	t left (s)	t right (s)
Window	0.145	25.04	-1.024	65.83	79.61
Proposed	**0.016**	**32.88**	**0.045**	**59.86**	**85.58**
Ground truth	-	31.93	0.019	51.55	93.89

Tab. 4.1.: Performance statistics of the predictions based on the learned model.

absence of obstacles, the demonstrations suggested driving on the right-most lane. In a similar fashion, Figure 4.6 shows that the predicted behavior when approaching slow traffic ahead with no vehicles blocking the left lane is an overtake. In this case the lane change trajectory is preferred because it allows an acceleration towards the speed limit, which in turn maximizes the traveled distance.

4.4.4 Quantitative results

Table 4.1 shows the performance metrics obtained on the roughly $150s$ of evaluation driving data. The proposed approach seems to clearly have captured the demonstrated behavior better than the "window" approach: it obtains a lower average MHD and resembles the ground truth more closely in the task-specific metrics.

4.5 Discussion

We have presented a driver modeling algorithm based on a combination of Maximum Entropy IRL and a spatiotemporal state lattice motion planner. The proposed approach is suitable for learning behavior models from demonstrations in highly dynamic environments. We validated our proposal using demonstrated suboptimal highway driving data gathered with an instrumented vehicle. Despite having been trained with general driving data containing all kinds of situations such as merges to the right, overtakes, getting stuck behind slow traffic and free driving, the model successfully captured the main characteristics of the demonstrated behavior. We show this result in both quantitative and qualitative experiments.

Several works in the literature are close to the presented approach (Perez-Higueras et al., 2018; Lee and Seo, 2017; Kuderer et al., 2015). In the first place, the approach from Perez-Higueras et al. approximates the feature expectations with multiple trajectories obtained by re-running an RRT* planner (Perez-Higueras et al., 2018). The approach is applied to learn a model for social robotics navigation among humans. However, the scenarios from where the model is learned are static. For highly dynamic and structured environments like the highway, the proposed lattice-based spatiotemporal planner constitutes a more efficient alternative to obtain representative trajectories of the model. Lee and Seo propose an approach in the same lines where the goal is to learn a driver model for overtaking illegally parked vehicles (Lee and Seo, 2017). In this case, they propose to generate the trajectories by sampling them from a previously learned GP. The main drawback of this approach is precisely that it requires learning the GP to model the trajectory distribution, which is not straightforward in practice. Moreover, for more complicated scenarios with an increased number of maneuvers, this approach would

require learning multiple GP models. Additionally, the trajectories sampled from the GPs do not respect the kinematic constraints of the vehicle. In general, since no explicit planning is performed, it might require many samples to find an adequate trajectory representative of the current model. Finally, the approach proposed by Kuderer et al. also addresses driver behavior modeling in highway scenarios (Kuderer et al., 2015). They propose to approximate the feature expectations using a single trajectory obtained using spline-based optimization. This approach presents two main drawbacks. The first one is that the spline optimization based on the RPROP algorithm may converge to local minima. A second issue is that the optimization procedure requires to set the endpoint of the spline in the same lane where the corresponding demonstrated trajectory ended. The first drawback is shared with our approach, which performs the search on a discretization of the underlying continuous space and additionally prunes the search space using the model as heuristic. However, in contrast to their approach, the trajectories obtained from the lattice-based planner are not constrained to end at any point in space (beyond remaining in the lattice structure). In other words, if the model is clearly wrong at any point of the optimization, the planned trajectories are free to deviate largely from the demonstrated ones. This makes the lattice-based spatiotemporal planner particularly suitable for the IRL modeling task.

5 Model-Based Driver Behavior Prediction

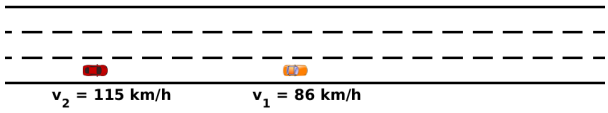
5.1 Introduction and related work

In the previous chapter, we proposed an approach capable of learning a driver behavioral model—a cost function—from highway driving demonstrations. We saw how using the learned model for motion planning would result in trajectories very close to those that had been demonstrated, i.e. human-like behavior. Our goal in this chapter is to exploit a similar feature-based model learned from demonstrations for motion prediction. In particular, we want to predict the lane and speed changes of all vehicles involved in a highway scene. For that, we will make the assumption that the behavior of each driver in the scene can be explained by the learned model. For this reason, we refer to this kind of approach as model-based behavior prediction.

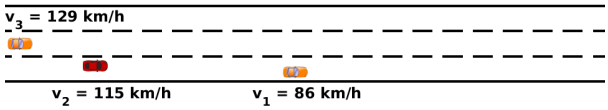
The idea of exploiting a cost function learned from demonstrations for the prediction of human-like motion has been recently explored under the name of planning-based motion prediction (Ziebart et al., 2009; Kitani et al., 2012; Vasquez, 2016). Planning-based approaches assume that people, when they move, they do so by minimizing a cost function that depends on their preferences and the context. Thus, given the cost function and the situational context, their future actions can be predicted. Unfortunately, most of the work in this area is limited to motion prediction for pedestrians in which the cost function is time-invariant and models the behavior of an agent in a static environment (i.e. an environment in which the obstacles do not move).

In contrast, the highway is a dynamic environment in which the actions of a driver are strongly conditioned by the states of the surrounding vehicles. Predicting the future becomes then a difficult task, as it involves predicting the actions of all agents in the scene while taking into account the interactions between them.

Despite the complexity of the task, humans are extremely good at predicting the future intents of other traffic participants. Moreover, we do this instinctively, without thinking too much about it. We know that their actions are constrained by the road network, by the traffic rules, and by a risk-aversive common sense. As an example of this, we can consider the situations shown in Figure 5.1. When a driver approaches the vehicle in front at a high relative speed, and there is no obstacle preventing him from switching lanes, we can predict with high probability that he will be indeed switching lanes (Figure 5.1a). However, if this maneuver carries a high risk of collision due to the presence of an obstacle (Figure 5.1b), the most likely response of the driver will be a braking maneuver. Similarly as in chapter 4, we model this "not so complex" behavior of drivers with a cost function that is a linear combination of (a) static features, which model the preferences of the driver (e.g. how fast to drive); and (b) dynamic features, which model the risk-aversive behavior of the driver and take into account the dynamic environment (e.g. the time to reach the closest vehicle in front at the current speed).



(a) Clear overtake



(b) Delayed overtake

Fig. 5.1.: Two traffic situations commonly found in highway driving.

In this chapter, we consider a discrete version of the highway navigation problem and apply the standard MaxEnt IRL algorithm from Ziebart et al. to learn the cost function from demonstrated driving data (Ziebart et al., 2008)[1]. We use such cost function to predict the behavior of each highway traffic participant up to the long-term future. In the next section we formalize the proposed approach.

5.2 Model-based prediction of highway scenes

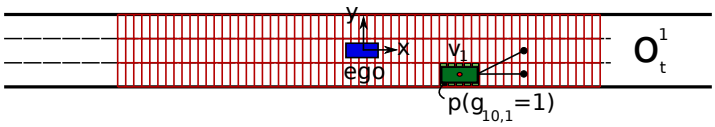
5.2.1 Discretization and notation

Our framework for motion prediction in high-speed highway scenes $(\mathcal{V}, \mathcal{G}, \{\mathcal{S}^i\}, \{\mathcal{A}^i\}, \{\mathcal{T}^i\}, \{S_0^i\})$ is a weakly-coupled multi-agent MDP given by:

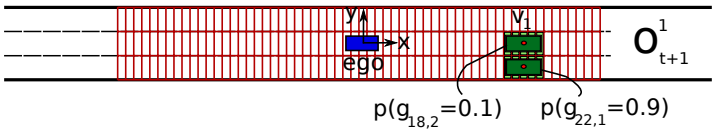
- $\mathcal{V}$ a finite set of $|\mathcal{V}|$ vehicles involved in the traffic scene.
- A grid $\mathcal{G}$ centered in the position of the ego-vehicle.
- $\mathcal{S}^i$ a finite set of states for vehicle $i \in \{1, 2, \ldots, |\mathcal{V}|\}$. Each vehicle i's state s^i is a tuple $((x^i, y^i), v^i)$, where (x^i, y^i) denotes the position of the vehicle i in grid $\mathcal{G}$, and v^i is its speed. The driver's desired speed $v_\bullet^i$ is assumed to be background knowledge.
- $\mathcal{A}^i$ a finite set of actions (or maneuvers) available to vehicle $i \in \{1, 2, \ldots, |\mathcal{V}|\}$.
- $\mathcal{T}^i : \mathcal{S}^i \times \mathcal{A}^i \mapsto \mathcal{S}^i$ denotes the transition rule, which describes the next-state $s'^i = \mathcal{T}^i(s^i, a^i)$ of vehicle i after taking action a^i in state s^i.
- $\mathcal{S}_0^i$ the initial state for vehicle $i \in \{1, 2, \ldots, |\mathcal{V}|\}$.

The cell (x^i, y^i) occupied by vehicle i in grid $\mathcal{G}$ can be uniquely identified given: a) the lane occupied by vehicle i; and b) the relative distance between the center of mass of vehicle i and that of the ego-vehicle (Figure 5.2). In the tuple (x^i, y^i), x^i indicates the cell index along the longitudinal dimension of the road, and y^i indicates the lane index. We denote by r_x the longitudinal resolution of the grid.

[1] Chronologically, chapters 5 and 6 were developed before chapter 4. Both chapters build upon a driver model learned on a simulator from a simplified driving task (section 5.3). Chapter 4 was developed afterwards in order to scale IRL and learn a model from real-world driving data. However, we decided upon the current organization of the document as it makes conceptually more sense.



(a) Occupancy distribution for vehicle $i = 1$ at time step t. The red dot marks the center of mass of the vehicle's state.



(b) Occupancy distribution for vehicle $i = 1$ at time step $t + 1$

Fig. 5.2.: Evolution of the occupancy distribution of a vehicle over the cells of grid $\mathcal{G}$ in one time step. The occupancy probabilities at time step t + 1 will depend on the available actions in the previous time step.

5.2.2 Feature-based cost function in dynamic environments

For any state $s^i \in \mathcal{S}^i$, we can calculate a feature vector $\mathbf{f}(s^i)$ that describes the vehicle's state and its context. The highway is a dynamic environment and therefore we need to consider static and dynamic features. Static features are time-invariant and can be directly calculated given a state. Dynamic features are those that change over time in response to the actions of the dynamic agents in the environment. Consequently, the cost associated to a given state, which is a weighted linear combination of the features, will also change over time. The cost of a state can then be expressed as:

$$\mathcal{C}_t^i(s^i) = \underbrace{\mathbf{w}_s^i \cdot \mathbf{f}_s(s^i)}_{\mathcal{C}_s^i(s^i)} + \underbrace{\mathbf{w}_d^i \cdot \mathbf{f}_{d,t}(s^i)}_{\mathcal{C}_{d,t}^i(s^i)} \quad \forall s^i \in \mathcal{S}^i \tag{5.1}$$

where $\mathbf{w}_s^i$ and $\mathbf{w}_d^i$ are the components of the weight vector associated to the static and dynamic features respectively, $\mathbf{f}_s : \mathcal{S}^i \mapsto \mathbb{R}^{K_s}$ is the time-invariant vector of static features, and $\mathbf{f}_{d,t} : \mathcal{S}^i \times \mathbb{T} \mapsto \{0,1\}^{K_d}$ is the vector of dynamic features at the different timesteps $t \in \{0, 1, \ldots, T-1\}$.

5.2.3 Feature selection

In order for the model to capture the preferences of a highway driver we have selected the following static features:

- Lane: These features aim to capture the preference of an agent to drive on a particular lane. For a highway with n_l lanes, the vector of lane features consists of n_l mutually-exclusive binary features: $\mathbf{f}_{s1} : \mathcal{S}^i \mapsto \{0,1\}^{nl}$ for any vehicle $i \in \{1, \ldots, |\mathcal{V}|\}$
- Speed deviation: This feature encodes the penalty of deviating from the agent's desired speed: $\mathbf{f}_{s2} : \mathcal{S}^i \mapsto \mathbb{R}$ for any vehicle $i \in \{1, \ldots, |\mathcal{V}|\}$, where

$$\mathbf{f}_{s2}(s^i) = v^i - v_\bullet^i$$

for each state $s^i \in \mathcal{S}^i$. The preferred speed of a driver $v^i_\bullet$ is calculated as the maximum speed observed since the last change in the speed limit.

The dynamic features capture the risk-aversive behavior of drivers. We propose the following features:

- Time-headway: We define the time-headway as the time elapsed between the back of the lead vehicle passing a point and the front of the following vehicle passing the same point. It indicates potentially dangerous situations (Vogel, 2003). To calculate the time-headway we define function $\phi : \mathcal{S}^i \times \mathcal{S}^i \mapsto \mathbb{R}$ that calculates the time-headway value between any two vehicles $i, j \in \{1, \ldots, |\mathcal{V}|\}$:

$$\phi(s^i, s^j) = \begin{cases} \frac{(x^i - x^j)\, r_x}{v^j} & \text{if } x^i - x^j \geq 0 \\ \frac{(x^j - x^i)\, r_x}{v^i} & \text{otherwise} \end{cases} \tag{5.2}$$

  We consider two separate dynamic features: the time-headway to the closest car in front in the current lane $\mathbf{f}_{d_1,t} : \mathcal{S}^i \times \mathbb{T} \mapsto \{0,1\}^{n_{th}}$, and the time-headway to the closest trailing car in the current lane $\mathbf{f}_{d_2,t} : \mathcal{S}^i \times \mathbb{T} \mapsto \{0,1\}^{n_{th}}$. They are defined as:

$$\mathbf{f}_{d_1,t}(s^i_t) = d\Big(\min_{\substack{j: \\ x^i_t - x^j_t \leq 0 \\ y^i = y^j}} \phi(s^i_t, s^j_t) \Big) \tag{5.3}$$

$$\mathbf{f}_{d_2,t}(s^i_t) = d\Big(\min_{\substack{j: \\ x^i_t - x^j_t \geq 0 \\ y^i = y^j}} \phi(s^i_t, s^j_t) \Big) \tag{5.4}$$

  where s^i_t and s^j_t are the states of vehicles i and j at time step t, and d is a discretization operator $d : \mathbb{R} \mapsto \{0,1\}^{n_{th}}$ that returns mutually exclusive binary vectors of dimension n_{th}. If there is no other vehicle at the front or at the back the time-headway is set to infinity.

5.2.4 Occupancy distribution

The generic driver model obtained via MaxEnt IRL enables us to predict what action is a driver likely to perform given his preferences and the current state of the environment. In order to keep track of the state of the environment in the current and future timesteps, we maintain a distribution over the cells of grid $\mathcal{G}$ for each vehicle. We shall call such distribution an *occupancy distribution of vehicle* i denoted $o^i_{t+1} \triangleq \chi(b^i_{t+1})$ and given by:

$$o^i_{t+1}(x^i, y^i) = \sum_{v^i} \sum_{s'^i = (x^i, y^i, z^i)} b^i_{t+1}(s'^i) \tag{5.5}$$

where $b^i_{t+1} \triangleq \tau(b^i_t, \pi^i_t)$ is the probability distribution over the state space of vehicle i, namely the belief state of vehicle i and given by:

$$b^i_{t+1}(s'^i) = \sum_{s^i} \sum_{a^i} P(s'^i | s^i, a^i)\, \pi^i_t(a^i | s^i)\, b^i_t(s^i) \tag{5.6}$$

We assume deterministic transitions and thus the transition probability term $P(s'^i|s^i, a^i) = 1\{s'^i = \mathcal{T}^i(s^i, a^i)\}$ only takes values 0 or 1, where $1\{\cdot\}$ is an indicator function. The term $\pi_t^i(a^i|s^i)$ is the policy model.

5.2.5 Policy model

The learned driver model tells us how comfortable a driver feels in a given state. For example, if the learned model associates high costs to states in which the distance to the car in front is too small, which is usually an indicator of dangerous situations, the driver will tend to avoid taking actions that would lead to such states. Thus, we propose the following heuristic as a policy model that captures the risk-aversive behavior of drivers:

$$\pi_t^i(a^i|s^i) \propto \exp\left(-\mathcal{C}_{t+1}^i(\mathcal{T}^i(s^i, a^i))\right) \tag{5.7}$$

The probability of taking one action will be low if it transitions the agent to a state s'^i in which the cost $\mathcal{C}_{t+1}^i(s'^i)$ is high and there are other actions available that would lead to states with a lower cost. In this chapter, we consider only a one time step look-ahead.

5.2.6 Dynamic cost estimation

In order to forecast the development of a traffic scene, we need to predict the state transitions of each agent following Equation 5.7. This implies calculating the cost associated to each state and by extension, the value of the dynamic features of the state, in the future. In subsection 5.2.3, we defined the dynamic features $\mathbf{f}_{d_1,t}(s_t^i)$ and $\mathbf{f}_{d_2,t}(s_t^i)$ for the deterministic case in which each vehicle occupies a single state. The calculation of the cost associated to the dynamic features (hereafter, dynamic cost) is then trivial:

$$\mathcal{C}_{d,t}^i(s^i) = \mathbf{w}_d^i \cdot \mathbf{f}_{d,t}(s^i) = \underbrace{\mathbf{w}_{d_1}^i \cdot \mathbf{f}_{d_1,t}(s^i)}_{\mathcal{C}_{d_1,t}^i(s^i)} + \underbrace{\mathbf{w}_{d_2}^i \cdot \mathbf{f}_{d_2,t}(s^i)}_{\mathcal{C}_{d_2,t}^i(s^i)}$$

where we have decomposed the dynamic cost into its components $\mathcal{C}_{d_1,t}^i(s^i)$ and $\mathcal{C}_{d_2,t}^i(s^i)$.

However, the concept of time-headway to the closest car is no longer defined in future timesteps where we have a state distribution for each vehicle in the scene. Instead, we traverse the occupancy distributions of all opposite vehicles calculating the dynamic costs associated to each possible location, and weighting them by the occupancy probability:

$$\begin{aligned} \varphi_{d1,t}(s_t^i) &\doteq \left\{ o_t^j(x^j, y^j) \;\; \mathbf{w}_{d_1}^i \cdot d\left(\phi(s_t^i, \hat{s}_t^j)\right) \right\} \\ \forall \hat{s}_t^j &= ((x^j, y^j), z_t'^j) : j \neq i,\, y^j = y^i,\, x^j - x^i \geq 0 \end{aligned} \tag{5.8}$$

$$\begin{aligned} \varphi_{d2,t}(s_t^i) &\doteq \left\{ o_t^j(x^j, y^j) \;\; \mathbf{w}_{d_2}^i \cdot d\left(\phi(s_t^i, \hat{s}_t^j)\right) \right\} \\ \forall \hat{s}_t^j &= ((x^j, y^j), z_t'^j) : j \neq i,\, y^j = y^i,\, x^j - x^i \leq 0 \end{aligned} \tag{5.9}$$

where in the state $\hat{s}_t^j$ induced by cell (x^j, y^j) we calculate the speed $z_t'^j$ as the weighted average of the states that contributed to the occupancy probability $o_t^j(x^j, y^j)$. The dynamic costs

$\mathcal{C}^i_{d_1,t}(s^i)$ and $\mathcal{C}^i_{d_2,t}(s^i)$ are then set to be the maximum of the weighted costs obtained above, that is:

$$\mathcal{C}^i_{d_1,t}(s^i) = \|\varphi_{d1,t}(s^i_t)\|_\infty \tag{5.10}$$

$$\mathcal{C}^i_{d_2,t}(s^i) = \|\varphi_{d2,t}(s^i_t)\|_\infty \tag{5.11}$$

Note that this is a maximum both across vehicles and across possible locations in grid $\mathcal{G}$.

This approach aims to mimic the way human drivers deal with risk: for a given vehicle i, if the probability of another driver cutting in front (and causing thus a low front time-headway feature value) is high, a risk-aversive preemptive maneuver (e.g. braking, changing lane) is expected. Equivalently, if a cell close in front of the vehicle has a high probability of being occupied, the policy model should encourage actions that transition the vehicle to low risk states (either by reducing the speed or changing lanes). In contrast, if a vehicle can cut in front but this is not likely to happen (i.e. there is no reason or evidence that this will happen), vehicle i will not alter its trajectory. In other words, cells with very low occupancy probability should generate a low dynamic cost. The occupancy probability acts here as a gate, letting through the cost associated to dangerous situations when they are likely to occur.

5.2.7 Highway scene prediction algorithm

Algorithm 3 summarizes the discussed steps of our approach. For each prediction time step within the desired time horizon, and for each vehicle $i \in \mathcal{V}$ from the front to the back of the scene we calculate the cost function $\mathcal{C}^i_t(s^i)$ using Equation 5.1 for all states $s^i \in \mathcal{S}^i$. To calculate the dynamic component of the cost, we use Equations 5.10 and 5.11. Once the costs are computed, we can calculate each vehicle's risk-aversive policy using Equation 5.7. Finally, we update the probability distribution over the state space b^i_{t+1} and the occupancy distribution $o^i_{t+1} = \chi(b^i_{t+1})$.

Algorithm 3: Highway scene prediction with feature-based driver models.

Input: time-horizon T, cost function $\mathcal{C}$
for $t = 0 \dots T-1$ **do**
  forall $i \in \mathcal{V}$ *from front to back* **do**
    Update cost function $\mathcal{C}^i_t$ using Eqs. 5.1, 5.10, 5.11;
    Compute vehicle i's policy π^i_t using Eq. 5.7;
    Maintain belief and occupancy distributions using Eqs. 5.6, 5.5;

The algorithm outputs at each time step the expected occupancy of the road. This could be be used for example by a planner to estimate the risk of different candidate trajectories.

5.2.8 Complexity analysis

The algorithm consists of three main operations (5.1), (5.7) and (5.5). Let $|\mathcal{A}^*|$ and $|\mathcal{S}^*|$ be the maximum number of actions and states for all vehicles; and G_x and G_y the transversal and longitudinal sizes of grid $\mathcal{G}$, respectively.

Complexity of cost-function updates The first operation (5.1) is the update of the cost function for all states. This involves the update of the static and dynamic parts of the cost function for all states. However, the static part can be computed offline so it is not considered. The complexity of computing the dynamic part of the cost function is linear in the number of states for each vehicle $i \in \mathcal{V}$ and each time step $t \in \{0, 1, \ldots, T-1\}$. As seen in subsection 5.2.6, to calculate these costs it is necessary to traverse longitudinally the occupancy grids of all opponent vehicles. Hence, it results in complexity $O(T \cdot |\mathcal{V}||S^*||\mathcal{V}-1|G_x)$.

Complexity of decision-rule updates The second operation is the computation of the decision rule for all state and action of a vehicle $i \in \mathcal{V}$ and each time step $t \in \{0, 1, \ldots, T-1\}$. That is about time complexity $O(|S^*||\mathcal{A}^*|)$. Notice that the denominator of (5.7) is computed only once for each state and vehicle, i.e., $O(|S^*||\mathcal{A}^*|)$. Overall the complexity of maintaining the decision rules for all vehicles and all timesteps is about $O(T \cdot |\mathcal{V}||\mathcal{S}^*||\mathcal{A}^*|)$

Complexity of information-measure updates The belief- and occupancy-update rules require time complexities $O(|\mathcal{A}^*||\mathcal{S}^*|^2)$ and $O(|\mathcal{S}^*|)$, respectively for each vehicle at every time step. That results in an overall complexity of about $O(T \cdot |\mathcal{V}||\mathcal{A}^*||\mathcal{S}^*|^2)$.

By aggregating together each operation, we end up with a complexity for Algorithm 3 that is linear with respect to the planning horizon T and the number of actions per vehicle $|\mathcal{A}^*|$, and quadratic in the number of vehicles $|\mathcal{V}|$, and the number of states $|\mathcal{S}^*|$ — i.e., $O(T \cdot |\mathcal{A}^*||\mathcal{S}^*|^2|\mathcal{V}|^2 G_x)$.

5.3 Experimental evaluation

To evaluate the presented approach we developed a 3-lane highway simulator inspired by the one presented in (Abbeel and Ng, 2004). We used this simulator to generate demonstrated trajectories from which a driver model was learned. Using this model, we tested our scene prediction algorithm on simulated highway traffic scenarios and on real-world traffic scenes from data gathered on a French highway.

5.3.1 Settings

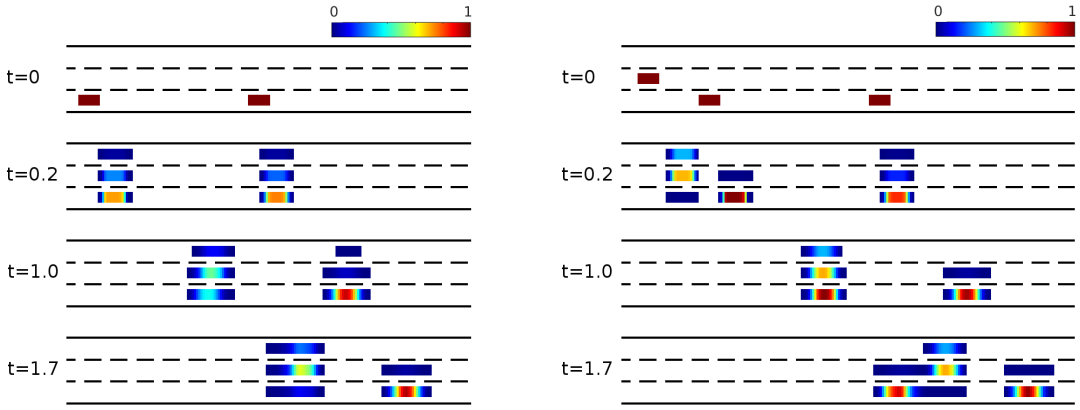
In the simulator, the discrete set of possible actions $\mathcal{A}^i$ for each vehicle i includes lane and speed changes, and any combination of the two. The vehicle speeds are discretized into a set of 11 evenly spaced bins between 0 m/s and 40 m/s (144 km/h). We assume that a vehicle can switch to an adjacent lane or speed per time step.

The simulator was used to generate training data by driving a vehicle through different highway traffic scenes. A total of 2 minutes of demonstrations were recorded and sampled at 10Hz. Using this dataset, we trained a driver model following the procedure described in subsection 5.2.2.

5.3.2 Simulation results

The driver model learned was used to predict the evolution of different simulated highway traffic scenes. The length of the grid was set to twice the range of a Velodyne LiDAR, i.e. $240\,\mathrm{m}$, with a resolution in the longitudinal dimension of $0.5\,\mathrm{m}$. Figure 5.3 shows the results obtained

for the two highway traffic situations discussed in the introduction of the chapter in Figures 5.1a and 5.1b. The figure shows the result of summing up across the occupancy distributions of all the vehicles in the scene (note that the result is no longer a probability distribution, but it is convenient to visualize the results). We assume that a vehicle changes lanes each time a new lane contains the majority of occupancy probability mass for that vehicle. The model captured correctly the demonstrated behavior. In Figure 5.3a, the system predicted that the trailing car would switch lanes in order to overtake the slower vehicle and maintain its speed. In Figure 5.3b, the system anticipated that the trailing vehicle on lane 1, would not switch lanes to overtake the slower vehicle in front, but instead slow down to maintain a security distance while waiting for lane 2 to be free of obstacles.



(a) Simulation of the traffic scene presented in Figure 5.1a. The model predicts an overtaking maneuver from the trailing vehicle.

(b) Simulation of the traffic scene presented in figure 5.1b. The model predicts that the trailing vehicle on lane 1 will slow down (note the low occupancy probabilities on lane 1 at $t = 1.7\,\mathrm{s}$ that correspond to states in which the vehicle would maintain/increase its speed).

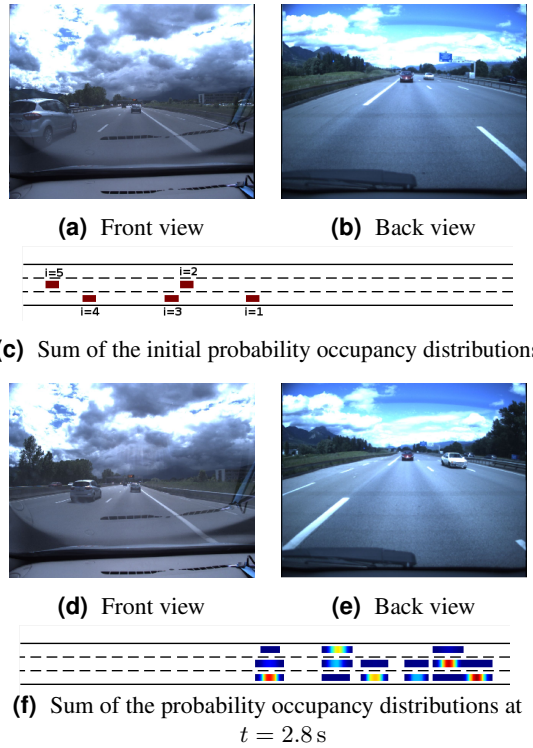
Fig. 5.3.: Prediction of the traffic scenes from figures 5.1a and 5.1b using the driver model learned with the simulator. We show the result of summing up the occupancy probabilities across the occupancy distributions of all vehicles in the scene. One time step corresponds to $0.1\,\mathrm{s}$. Each action (lane change, speed change) takes one time step to be completed.

5.3.3 Real-world highway traffic scenario

In order to verify if the intuitive results obtained in simulation apply as well to real-world situations, we used an instrumented vehicle to gather data on a French 3-lane highway. Figure 5.4 shows a given traffic scene encountered during our data gathering. Aside from the ego-vehicle ($i = 3$), 4 other vehicles are in the scene. Their velocities and positions were tracked

Tab. 5.1.: Data from the traffic scene evaluated.

i	z^i[km/h]	rel. dist. [m]	lane
1	86	24.2	1
2	101	2.1	2
3	86	0	1
4	84	-26.6	1
5	107	-36.7	2



(a) Front view **(b)** Back view

(c) Sum of the initial probability occupancy distributions

(d) Front view **(e)** Back view

(f) Sum of the probability occupancy distributions at $t = 2.8\,\mathrm{s}$

Fig. 5.4.: Typical 3-lane highway scene ($t = 2.8\,\mathrm{s}$). Our approach predicted a lane change for vehicle $i = 4$, that was advancing fast in the middle lane.

using the Conditional Monte Carlo Dense Occupancy Tracker (Rummelhard et al., 2015) and are shown in Table 5.1 (the vehicles are indexed from front to back). Vehicle $i = 5$ was approaching fast in the middle lane. Our approach successfully predicted the evolution of this traffic scene. In particular, it was predicted that vehicle $i = 5$ would change lanes to the left around $t = 2.8s$ in order to keep its high speed and overtake the vehicle in front (Figure 5.4f).

5.4 Discussion

In this chapter, we proposed a novel highway scene prediction framework that takes into account the interactions between traffic participants through the use of a driver model learned from a demonstrated driving task. In contrast to other approaches that take into account the mutual influence between drivers, the complexity of our approach does not grow exponentially in the number of vehicles in the scene (Lawitzky et al., 2013). Instead, we showed that the complexity scales quadratically in the number of vehicles, making the approach relatively lightweight from a computational standpoint.

The results presented in section 5.3 intuitively showed that the proposed approach can produce sensible predictions of the development of highway traffic scenes. The predictions are founded principally on the idea that highway drivers try to maintain their desired speeds while, at the same time, avoiding potentially dangerous states in the proximity of other vehicles. These are, essentially, the ideas upon which human drivers rely to anticipate the long-term future in highway scenarios.

While useful for the long-term prediction of highway scenes, the proposed prediction approach presents two main shortcomings that prevent its applicability in short-term horizons. These shortcomings are shared with most model-based prediction approaches (subsection 3.2.3). In the first place, the assumption under which drivers avoid, at all times, getting into dangerous states does not hold in reality—otherwise there would be no accidents. Secondly, disregarding the lateral dynamics of the vehicles throws away critical information, particularly for the detection of lane changes in the short-term.

To address the above-mentioned issues, we propose in the following chapter a probabilistic driver behavior estimation framework that fully leverages the dynamics of the target vehicles. Moreover, the proposed framework relies on a variation of the prediction approach presented in this chapter to model the interactions between traffic participants, which brings a degree of scene understanding into the behavior estimations.

6 Human-Like Driver Behavior Estimation

6.1 Introduction and related work

Model-based scene prediction approaches typically assume a risk-minimizing behavior for all agents, whether it is selfish (such as the one presented in the previous chapter) or altruist (Lawitzky et al., 2013; Schwarting and Pascheka, 2014). As a consequence, these approaches fail to predict dangerous maneuvers that the model cannot explain (e.g., a dangerous lane change maneuver in the highway). The end-goal of this thesis is to develop a highway decision-making system that can make safe and foresighted plans for the host vehicle. To do that, the system will require an accurate prediction of the short-term future, something that cannot be achieved with the prediction model from the previous chapter.

As we saw in section 3.2, an straight-forward way to predict the short-term future is to directly leverage the dynamics of the surrounding vehicles to identify the motion in execution (Weiss et al., 2004; Kaempchen et al., 2004; Toledo-Moreo and Zamora-Izquierdo, 2009). However, most of these dynamics-based approaches do not perform any reasoning about the traffic scene (i.e., the interactions between vehicles are disregarded). As a result, they are limited to predicting only the most immediate future and are prone to false positives under real-world conditions (in the context of highway lane change prediction, consider the response of one such system to the deviations from the lane center characteristic of human lane keeping).

In this chapter, we take inspiration once again from human drivers, who are extremely skillful at predicting the short-term intentions of surrounding vehicles. We hypothesize that this is due to their innate ability to: 1) interpret driver motion cues, and 2) put themselves in the place of other drivers to reason over their most likely behavior. Hence, in order to produce accurate, human-like, short-term predictions we present a probabilistic framework that integrates these two key ideas. In the first place, we make inference on an interaction-aware augmented Switching State-Space Model (aSSSM) (Barber, 2006; Agamennoni et al., 2012), which allows us to take into account the motion cues of drivers. Additionally, we apply model-based prediction to identify the most likely, risk-aversive, anticipatory maneuver for each driver in the scene. By integrating both sources of information, dynamic evidence is combined with the scene understanding brought in by the model-based prediction, overcoming thus the main drawbacks of both model- and dynamics-based vanilla approaches.

The proposed model from this chapter builds upon the work from Agamennoni et al., adopting their DBN structure and inference engine (Agamennoni et al., 2012). In their work, which was applied to track interacting trucks in an opencast mine, a fairly straight-forward model (essentially building upon the time-to-collision and the right-of-way) is used to describe the interactions between agents. This model is used to answer the question: "what will be the next action of a given agent given our current state and intention estimations for all

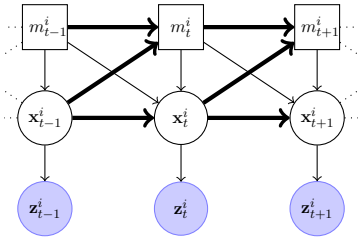


Fig. 6.1.: Graphical model representation of the proposed switching state-space model. A slice of the Bayesian network corresponding to vehicle i is shown. Bold arrows indicate multi-vehicle dependencies.

agents?", which plays a key role in the estimation procedure. In contrast, in this chapter we apply anticipative, interaction-aware, model-based prediction to answer this question. As we discussed in the previous chapter, this is in line with how humans reason about the behavior of surrounding drivers. Since we are dealing with highway scenarios, we will apply the proposed approach to estimate the lane change intentions of surrounding drivers. In the next section, we formalize our approach.

6.2 Interaction-aware model for driver behavior estimation

6.2.1 Overview and Notation

We propose a state and maneuver estimation framework based on a combination of discrete model-based maneuver prediction and discrete-continuous Bayesian filtering. More generally, the probabilistic model proposed can be categorized as a Switching State Space Model (SSSM), in which a high-level layer reasons about the maneuvers being performed by the different interacting vehicles and determines the evolution of the low-level dynamics. The high-level reasoning is performed through the fusion of: 1) an anticipatory, interaction-aware, model-based prediction; and 2) probabilistic messages coming from the bottom nodes of our SSSM and representing evidence about low level dynamics. Figure 6.1 shows the graphical model that specifies the conditional independence assumptions of our model. Bold arrows indicate multi-vehicle dependencies. Focusing on the slice of the graphical model for vehicle i, we can observe three layers of abstraction:

- The highest level corresponds to the maneuver m^i_t being executed by the vehicle. This is a discrete hidden random variable. The different maneuvers considered in this work and their corresponding dynamics are specified in subsection 6.2.3.
- The second level describes the state of the vehicle in a curved road frame through the continuous state vector $\mathbf{x}^i_t$. The state is not directly observable.
- Finally, the shaded nodes in the graphical model are the observations. The observation vector $\mathbf{z}^i_t$ is a noisy measurement of the state $\mathbf{x}^i_t$.

The factorization of the joint distribution given the model assumptions is the following:

$$P\left(\mathbf{x}_{1:T}^{1:N}, m_{1:T}^{1:N}, \mathbf{z}_{1:T}^{1:N}\right) = P\left(\mathbf{x}_{1}^{1:N}, m_{1}^{1:N}, \mathbf{z}_{1}^{1:N}\right) \prod_{t=2}^{T} \prod_{i=1}^{N} [P(\mathbf{x}_t^i | m_{t-1:t}^i, \mathbf{x}_{t-1}^{1:N})\ P(m_t^i | m_{t-1}^{1:N}, \mathbf{x}_{t-1}^{1:N})\ P(\mathbf{z}_t^i | \mathbf{x}_t^i)] \tag{6.1}$$

where the notation $\mathbf{x}_{1:T}^{1:N}$ is shorthand for the tuple $(\mathbf{x}_1^1, \ldots, \mathbf{x}_T^1, \ldots, \mathbf{x}_1^N, \ldots, \mathbf{x}_T^N)$, T indicates the number of timesteps considered and N denotes the number of vehicles involved.

The term $P(\mathbf{x}_t^i | m_{t-1:t}^i, \mathbf{x}_{t-1}^{1:N})$ describes the dynamic evolution of the state of a target given the distribution over maneuvers at the current and previous timesteps, and the previous states of all vehicles. Detailed descriptions of the motion model considered and of the dynamics associated to the different maneuvers are given in subsections 6.2.2 and 6.2.3 respectively.

The maneuvering behavior of drivers is described in the predictive term $P(m_t^i | m_{t-1}^{1:N}, \mathbf{x}_{t-1}^{1:N})$. The probability of a driver choosing a maneuver depends heavily on the state of the other traffic participants. We take advantage of the risk-aversive IRL driver model to take into account these interactions and to forecast the most likely distribution over maneuvers at each timestep. This process is detailed in section 6.2.4.

The term $P(\mathbf{z}_t^i | \mathbf{x}_t^i)$ is the measurement model and relates the hidden states of the vehicles in the scene with the observations through a linear measurement equation with added Gaussian noise $\mathbf{z}_t^i | \mathbf{x}_t^i \sim \mathcal{N}(\boldsymbol{C}\mathbf{x}_t^i, \boldsymbol{R})$, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$; $\boldsymbol{C}$ is the output matrix, and $\boldsymbol{R}$ is the covariance of the Gaussian observation noise.

To complete the description of the framework we present the inference engine to estimate the filtered posterior probability distribution over states and maneuvers for all agents in subsection 6.2.5.

6.2.2 Vehicle dynamics and maneuvers

We assume that vehicles move on a 2D environment and that the configuration of a vehicle i in a curved road frame is given by the state vector $\mathbf{x}_t^i = [x, y, \psi, v, \omega]^T \in \mathbb{R}^5$, where x and y are the target coordinates, v is the vehicle's absolute linear speed along its direction of travel ψ, and ω is the yaw rate. All variables in the state vector except for the yaw rate are observed. We assume that the dynamics satisfy the following differential equations:

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\psi(t) \\ \dot{y}(t) &= v(t)\sin\psi(t) \\ \dot{\psi}(t) &= \omega(t) \\ \dot{v}(t) &= a_{idm} \\ \dot{w}(t) &= 0 \end{aligned} \tag{6.2}$$

where we have neglected the influence of the yaw rate ω on the position. The term a_{idm} is the longitudinal acceleration of the target, which is set using the Intelligent Driver Model (IDM)

(Treiber et al., 2000). Hence, the acceleration is calculated at each timestep as a function of the states of all the vehicles in the scene. A maneuver-dependent predictive function $\mathbf{g}_\lambda$ integrates Equation 6.2 over an interval of time Δt to obtain the next state of the vehicle. The motion is assumed to be perturbed by Gaussian noise to account for the maneuver-specific modeling errors:

$$P(\mathbf{x}_t^i|\mathbf{x}_{t-1}^{1:N}, m_t^i{=}M_\lambda) \sim \mathcal{N}\Big(\mathbf{g}_\lambda\big(\mathbf{x}_{t-1}^{1:N}\big), \boldsymbol{Q}_\lambda\Big) \tag{6.3}$$

where $\boldsymbol{Q}_\lambda$ is the noise covariance matrix, which is dependent on the maneuver being performed.

Notice that it makes a lot of sense to have the motion being dependent not only on the maneuver being performed, but on the states of the other traffic participants. For instance, a lane keeping maneuver involves adjusting the acceleration of the vehicle depending on the state of the preceding traffic.

6.2.3 Maneuvers

In this work, we consider only two possible high-level maneuvers for the vehicles in the scene: lane keeping and lane changing. Their acceleration is set, as described above, using the IDM (Treiber et al., 2000):

$$a_{idm} = a_0 \left[1 - \left(\frac{v_{t-1}}{v_0}\right)^\delta - \left(\frac{g^*(v_{t-1}, \Delta v_{t-1})}{g_{t-1}}\right)^2\right] \tag{6.4}$$

where the desired minimum gap g^* is given by:

$$g^*(v, \Delta v) = g_0 + v\,T + \frac{v\Delta v}{2\sqrt{a_0 b_0}} \tag{6.5}$$

In Equations 6.4 and 6.5, g represents the gap with the leading vehicle, a_0 the maximum acceleration, b_0 the maximum comfortable deceleration, v_0 the desired speed, T the desired front time-headway, g_0 the traffic jam minimum distance, and $\Delta v = v - v_l$ the velocity difference with the leading vehicle. The parameter δ controls the change in acceleration as a function of velocity.

The IDM encodes different driving modes such as a free-road acceleration strategy when the gap to the leading vehicle is sufficiently large, or an "intelligent" braking strategy that keeps the deceleration below the comfort threshold b_0 unless an emergency situation takes place. More details can be found in (Treiber et al., 2000).

The details regarding the dynamics of each of the two available maneuvers are detailed below:

Lane change (LC)

The target vehicle turns and moves towards the neighboring lane. The dynamics of this maneuver are perfectly specified with the differential Equations (6.2) and its process noise covariance matrix $\boldsymbol{Q}_{LC}$.

Lane keeping (LK)

The vehicle of interest remains on its lane, aligned with the direction of the road, and driving at its desired speed unless it is slowed down by the leading vehicle. We do not adopt the constant yaw assumption of other approaches for this maneuver (Toledo-Moreo and Zamora-Izquierdo, 2009). Due to the particularities of the inference engine used in this work, the constant yaw assumption does not suffice to tell apart the LK and the LC dynamics. Instead, we parametrize our lane keeping motion model using Equations 6.2 and the corresponding process noise covariance matrix $\boldsymbol{Q}_{LK}$, along with an artificial observation ω' for the yaw rate:

$$\omega' = -\omega_{max}(\psi(t)/\psi_{max}) \tag{6.6}$$

The intuition behind this artificial observation is simple: if a target vehicle that is performing a lane keeping maneuver has a high yaw (in road coordinates, i.e. it is not aligned with the road), a steering action is expected in order to re-align the vehicle with the road. The magnitude of the expected yaw rate will be proportional to the misalignment of the target with the road, and is parametrized by ψ_{max} and ω_{max}, which have been obtained experimentally.

6.2.4 Interaction-aware, model-based maneuver forecasting

By means of the risk-aversive driver model obtained with IRL, we can forecast the probability of each driver's next maneuver in response to the states and maneuvers of the other traffic participants. This full-fledged single driver model contrasts with the approach presented in (Agamennoni et al., 2012), where each maneuver has an associated simple feature-based model. The driver model used here balances the (navigational and risk) preferences of drivers and enables us to predict their anticipatory behavior. A driver will perform a maneuver at the current timestep if, given his prediction for the behavior of the surrounding drivers, this leads to a sequence of F future states that accrue a lower cost than those of any potential alternative maneuver:

$$P(m^i_{t+1} = M|\mathbf{x}^{1:N}_t, m^{1:N}_t) \propto$$
$$\mathbb{E}_{(\mathbf{x}^{1:N}_t, m^{-i}_t)\sim P(\mathbf{x}^{1:N}_t, m^{1:N}_t|z^{1:N}_{1:t})} \exp\left(-\underbrace{\sum_{k=0}^{F-1} \mathbf{w}^T \mathbf{f}^i([\mathbf{x}^i_{t+k,M}, \mathbf{x}^{-i}_{t+k,m^{-i}_t}])}_{\substack{\text{Cost accrued over F time steps by vehicle } i. \\ \text{Vehicle } i \text{ executes maneuver } M. \\ \text{The obstacles execute sampled maneuvers } m^{-i}_t.}}\right) \tag{6.7}$$

where we have overloaded the notation for the state to explicitly indicate the maneuver being used to propagate it between timesteps, and the notation m^{-i} indicates the maneuvers for all agents except agent i. The superscript i in the feature function $\mathbf{f}^i$ indicates that the features are being calculated for the input joint state from the point of view of the ith vehicle. The expectation is taken with respect to the posterior at the previous timestep, which factorizes across agents, and solved using Monte Carlo sampling.

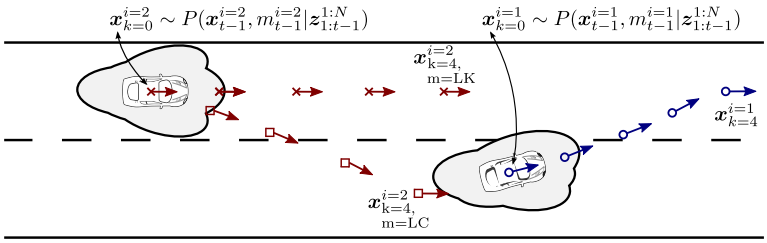


Fig. 6.2.: Schematic showing the anticipatory, model-based maneuver forecasting approach.

Figure 6.2 illustrates the procedure encoded in Equation 6.7. At each Monte Carlo sampling step, we draw a state and a maneuver from the posterior distribution of each traffic participant. Vehicles other than the vehicle of interest are propagated forward F timesteps according to the sampled maneuver. The lane change maneuver is propagated by setting a predefined angular speed ω_{LC} until the yaw reaches a given threshold. The lane keeping maneuver fixes the relative lateral position of the vehicle of interest and propagates it only longitudinally. When the maneuver being performed is a lane change and the vehicle reaches the centerline of the neighboring lane, the maneuver is switched automatically to lane keeping.

In contrast, the vehicle of interest is propagated forward F timesteps with all available maneuvers. For each of them, the cost associated to the resulting sequence of states is calculated. The distribution over maneuvers is given by the Boltzmann distribution, with the maneuver producing the lowest sum of costs being exponentially preferred. This captures the planning-based behavior of drivers, and overcomes one of the limitations of the approach from chapter 5, where only one time step look-ahead was considered.

As a particular example to illustrate the intuition behind Equation 6.7, we can imagine a vehicle being slowed down behind a truck in the highway and considering whether to keep driving behind the truck or to overtake. Using the features presented in chapter 5, we know that if the driver keeps driving behind the truck he will be penalized for deviating from his desired speed. In contrast, a lane change will mean a small penalty for not driving in the right-most lane but will enable the driver to accelerate, minimizing thus the cost due to speed. This example highlights the importance of: 1) The need to consider a sufficiently large planning-horizon when evaluating different maneuvers; and 2) Using a cost function that realistically balances the different features.

6.2.5 Approximate Inference

Exact inference of the posterior $P(\mathbf{x}_t^i, m_t^i | \mathbf{z}_{1:t}^i)$ is intractable in both the standard SLDS (Switching Linear Dynamical System) and in the aSLDS, scaling exponentially with time (Barber, 2006; Agamennoni et al., 2012). The proposed approximate inference engine is similar to the filtering approach presented in (Barber, 2006), with an extension to account for non-linear dynamics. Such an engine was already used in (Agamennoni et al., 2012), although with a non-Gaussian emission model. Approximate inference is performed individually per agent, which makes the algorithm highly parallelizable. The key idea is to approximate the intractable posterior with a simpler distribution (a Gaussian mixture), and to establish a recursion to track

it over time. More precisely, the posterior is decomposed into two terms, where the first one is approximated by a mixture of Gaussians:

$$P(\mathbf{x}_t^i, m_t^i|\mathbf{z}_{1:t}) = P(\mathbf{x}_t^i|m_t^i, \mathbf{z}_{1:t})P(m_t^i|\mathbf{z}_{1:t}) \tag{6.8}$$

$$\approx \underbrace{[\underbrace{\sum_{c_t} \underbrace{P(\mathbf{x}_t^i|c_t, m_t^i, \mathbf{z}_{1:t})}_{\substack{\text{Gaussian component}\\ \text{Mean: } f(m_t^i,c_t)\\ \text{Cov: } F(m_t^i,c_t)}} \underbrace{P(c_t|m_t^i, \mathbf{z}_{1:t})}_{\substack{\text{Component's}\\ \text{weight}}}}_{\text{Mixture of Gaussians}}] \underbrace{P(m_t^i|\mathbf{z}_{1:t})}_{\substack{\text{Marginal } m_t^i}} \tag{6.9}$$

We will present separately the recursions for the Gaussian mixture and for the manuever's marginal probability. Similarly as for the Bayes filter (subsection 2.2.4), the recursion can be divided into a predictive and a measurement step.

Predictive step

We search for $P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t}) = P(\mathbf{x}_{t+1}^i|m_{t+1}^i, \mathbf{z}_{1:t})P(m_{t+1}^i|\mathbf{z}_{1:t})$.

1. Update for the Gaussian mixture

$$P(\mathbf{x}_{t+1}^i|m_{t+1}^i, \mathbf{z}_{1:t}) = \sum_{m_t^i, c_t} P(\mathbf{x}_{t+1}^i, m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t}) \tag{6.10}$$

$$= \sum_{m_t^i, c_t} \underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Gaussian component}\\ \text{Mean: } \bar{f}(m_t^i,c_t,m_{t+1}^i)\\ \text{Cov: } \bar{F}(m_t^i,c_t,m_{t+1}^i)}} \underbrace{P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Weight}\\ \bar{w}(m_t^i,c_t,m_{t+1}^i)}} \tag{6.11}$$

The Gaussian component indexed by (m_t^i, c_t, m_{t+1}^i) is obtained by propagating forward the component (m_t^i, c_t) of $P(\mathbf{x}_t^i|c_t, m_t^i, \mathbf{z}_{1:t})$ with the dynamics of maneuver m_{t+1}^i using the EKF. This corresponds to the following equations:

$$\bar{f}(m_t^i, c_t, m_{t+1}^i) = \mathbf{g}_{m_{t+1}^i}\left(f(m_t^i, c_t)\right) \tag{6.12}$$

$$\bar{F}(m_t^i, c_t, m_{t+1}^i) = \nabla\mathbf{g}_{m_{t+1}^i}\left(f(m_t^i, c_t)\right)^T F(m_t^i, c_t)\nabla\mathbf{g}_{m_{t+1}^i}\left(f(m_t^i, c_t)\right) + \boldsymbol{Q}_{m_{t+1}^i} \tag{6.13}$$

where $\nabla\mathbf{g}_{m_{t+1}^i}\left(f(m_t^i, c_t)\right)$ denotes the Jacobian of the predictive function associated to maneuver m_{t+1}^i, evaluated at the mean of the Gaussian at time step t.

The weight component is calculated as:

$$\bar{w}(m_t^i, c_t, m_{t+1}^i) = P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t}) \tag{6.14}$$

$$\propto P(m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t}) \tag{6.15}$$

$$= P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t})P(c_t|m_t^i, \mathbf{z}_{1:t})P(m_t^i|\mathbf{z}_{1:t}) \tag{6.16}$$

where $P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t})$ is calculated using Equation 6.7, and $P(c_t|m_t^i, \mathbf{z}_{1:t})$ and $P(m_t^i|\mathbf{z}_{1:t})$ are available from the previous time step.

2. Update for the maneuver's marginal

The maneuver's marginal is predicted as:

$$P(m_{t+1}^i|\mathbf{z}_{1:t}) = \sum_{m_t^i, c_t} P(m_{t+1}^i, m_t^i, c_t|\mathbf{z}_{1:t}) \tag{6.17}$$

$$\propto \sum_{m_t^i, c_t} P(m_{t+1}^i, m_t^i, c_t, \mathbf{z}_{1:t}) \tag{6.18}$$

$$= \sum_{m_t^i, c_t} P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t}) \tag{6.19}$$

Measurement step

We search for $P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t+1}) = P(\mathbf{x}_{t+1}^i|m_{t+1}^i, \mathbf{z}_{1:t+1}) P(m_{t+1}^i|\mathbf{z}_{1:t+1})$.

1. Update for the Gaussian mixture

$$P(\mathbf{x}_{t+1}^i|m_{t+1}^i, \mathbf{z}_{1:t+1}) = \sum_{m_t^i, c_t} P(\mathbf{x}_{t+1}^i, m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1}) \tag{6.20}$$

$$= \sum_{m_t^i, c_t} \underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Gaussian component} \\ \text{Mean: } f(m_t^i, c_t, m_{t+1}^i)) \\ \text{Cov: } F(m_t^i, c_t, m_{t+1}^i)}} \underbrace{P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Weight} \\ w(m_t^i, c_t, m_{t+1}^i)}} \tag{6.21}$$

The Gaussian component indexed by (m_t^i, c_t, m_{t+1}^i) is obtained from $P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})$ by executing the measurement step of the Kalman filter with observation $\mathbf{z}_{t+1}$. This is achieved by the following equations:

$$\boldsymbol{S}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)} = \boldsymbol{C}\bar{F}(m_t^i, c_t, m_{t+1}^i)\boldsymbol{C}^T + \boldsymbol{R} \tag{6.22}$$

$$\boldsymbol{K}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)} = \bar{F}(m_t^i, c_t, m_{t+1}^i)\boldsymbol{C}^T(\boldsymbol{S}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)})^{-1} \tag{6.23}$$

$$f(m_t^i, c_t, m_{t+1}^i) = \bar{f}(m_t^i, c_t, m_{t+1}^i) + \boldsymbol{K}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)}(\mathbf{z}_{t+1}^i - \boldsymbol{C}\bar{f}(m_t^i, c_t, m_{t+1}^i)) \tag{6.24}$$

$$F(m_t^i, c_t, m_{t+1}^i) = (\boldsymbol{I} - \boldsymbol{K}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)}\boldsymbol{C})\bar{F}(m_t^i, c_t, m_{t+1}^i) \tag{6.25}$$

In the first place, the innovation covariance matrix $\boldsymbol{S}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)}$, which is the estimated covariance matrix of the measurements, is calculated using the emission matrix $\boldsymbol{C}$ and the noise covariance matrix $\boldsymbol{R}$. This matrix is used to calculate the Kalman gain matrix $\boldsymbol{K}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)}$, which balances the confidence between the predictions and the measurements. Using these elements, the corrected mean $f(m_t^i, c_t, m_{t+1}^i)$ and covariance $F(m_t^i, c_t, m_{t+1}^i)$ are obtained.

The corrected weight component is calculated as:

$$w(m_t^i, c_t, m_{t+1}^i) = P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1}) \tag{6.26}$$

$$\propto P(m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t+1}) \tag{6.27}$$

$$= P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t}) P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t}) \tag{6.28}$$

where the only new element is $P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t})$, which is the likelihood of $\mathbf{z}_{t+1}$ under the Gaussian $\mathcal{N}(\boldsymbol{C}\bar{f}(m_t^i, c_t, m_{t+1}^i), \boldsymbol{S}_{t+1}^{(m_t^i, c_t, m_{t+1}^i)})$. Intuitively, the observation is compared to the predicted Gaussian projected onto observation space. If the prediction using the dynamics of maneuever m_{t+1}^i corresponds to the movement of the target, the likelihood will be high and, by extension, also the weight.

2. Update for the maneuver's marginal

The corrected maneuver's marginal is predicted as:

$$P(m_{t+1}^i|\mathbf{z}_{1:t+1}) = \sum_{m_t^i, c_t} P(m_{t+1}^i, m_t^i, c_t|\mathbf{z}_{1:t+1}) \tag{6.29}$$

$$\propto \sum_{m_t^i, c_t} P(m_{t+1}^i, m_t^i, c_t, \mathbf{z}_{1:t+1}) \tag{6.30}$$

$$= \sum_{m_t^i, c_t} P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t}) P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t}) \tag{6.31}$$

which is calculated with the same terms as Equation 6.16.

Remark: it should be noted that it is in Equations 6.31 and 6.28 where the fusion between dynamic evidence (the likelihood of the observation under each of the available dynamics) and scene understanding (the maneuver forecasting using model-based prediction) takes place. More insights on the effects of this fusion are given in the experimental section 6.3.

Mixture components collapse step

Each recursive step starts with $M \times C$ Gaussian components, where M denotes the number of available maneuvers and C denotes the number of Gaussian components used for the approximation. After the prediction and measurement steps, the number of Gaussian components, indexed with (m_t^i, c_t, m_{t+1}^i), is $M^2 \times C$. Clearly, for each maneuver m_{t+1}^i, there are $M \times C$ associated Gaussian components, which need to be reduced to C. This is achieved by retaining, for each maneuver $m_{t+1}^i = m'$ the $(C-1)$ components with the highest weights $w(m_t^i, c_t, m_{t+1}^i = m')$ and merging the rest to a single Gaussian using the procedure mentioned in (Barber, 2006), and shown in appendix B. The resulting C components per maneuver are indexed by c_{t+1}.

After the collapse, we obtain:

$$P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t+1}) = P(\mathbf{x}_{t+1}^i|m_{t+1}^i, \mathbf{z}_{1:t+1}) P(m_{t+1}^i|\mathbf{z}_{1:t+1}) \tag{6.32}$$

$$\approx \underbrace{\Big[\underbrace{\sum_{c_{t+1}} \underbrace{P(\mathbf{x}_{t+1}^i|c_{t+1}, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Gaussian component} \\ \text{Mean: } f(m_{t+1}^i, c_{t+1}) \\ \text{Cov: } F(m_{t+1}^i, c_{t+1})}} \underbrace{P(c_{t+1}|m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Component's} \\ \text{weight, obtained} \\ \text{through merging}}}\Big]}_{\text{Mixture of Gaussians}} \underbrace{P(m_{t+1}^i|\mathbf{z}_{1:t+1})}_{\text{Marginal } m_{t+1}^i}$$

The discussed predictive, measurement and mixture steps are summarized in Algorithm 4.

Algorithm 4: Variational filtering on an SSSM for tracking the states and intentions of drivers in highway traffic scenes.

input : $\left[\sum_{c_t} P(\mathbf{x}_t^i|c_t, m_t^i, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t})\right] P(m_t^i|\mathbf{z}_{1:t})$ with $t = 1$
EKF and dynamics parameters for all maneuvers

SSSMfiltering
  for $t = 1$ **to** T **do**
    `/* Predictive step */`
    for *obstacle* $i = 1$ **to** N **do**
      for $m_{t+1}^i = 1$ **to** M **do**
        for $m_t^i = 1$ **to** M **do**
          for $c_t = 1$ **to** C **do**
            $\underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Mean: } \bar{f}(m_t^i, c_t, m_{t+1}^i) \\ \text{Cov: } \bar{F}(m_t^i, c_t, m_{t+1}^i)}} \leftarrow$ `EKF-predict` $(P(\mathbf{x}_t^i|c_t, m_t^i, \mathbf{z}_{1:t})$, dyn: $m_{t+1}^i)$
            Calculate $P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t})$ using Equation 6.8
            `// Subscript` $u \equiv$ `unnormalized prob.`
            $\bar{w}_u(m_t^i, c_t, m_{t+1}^i) = P_u(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t}) = P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t})$
        $P_u(m_{t+1}^i|\mathbf{z}_{1:t}) = \sum_{m_t^i, c_t} P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t})$
      $\bar{w}(m_t^i, c_t, m_{t+1}^i) = P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t}) \leftarrow$ `normalize`$(\bar{w}_u(m_t^i, c_t, m_{t+1}^i))$
      $P(m_{t+1}^i|\mathbf{z}_{1:t}) \leftarrow$ `normalize`$(P_u(m_{t+1}^i|\mathbf{z}_{1:t}))$
      `/* Predictive distribution */`
      $P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t}) \approx$
      $[\sum_{m_t^i, c_t} \underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Mean: } \bar{f}(m_t^i, c_t, m_{t+1}^i) \\ \text{Cov: } \bar{F}(m_t^i, c_t, m_{t+1}^i)}} \underbrace{P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})}_{\bar{w}(m_t^i, c_t, m_{t+1}^i)}] P(m_{t+1}^i|\mathbf{z}_{1:t})$
    `/* Measurement step */`
    Receive observation $\mathbf{z}_{t+1}$
    for *obstacle* $i = 1$ **to** N **do**
      for $m_{t+1}^i = 1$ **to** M **do**
        for $m_t^i = 1$ **to** M **do**
          for $c_t = 1$ **to** C **do**
            $\underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Mean: } f(m_t^i, c_t, m_{t+1}^i)) \\ \text{Cov: } F(m_t^i, c_t, m_{t+1}^i)}} \leftarrow$ `EKF-measure` $(P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})$, obs: $\mathbf{z}_{t+1})$
            $P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t}) \leftarrow \mathcal{N}(\mathbf{z}_{t+1}|\boldsymbol{C}\bar{f}(m_t^i, c_t, m_{t+1}^i), \boldsymbol{C}\bar{F}(m_t^i, c_t, m_{t+1}^i)\boldsymbol{C}^T + \boldsymbol{R}_{m_{t+1}^i})$
            `// Subscript` $u \equiv$ `unnormalized prob.`
            $w_u(m_t^i, c_t, m_{t+1}^i) = P_u(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1})$
            $= P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t}) P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t})$
        $P_u(m_{t+1}^i|\mathbf{z}_{1:t+1}) = \sum_{m_t^i, c_t} P(\mathbf{z}_{t+1}|m_t^i, m_{t+1}^i, c_t, \mathbf{z}_{1:t}) P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t})$
      $w(m_t^i, c_t, m_{t+1}^i) = P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1}) \leftarrow$ `normalize`$(w_u(m_t^i, c_t, m_{t+1}^i))$
      $P(m_{t+1}^i|\mathbf{z}_{1:t+1}) \leftarrow$ `normalize`$(P_u(m_{t+1}^i|\mathbf{z}_{1:t+1}))$
      `/* Corrected distribution (not collapsed) */`
      $P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t+1}) \approx [\sum_{m_t^i, c_t} \underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Mean: } f(m_t^i, c_t, m_{t+1}^i)) \\ \text{Cov: } F(m_t^i, c_t, m_{t+1}^i)}} \underbrace{P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t+1})}_{w(m_t^i, c_t, m_{t+1}^i)}] P(m_{t+1}^i|\mathbf{z}_{1:t+1})$
      `/* Collapse Gaussian components to obtain final posterior distribution */`
      $P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t+1}) \approx$
      $[\underbrace{\sum_{c_{t+1}} P(\mathbf{x}_{t+1}^i|c_{t+1}, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Mean: } f(m_{t+1}^i, c_{t+1}) \\ \text{Cov: } F(m_{t+1}^i, c_{t+1})}} \underbrace{P(c_{t+1}|m_{t+1}^i, \mathbf{z}_{1:t+1})}_{\substack{\text{Component's} \\ \text{weight, obtained} \\ \text{through merging}}}] P(m_{t+1}^i|\mathbf{z}_{1:t+1})$

6.3 Experimental evaluation

We have presented a filtering framework that keeps track of the posterior distribution over states and maneuvers of any target within sensor range of the ego-vehicle in highway scenarios. Low-level dynamic evidence is combined with a model-based, risk-aversive prediction, as shown in Equations 6.28 and 6.31. By fusing the dynamic evidence with an anticipatory, model-based prediction some degree of scene understanding is brought into the estimate. In this section, we focus on analyzing: 1) how both sources of information contribute to the final maneuver estimate; and 2) the ability of the proposed framework to estimate lane change maneuvers in real highway scenes.

To perform the analysis, we have gathered data on a French 2-lane highway using an instrumented vehicle equipped with a Velodyne HDL-64E LiDAR sensor, an Xsens MTi-100 IMU, and forward- and rear-facing IDS cameras. By means of a grid-based target tracker (Rummelhard et al., 2015) and a lane tracker, we are able to localize the targets with respect to the ego-vehicle (EV) and the road network. A total of 10 lane-change highway scenes have been annotated, accounting for a total of $152\,\mathrm{s}$ of driving data. The selected scenes include a variety of situations with 6 cut-in and 4 cut-out lane changes, taking place both behind (3) and in front (7) of the EV.

This section is organized as follows: first, the values of all the framework parameters used in the evaluation are presented. Next, the maneuver filtering performance of our approach is analyzed qualitatively on two of the traffic scenes of the dataset. Finally, a quantitative analysis is presented in order to be able to compare the lane-change detection performance of our framework against alternative approaches.

6.3.1 Framework parameters

The values of the framework parameters used in our evaluation are discussed here in roughly the same order that they have been introduced in the chapter. In the first place, we use the same driver model that was obtained in chapter 5 from simulated driving data.

The values for the rest of parameters are shown in Table 6.1. The parameters for the IDM are similar to those suggested by Treiber et al. (Treiber et al., 2000) with the exception of the desired time-headway T, which has been reduced from $1.6\,\mathrm{s}$ to $1.0\,\mathrm{s}$.

The noise covariance matrices have been set using the discrete white noise approach (Kaempchen et al., 2004; Bar-Shalom and Li, 2001). The process noise standard deviations to account for the constant acceleration and constant yaw-rate assumptions have been adopted from (Toledo-Moreo and Zamora-Izquierdo, 2009) and are shown in Table 6.1. The parameters ω_{max} and ψ_{max} used to generate the synthetic observations that encode the lane keeping behavior have been chosen empirically, as well as the components of the measurement noise covariance matrix $\boldsymbol{R}$, which is set to be diagonal.

The time horizon T_h and timestep dt for the model-based maneuver prediction were set by considering the trade-off between foresight, accuracy, and computation time. Finally, the number of Gaussian mixture components used during filtering has been set to $C = 3$. With this

IDM	$a_0 = 1.5\,\mathrm{m/s^2}$	$b_0 = 1.67\,\mathrm{m/s^2}$	$T = 1.0\,\mathrm{s}$
	$g_0 = 2\,\mathrm{m}$	$\delta = 4$	
Noise LK	$\sigma_a = 4.0\,\mathrm{m/s^3}$	$\sigma_w = 0.0205\,\mathrm{rad/s^2}$	
Noise LC	$\sigma_a = 4.0\,\mathrm{m/s^3}$	$\sigma_w = 0.15\,\mathrm{rad/s^2}$	
LK dynam.	$\psi_{max} = 0.04\,\mathrm{rad}$	$\omega_{max} = 0.28\,\mathrm{rad/s}$	
R	$\sigma_x = 0.2\,\mathrm{m}$	$\sigma_y = 0.2\,\mathrm{m}$	$\sigma_\psi = 0.01\,\mathrm{rad}$
	$\sigma_v = 0.2\,\mathrm{m/s^2}$	$\sigma_{w'} = 0.06\,\mathrm{rad/s^2}$	
Model pred.	$T_h = 3\,\mathrm{s}$	$dt = 0.1\,\mathrm{s}$	
Filtering	$C = 3$		

Tab. 6.1.: Framework parameters.

configuration, each filtering step took an average of $53\mathrm{ms}$ in a Python implementation of our framework executed on an Intel i7-6700HQ CPU running at $2.60\mathrm{GHz}$.

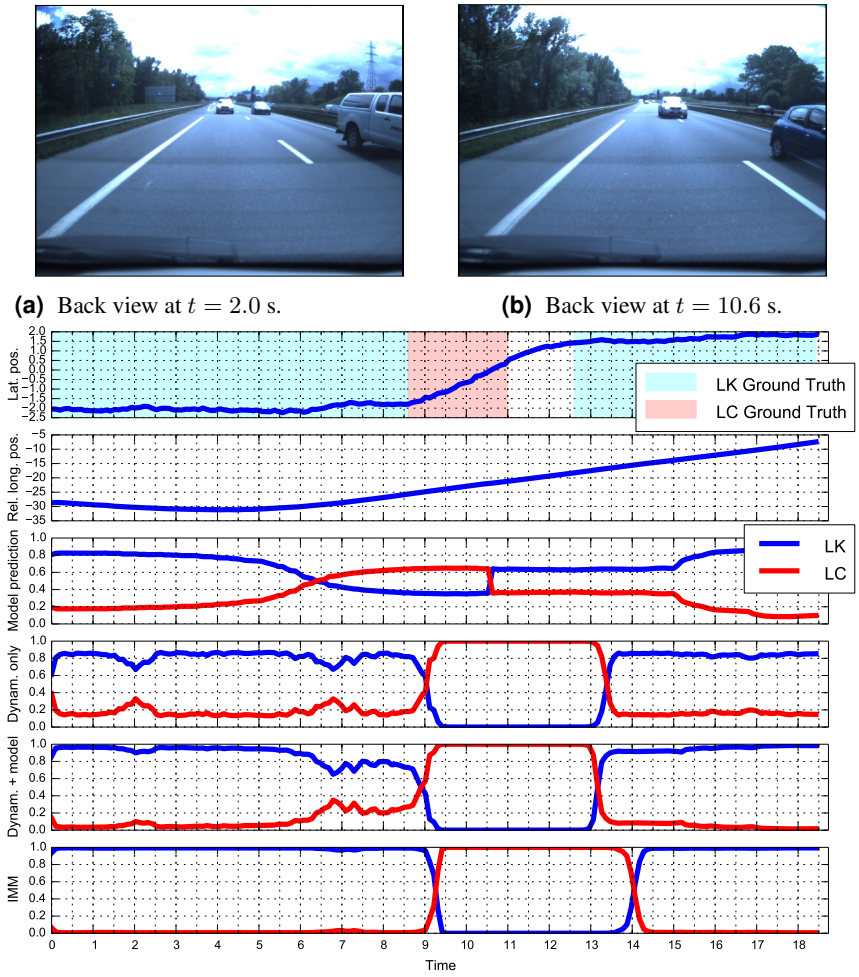
6.3.2 Qualitative analysis

We analyze the performance of our framework on two of the annotated highway scenes and compare the results obtained against the IMM-based lane change prediction approach from (Toledo-Moreo and Zamora-Izquierdo, 2009) (Model-set A).

Figures 6.3 and 6.4 show the results obtained for the two scenes. In the first one, the target vehicle drives behind the EV until the left lane is free of traffic and then performs a lane change to overtake. The first row in Figure 6.3c shows the lateral position of the target in road coordinates (the center lane-marking corresponds to $y = 0$); the second row shows the relative longitudinal position of the target with respect to the EV. As can be seen in the third row, the model-based prediction initially concedes a low probability to the LC maneuver due to the presence of traffic in the neighboring lane. Once the left lane becomes free, the probability for a LC begins to grow as the distance with the preceding traffic increases. The increase in the LC maneuver probability is due to the cost penalty induced by driving behind the EV at a speed lower than the target's desired speed, which had been set earlier during the target's approach.

Our framework's dynamics-based maneuver filtering estimate is shown in the fourth row of Fig. 6.3c. This estimate is obtained by using an uninformative model-based prediction, i.e. a prediction that assigns the same probability to all maneuvers. The lane change maneuver is detected only $0.51s$ after it begins and roughly $1.5s$ before the target crosses the lane marking. This is $0.2s$ faster than the IMM-based estimate, which is shown in the last row. Our framework's final maneuver estimate is shown in the fifth row. As it can be seen, the effect of the model-based prediction in the LC maneuver estimate is a slightly quicker detection both of the beginning and of the end of the maneuver.

Figure 6.4 shows the filtering results for a different scene, in which the target vehicle overtakes the EV and then merges back to the right lane. Focusing on Figure 6.4c, we can observe again how integrating the model-based prediction into the filtering framework results in a quicker detection of the start and the end of the LC maneuver. An additional effect can be observed in this figure. The slight oscillations in the lateral position typical of a human-driven vehicle following a highway lane can be observed here around $t = 3.5s$. As a consequence



(a) Back view at $t = 2.0$ s.

(b) Back view at $t = 10.6$ s.

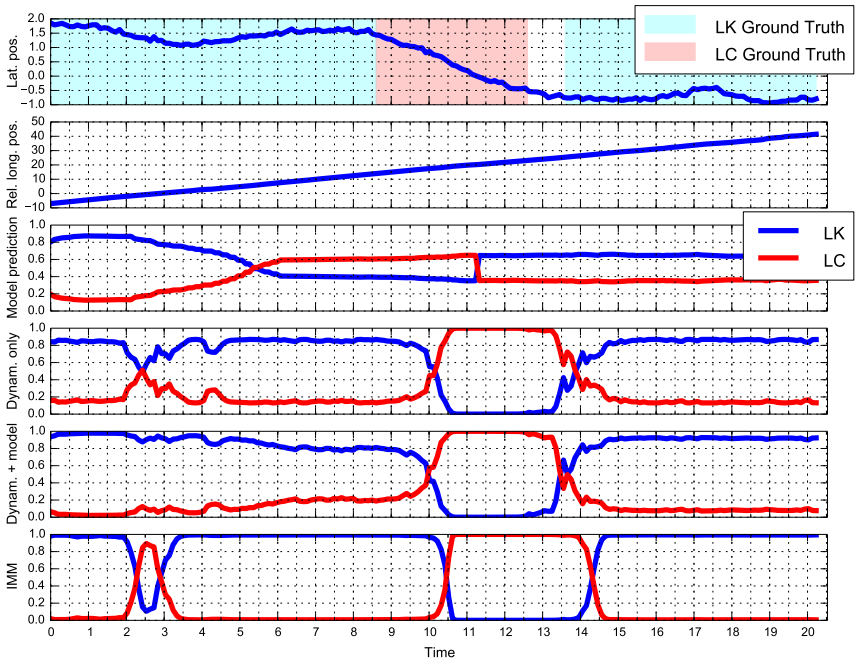
(c) Filtering results obtained for scene 3. The results of the proposed approach that leverages dynamic evidence and model-based prediction are shown in the penultimate row. Compared to an IMM filter (last row), we can observe a slightly quicker detection both of the beginning and of the end of the lane change maneuver of the target.

Fig. 6.3.: Maneuver filtering results obtained for a rear cut-out lane change. The vehicle trailing behind the ego-vehicle does a left lane change when the left lane becomes free of traffic.

of this lateral movement, both the dynamics-based and the IMM LC estimate produce a false positive. This does not occur in the final estimate of our approach, since the model-based prediction assigns a very low probability to a lane change maneuver around $t = 3$, moment in which the target drives right next the EV. This highlights the potential of our approach to reduce the number of false positives through the scene understanding brought in by the model-based prediction. However, does it imply that our framework cannot predict dangerous maneuvers? To answer this question we fixed the output of the model-based prediction to $P(m_t = \text{LK}) = 0.8$ and $P(m_t = \text{LC}) = 0.2$, and obtained again the filtered maneuver estimate for the two scenes discussed, showing the results in Fig. 6.5. It can be observed that the lane changes are still detected, with only a slight delay with respect to the estimate obtained with the true model prediction. The detections are still as fast as those of the IMM approach.

(a) Front view at $t = 4.0$ s.

(b) Front view at $t = 11.2$ s.



(c) Filtering results obtained for scene 7. The results of the proposed approach that leverages dynamic evidence and model-based prediction are shown in the penultimate row. Compared to an IMM filter (last row), we can observe an increased robustness against false positive lane change detections ($t = 2.5$s) and a lower latency in the detection of actual lane changes ($t = 10$s).

Fig. 6.4.: Maneuver filtering results obtained for a forward cut-in lane change. The target vehicle overtakes the ego-vehicle and merges in front of it by doing a right lane change.

6.3.3 Quantitative analysis

In order to produce a quantitative analysis, we can interpret the filtered estimate as the output of a binary classifier in which the LC maneuver is the positive class, and each timestep corresponds to a classification sample. A prediction is counted as positive when $P(m_t = \text{LC}) > 0.5$. We have manually annotated the maneuver ground truth for the 10 test scenes, as shown in the first row of Figure 6.3c and 6.4c. In the annotation, the lane change is assumed to start at the moment in which the target begins to move towards the neighboring lane. The end of the LC maneuver is assumed to occur when the center of mass of the target has reached $0.5m$ into the new lane. The LK maneuver is resumed once the target stabilizes its lateral position. The time between the end of the LC maneuver and the beginning of the LK maneuver is not considered in our analysis. It is unimportant whether the algorithm predicts the maneuver switch at the

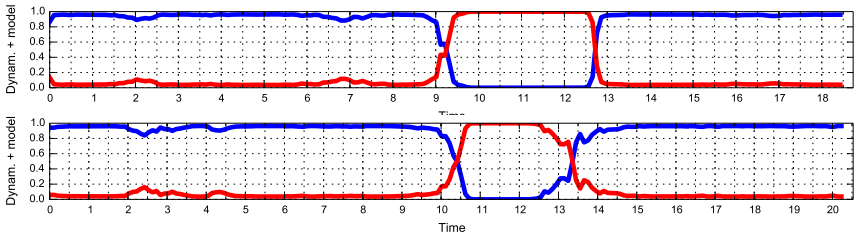


Fig. 6.5.: Filtering results obtained with an incorrect model prediction for the scenes shown in Figure 6.3c (top), and Figure 6.4c (bottom). The model prediction is manually set to $P(m_t = \text{LK}) = 0.8$ and $P(m_t = \text{LC}) = 0.2$ throughout the whole scene. Despite this, the lane change maneuvers are still detected.

moment in which the target begins to steer or at the moment in which it has finally aligned with the road.

With this setup, we can calculate the following metrics to evaluate the robustness of the estimation: 1) Accuracy: proportion of samples correctly classified; 2) Precision: number of correct positive predictions divided by the total number of positive class values predicted; 3) Recall: number of positive predictions divided by the number of positive class values in the test data; and 4) False Positive Rate: number of false positives divided by the total number of negative class values in the test data. The results are presented in Table 6.2. The overall performance of our approach is satisfactory, beating the IMM filter in every metric. The effect of the model-based prediction is also clear in the results, improving the performance with respect to the dynamics-based prediction in all four metrics. A result of particular relevance for the potential application of this system as a collision detection warning system is the extremely low false-positive rate.

	Accuracy	**Precision**	**Recall**	**FPR**
Model	0.7463	0.4455	0.7305	0.2493
Dynamics	0.8945	0.7557	0.7532	0.0668
IMM	0.8029	0.5332	0.6786	0.1630
Dynamics+Model	**0.9203**	**0.8277**	**0.7955**	**0.0454**

Tab. 6.2.: Performance metrics from a classification perspective.

Another relevant performance measure that needs to be quantified is the lane change detection delay, that is, the delay between the beginning of the lane change maneuver and the actual detection. The individual detection delays for each of the test scenes and for each approach are presented in Table 6.3. Again, the proposed framework beats the IMM filter by detecting the beginning of a lane change $0.39\,\text{s}$ earlier on average. Integrating the model-based prediction into the filtering framework leads to detecting a lane change $0.13\,\text{s}$ earlier on average in comparison with the dynamics-only estimate.

6.4 Discussion

In this chapter we have presented a probabilistic filtering framework to infer the maneuvers of drivers in highway scenarios. The proposed approach integrates the anticipatory, risk-aversive

Detection delay: t_D [s]											
Scene	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**
IMM	1.40	0.78	0.71	0.84	0.46	1.27	2.01	1.25	1.05	0.78	**1.05**
Dyn	1.09	0.48	0.51	0.74	0.15	1.07	1.71	0.95	0.75	0.48	**0.79**
Dyn+Mod	0.99	0.28	0.31	0.64	0.15	1.07	1.51	0.74	0.65	0.28	**0.66**

Tab. 6.3.: Lane change detection delay.

maneuver prediction capabilities of a driver model learned with IRL into a filtering framework. Experimental results on real highway data gathered with an instrumented vehicle show that our approach can detect a lane change maneuver $0.39\,\mathrm{s}$ earlier on average than a state-of-the-art IMM filter.

Including the model-based prediction into the filtering framework has been shown to produce faster detections and increased robustness in the maneuver estimations in comparison with a filter relying only on dynamics. The intuition behind the increased robustness was illustrated by the scene shown in Figure 6.4. The model-based prediction encodes the behavior that is normally expected from a target driver in a given situation. When the dynamics of the target contradict the model-based prediction, the detection of the maneuver that matches the target's dynamics is delayed as per Equations 6.28 and 6.31. During this delay, the dynamics of the target can change (evidencing a false positive, case shown in Figure 6.4) or continue the same, in which case the maneuver in still detected despite contradicting the model-based prediction (case shown in Figure 6.5).

In practice, the overall high accuracy and low false positive rate in the detection of lane changes obtained in our experiments open the door to the real-world application of this work as part of a collision warning system.

We envision several potential avenues to improve the proposed approach. In the first place, the generation of trajectories during the model-based maneuver forecasting step is currently performed on a rather rudimentary manner. More realistic trajectories could be obtained, for example, using a state-lattice trajectory planner, as shown in chapter 4. Furthermore, states are sampled along each trajectory to evaluate the cost of the corresponding maneuver. In this process, potential collisions between the candidate trajectories of different vehicles could be missed, leading to an incorrect prediction of the most-likely maneuvers. A trajectory collision checker could be used here to address this issue; however, this step needs to remain computationally light-weight as it is repeated many times per filtering step.

Another aspect that will be considered in future work is the addition of new maneuvers. Clearly, to handle multi-lane highways we will need to differentiate between left and right lane changes. This will involve the tuning of the noise models of each maneuver. Furthermore, currently we are modeling the longitudinal motion of each vehicle using the IDM car-following model. However, this is only valid if the host vehicle's sensors can detect the leading vehicle of the target. Of particular interest is the case where the target vehicle is being slowed down by a

leading vehicle that we cannot sense. This case could be modeled by adding a simple "braking" maneuver, i.e. slowing down from the desired speed without any apparent cause to do so.

Finally, while the model-based prediction brings a degree of scene understanding into the estimates, no reasoning is done about the navigational goals of the targets. An example of navigational goal could be, for example, exiting the highway. If we observe that a vehicle performs a lane change to join a lane that ultimately exits the highway, this provides extra information that could be exploited in the future while reasoning about the potential maneuvers of this target. This idea could be perhaps modeled as an extra layer in the graphical model from Figure 6.1. The difficulty here lies on how to maintain our belief in the navigational goals of the surrounding vehicles from their estimated maneuvers.

7 Towards Human-Like Tactical Planning

7.1 Introduction and related work

So far, in this thesis we have presented how to learn a cost function that encodes the driving behavior of a human driver (chapter 4), how to estimate the state and intentions of surrounding vehicles in highways (chapter 6), and how to predict the long-term development of highway scenes using model-based prediction (chapter 5). In this chapter, we present a POMDP-based tactical decision-making approach that integrates all these elements to take safe, foresighted actions in highways.

As discussed in the introductory chapter, the decision-making system of an automated vehicle should take actions that are foresighted, predictable by other road users, and coherent. Modeling the decision-making task as a POMDP enables us to meet all the desired requirements. First, by definition a POMDP looks for actions that minimize the long-term cost accrued by the agent. To determine the future impact of any action in the accrued cost, we can leverage the predictive models from chapters 5 and 6—these models can provide an estimate of the future state of the world as a reaction to the agent's action. Secondly, learning the cost model directly from human-driven demonstrations should encourage the agent to act in a human-like manner, making its actions predictable to other human road users. Finally, as a consequence of the previous two points, the decisions made by the POMDP should be coherent, avoiding an unstable switching between contradictory actions.

Another major advantage of using a POMDP for decision making is its ability to reason in the presence of uncertainty. In this chapter, we model as uncertain the physical state of all vehicles in the scene (including the ego-vehicle) and the lane change intentions of surrounding drivers. In chapter 6, we showed how to accurately estimate the lane change intentions of any vehicle in a highway traffic scene by reasoning about the interactions between the traffic participants and about their dynamics. A POMDP can exploit this knowledge to produce safe and comfortable driving decisions. Figure 1.3a showed an scenario that perfectly illustrates this idea. If the lane keeping/changing intention estimations of the vehicle trailing in the right lane are uncertain, the POMDP guiding the actions of the ego-vehicle can choose to take an anticipatory smooth deceleration action to avoid falling into a potentially dangerous, high-cost, situation. On the other hand, if the next observations of the obstacle lead to a confident lane keeping intention estimation, the POMDP should make the ego-vehicle return to its desired velocity. This example also highlights the importance of having reliable intention estimation capabilities, which have a direct impact on the behavior of the automated vehicle.

The POMDP approach that we propose is based on Monte Carlo tree search and closely resembles the POMCP algorithm (Browne et al., 2012; Silver et al., 2008). Both in our

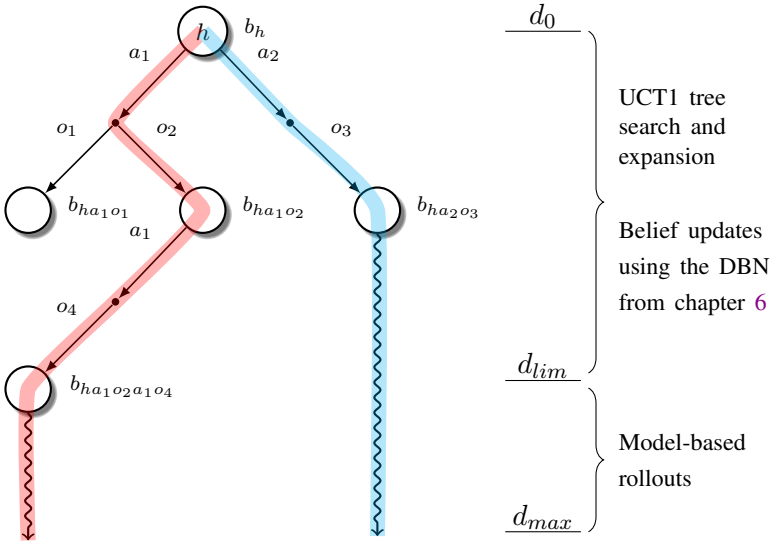


Fig. 7.1.: Search tree of histories. Each history h, has an associated parametric belief b_h. The beliefs are updated using the DBN model from chapter 6. The subscripts in the actions and observations indicate particular instantiations rather than time. The depth of the tree is truncated to d_{lim}. The model-based rollouts are executed until d_{max}. In this chapter, we will refer to the circular tree nodes as belief nodes, and to the black dots as action nodes. The red and cyan transparent lines indicate exemplary simulations executed in the tree.

approach and in POMCP a search tree of histories is built by sampling sequences of actions and observations. However, while in POMCP the belief state updates are performed using particles, in our approach we rely for the updates on the DBN-based estimation model from chapter 6, which represents the beliefs as parametric distributions. Since this model takes into account the interactions between vehicles, we also use it as the informed generative model from where observations are sampled, guiding thus the construction of the tree.

Unfortunately, updating the belief with the interaction-aware estimation model is computationally expensive. For this reason, the tree expansion is truncated at a given depth d_{lim} in the tree (Figure 7.1). The values of the leaf nodes are then approximated using a simulation (rollout) policy. As noted by Silver and Tesauro, "a simulation policy with appropriate domain knowledge can dramatically outperform a uniform random simulation policy" (Silver and Tesauro, 2009). In our particular domain, we are interested in lightweight rollouts that take into account the interactions between vehicles and that encode the normal behavior of highway drivers. We meet both criteria by producing the rollouts using a model-based scene prediction algorithm, where it is assumed that vehicles will avoid risk and take actions according to their driving preferences.

Despite being developed for discrete action, state and observation spaces, POMCP has been applied to continuous domains such as mobile robotics and autonomous driving (Goldhoorn et al., 2014; Bouton et al., 2017; Sunberg et al., 2017). Goldhoorn et al. apply POMCP on a find-and-follow mobile robotics application (Goldhoorn et al., 2014). They consider only discrete movement actions for the robot and propose to discretize the observation space. This is

a major drawback, as finding a discretization with an adequate trade-off between search space size and performance is generally hard.

For this reason, Sunberg and Kochenderfer propose instead to apply POMCP in combination with double progressive widening to cope with the continuous action and observation spaces (Sunberg and Kochenderfer, 2018). The use of progressive widening limits the number of actions and observations sampled from their corresponding continuous spaces and enables thus a controlled growth of the search tree. This has the inconvenience of having to restart the tree after each execution step of the planner. In our approach, we also choose to apply progressive widening in the observations and to restart the tree after each execution step. Our main goal in this chapter is to validate the behavioral model learned from demonstrations and the interaction-aware behavior estimation model as reliable elements to make safe driving decisions. In consequence, achieving real-time performance is not critical for us, although we discuss in section 7.4 possible options to accelerate the computation.

Sunberg et al. apply the variant of POMCP with progressive widening to a highway tactical navigation task, where the ego-vehicle has to change lanes until it reaches a particular target lane (Sunberg et al., 2017). The behavior of surrounding traffic is simulated and modeled in the POMDP using the IDM (car-following) and MOBIL (lane-change gap acceptance) models so, as occurred with our model-based prediction algorithm from chapter 5, the POMDP is not able to anticipate and react to potential accidents. In contrast, by using the interaction-aware DBN as the generative model used to construct the tree, we ensure that any dangerous maneuvers evidenced by the dynamics of the targets are represented in the tree, and can therefore be taken into account to make safe driving decisions.

A closer approach to ours is presented by Bouton et al., who apply POMCP with observation progressive widening for an IV decision-making application focused on unsignaled intersections (Bouton et al., 2017). They represent the belief on the physical state and dynamic mode (CA or CV) of surrounding vehicles using Gaussian distributions. The main difference with our approach is that they perform the belief updates using an IMM filter, which in contrast to our DBN model does not consider the interactions between vehicles. They evaluate the POMDP planner on its ability to merge on a T-intersection with varying degrees of traffic density. As the traffic density increases, the calculated policies lead to collisions. They hypothesize that this is caused by the limitations of the IMM as the internal state estimator and generative model. Another possible cause is the fairly simple, hand-tuned reward model, mainly composed of a term to reward merging into the intersection and another to penalize collisions. The differences between this approach and ours are summarized in Table 3.1.

The presented high-level idea of our approach should already explain the title of the chapter, "Towards Human-Like Tactical Planning". Human drivers closely monitor their environment to infer the behavior of surrounding vehicles. Based on this intention estimation, they can take foresighted driving actions that help them avoid dangerous situations. This idea closely resembles the proposed approach, in which the best action for the ego-vehicle is found by

	(Bouton et al., 2017)	**Proposed approach**
Application	Unsignaled intersections navigation	Highway navigation
Approach	POMCP+PW with parametric beliefs	POMCP+PW with parametric beliefs
Uncertainties	State and longitudinal dynamic mode of surrounding traffic	State of all vehicles and lane change intentions of surrounding traffic
Belief updater	IMM with two dynamic modes, CA and CV. Interactions between vehicles are disregarded.	DBN that models the interactions between vehicles.
Reward model	Hand-tuned model. Rewards crossing the intersection and penalizes collisions.	Human-like model learned from demonstrations. Contains terms that model the risk and navigational preferences of drivers.
Simulation policy	Rule-based considering TTC + IDM	Model-based

Tab. 7.1.: Characteristics comparison between the proposed decision-making approach and the closest work in the literature (Bouton et al., 2017).

searching on a history tree of 'alternative futures', which is constructed based on the estimated intentions of the surrounding drivers.

Furthermore, human drivers also consider the long-term consequences of their actions while driving. They do so by predicting approximately the long-term development of traffic scenes (remember for instance the case presented in Figure 1.3d, where we hypothesized that a human driver would perform an early lane change to avoid getting stuck behind a much slower vehicle). This is exactly the behavior captured by the proposed rollout policy. For these reasons, plus the fact that we use a cost model learned from human driving demonstrations, we refer to our approach as *human-like tactical planning*. The approach is properly formalized in the following section.

7.2 Foresighted and interaction-aware decision-making

In this section, we formalize the proposed POMDP model for decision-making in highways.

7.2.1 POMDP model

The proposed POMDP model is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, \mathcal{Z}, \mathcal{O}, \mathcal{V} \rangle$ where:

- A state $s \in \mathcal{S}$ contains:
  1. The ego-vehicle's physical state $\mathbf{x}^e = [x^e, y^e, \dot{x}^e]^T \in \mathbb{R}^3$, where x^e, y^e represent the longitudinal and lateral positions of the vehicle's center of mass in road coordinates, and $\dot{x}^e$ represents the longitudinal velocity.
  2. The obstacles' physical states $\mathbf{x}^{1:N}$, each of them with the same components as the ego-vehicle's. We assume the constant presence of N obstacle vehicles throughout any traffic scene considered. The space of all joint physical states of the vehicles in the scene is denoted by $\mathcal{X}$.
  3. The obstacles' maneuver intentions $m^{1:N} \in \{\text{LCL, LCR, LK }\}^N$, where LK represents a lane keeping maneuver, and LCL/LCR represent lane change maneuvers to the left and to the right, respectively.

  That is, at any discrete time step t we have $s_t = [\mathbf{x}_t^e, \mathbf{x}_t^{1:N}, m_t^{1:N}]$.
- In this chapter, the actions $m_t^e \in \mathcal{A}$ correspond to the maneuvers of the ego-vehicle. We consider two different experimental scenarios, each with a different action space:

1. In the first scenario, the ego-vehicle can perform lane changes, but its acceleration is set automatically using the IDM car-following model. That is, $\mathcal{A}_1 = \{\text{LCL}, \text{LCR}, \text{LK}\}$.
2. In the second scenario, the ego-vehicle drives in a given lane and it needs to adjust its speed so as to drive comfortable and safe. The action space for this scenario consist of three different discrete acceleration values $\mathcal{A}_2 = \{-1, 0, 1, \} \ \mathrm{m/s^2}$.

- The state transition function $\mathcal{T}(s, a, s')$ is given by the dynamics of the system, detailed in subsection 7.2.2.
- The cost function $\mathcal{C}^i : \mathcal{X} \mapsto \mathbb{R}$ provides the cost associated to a particular joint physical state of all vehicles in the scene, from the point of view of the ith vehicle. It corresponds to the model learned from demonstrations in chapter 4.
- The observation $\mathbf{z}^i$ of the physical state of any vehicle i is composed of noisy versions of all the components in its physical state. The intentions of surrounding traffic are not observable. The joint observation at each time step is $\mathbf{z}_t = [\mathbf{z}_t^e, \mathbf{z}_t^{1:N}]$. The continuous space of all possible observations is denoted by $\mathcal{Z}$.
- The observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \mathscr{P}(\mathcal{Z})$ is given by the predictive step of the model from chapter 6. Given an action and resulting state, it returns a probability distribution over the possible observations. Further details are provided in subsection 7.2.3.
- A model $\mathcal{V}$ that provides the desired speed of the ego-vehicle $v_\bullet^e$ and of all vehicles in the scene $v_\bullet^{1:N}$ using background knowledge (such as maps) and the most recent belief distribution. These values are necessary to determine the *desired-speed deviation* feature, which is part of the cost function as described in chapter 4. For the ego-vehicle, this value is set to the road speed limit. For any other target in the scene, the desired speed is set to its highest estimated longitudinal velocity since it has entered the scene. These values are assumed to be constant during the planning cycle of the POMDP and updated only after each execution step.

7.2.2 Dynamics and belief updates

The dynamics of the system are modeled using the DBN presented in chapter 6, where the posterior distribution $P(\mathbf{x}_t^i, m_t^i | \mathbf{z}_{1:t})$ is recursively estimated at each time step for each vehicle i in the scene (Equation 6.32). For the ego-vehicle, the maneuver is known and only the state is estimated.

However, in this chapter we treat vehicles as point objects, which simplifies the dynamic equations with respect to what was presented in chapter 6. This enables faster belief updates and, by extension, increased exploration during each POMDP planning cycle. The point dynamics satisfy the following equations:

$$\begin{aligned} x_{t+1} &= x_t + \Delta t \ \dot{x}_t \\ y_{t+1} &= y_t + \Delta t \ v_{\text{lat}} \\ \dot{x}_{t+1} &= \dot{x}_t + \Delta t \ a_{\text{long}} \end{aligned} \tag{7.1}$$

where a_{long} and v_{lat} are the longitudinal acceleration and lateral velocity control inputs, which are maneuver-dependent values. These linear equations determine the evolution of the physical state of a vehicle. They can be expressed in matrix form:

$$\underbrace{\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x}_{t+1} \end{bmatrix}}_{\mathbf{x}_{t+1}} = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \end{bmatrix}}_{\mathbf{x}_t} + \underbrace{\begin{bmatrix} 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{bmatrix} v_{\text{lat}} \\ a_{\text{long}} \end{bmatrix}}_{\mathbf{u}_t} \tag{7.2}$$

Note that these equations resemble the predictive step of the Kalman filter (Equation C.1). Similarly as in the Kalman filter, the physical state transitions are assumed to be perturbed by Gaussian noise (Equation 6.3). In this case, the maneuver-dependent predictive function $\mathbf{g}_\lambda$ integrates the linear equations 7.1 to produce the mean state prediction. Integration is necessary when the targets are highly maneuverable relative to the sampling rate (Agamennoni et al., 2012). The lane keeping and lane change maneuvers are defined by their associated control input values and the noise covariance matrix $\boldsymbol{Q}_\lambda$.

Lane change maneuver When a vehicle performs a lane change maneuver, its lateral velocity v_{lat} is set to a fixed v_{LC} value (with a sign that depends on the lane change direction) until the mean lateral position of the vehicle is at the center of the target lane, moment in which v_{lat} is set to 0. The longitudinal acceleration is always set using the IDM (Equation 6.4). This setting was also used in (Sunberg et al., 2017).

Lane keeping maneuver For a lane keeping maneuver, v_{lat} is set to be proportional to the lateral distance from the lane center, i.e. $v_{\text{lat}} = 2(y - y_C)\ v_{LC}/L$, where L denotes the lane width and y_C the lateral position of the lane center. The position of the lane markings and of the lane centers is assumed to be background knowledge and not included in the state. The longitudinal acceleration of all vehicles is always predicted using the IDM (Equation 6.4), except for the ego-vehicle in the second experimental setting.

In Algorithm 4, which was used to update the beliefs in the previous chapter, the EKF was used to perform the prediction and measurement steps. This involved a repetitive calculation of the Jacobian matrices, which is no longer necessary with the simplified linear dynamics proposed in this chapter. The only modifications happen in lines 7 and 19, which are replaced by the KF counterparts. This simply involves using matrix $\boldsymbol{A}$ instead of the Jacobian (see Appendix C).

7.2.3 Observation model

The belief state contains the current estimation of the physical state of the ego-vehicle and all surrounding obstacles, as well as the estimation of the obstacles' maneuver intentions. As discussed in the introduction of this chapter, we propose a POMCP-inspired planning approach where a tree of histories is built by sampling sequences of actions and observations. The belief associated to each history is maintained using algorithm 4. This algorithm recursively updates a parametric distribution $P(\mathbf{x}_t^i, m_t^i | \mathbf{z}_{1:t})$ for each obstacle in the scene. For the ego-vehicle, we

maintain only $P(\mathbf{x}_t^e|\mathbf{z}_{1:t})$ since the maneuvers of the ego-vehicle are the actions of the POMDP. The algorithm consists of a predictive and a measurement step. The distribution used to sample observations is obtained after the predictive step as follows:

1. We start off at time step t with:

   a) The posterior over the physical state and maneuver intention for each obstacle:

$$P(\mathbf{x}_t^i, m_t^i|\mathbf{z}_{1:t}) \approx \left[\sum_{c_t} \underbrace{P(\mathbf{x}_t^i|c_t, m_t^i, \mathbf{z}_{1:t})}_{\substack{\text{Mean: } f(m_t^i,c_t) \\ \text{Cov: } F(m_t^i,c_t)}} \underbrace{P(c_t|m_t^i, \mathbf{z}_{1:t})}_{\text{Weight: } w(m_t^i,c_t)}\right] P(m_t^i|\mathbf{z}_{1:t})$$

   b) The posterior over the state of the ego-vehicle:

$$P(\mathbf{x}_t^e|\mathbf{z}_{1:t}) = \mathcal{N}\left(\mathbf{x}^e; \boldsymbol{\mu}_t^e, \boldsymbol{\Sigma}_t^e\right)$$

2. An action m_t^e is sampled for the ego-vehicle (more details on the sampling procedure on subsection 7.2.5). Then, the following predictive distributions are obtained:

   a) For the obstacles:

$$P(\mathbf{x}_{t+1}^i, m_{t+1}^i|\mathbf{z}_{1:t}) = \left[\sum_{m_t^i, c_t} \underbrace{P(\mathbf{x}_{t+1}^i|m_t^i, c_t, m_{t+1}^i, \mathbf{z}_{1:t})}_{\substack{\text{Mean: } \bar{f}(m_t^i,c_t,m_{t+1}^i) \\ \text{Cov: } \bar{F}(m_t^i,c_t,m_{t+1}^i)}} \underbrace{P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})}_{\bar{w}(m_t^i,c_t,m_{t+1}^i)}\right] P(m_{t+1}^i|\mathbf{z}_{1:t})$$

   b) For the ego-vehicle, the Gaussian representing the state is propagated forward with the dynamics of the sampled action:

$$P(\mathbf{x}_{t+1}^e|\mathbf{z}_{1:t}) = \mathcal{N}\left(\mathbf{x}^e; \bar{\boldsymbol{\mu}}_{t+1}^e, \bar{\boldsymbol{\Sigma}}_{t+1}^e\right)$$

3. To produce the observation $\mathbf{z}_{t+1} = [\mathbf{z}_{t+1}^e, \mathbf{z}_{t+1}^{1:N}]$ we need to sample the physical state of the ego-vehicle and of all obstacles. The procedure is the following:

   a) For the obstacles:

   - Sample from $P(m_{t+1}^i|\mathbf{z}_{1:t})$, which is calculated following Equation 6.19.
   - Given $\hat{m}_{t+1}^i$, sample from $P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})$, which follows Equation 6.16.
   - The sampled $\hat{m}_{t+1}^i$, $\hat{m}_t^i$ and $\hat{c}_t$ index the corresponding predictive Gaussian distribution $P(\mathbf{x}_{t+1}^i|\hat{m}_t^i, \hat{c}_t, \hat{m}_{t+1}^i, \mathbf{z}_{1:t})$. To sample from this predictive distribution, we project it into observation space and include the observation noise:

$$\mathbf{z}_{t+1}^i \sim \mathcal{N}(\boldsymbol{C}\bar{f}(\hat{m}_t^i, \hat{c}_t, \hat{m}_{t+1}^i), \boldsymbol{C}\bar{F}(\hat{m}_t^i, \hat{c}_t, \hat{m}_{t+1}^i)\boldsymbol{C}^T + \boldsymbol{R}) \tag{7.3}$$

   b) For the ego-vehicle:

$$\mathbf{z}_{t+1}^e \sim \mathcal{N}(\boldsymbol{C}\bar{\boldsymbol{\mu}}_{t+1}^e, \boldsymbol{C}\bar{\boldsymbol{\Sigma}}_{t+1}^e\boldsymbol{C}^T + \boldsymbol{R}) \tag{7.4}$$

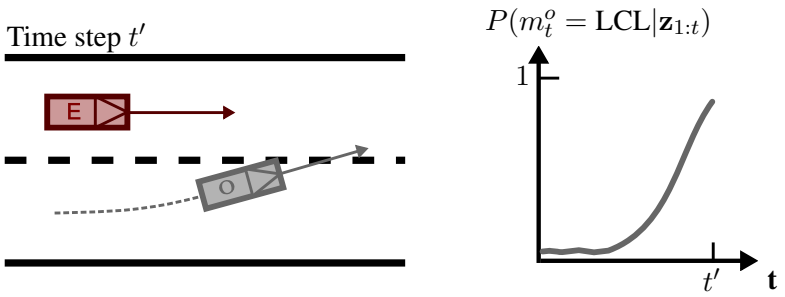


Fig. 7.2.: Example that motivates the proposed sampling procedure. In this scene, the model-based maneuver prediction from Equation 6.7 would predict with high probability a lane keeping maneuver for obstacle o given the presence of the ego-vehicle in the adjacent lane. However, the maneuver of this obstacle had been estimated to be a lane change based on its dynamics (graph on the right) and therefore this situation should be adequately represented in the history tree. Relying exclusively on *ideal* driver models for sampling during the tree construction can lead to an incorrect estimation of the consequences of an action.

In the third step of the sampling procedure, the distributions $P(m_{t+1}^i|\mathbf{z}_{1:t})$ and $P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})$ are dependent on the term $P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t})$, which encodes the model-based probability of executing maneuver m_{t+1} from the Gaussian physical state indexed by m_t^i and c_t (Equation 6.7). For example, the calculation of the maneuver's marginal before the measurement step is done using Equation 6.19, which is restated here for convenience:

$$P(m_{t+1}^i|\mathbf{z}_{1:t}) = \sum_{m_t^i, c_t} P(m_{t+1}^i|m_t^i, c_t, \mathbf{z}_{1:t}) P(c_t|m_t^i, \mathbf{z}_{1:t}) P(m_t^i|\mathbf{z}_{1:t})$$

In the equation above, the last two terms are available from the previous time step in the recursion and basically play the role of weights for each Gaussian component indexed by m_t^i and c_t. As a result, the calculation of $P(m_{t+1}^i|\mathbf{z}_{1:t})$ above corresponds to a weighted combination of model-based predictions. The case for $P(m_t^i, c_t|m_{t+1}^i, \mathbf{z}_{1:t})$ is similar. In this situation, the observations sampled to construct the history tree will be obtained assuming a development of the traffic scene where each vehicle behaves in the risk-averse manner encoded in the model. However, there might be situations where the behavior of a vehicle deviates from what is encoded in the behavioral model. One example of such situation is illustrated in Figure 7.2. In this scenario, a vehicle is performing a dangerous lane change just in front of the ego-vehicle. A risk-averse model cannot explain this behavior and thus the term $P(m_{t+1}^i|\mathbf{z}_{1:t})$ calculated with Equation 6.7 would heavily bias the sampling (and by extension the construction of the history tree) assuming a lane keeping maneuver for the obstacle. It is obvious to see how this can lead to a dangerously incorrect estimation of the consequences of the ego-vehicle's actions.

In contrast to the inability of the model-based prediction to anticipate unsafe driving behavior, the maneuver estimation approach presented in chapter 6 was able to detect lane change maneuvers by relying on the dynamic evidence of the target, even when the model indicated a different behavior (Figure 6.5). This is illustrated in the graph on the right side of Figure 7.2, which shows the posterior marginal probability of a left lane change maneuver. In

our online POMDP setting, each time the vehicle takes an action it receives a true observation of the world, which is then used to update the belief. Hence, the accurate dynamics-aware state and maneuver estimations are available at the beginning of each planning cycle and should therefore be exploited to bias the construction of the tree. We achieve this by modifying Equation 6.7 to include a term than promotes maneuver continuity across timesteps:

$$P(m_{t+1}^i = M | \mathbf{x}_t^i, m_t^i) \propto$$
$$\mathbb{E}\left[\exp\left(-\underbrace{\left[\begin{array}{c}\text{Cost accrued by executing}\\ \text{maneuver M over } T_m \text{ time steps}\end{array}\right]}_{\substack{\text{Model-based,}\\ \text{risk-aversive component}}} + \underbrace{T_m \;\; w_{mc}\,\delta_{(M,\hat{m}_t^i)}\left(\frac{\tau - t}{\tau - t_0}\right)}_{\substack{\text{Term to promote}\\ \text{maneuver continuity}}}\right)\right] \tag{7.5}$$

where the expectation is with respect to the posterior distributions at time step t over the physical state of the ego-vehicle $P(\mathbf{x}_t^e | \mathbf{z}_{1:t})$ and over the physical state and maneuver intention of all obstacles $P(\mathbf{x}_t^{1:N}, m_t^{1:N} | \mathbf{z}_{1:t})$. The term $w_{mc} \in \mathbb{R}^+$ is a reward term, $\tau \in \mathbb{N}$ indicates the lifetime in time steps of a linear decay term, and δ denotes the Kronecker delta function:

$$\delta_{(i,j)} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{7.6}$$

Equation 7.5 calculates the probability that the ith vehicle executes maneuver M at time step $t+1$ following the same idea as Equation 6.7. That is, a maneuver of vehicle i will be given an exponentially higher probability if, given the estimated states and maneuvers of all other vehicles, it accrues a lower expected cost over T_m timesteps compared to all other available maneuvers. However, in this case we consider an additional term that rewards maneuver continuity across time steps, increasing thus the predicted probability of maneuvers that were estimated likely for obstacle i. This effect is reduced as the depth in the tree increases using a linear decay term. When $t \geq \tau$ this reward term is set to 0 and the observations are sampled according only to what the model considers reasonable driving behavior. This conceptually makes sense. At the root of the history tree, the belief over the state and maneuver intentions of all obstacles has been estimated using the model and the real dynamic observations provided by the perception system. However, as we go deeper into the history tree (that is, into the future), the observed dynamics of the targets at the root node become less relevant and so we rely increasingly in the model to predict the behavior of the obstacles.

The expectation in Equation 7.5 is solved using Monte Carlo sampling. The procedure is detailed in Algorithm 5. In line 5 of the algorithm, a state and maneuver are sampled for all obstacles in the scene from the available posterior distributions at time t. The maneuver sampled for the ith obstacle, for which the predictive distribution is being calculated, is then used in line 11 to determine the activation of the maneuver continuity reward term. Consider the case in the right-side graph from Figure 7.2; if the target has been estimated to be performing a left lane change with high probability, this will be the maneuver that is consistently sampled and promoted in the predicted probability. As a consequence, given a correct tuning of the reward

Algorithm 5: Maneuver forecasting algorithm to be used during the sampling procedure.

Inputs : $P(\mathbf{x}_t^e|\mathbf{z}_{1:t})$: posterior over the physical state of the ego-vehicle
$P(\mathbf{x}_t^{1:N}, m_t^{1:N}|\mathbf{z}_{1:t})$: posterior over the physical state and maneuver intention of all obstacles
$h_t m_{t+1}^e$: history at time t appended with the maneuver for the ego-vehicle at $t+1$
N_{sm}: number of samples to approximate the expectation
T_m: number of simulation steps to determine the consequences of a maneuver
i: target obstacle
τ: duration in time steps of the decay term
$\mathbf{w}$: weights of the cost function learned from demonstrations
w_{mc}: additional weight term to reward maneuver continuity

Output : $P(m_{t+1}^i|\mathbf{x}_t^i, m_t^i)$

ManeuverForecasting
  for $m_{t+1}^* \in m_{t+1}^i$ **do**
    for *sample* $= 1$ **to** N_{sm} **do**
      `/* The hat notation indicates samples */`
      $\hat{\mathbf{x}}_t^e \leftarrow$ `sample(`$P(\mathbf{x}_t^e|\mathbf{z}_{1:t})$`)`
      $\hat{\mathbf{x}}_t^{1:N}, \hat{m}_t^{1:N} \leftarrow$ `sample(`$P(\mathbf{x}_t^{1:N}, m_t^{1:N}|\mathbf{z}_{1:t})$`)`
      accum_cost $= 0$ `/* Accumulates the cost over simulation steps */`
      for *simulation time step* $k = 1$ **to** T_m **do**
        $\hat{\mathbf{x}}_{t+k}^e \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^e$, *maneuver* $= m_{t+1}^e$`)`
        $\hat{\mathbf{x}}_{t+k}^{-(i,e)} \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^{-(i,e)}$, *maneuver* $= \hat{m}_t^{-(i,e)}$`)`
        $\hat{\mathbf{x}}_{t+k}^i \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^i$, *maneuver* $= m_{t+1}^*$`)`
        accum_cost+ $= \mathbf{w}^T \mathbf{f}^i \left([\hat{\mathbf{x}}_{t+k}^i, \hat{\mathbf{x}}_{t+k}^{-(i,e)}, \hat{\mathbf{x}}_{t+k}^e] \right) - w_{mc}\, \delta_{(m_{t+1}^*, \hat{m}_t^i)} \left(\frac{\tau - t}{\tau} \right)$
      $P(m_{t+1}^i = m_{t+1}^*|\mathbf{x}_t^i, m_t^i)$ += exp (−accum_cost)
    $P(m_{t+1}^i = m_{t+1}^*|\mathbf{x}_t^i, m_t^i)$ /= N_{sm}
  `normalize(`$P(m_{t+1}^i|\mathbf{x}_t^i, m_t^i)$`)`
  return $P(m_{t+1}^i|\mathbf{x}_t^i, m_t^i)$

parameter w_{mc}, the upper layers of the history tree will sufficiently cover the case in which the obstacle continues its dangerous lane change, even if it contradicts the model-based prediction. Although not explicitly stated in the algorithm, we can choose to apply the maneuver continuity reward only to particular maneuvers that might be underrepresented in the tree otherwise. In the experimental section from this chapter, we will apply this term only to the lane change maneuver.

7.2.4 Rollouts

In POMCP, as in our proposed approach, the value function for each action node is maintained using the mean return of all simulations passing through that node (further details on this will be provided in subsection 7.2.5). We use the notation $\bar{V}(h_t)$ to refer to the estimated value of a history, differentiating it from the value notation given in Equation 2.31. These estimated values are important in our online POMDP setting, as they are used at each execution step to select the most advantageous action for the robot. Hence, these values should be accurately estimated. However, as discussed in the introductory section, the tree expansion is truncated to a given depth d_{lim} for computational tractability. Beyond that point, the estimation of the accrued cost is done using a rollout policy.

Although using random rollout policies can provide adequate value estimates (Goldhoorn et al., 2014), it is accepted in general that exploiting domain knowledge leads to superior

performance (Silver et al., 2008). In our highway driving domain, we use a variant of the model-based prediction approach from chapter 5 to determine both the rollout policy and the environment dynamics (i.e. how the environment reacts to the ego-vehicle's actions). The intuition of these model-based rollouts is shown in the two-vehicle situation from Figure 7.3. In this case, we address the first experimental scenario, in which the ego-vehicle can choose to perform lane change maneuvers, but its acceleration is set automatically using the IDM car-following model. The evolution of the scene is estimated by predicting the actions of the involved vehicles from the front to the back of the scene.

The procedure is as follows. First, in Figure 7.3, states are sampled for all vehicles in the scene from the starting belief distribution (shown as a gray blob in the figure). Once the states are sampled, we process the vehicles in order from the front to the back. For the leading vehicle, we evaluate the cost associated to each of its available maneuvers (left lane change or lane keeping) in steps 1 and 2. To calculate the cost of a maneuver, the trailing vehicles are always assumed to be performing a lane keeping maneuver, and their acceleration adjusted using the IDM.

In the rollouts, we employ a time step that can be a multiple of the time step used during the tree search:

$$dt_{\text{ROLL}} = \Psi \cdot dt_{\text{TREE}} \tag{7.7}$$

where $\Psi \in \mathbb{N}$. The idea here is to have a time step longer than the duration of any maneuver (e.g. a lane change). However, during the simulation of a maneuver, intermediate states are sampled each dt_{TREE}. These intermediate states are used to update the acceleration of the vehicles. The cost of a maneuver is calculated as the sum of costs at the corresponding intermediate states. After obtaining the costs of the different maneuvers, the policy for the current time step is obtained using the Boltzmann distribution (step 3). Maneuvers with lower associated costs will be exponentially more likely for each vehicle. In step 4, a maneuver $\hat{m}_t^i$ is sampled from the calculated policy for obstacle i. In the figure, is it assumed that the sampled maneuver is lane keeping. In the cost calculations of all remaining vehicles, it will be assumed that vehicle i is performing a lane keeping maneuver.

The presented procedure for vehicle i is repeated for all vehicles in the scene. After all vehicles have been processed, their states are propagated forward using the sampled maneuvers and a new rollout step begins. The procedure continues until the maximum depth d_{max} is reached. The rollout simulation is repeated N_{sr} times, and the average discounted cost accrued by the ego-vehicle is returned. The whole process we have described is detailed in Algorithm 6.

7.2.5 Algorithm

Our tactical decision-making approach is formulated in Algorithm 7. This algorithm resembles the POMCP and POMCP-DPW algorithms (Silver and Veness, 2010; Sunberg and Kochenderfer, 2018), using MCTS to explore a search tree of histories. However, in these algorithms the belief is maintained using particle filtering, whereas we rely on a variational approach to maintain parametric beliefs for each history in the tree.

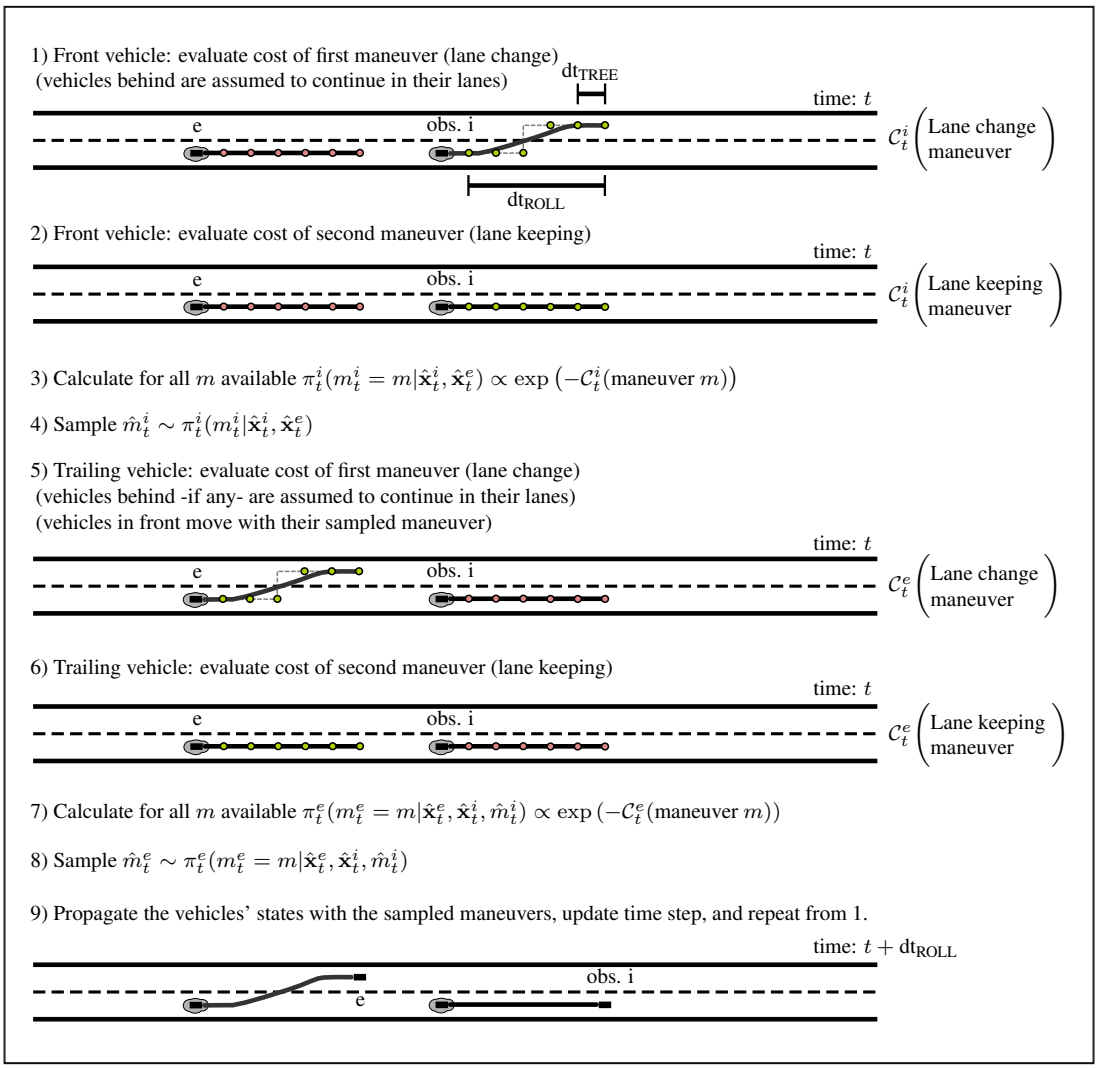


Fig. 7.3.: Example of the first rollout step on a traffic situation with two vehicles (the ego-vehicle and one obstacle). In this scenario, the ego-vehicle can choose between lane keeping and lane changing maneuvers. The step duration in the rollouts $dt_{\text{ROLL}} = \Psi \cdot dt_{\text{TREE}}$ is a multiple of the step duration in the history tree, allowing a complete lane change in one step. The rollout is produced in a similar way as the model-based prediction algorithm from chapter 5. From the front to the back of the scene, the cost associated to the different available maneuvers is calculated. Based on this cost, a model-based policy that encodes the normal behavior of a driver is obtained, from where a maneuver is sampled. The process continues until all vehicles have been processed. After that, the state of the vehicles is propagated using the sampled maneuver, and a new rollout step starts. The rollout simulation executes until the maximum depth d_{max} is reached. In the figure, the dots indicate the sampled states considered during the cost evaluation of each maneuver.

Algorithm 6: Algorithm to estimate accrued cost using rollout simulations.

Inputs : h - history at which the rollout starts.
b_h - belief distribution at history h
d - depth on the tree of h

Output : c_{avg} - average discounted cost accrued by the ego-vehicle across rollout simulations

Rollout
  $c_{avg} \leftarrow 0$ `/* Initialize average discounted cost */`
  for *sim* $= 1$ **to** N_{sr} **do**
    $c_{sim} \leftarrow 0$ `/* Stores this simulation's cost */`
    $\gamma_{sim} \leftarrow \gamma$ `/* γ is the POMDP's discount factor */`
    $\hat{\mathbf{x}}_t^e, \hat{\mathbf{x}}_t^{1:N} \leftarrow$ `sample(`b_h`)`
    while $d \leq d_{max}$ **do**
      for $i = 1$ *(front vehicle)* **to** $N + 1$ *(last vehicle)* **do**
        accum_cost $\leftarrow$ `initialize_array_zeros(`*size* $=$ *available_maneuvers*($\hat{\mathbf{x}}_t^i$))
        for *sim_step* $k = 1$ **to** Ψ **do**
          `/* Each step lasts` dt_{tree} `*/`
          for $m \in$ *available_maneuvers*($\hat{\mathbf{x}}_t^i$) **do**
            $\hat{\mathbf{x}}_{t+k}^i \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^i$, *maneuver* $= m$)
            $\hat{\mathbf{x}}_{t+k}^{j<i} \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^j$, *maneuver* $= \hat{m}_t^j$)
            $\hat{\mathbf{x}}_{t+k}^{j>i} \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^j$, *maneuver* $= LK$)
            accum_cost$[m]$ $+= \mathbf{w}_m^T \mathbf{f}^i \left([\hat{\mathbf{x}}_{t+k}^i, \hat{\mathbf{x}}_{t+k}^{-i}]\right)$
        $\pi_t^i(m_t^i = m | \hat{\mathbf{x}}_t^e, \hat{\mathbf{x}}_t^{1:N}) \propto \exp\left(-\text{accum\_cost}[m]\right)$
        $\hat{m}_t^i \leftarrow$ `sample(` $\pi_t^i(m_t^i | \hat{\mathbf{x}}_t^e, \hat{\mathbf{x}}_t^{1:N})$)
      `/* Get new states:` $\hat{\mathbf{x}}_{t+dt_{\texttt{roll}}} = \hat{\mathbf{x}}_{t+\Psi dt_{\texttt{tree}}}$ `*/`
      for *sim_step* $k = 1$ **to** Ψ **do**
        for $i = 1$ *(front vehicle)* **to** $N + 1$ *(last vehicle)* **do**
          $\hat{\mathbf{x}}_{t+k}^i \leftarrow$ `propagate(`$\hat{\mathbf{x}}_{t+k-1}^i$, *maneuver* $= \hat{m}_t^i$)
          if $k == \Psi$ **then**
            $\hat{\mathbf{x}}_{t+dt_{roll}}^i = \hat{\mathbf{x}}_{t+k}^i$
        $c_{sim} += \gamma_{sim}^k \mathbf{w}_m^T \mathbf{f}^e \left([\hat{\mathbf{x}}_{t+k}^e, \hat{\mathbf{x}}_{t+k}^{-e}]\right)$
      `/* Update depth, time step, and` γ_{sim} `*/`
      $d = d + 1$
      $t = t + dt_{roll}$
      $\gamma_{sim} = \gamma^{\Psi}$
    $c_{avg} += c_{sim}$
  return c_{avg}/N_{sr}

Online tactical planning The tactical planning starts in the `PLAN` procedure. During the allocated planning time, simulations are continuously run from the initial history h_0 and corresponding belief b_0 to construct and explore the search tree of histories. A simulation consists of a sequence of sampled actions and observations, ending on a leaf node from where a rollout is executed (see Figure 7.1 for two exemplary simulations). The goal of the tree exploration is to maintain accurate estimates of the value $\bar{V}(h_0 a)$ of the actions that can be taken from the current belief state, as they will be used when the planning time runs out to select the next action for the robot (line 4).

History tree construction and exploration The simulations are performed in the `SIMULATE` procedure, which takes as parameters a history h, and its corresponding belief

Algorithm 7: Algorithm for human-like tactical decision-making under uncertainty in highways.

1 **procedure** PLAN (h_0, b_0) **:**
2 **while** *not timeout* **do**
3 SIMULATE $(h_0, b_0, 0)$
4 **return** $\arg\min_a \bar{V}(ha)$

5 **procedure** SIMULATE (h, b_h, d) **:**
6 **if** $d \geq d_{lim}$ **then**
7 **return** ROLLOUT (h, b_h, d) // Algorithm 6
8 **else**
9 **if** $h \notin$ *history-tree* **then**
10 add-to-tree (h)
11 **for** $\forall a \in$ available-actions (b_h) **do**
12 init $(\bar{V}(ha), N(ha), N(h))$
13 **return** ROLLOUT (h, b_h, d) // Algorithm 6
14 **else**
15 $a \leftarrow \arg\min_{a'} \bar{V}(ha') - c\sqrt{\frac{\log N(h)}{N(ha')}}$
16 **if** $|$children $(ha)| \leq$ floor $(k_0 N(ha)^{\alpha_0})$ **then**
17 $b_{hao}, o \leftarrow$ dbn-model (b_h, a) // Algorithms 4,5; Eqs 7.3, 7.4
18 children $(ha) \leftarrow$ children $(ha) \cup \{o\}$
19 accum $\leftarrow$ COST (b_{hao}, a) + γ SIMULATE $(hao, b_{hao}, d+1)$
20 **else**
21 $o' \leftarrow$ sample-observation(children (ha))
22 accum $\leftarrow$ COST $(b_{hao'}, a)$ + γ SIMULATE $(hao', b_{hao'}, d+1)$
23 $N(h) \leftarrow N(h) + 1$
24 $N(ha) \leftarrow N(ha) + 1$
25 $\bar{V}(ha) \leftarrow \bar{V}(ha) + \frac{\text{accum} - \bar{V}(ha)}{N(ha)}$
26 **return** *accum*

27 **procedure** COST (b, a) **:**
28 cost $\leftarrow 0$
29 **for** $i = 1$ **to** N_c **do**
30 $\hat{\mathbf{x}}^e, \hat{\mathbf{x}}^{1:N} \leftarrow$ sample-states (b)
31 cost += $\mathcal{C}_a^e([\hat{\mathbf{x}}^e, \hat{\mathbf{x}}^{1:N}])$
32 **return** $cost/N_c$

b_h and tree-depth d. To explore the tree and estimate the value functions, this procedure recursively calls itself or the routine ROLLOUT, which is executed using Algorithm 6. Rollouts are performed in two cases. First, when the tree maximum depth d_{lim} is reached (line 7); and second, each time a new history is found (line 13). In the latter case, before the rollout is executed, the new history found is added to the tree (line 10) and its children (action nodes) initialized (line 12). We initialize the value function of the action nodes $\bar{V}(ha)$ and the number of times they have been visited both to 0. Different initialization schemes to leverage domain knowledge would also be possible (Silver and Veness, 2010).

The history tree exploration is based on selecting actions using the UCT algorithm (Kocsis and Szepesvári, 2006), and sampling observations using the method presented in subsection 7.2.3. From any given history node h, actions are selected based on the value of their associated action node $\bar{V}(ha)$ and on an exploration bonus that promotes less visited actions (line 15). The value of $c \in \mathbb{R}$ controls the exploration (searching in unexplored areas of the tree) and exploitation (searching in promising parts of the tree) balance. Note that the exploration

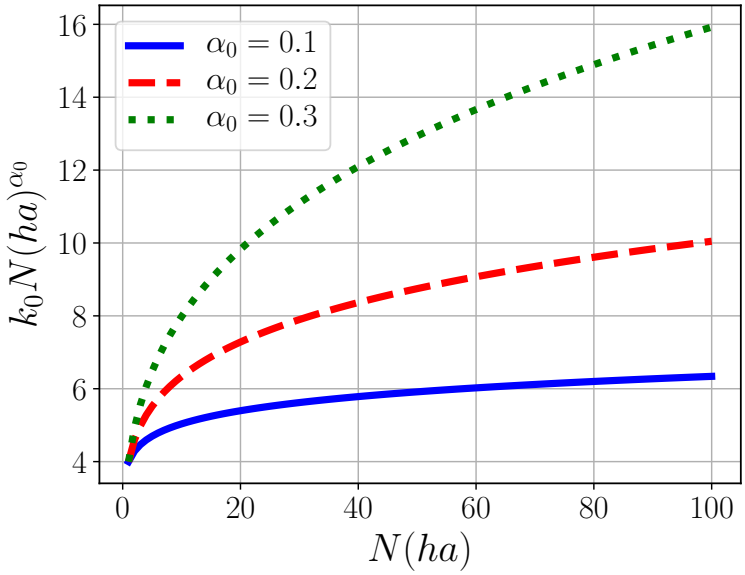


Fig. 7.4.: Progressive widening: Number of observations allowed per action node as a function of the number of visits to the node. The results are shown for $k_0 = 4$ and different settings of the parameter α_0. In practice, the observation limit is rounded down to the nearest integer.

bonus for untried actions is $-\infty$, and therefore no action resampling occurs until all actions have been tried. The next step after selecting an action is to sample an observation from the predictive distribution resulting of the action execution. In order to focus the sampling towards likely traffic situations, we follow the procedure discussed in subsection 7.2.3. The sampled observation is then used to obtain the belief b_{hao} of the new history node using Algorithm 4 (with the modifications discussed in subsection 7.2.2).

Sampling from a continuous observation space The observations are sampled from a continuous predictive distribution (Equations 7.3 and 7.4). Sampling twice the same observation from this continuous distribution is unlikely and, in consequence, applying the vanilla POMCP algorithm results on a shallow history tree that does not grow beyond the first layer of history nodes. To circumvent this problem, we follow the approach proposed by Sunberg and Kochenderfer, artificially limiting the number of observations via progressive widening (Sunberg and Kochenderfer, 2018; Couëtoux et al., 2011). This technique limits the number of children of an action node to $k_0 N(ha)^{\alpha_0}$, where $N(ha)$ is the number of times that the action node with history ha has been visited, and k_0 and α_0 are the parameters that control the widening. To illustrate how this works, Figure 7.4 shows the increasing number of allowed observations under the action node ha as the visits to this node $N(ha)$ increase, for $k_0 = 4$ and different values of α_0. The application of progressive observation widening enables the exploration of the lower layers of the history, while progressively *unpruning* the upper layers to permit further exploration. For this reason, this technique is also known as progressive unpruning (Browne et al., 2012).

In Algorithm 7, the condition that checks if the limit in the number of observations for an action node has been reached corresponds to line 16. If more observations are allowed, we

sample a new observation, add it to the list of children of the corresponding action node, and recursively continue the simulation from the new history (lines 17-19). Otherwise, we randomly sample an observation from the previously obtained samples and continue the tree exploration in that branch (lines 21-22).

Estimating the values at the action nodes The key idea of this algorithm, as in POMCP, is to use Monte Carlo simulations through the history tree to estimate the values of the action nodes. The value estimate at each action node is calculated as the average discounted cost achieved by the simulations passing through that node (line 25). To calculate the immediate cost at each step of the simulation (lines 19 and 22) we need to evaluate the cost of a belief. We do this by sampling physical states for all vehicles (line 30), which enables us to evaluate the cost using the model learned in chapter 4.

Computational complexity We can bound the complexity of each tree simulation. Let $|N|$ denote the number of vehicles in the scene and $|M|$ the number of available maneuvers (we assume the same maneuvers are available for all vehicles). In the first place, the complexity of performing maneuver forecasting (that is, calculating $P(m_{t+1}^i|\mathbf{x}_t^i, m_t^i)$) for a single vehicle with Algorithm 5 is $O(|M|N_{sm}T_m)$. Then, the complexity of performing a belief update with Algorithm 4 is about $O(|N||M|^2CN_{sm}T_m)$. Finally, the complexity of a rollout is given by $O(N_{sr}|N||M|\Psi d_{max})$. Aggregating all these results, we can approximately bound the complexity of a tree simulation by $O(d_{max}|N||M|^2CN_{sm}T_mN_{sr}\Psi)$. That is, the time complexity of the algorithm grows quadratically with the number of possible maneuvers and linearly on the number of vehicles and the depth of the tree.

7.3 Experimental evaluation

The goal from this chapter is to integrate the modeling and prediction results from the previous chapters into a tactical decision-making model in order to make foresighted and safe driving decisions in highways. To analyze the ability of the proposed approach to make such driving decisions, we rely on a simulation platform capable of realistically simulating the behavior of surrounding drivers as a response to the ego-vehicle's actions. This platform is presented in subsection 7.3.1. In particular, we will analyze two specific traffic situations that can help visualize very clearly the operation of our approach. These situations are presented in detail in subsection 7.3.2, and the results are presented in subsection 7.3.3.

7.3.1 Simulation platform

For the initial evaluation of a decision-making approach for autonomous vehicles it seems sensible to rely on a simulation platform. The main requirement for this simulator will be to generate human-like behaviors for all the vehicles in a given traffic scene. The vehicles should react to the actions of the surrounding traffic, including those of the ego-vehicle. Additionally, the model to generate this human-like behavior should ideally be different from the one we use for prediction. Under consideration of these constraints, we have chosen to run the performance analysis on a customized version of an open-source driving simulator (Garzón and Spalanzani, 2018). This simulator builds upon two existing open-source simulation packages:

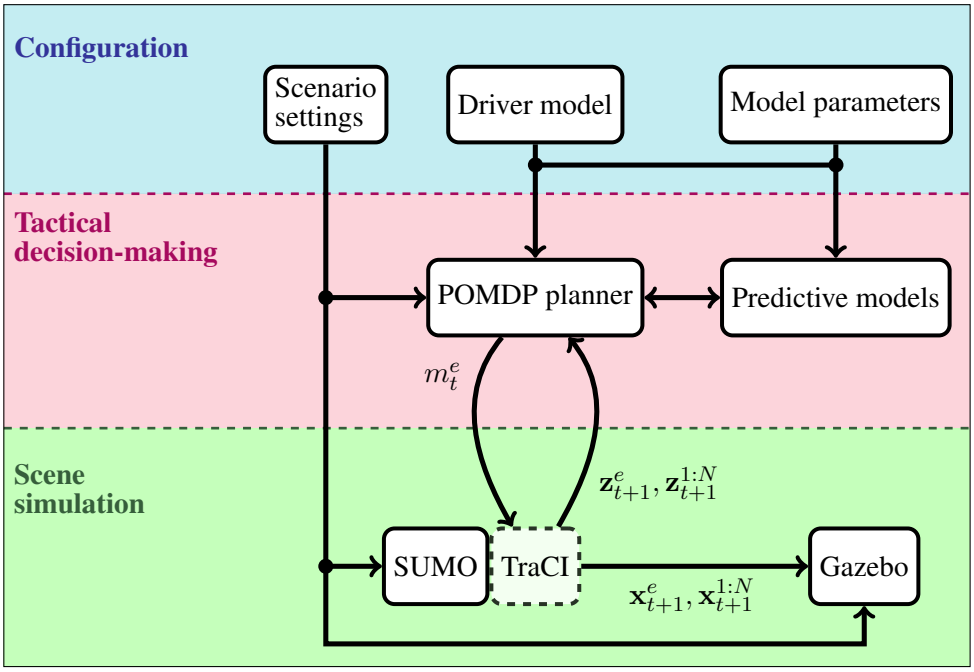


Fig. 7.5.: Architecture of our highway simulation platform. The user can configure the settings of the traffic scene (number of lanes and connectivity, number of obstacles and their preferences, speed limits, etc.) and the parameters of the driver model used in the POMDP and predictive models. In our case, this model is learned from demonstrated driving data as shown in chapter 4. At each time step, our POMDP planner selects a maneuver for the ego-vehicle and then SUMO simulates the state transitions for all vehicles. The resulting physical states of all vehicles are passed to Gazebo for visualization. The POMDP planner receives a noisy observation of the physical states and then proceeds to search for the next maneuver to execute.

1. Simulation of Urban MObility (SUMO) (Krajzewicz et al., 2012). SUMO is an open-source microscopic road traffic simulator. As a microscopic model, SUMO simulates traffic flows by explicitly considering the interactions that drivers have with the road network and with other drivers.

2. Gazebo (Koenig and Howard, 2004). Gazebo is an advanced 3D simulation environment compatible with the Robot Operating System (ROS). It enables the simulation of the vehicle's dynamics and sensors, and the 3D visualization of the traffic scene.

Figure 7.5 shows a diagram that illustrates how the simulator's components interact with the proposed decision-making system. The diagram distinguishes three layers in the system. The upper layer contains all aspects of the simulation that can be configured by the user. In the first place, we can finely configure the settings of the highway traffic scene. The configurable parameters include the number of lanes, their lengths, widths and connections to each other, the direction of traffic in the lanes, and the number of vehicles and their particular routes and behavioral models. Another aspect that can be configured by the user is the driver model used for selecting actions in the POMDP planner and to predict the behavior of surrounding drivers. In our experiments, we will use the model that was learned from demonstrated driving data in chapter 4.

The intermediate layer contains the POMDP planner and the predictive models used for the belief updates and the rollouts. After each planning cycle, the POMDP planner chooses the action m_t^e to be executed by the ego-vehicle. This action is passed to SUMO via the Traffic Control Interface (TraCI), which allows online manipulation of the states and behaviors of the simulated objects. A simulation step is then executed by SUMO, leading to the new physical states $(\mathbf{x}_{t+1}^e, \mathbf{x}_{t+1}^{1:N})$ of all vehicles, which are passed to Gazebo to update the visualization. The POMDP planner receives noisy versions of the new physical states and proceeds to plan the next action for the ego-vehicle.

An important aspect of the simulation platform is how SUMO determines the interaction-aware behavior of each highway traffic participant, which involves mainly the longitudinal acceleration and the lane-change actions. To set the acceleration of a vehicle, SUMO relies on a car-following model, offering several options to choose from. For our experiments, we selected the Krauß model (Krauß, 1998). On the other hand, the lane-changing model from SUMO is based on a large, intricate combination of rules (Erdmann, 2015). Essentially, a vehicle can perform a lane change due to four different reasons:

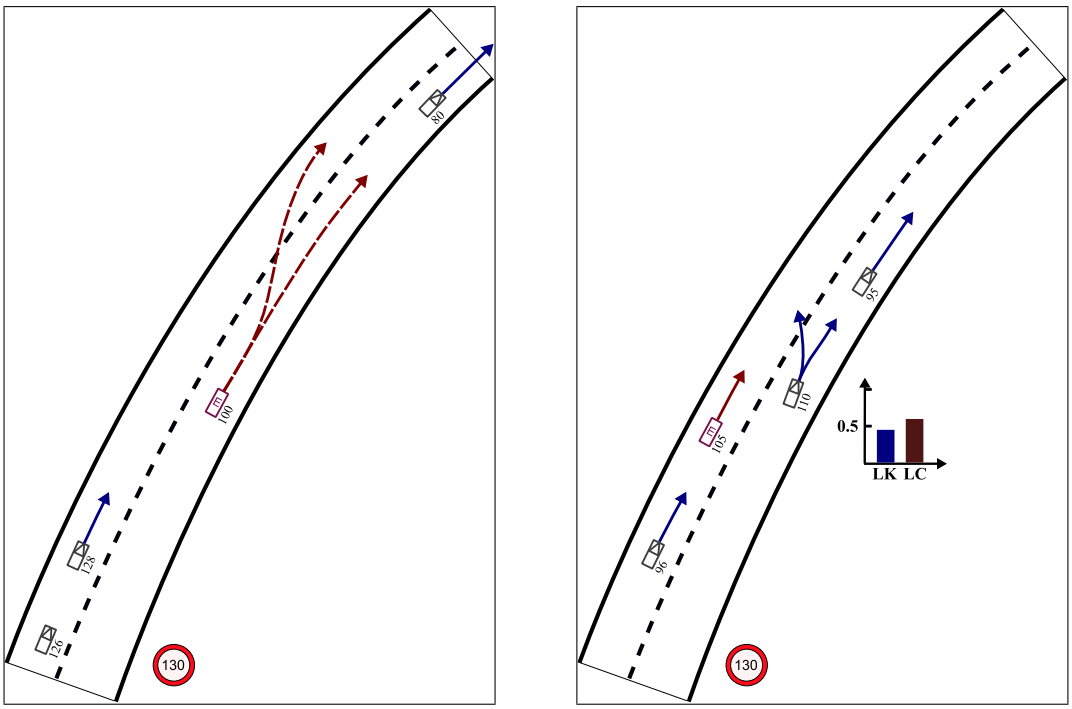
1. Strategic: necessary lane-change so that the vehicle can follow its predefined route (for example, at a highway interchange).
2. Cooperative: to help another vehicle change lanes (for example, at an entrance ramp).
3. Tactical: to avoid driving behind a slow vehicle.
4. Regulatory: to abide by the traffic rules (for example, avoid blocking the left-most lane).

The situations that will be considered in our experiments involve mainly tactical and regulatory lane changes. This is in line with our models from chapters 5 and 6 and constitutes thus a fair evaluation setting for our decision-making approach.

7.3.2 Traffic situations considered

We consider two different tasks for the evaluation of the proposed approach. In the first evaluation task, the planner takes control of the lane change maneuvers of the ego-vehicle. In this task, the longitudinal acceleration is automatically set using the IDM car-following model. This task is illustrated in Figure 7.6a. The goal of this task is to evaluate the proposed approach for its ability to consider the long-term consequences of any action. This was one of the key requirements set for the planner in chapter 1.

In the second task, the ego-vehicle navigates in the left lane of a two-lane highway and the goal of the POMDP planner is to choose its acceleration commands following the driving preferences learned in chapter 4. To drive safely and avoid falling into dangerous situations, the planner is required to anticipate the lane change intentions of the vehicles circulating on the right lane, even when these intentions do not fit with a risk-averse driving style. This task, illustrated in Figure 7.6b, will test the ability of the proposed approach to consider the short-term consequences of the selected actions by exploiting the dynamics-based behavior estimations.



(a) In the first evaluation task, the POMDP planner takes control of the lane-changing actions of the ego-vehicle, while the longitudinal control is delegated to a car-following model. This scenario helps to illustrate the ability of the proposed approach to select actions by considering their long-term consequences.

(b) In the second evaluation task, the POMDP planner selects acceleration actions for the ego-vehicle by considering the uncertain lane-change intentions estimated for the vehicles driving in the right lane. This scenario helps us illustrate how the planner leverages available behavior estimations to construct the history tree and select safe actions for the ego-vehicle.

Fig. 7.6.: Highway scenarios proposed for the evaluation of the tactical decision-making approach.

7.3.3 Configuration and results

The meaning of the more relevant parameters of our planning approach, as well as the values selected for them in each of the two experimental tasks are shown in Table 7.2. In this subsection we discuss the configuration selected for each of the two experimental tasks and present the results obtained in different traffic scenes.

Task 1: Planning lane change actions

Configuration The main goal of this task is to show that the decisions made by the planner are:

- Foresighted: the long-term consequences of any decision are considered.
- Human-like: the resulting driving behavior is consistent with the driving demonstrations used to obtain the driver model in chapter 4.
- Coherent: the planner does not continuously 'change its mind' about the best course of action to take in a given traffic situation.

With this goal in mind, we set the time step of the POMDP to $2\,\mathrm{s}$, the tree depth d_{lim} to 3, the maximum depth of the rollouts d_{max} to 7 and the discount factor γ to 0.9. With this setup, the

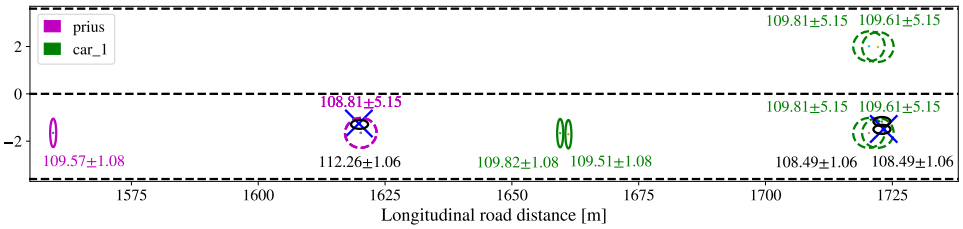
Scope	Parameter	Meaning	Task 1	Task 2
Tree construction and exploration	N_{sim}	Number of tree simulations	100	400
	d_{lim}	Maximum depth for tree exploration	3	4
	d_{max}	Maximum depth of rollouts	7	5
	c	Exploration constant	50	50
	k_0	Progressive widening constant	2	2
	α_0	Progressive widening exponent	0.3	0.2
	N_s	Number samples belief cost evaluation	5	5
POMDP	dt	Time step	2.0	1.0
	γ	Discount factor	0.9	0.9
	$\mathbf{w}$	Cost weight vector	Chap. 4	
Maneuver forecasting	w_{mc}	Reward for maneuver continuity	0	25
	τ	Linear decay term steps	-	-
	T_m	Number of steps maneuver evaluation	16	16
	dt_m	Time step maneuver evaluation	0.25	0.25
	N_{sm}	Number of samples maneuver evaluation	5	30
Rollouts	N_{sr}	Number of rollout executions	5	5
	Ψ	Rollout time step multiplicator	1	1
Dynamics	v_{LC}	Lateral velocity of lane change maneuver	2.2	2.2
	L	Lane width	3.3	3.3
	dt_d	Time step dynamics integration	0.25	0.25
	IDM	IDM car-following model parameters	Chap. 6	
Intention estimation	C	Number of Gaussian components	1	1

Tab. 7.2.: Recap of the parameters in the proposed tactical decision-making approach and values selected for the two experimental tasks.

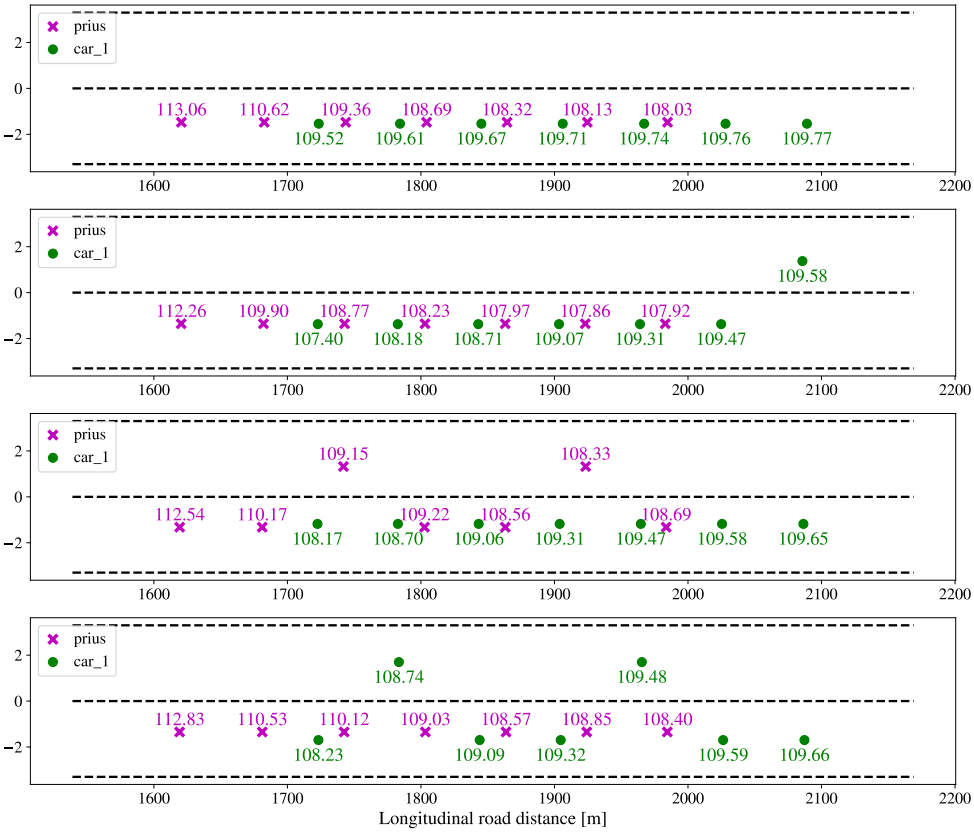
consequences of any action up to 14s into the future will influence the decision taken by the planner. The parameter w_{mc} is set to 0 in this task, indicating that in the simulations the actions of surrounding traffic will be distributed according to the model-based prediction, disregarding thus potentially dangerous situations in the short-term future. In this experimental task, we consider two possible maneuvers: lane keeping (LK) or lane changing (LC).

Although not shown in Table 7.2, the noise parameters of the dynamics have been configured so that there is sufficient uncertainty in the velocity and longitudinal position of the vehicles to account for the imperfection of the car-following model, which does not necessarily reflect the behavior of surrounding drivers (in simulation or in real life). As stated before, we rely on the IDM car-following model to control the acceleration of the ego-vehicle and to predict the transitions of the obstacles. However, we configured SUMO to control all obstacles using a different car-following model, namely the Krauß model (Krauß, 1998). The effects of this mismatch should be neutralized by the proposed Gaussian dynamics.

An exemplary belief update step is shown in Figure 7.7a. In this scene, the ego-vehicle (marked with the label 'prius') is trailing behind an obstacle (label 'car_1'). The narrow ellipses at around 1550m and 1660m indicate the 2σ error ellipses for the ego-vehicle and the obstacle at the initial time step, respectively. These ellipses indicate the initial uncertainty in the x-y



(a) Belief update step after selecting a LK maneuver for the ego-vehicle (prius). The predictive distributions are illustrated by the dashed 2σ error ellipses. An observation is sampled from the predictive distributions (blue crosses indicate the sampled physical states) and used to perform the KF measurement step. The expanded posterior distribution for the obstacle is collapsed back to two components. The x-y components of the final belief are illustrated by the black ellipses.



(b) Example of possible rollouts. For each vehicle, the rollout is illustrated as a sequence of markers showing its states at the different rollout steps. A label showing the velocity of the vehicle is also shown next to the markers. In the proposed rollout policy, a maneuver is exponentially more likely when it leads to states with a lower cost. Given that in this traffic scene the difference in cost between driving in the right or left lane is not large, sometimes a lane change maneuver is sampled.

Fig. 7.7.: Examples of a belief update step and possible rollouts in a traffic scene involving two vehicles.

location of the vehicles. Next to them, a label shows their mean velocity and its standard deviation. Note that since we are using $C = 1$ to track the posterior of the obstacles, i.e., a single Gaussian component per maneuver, two ellipses are shown for the obstacle. In this setting, a LK maneuver is selected for the ego-vehicle and its state is propagated forward accordingly—the dashed magenta ellipse is the predictive Gaussian state distribution. For the obstacle, each Gaussian component is propagated forward with each of the available maneuvers,

resulting in 4 predictive Gaussian distributions. Due to the scaling of the figure, these predictive distributions look rather circular. However, it should be clear that they are elongated along the longitudinal direction of the road to account for the uncertainty in the longitudinal velocity and position of the vehicles. Once all predictive distributions are obtained, an observation is sampled (blue large crosses) and used to perform the measurement step of the Kalman filter. For the obstacle, the 4 Gaussians are also collapsed back to the initial 2 components (shown in black). The sampling of the observation for the obstacle is biased by the predictive probability $P(m^i_{t+1}|\mathbf{x}^i_t, m^i_t)$ (Equation 7.5). In this case, with no reason to perform a lane change, the probability of sampling a physical state for the obstacle corresponding to a LK maneuver was significantly larger, and that is the case that was represented in the figure.

In each tree simulation, after sampling a novel observation and performing a belief update, a rollout simulation is executed. Figure 7.7b shows 4 different examples of rollouts executed from the final posterior distribution shown in Figure 7.7a. In this case, the rollout starts at the second layer of the tree and consists therefore of 6 time steps. For each vehicle, the rollout is illustrated as a sequence of markers showing its states at the different rollout steps. A label showing the velocity of the vehicle is also shown next to the markers. As discussed in subsection 7.2.4, the rollout policy is determined by calculating the cost associated to each possible maneuver. A maneuver will be exponentially more likely when it leads to states with a lower cost. In this relatively simple traffic scene, there is no reason for any of the vehicles to perform a lane change. This makes the first rollout example the most likely option. However, the cost of driving in the left lane is only marginally higher than doing so in the right lane. For this reason, sometimes the lane change maneuver is also sampled in the rollouts, as illustrated in the examples from Figure 7.7b. In more complex scenarios involving a larger number of vehicles, as we will see for instance in Figure 7.11, the gap in probability between the different alternative maneuver will be significantly larger, leading to more consistent rollouts.

Results obtained For this first experimental task, we evaluate the performance of our planner in three different typical highway scenarios. In the first one, illustrated in the top row of Figure 7.8, the ego-vehicle approaches a slow vehicle in the right lane. The numbers in brackets indicate the desired velocity of each vehicle in km/h. The remaining rows show, in order, the lateral position of the ego-vehicle in the road, the longitudinal distance to obstacle 'car_1', and the velocity of the ego-vehicle. The decisions of our planner are compared to those taken by the reactive SUMO model, which was introduced in subsection 7.3.1. In the lateral position subfigure, the triangular markers show the time instants at which our planner chose an action—every 2 seconds. As the ego-vehicle approaches the obstacle in front, the car-following acceleration model starts to decelerate the vehicle. The planner evaluates the cost of continuing in the same lane with a deceleration speed profile versus doing a lane change and, at a distance to the vehicle in front of around 70m, decides to execute a lane change. In contrast, the SUMO model continues for longer in the right lane, reducing its speed below 100km/h. This behavior can be explained by the method SUMO uses to avoid oscillations when making lane changing decisions. When a lane change is beneficial to the ego-vehicle, SUMO adds value to a counter at each simulation step. Then, when the value of that counter exceeds a given threshold, the

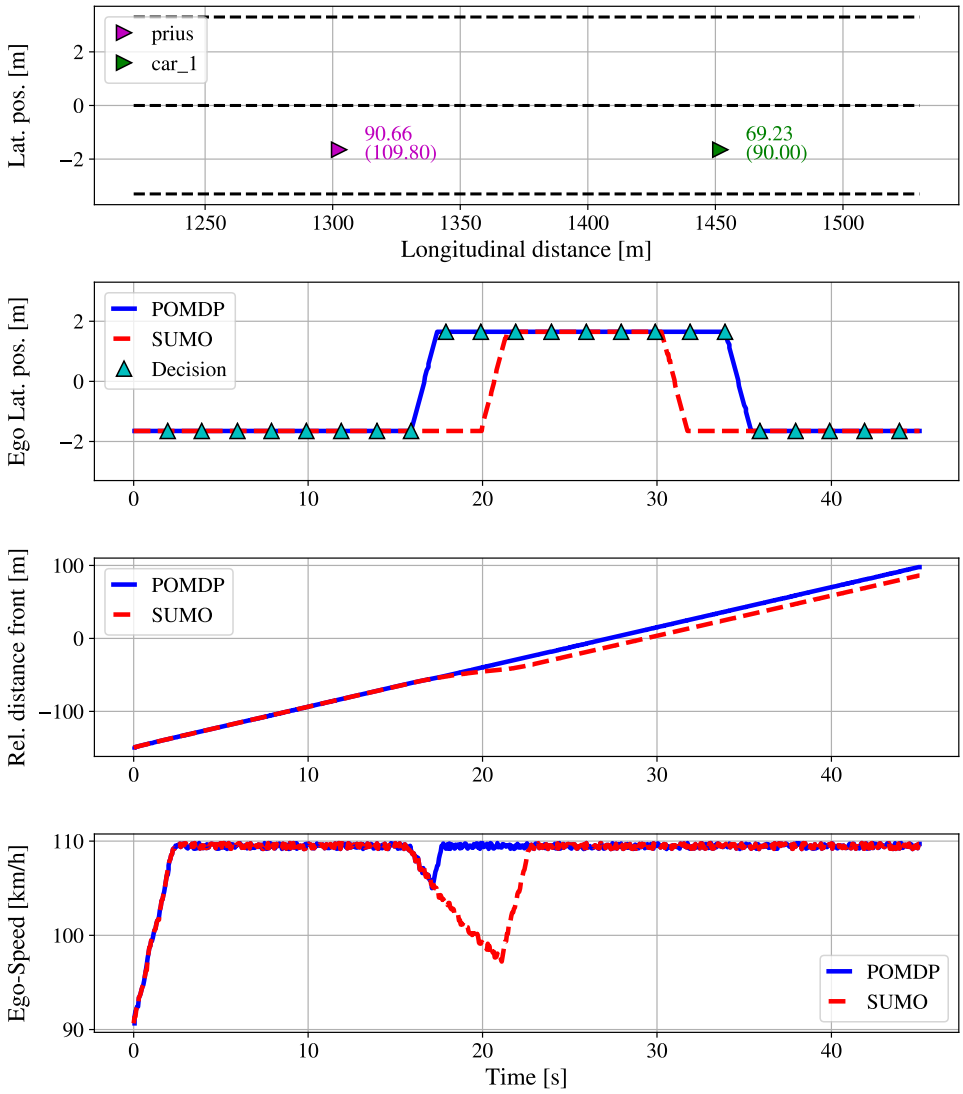


Fig. 7.8.: Planning results obtained for an easy highway overtake scenario. The POMDP planner recognizes the advantage in doing a lane change to avoid deviating too much from the desired speed.

lane change is executed and the counter reset. In contrast, our planner reaches stability by considering the repercussions of taking an action up to a sufficiently large time horizon.

The second scenario considered in shown in Figure 7.9. The situation resembles the previous traffic scene, except for the fact that the left lane is occupied by an obstacle, which makes the lane change maneuver extremely risky. In this case, both SUMO and our POMDP planner decelerate the ego-vehicle to match the velocity of the leading vehicle. Once the left lane is clear both models perform a lane change.

In the two exemplary traffic scenes discussed so far, there have not been significative differences between the decisions taken by the proposed POMDP planner and the reactive SUMO model. This is due to the fact that neither of those scenes truly required to analyze the long-term consequences of a maneuver. Figure 7.10 presents an scenario where neglecting the

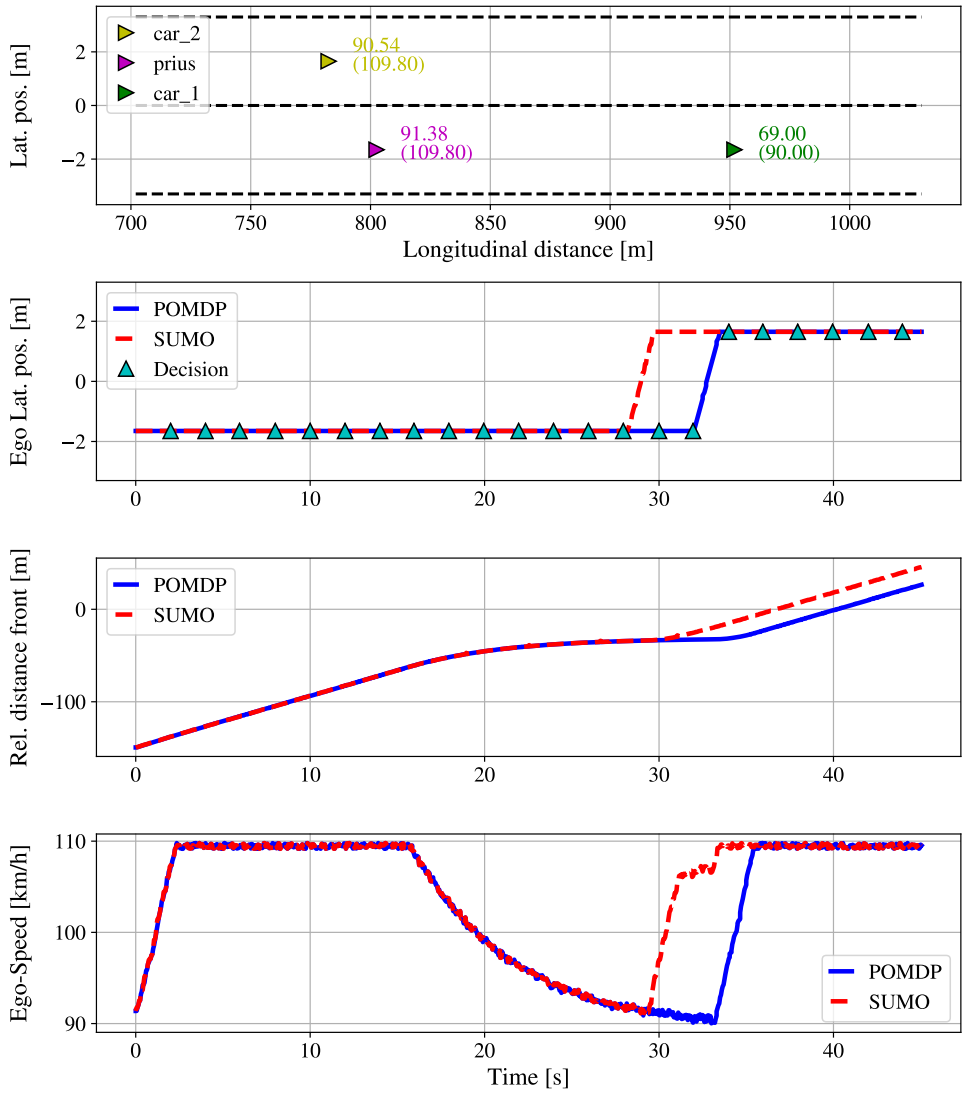


Fig. 7.9.: Planning results obtained for a delayed overtake scenario. As the left lane is occupied, both SUMO and our POMDP planner decelerate the ego-vehicle to match the velocity of the slow leading vehicle.

long-term future leads to a suboptimal decision. In this scenario, which was firstly discussed in subsection 1.2.2, the ego-vehicle is approaching a slow vehicle in lane 1 at a high relative velocity. The distance between both vehicles is still large (roughly 150m) but the gap is closing quickly. At the same time, lane 2 is free but some vehicles are approaching fast from behind. This is a traffic situation often faced by drivers in highways, where the slow vehicle is normally a truck. In this scene, SUMO's reactive model does not consider the need to perform a lane change until it approaches the vehicle in front, which is forcing the ego-vehicle to decelerate. Unfortunately, at that point the left lane is already blocked and the lane change is unfeasible. SUMO proceeds then to decelerate the ego-vehicle to match the velocity of the front vehicle. In contrast to SUMO's decision, the POMDP planner anticipates the high-cost associated to continuing in lane 1 and selects a lane change maneuver at the beginning of the scene. To visualize how the planner could have arrived to that decision, we show in Figure 7.11 two

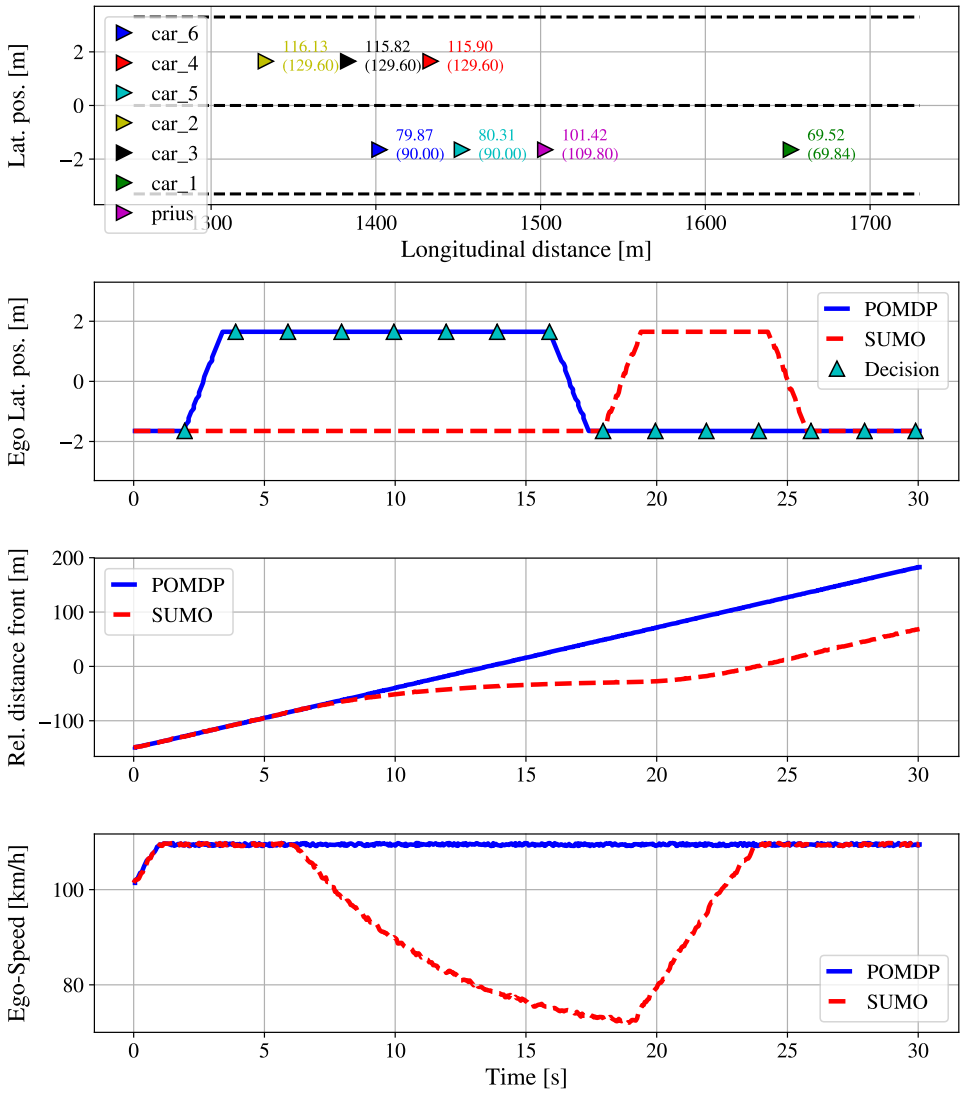


Fig. 7.10.: Planning results obtained for an overtake scenario where the long-term development of the scene needs to be considered. SUMO's reactive model leads the ego-vehicle to get stuck behind obstacle 'car_1' and deviating significantly from its desired velocity. In contrast, the proposed POMDP planner considers the long-term consequences of any decision and decides to switch lanes at around $t = 2s$, avoiding thus to get stuck.

exemplary tree simulations. For clarity, we have chosen to show the colors for only three vehicles in the scene, namely, the ego-vehicle (magenta), the leading obstacle in lane 1 (green), and the closest vehicle approaching from behind in lane 2 (red). In the first simulation, three LK actions are selected for the ego-vehicle, arriving to a belief node from where the rollouts are executed. This simulation is shown on the right-hand side of Figure 7.11. As can be seen in the belief obtained after executing the three LK maneuvers, the ego-vehicle is already predicted to be blocked behind 'car_1', with 'car_4' in parallel in lane 2. The rollout then shows that the ego-vehicle will continue to be blocked in lane 1 for the foreseeable future, and forced to decelerate further in order to keep its distance with the leading vehicle.

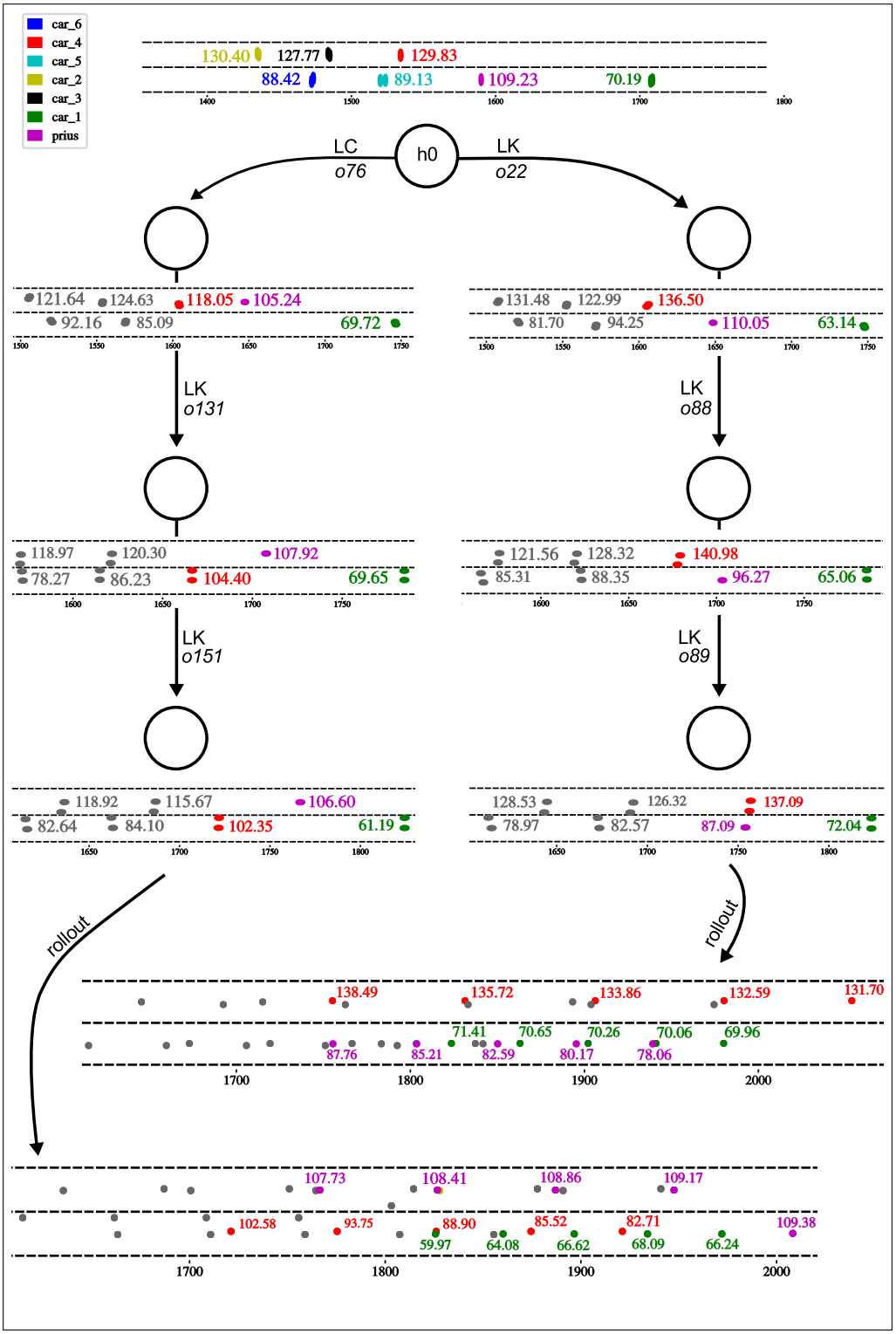


Fig. 7.11.: Exemplary tree simulations for the third traffic scene of task 1. Only the three most relevant vehicles in the scene are depicted in color. The POMDP's time step is 2 seconds. A simulation in which three lane keeping actions are selected shows the ego-vehicle (prius) stuck on lane 1 behind 'car_1' (right-hand side of the figure). In contrast, if a lane change action is selected followed by two lane keeping actions, the ego-vehicle is predicted to overtake 'car_1' and merge in front of it.

A different tree simulation considers first selecting a LC maneuver, and then follow up with two LK maneuvers. This simulation is shown on the left-hand side of Figure 7.11. The predicted belief after taking the LC action and receiving observation *o76*, is that the ego-vehicle successfully merges into lane 1, forcing 'car_4' to decelerate slightly. After the two subsequent LK actions, the predicted belief shows the ego-vehicle in lane 2, gaining terrain on 'car_1'. In contrast to the previous simulation, 'car_4' is predicted to merge into lane 1, due to the gap left by the ego-vehicle's lane change. The rollout that follows shows the ego-vehicle reaching its desired speed and merging back into lane 1 after overtaking 'car_1'. In this case, 'car_4' is the vehicle getting blocked behind 'car_1'.

Clearly, from the two exemplary simulations that we have discussed, and according to the driver model learned in chapter 4, the simulation in which the ego-vehicle is allowed to continue at its desired velocity will accrue the lowest cost. Through the 100 simulations per decision step, this information is backpropagated into the root node of the tree, leading to a lower value for the lane change action, which was ultimately selected.

Task 2: Controlling the acceleration of the vehicle

Configuration In this second experimental task, the goal is to test the ability of the proposed planner to leverage the intention estimations of surrounding traffic in order to make safe driving decisions. In terms of maneuvers, the ego-vehicle can perform three different acceleration actions $\{-1, 0, 1, \}$ $^{\mathrm{m}}/_{\mathrm{s}^2}$. The configuration selected for this task includes a time step for the POMDP of $1\mathrm{s}$, a maximum tree depth d_{lim} of 4, and only one rollout step with $d_{max} = 5$. The discount factor γ is maintained at 0.90. The considered time horizon is therefore 5s. The parameter w_{mc} that rewards maneuver continuity during the tree construction was set to 25 (and applied only to the LC maneuver). This value was selected as it is close to the weight component associated to the narrow TTC term of the cost function. The linear decay term τ was not used here; as a consequence, the estimated intentions for the obstacles will remain relevant during the complete planning horizon.

While our action space consists of acceleration commands, SUMO does not allow to fix the acceleration of a vehicle. In practice, our acceleration commands will be executed immediately, resulting in discontinuous changes in the ego-vehicle's velocity.

Results obtained We consider two different traffic scenes to test the performance of the proposed planner in this experimental task. In both scenes, the ego-vehicle navigates in the left lane of a 2-lane highway and is about to overtake a slower driver. The difference between the scenes is that in the first one, illustrated in Figure 7.12, the obstacle is never estimated to be performing a lane change. The lane change probability of the obstacle, shown in the second row (right y-axis), is consistently set to 0.05 for the duration of the scene. In contrast, in the second scene, pictured in Figure 7.13, the obstacle is estimated to be performing a lane change with probability 0.6 in the time range $5 - 25\mathrm{s}$.

The results obtained in the first traffic scene (Figure 7.12) show how the POMDP planner simply accelerates the ego-vehicle to its desired velocity, overtaking the obstacle. The results are

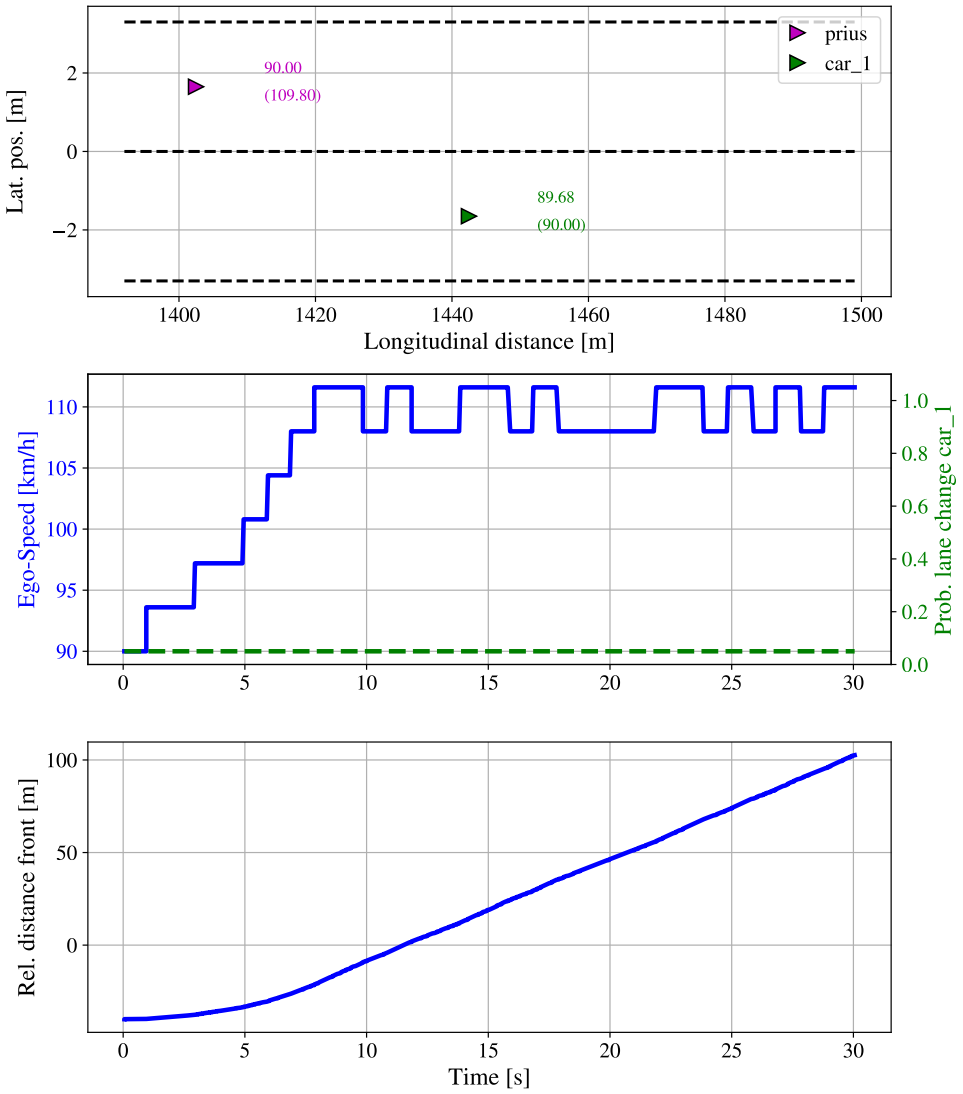


Fig. 7.12.: In this overtake scenario, the POMDP planner selects the acceleration commands for the ego-vehicle. The obstacle in lane 1 is never estimated to be executing a lane change maneuver.

however different in the second traffic scene (Figure 7.13). We discuss first the results obtained for the setting $w_{mc} = 25$. Initially, the POMDP planner also accelerates the ego-vehicle, aiming to reach the desired velocity. However, at time $t = 5\text{s}$, when the ego-vehicle is roughly 30m behind the obstacle (third row), a lane change intention is estimated for the obstacle with probability 0.6. At that point, the POMDP planner begins to decelerate the ego-vehicle in order to increase the distance gap with the obstacle. Once the distance has been sufficiently increased according to the model, the planner accelerates the ego-vehicle to match the obstacle's velocity (time $18 - 26\text{s}$). Once the obstacle, which remains the whole time in lane 1, is no longer estimated to be doing a lane change, the ego-vehicle accelerates and completes the overtake.

The reason behind the defensive behavior chosen by the planner is the parameter w_{mc}, which biases the sampling performed during the construction of the tree to have a sufficient representation in the beliefs of situations where the obstacle actually does perform a lane change.

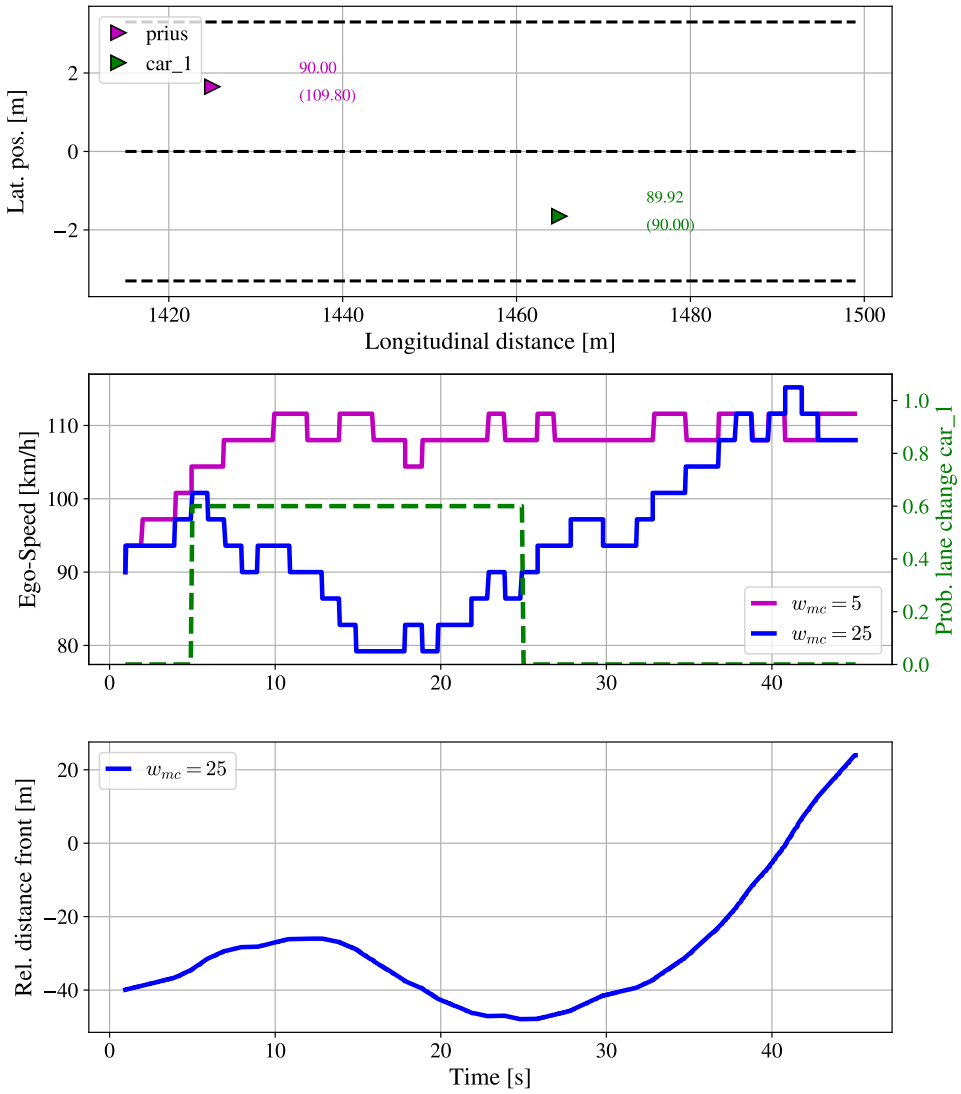


Fig. 7.13.: In this overtake scenario, the POMDP planner selects the acceleration commands for the ego-vehicle. The obstacle in lane 1 is estimated to be executing a lane change maneuver with probability 0.6 between seconds 5 and 25. As a response, the planner (with $w_{mc} = 25$) slows down the ego-vehicle to allow the lane change and avoid falling into high-cost states. Once it is clear that the obstacle will remain on its current lane, the planner (with $w_{mc} = 25$) accelerates to reach the desired speed of the ego-vehicle. However, if $w_{mc} = 5$ the danger is ignored and the ego-vehicle simply accelerates to overtake the obstacle.

In consequence, acceleration actions are estimated to lead to high-cost situations, promoting the conservative behavior of the ego-vehicle. Lowering w_{mc} to 5, as shown in Figure 7.13, makes the ego-vehicle ignore the lane change intentions of the obstacle and simply proceed with the overtake. The parameter w_{mc} determines therefore the behavior of the ego-vehicle as a response to the uncertain intentions of other drivers and should be carefully tuned.

7.4 Discussion

The goal of this chapter was to develop a human-inspired decision-making system for highway scenarios. Much like human drivers, the system should anticipate the short-term intentions of

surrounding vehicles and take them into consideration while selecting actions. Additionally, the system should also incorporate the human ability of considering the long-term consequences of a given maneuver by approximately predicting the long-term development of the traffic scene.

To achieve those goals, we have modeled the decision-making task as an online POMDP. During the planning process, we explicitly consider the uncertainties in the physical states and intentions of surrounding vehicles. In practice, the proposed approach constitutes a variation of the POMCP algorithm (Silver and Veness, 2010), where a search tree of histories is constructed by sampling state transitions and observations using a black box simulator. More specifically, POMCP relies on Monte-Carlo simulations to maintain the belief at each node and to estimate the value of each history. Sunberg et al. apply such an approach for a highway navigation task (Sunberg et al., 2017). The black box simulator here is given by a car-following (IDM) and a lane-change gap acceptance (MOBIL) model. The main drawback of this approach is that these models, which are in charge of predicting the environment dynamics, cannot predict by definition dangerous situations. In consequence, their POMDP cannot anticipate dangerous situations and can therefore lead to unsafe driving maneuvers.

In contrast to POMCP, the proposed approach maintains the beliefs at each node in the history tree as parametric distributions using the DBN presented in chapter 6. This interaction-aware model is also used to sample the observations during the tree construction, focusing the search towards likely situations given the state and intention estimations of the surrounding vehicles. By doing this, we overcome the main limitation of approaches that rely exclusively on risk-averse models to construct the tree. A similar approach to ours but applied to T-intersections navigation has been proposed by Bouton et al., where an IMM model is used to maintain the beliefs and guide the observation sampling (Bouton et al., 2017). However, the IMM model does not consider the interactions between vehicles, which can lead to inaccurate state and intention estimations. This could probably explain the collisions that the authors obtained in their simulated experimental evaluation.

In our approach, the value estimations are based on the driver model encoding the driving preferences of a generic human driver that was learned from demonstrated driving data in chapter 4. Furthermore, the estimated values also account for the long-term consequences of any action. This is achieved by performing approximate long-term scene predictions following a model-based approach close to the one we presented in chapter 5. Hence, in principle, the proposed approach satisfies all the above mentioned human-like requirements expected of the decision system.

The main drawback of the proposed approach lies on its limitation to meet real-time constraints, which is absolutely necessary in the autonomous driving domain. The main reasons for this are the expensive belief updates and the fact that the history tree is reconstructed for each planning step due to the continuous observation space. To mitigate the first cause, an option would be to parallelize the computation of the belief updates across vehicles, which is currently not implemented. The Monte-Carlo simulations could likewise be parallelized, increasing thus the amount of tree exploration done for similar planning times. To address

the second issue, we could rely on discretizing the observation space, although this would bring additional challenges. Other alternatives that could get us closer to meet the real-time constraints would be combining offline and online planning (Brechtel et al., 2011), or even try to learn the complete policy offline (Porta et al., 2006).

Part III

Conclusions

Conclusion and Perspectives

8

In this chapter, we first briefly summarize the materials that have been presented in the thesis. Then, we present the conclusions and insights at which we have arrived while conducting this research work. We finalize the chapter by presenting potential perspectives of future work.

8.1 Summary

This thesis revolves around human-like driving in highway environments. We introduced the motivations and the context around the topic of the thesis in **chapter 1**. Our main goal was to explore how to make an autonomous vehicle drive like a human driver. We quickly arrived to the idea that we needed a model encoding the preferences of a human driver in order to act as one. Moreover, we should also have a model to infer the intentions and future motion of surrounding vehicles, something at which human drivers are extraordinarily skilled. However, all predictions are inherently uncertain. To drive like humans, we would also need to mimic their ability to make safe driving decisions when the intentions of surrounding traffic are uncertain.

Chapter 3 reviewed the state of the art on driver behavior modeling, motion prediction and decision-making for intelligent vehicles. It turns out that most approaches that model the preferences of drivers as cost functions rely on hand-tuning the model parameters. Alternatively, IRL has also been explored for this task, although mostly on simple, synthetic problems. In terms of motion prediction, the current state-of-the-art regarding predictive power are models that consider the interactions between vehicles, typically relying on DBNs to do so. Finally, a considerable number of approaches for decision-making under uncertainty was presented. Most of them rely on the POMDP framework, and differ in the assumptions made to keep the approach tractable (simplified state spaces, action spaces, uncertainties, etc.).

Chapter 4 presented our first contribution. We showed how to learn a cost function encoding the preferences of a highway driver from demonstrated real-world driving data using IRL. Traditional IRL approaches cannot scale to continuous domains or handle the presence of dynamic obstacles. In contrast, our approach successfully handles both cases by combining the IRL paradigm with a trajectory planner based on a spatiotemporal state lattice. This makes the proposed approach an optimal alternative for the systematic learning of behavioral models in the autonomous driving domain.

Chapter 5 presented our second contribution. We proposed a method to predict the long-term development of highway traffic scenes based on a driver model learned using IRL. In this method, it is assumed that each vehicle in the scene behaves in the risk-averse manner encoded by the model. While the method constitutes a sensible option to produce long-term predictions of highway scenes, it is not a reliable alternative for the short-term estimation of behavior, since it cannot predict dangerous actions that deviate from what is encoded in the model.

Chapter 6 presented our third contribution, a probabilistic filtering framework that extends the work from Agamennoni et al. to infer the maneuvers of drivers in highway scenarios. The inference procedure is based on combining an anticipatory model-based prediction with a term that represents how the observed movement of the target matches a given maneuver. The model-based component provides some degree of situation understanding, reducing the number of false alerts in the detections. On the other hand, the dynamics-based component enables the detection of maneuvers that deviate from the risk-averse behavior encoded in the driver model. Our results showed that combining both components during inference led to an improved accuracy and a reduced rate of false positives in the detection of lane change maneuvers.

Chapter 7 presented our fourth and final contribution. We proposed a variation of the POMCP algorithm that maintains parametric beliefs for each history in the search tree using the DBN model from chapter 6. This model is also used to bias the construction of the history tree towards likely situations given the state and intention estimations of the surrounding vehicles. Thanks to this, and in contrast to alternative POMCP-like approaches that rely on traditional car-following and gap-acceptance models to model the environment dynamics, our approach also explores situations in which the surrounding vehicles behave dangerously (if evidence of that behavior was previously observed). Having a history tree that contains such potential scene developments enables the anticipative defensive behavior of the ego-vehicle. Furthermore, the proposed planner leverages a variation of the model-based prediction approach from chapter 5 in order to make foresighted decisions.

8.2 Conclusions

During the development of this thesis, we arrived to the following conclusions:

- Approximated IRL approaches like the one presented in this thesis constitute a convenient avenue to systematically learn models of behavior for the intelligent vehicles domain. This technique helps us avoid the costly and inefficient manual hand-tuning of the model parameters by simply providing demonstrated driving data and a set of task-relevant features.
- The use of properly balanced driver models allows for sensible long-term predictions of behavior. Despite not being necessarily too accurate, having an idea of the approximate long-term development of a traffic scene can provide invaluable information for decision-making, as experimentally shown in chapter 7.
- Considering the context leads to improved motion predictions. We experienced that this is not only true for long-term predictions, but also in behavior estimation tasks. In chapter 6, we used a driver model to determine what the normal behavior of a driver would be given the context. We showed how leveraging this information led to improved estimations compared to an approach that relied exclusively on dynamic evidence.
- Considering both the uncertain intention estimations of other drivers and the long-term consequences of any driving maneuver are neccesary components to achieve safe, human-like automated driving.

8.3 Perspectives

To conclude the thesis, we propose possible future avenues to extend our work. We classify them by the corresponding topic, namely, behavior modeling, prediction, and tactical planning.

Behavior modeling

Additional models The proposed behavior modeling approach could be applied to obtain driver models of different behaviors such as those of aggressive drivers, transport vehicles, drivers looking for a parking spot, etc. Since the demonstrated driving data cannot be easily provided for some of those cases, the use of realistic driving simulators could be explored. Another idea would be to extend the proposed approach in order to extract multiple behavioral models from a single dataset of demonstrations (Morton and Kochenderfer, 2017).

Avoiding feature engineering The current modeling approach requires to provide the features that characterize the behavior for which the model is learned. This might not be straightforward to do in some cases. Instead, we could resort to the use of a DNN to automatically learn the relevant features. This approach has already been explored in the literature, although not in situations with dynamic obstacles (Wulfmeier et al., 2017).

Additional scenarios Driver models could be learned for scenarios others than highways, for example, city center driving. However, in order to handle such cases, the original trajectory planner based on spatiotemporal state lattices would need to be extended to address stop-and-go scenarios.

Motion prediction:

Goal-based prediction A possibility to improve the quality of our state and maneuver estimations would be to additionally infer the goal of the target. Having a correct estimation of the navigational goal of a vehicle, could only lead to more accurate and robust maneuver and state estimations. This idea was already contemplated in earlier work (Dagli and Reichardt, 2002; Dagli et al., 2003).

Anomaly detection The proposed filtering framework estimates the maneuver in execution by the target by considering a finite set of candidate maneuvers. However, if none of the candidate maneuvers matches the observations of the target, we could identify its behavior as *anomalous* (Galceran et al., 2017). This information could be exploited by the tactical planner, making the ego-vehicle behave in a more conservative manner around that target.

Exploit multiple models If multiple behavioral models are available (e.g. normal, aggressive, inexpert...), a step before prediction could be included in order to assign a particular model to each target. If adequately assigned, this could improve the quality of the predictions.

Tactical-planning:

Guaranteeing safety In this thesis, we have focused on achieving human-like autonomous driving. However, it has been shown that human driving is not necessarily safe (Althoff and Lösch, 2016). In contrast, several authors have focused on achieving provably safe navigation; for example, using reachability analysis (Althoff, 2010) or inevitable collision states (Fraichard and Asama, 2004). Unfortunately, these approaches typically result in unnatural, extremely conservative actions of the autonomous vehicle. A promising direction to extend the proposed work would be to combine it with existing approaches in order to achieve some degree of safety guarantees, while at the same time maintaining the non-disruptive and predictable behavior characteristic of human-inspired approaches.

Appendices

Analysis of Algorithms

A

Algorithms provide set of rules to solve computational problems. The analysis of an algorithm involves the determination of its computational complexity. This enables the comparison between different algorithmic approaches. The computational complexity is typically measured as the time that the algorithm needs to solve a problem (time complexity), or to the amount of memory that it requires (space complexity). Both the time and space complexities are usually expressed as a function of the size n of the input data to the algorithm. In general, algorithm designers care about how algorithms scale for large input sizes, and hence constant factors (which are system dependent) and low-order terms are disregarded without losing much predictive power (Roughgarden, 2017). These ideas crystallize in the asymptotic analysis of algorithms.

A.1 Asymptotic notation

The asymptotic analysis of algorithms provides a clear view of the performance of any algorithm on large inputs, enabling thus a fair comparison between different approaches. The $\mathcal{O}$, Ω and Θ notations constitute the basis of the asymptotic analysis of algorithms (Knuth, 1976).

The $\mathcal{O}$ notation Let $T(n)$ and $f(n)$ be functions from the size of the input $n = 1, 2, 3, \ldots$ to positive reals. Then, $T(n) = \mathcal{O}(f(n))$ if and only if there exist positive constants c and n_0 such that:

$$T(n) \leq c \cdot f(n) \tag{A.1}$$

for all $n \geq n_0$. In other words, $T(n) = \mathcal{O}(f(n))$ if and only if $T(n)$ is eventually bounded above by a constant multiple of $f(n)$ (Roughgarden, 2017).

The Ω notation Let $T(n)$ and $f(n)$ be functions from the size of the input $n = 1, 2, 3, \ldots$ to positive reals. Then, $T(n) = \Omega(f(n))$ if and only if there exist positive constants c and n_0 such that:

$$T(n) \geq c \cdot f(n) \tag{A.2}$$

for all $n \geq n_0$. In other words, $T(n) = \Omega(f(n))$ if and only if $T(n)$ is eventually bounded below by a constant multiple of $f(n)$ (Roughgarden, 2017).

The Θ notation Let $T(n)$ and $f(n)$ be functions from the size of the input $n = 1, 2, 3, \ldots$ to positive reals. Then, $T(n) = \Theta(f(n))$ if and only if there exist positive constants c_1, c_2 and n_0 such that:

$$c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n) \tag{A.3}$$

for all $n \geq n_0$. In other words, $T(n) = \Theta(f(n))$ if and only if $T(n)$ is eventually bounded between two different constant multiples of $f(n)$ (Roughgarden, 2017).

Properties and Manipulation of Gaussian Distributions

B

This appendix presents the more relevant properties of the Gaussian distribution. References: (Bishop, 2006; Prince, 2012).

Linear transformation The Gaussian distribution is closed under linear transformations. Let $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ be a random vector Gaussian distributed. Then, any random vector resulting of a linear tranformation $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ is distributed as: $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; \mathbf{A}\boldsymbol{\mu}_X + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}_X\mathbf{A}^\mathbf{T})$.

Marginalization Let $\mathbf{x}$ be a Gaussian distributed random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Without loss of generality, let us assume that $\mathbf{x}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be partitioned as follows:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \tag{B.1}$$

Then, the resulting marginal distributions of this partition are also Gaussian and they are distributed as:

$$\begin{aligned} P(\mathbf{x}_1) &\sim \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \\ P(\mathbf{x}_2) &\sim \mathcal{N}(\mathbf{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \end{aligned} \tag{B.2}$$

Conditioning Let $\mathbf{x}$ be a Gaussian distributed random vector $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ that, without loss of generality, can be partitioned as shown in equation B.1. Then, the conditional distribution of the subset of variables $\mathbf{x}_1$ given the values of the variables in subset $\mathbf{x}_2$ is also Gaussian distributed:

$$\begin{aligned} P(\mathbf{x}_1|\mathbf{x}_2 = \mathbf{x}_2') &\sim \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2' - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}) \\ P(\mathbf{x}_2|\mathbf{x}_1 = \mathbf{x}_1') &\sim \mathcal{N}(\mathbf{x}_2; \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1' - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}) \end{aligned} \tag{B.3}$$

Product of two Gaussians The product of two Gaussian PDFs is proportional to another Gaussian PDF:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = \kappa\, \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3) \tag{B.4}$$

where:

$$\boldsymbol{\mu}_3 = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) \tag{B.5}$$

$$\boldsymbol{\Sigma}_3 = (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1} \tag{B.6}$$

$$\kappa = |2\pi(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right) \tag{B.7}$$

Collapsing a mixture of Gaussians Barber suggests a simple procedure to collapse a mixture of Gaussians to a single component (Barber, 2006). Given the mixture of Gaussians $P(\mathbf{x}) = \sum_i c_i \, \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, it may be collapsed to a single Gaussian defined by:

$$\boldsymbol{\mu} = \sum_i c_i \, \boldsymbol{\mu}_i \tag{B.8}$$

$$\boldsymbol{\Sigma} = \sum_i c_i \, \left(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T\right) - \boldsymbol{\mu}\boldsymbol{\mu}^T \tag{B.9}$$

Kalman filter models

C

In this appendix, the derivation of equations for the Kalman and the Extended Kalman filter is provided. References: (Thrun et al., 2005; Bishop, 2006; Abbeel, 2011).

C.1 The Kalman filter

The Kalman filter is a specialization of the more generic Bayes filter discussed in subsection 2.2.4. In the Kalman filter, the transition model is linear with added Gaussian noise to account for the uncertainty in the state transitions. Similarly, the emission model is linear with added Gaussian noise to account for the error in the measurements. This setting results in what is known as a linear-Gaussian state space model. The hidden states $\mathbf{x}_t$ and the observations $\mathbf{z}_t$ in the Kalman filter are Gaussian random vectors. The transition and emission models are:

$$\mathbf{x}_t = \boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{B}\mathbf{u}_{t-1} + \boldsymbol{\epsilon} \qquad \text{with } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{Q}) \tag{C.1}$$

$$\mathbf{z}_t = \boldsymbol{C}\mathbf{x}_t + \boldsymbol{\delta} \qquad \text{with } \boldsymbol{\delta} \sim \mathcal{N}(0, \boldsymbol{R}) \tag{C.2}$$

where $\boldsymbol{A}$ is known as the transition matrix, $\boldsymbol{B}$ is the control matrix, $\mathbf{u}_{t-1}$ is the control input at $t-1$, $\boldsymbol{C}$ is the emission matrix, $\boldsymbol{Q}$ is the process noise covariance matrix and $\boldsymbol{R}$ is the measurement noise covariance matrix.

Prediction step To derive the equations for the prediction step, we first consider the partitioned representation of the joint distribution between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, given past observations and control inputs. To save space, we denote $\mathbf{o}_{1:t-1} \doteq (\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1})$:

$$P(\mathbf{x}_{t-1}, \mathbf{x}_t | \mathbf{o}_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{t-1|1:t-1} \\ \boldsymbol{\mu}_{t|1:t-1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{t-1|1:t-1} & \boldsymbol{\Sigma}_{t-1,t|1:t-1} \\ \boldsymbol{\Sigma}_{t,t-1|1:t-1} & \boldsymbol{\Sigma}_{t,t|1:t-1} \end{bmatrix} \right) \tag{C.3}$$

where the shorthand notation $\boldsymbol{\mu}_{t-1|1:t-1} = \mathbb{E}[\mathbf{x}_{t-1}|\mathbf{o}_{1:t-1}]$ has been used. The elements $\boldsymbol{\mu}_{t-1|1:t-1}$ and $\boldsymbol{\Sigma}_{t-1|1:t-1}$ are available from the previous timestep. We determine the expressions for the remaining terms:

$$\begin{aligned} \boldsymbol{\mu}_{t|1:t-1} &= \mathbb{E}[\mathbf{x}_t|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[\boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{B}\mathbf{u}_{t-1} + \boldsymbol{\epsilon}|\mathbf{o}_{1:t-1}] \\ &= \boldsymbol{A}\boldsymbol{\mu}_{t-1|1:t-1} + \boldsymbol{B}\mathbf{u}_{t-1} \end{aligned} \tag{C.4}$$

$$\begin{aligned} \boldsymbol{\Sigma}_{t,1:t-1} &= \mathbb{E}[(\mathbf{x}_t - \boldsymbol{\mu}_t)(\mathbf{x}_t - \boldsymbol{\mu}_t)^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{\epsilon} - \boldsymbol{A}\boldsymbol{\mu}_{t-1})(\boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{\epsilon} - \boldsymbol{A}\boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\epsilon})(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\epsilon})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T\boldsymbol{A}^T|\mathbf{o}_{1:t-1}] + \mathbb{E}[\boldsymbol{\epsilon}(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}))^T|\mathbf{o}_{1:t-1}] \\ &+ \mathbb{E}[\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\boldsymbol{\epsilon}^T|\mathbf{o}_{1:t-1}] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T|\mathbf{o}_{1:t-1}] \end{aligned}$$

$$= \boldsymbol{A}\mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}]\boldsymbol{A}^T + \boldsymbol{Q}$$
$$= \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T + \boldsymbol{Q} \tag{C.5}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{t-1,t,1:t-1} &= \mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_t - \boldsymbol{\mu}_t)^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{\epsilon} - \boldsymbol{A}\boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\epsilon})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T\boldsymbol{A}^T|\mathbf{o}_{1:t-1}] + \mathbb{E}[(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})\boldsymbol{\epsilon}^T|\mathbf{o}_{1:t-1}] \\ &= \boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T \end{aligned} \tag{C.6}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{t,t-1,1:t-1} &= \mathbb{E}[(\mathbf{x}_t - \boldsymbol{\mu}_t)(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\boldsymbol{A}\mathbf{x}_{t-1} + \boldsymbol{\epsilon} - \boldsymbol{A}\boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \boldsymbol{\epsilon})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \mathbb{E}[(\boldsymbol{A}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] + \mathbb{E}[\boldsymbol{\epsilon}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})^T|\mathbf{o}_{1:t-1}] \\ &= \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1} \end{aligned} \tag{C.7}$$

The joint distribution between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, given $\mathbf{o}_{1:t-1}$ can then be expressed as:

$$P(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{o}_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{t-1|1:t-1} \\ \boldsymbol{A}\boldsymbol{\mu}_{t-1|1:t-1} + \boldsymbol{B}\mathbf{u}_{t-1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{t-1|1:t-1} & \boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T \\ \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1} & \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T + \boldsymbol{Q} \end{bmatrix}\right) \tag{C.8}$$

By applying the marginalization property (equation B.2), the predictive step is obtained:

$$P(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) = \mathcal{N}\left(\boldsymbol{A}\boldsymbol{\mu}_{t-1|1:t-1} + \boldsymbol{B}\mathbf{u}_{t-1}, \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T + \boldsymbol{Q}\right) \tag{C.9}$$

Measurement step To derive the equations for the measurement step, we first consider the partitioned representation of the joint distribution between $\mathbf{x}_t$ and $\mathbf{z}_t$, given $\mathbf{o}_{1:t-1}$. This representation is obtained using the same techniques shown in the prediction step.

$$P(\mathbf{x}_t, \mathbf{z}_t|\mathbf{o}_{1:t-1}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{t|1:t-1} \\ \boldsymbol{C}\boldsymbol{\mu}_{t|1:t-1} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{t|1:t-1} & \boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T \\ \boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1} & \boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T + \boldsymbol{R} \end{bmatrix}\right) \tag{C.10}$$

By conditioning on $\mathbf{z}_t$ as in equation B.3, the posterior distribution is obtained:

$$P(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \mathcal{N}\left(\boldsymbol{\mu}_{t|1:t}, \boldsymbol{\Sigma}_{t|1:t}\right) \tag{C.11}$$

$$\begin{aligned}\boldsymbol{\mu}_{t|1:t} &= \boldsymbol{\mu}_{t|1:t-1} + \boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T + \boldsymbol{R})^{-1}(\mathbf{z}_t - \boldsymbol{C}\mu_{t|1:t-1}) \\ \boldsymbol{\Sigma}_{t|1:t} &= \boldsymbol{\Sigma}_{t|1:t-1} - \boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T + \boldsymbol{R})^{-1}\boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1}\end{aligned} \tag{C.12}$$

Complete Kalman filter algorithm The steps of the Kalman filter, starting from the belief state $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{1|1}, \boldsymbol{\Sigma}_{1|1})$ are given in algorithm 8. The equations in the measurement step have been rewritten to include the Kalman gain matrix $\boldsymbol{K}_t$, which intuitively indicates our confidence in the accuracy of the measurements.

Algorithm 8: Kalman filter algorithm

Input: $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{1|1}, \boldsymbol{\Sigma}_{1|1})$, $\boldsymbol{A}$, $\boldsymbol{B}$ $\boldsymbol{C}$, $\boldsymbol{Q}$, $\boldsymbol{R}$

for $t = 2, 3, \ldots$ **do**

  Prediction step:

  $\boldsymbol{\mu}_{t|t-1} = \boldsymbol{A}\boldsymbol{\mu}_{t-1|1:t-1} + \boldsymbol{B}\mathbf{u}_{t-1}$

  $\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{A}\boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{A}^T + \boldsymbol{Q}$

  Measurement step:

  Receive $\mathbf{z}_t$

  $\boldsymbol{K}_t = \boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{C}^T + \boldsymbol{R})^{-1}$

  $\boldsymbol{\mu}_{t|1:t} = \boldsymbol{\mu}_{t|1:t-1} + \boldsymbol{K}_t(\mathbf{z}_t - \boldsymbol{C}\mu_{t|1:t-1})$

  $\boldsymbol{\Sigma}_{t|1:t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C})\boldsymbol{\Sigma}_{t|1:t-1}$

C.2 The extended Kalman filter

The extended Kalman filter relaxes the linearity assumption from the transition and emission models of the Kalman filter:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \boldsymbol{\epsilon} \qquad \mathbf{z}_t = g(\mathbf{x}_t) + \boldsymbol{\delta} \tag{C.13}$$

where f and g are arbitrary non-linear functions. Directly performing the predictive and measurement steps of the Kalman filter with these non-linear functions will result in non-Gaussian distributions. Instead, the extended Kalman filter approximates the true belief state using a Gaussian distribution. The approximation is performed by linearizing at each time step the functions f and g using their Taylor expansion around the means of the Gaussians to be transformed:

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx f(\boldsymbol{\mu}_{t|1:t}, \mathbf{u}_t) + \left.\frac{\partial f(\mathbf{x}_t, \mathbf{u}_t)}{\partial \mathbf{x}_t}\right|_{\boldsymbol{\mu}_{t|1:t}} (\mathbf{x}_t - \boldsymbol{\mu}_{t|1:t}) = f(\boldsymbol{\mu}_{t|1:t}, \mathbf{u}_t) + \boldsymbol{F}_t(\mathbf{x}_t - \boldsymbol{\mu}_{t|1:t})$$

$$g(\mathbf{x}_t) \approx g(\boldsymbol{\mu}_{t|1:t-1}) + \left.\frac{\partial g(\mathbf{x}_t)}{\partial \mathbf{x}_t}\right|_{\boldsymbol{\mu}_{t|1:t-1}} (\mathbf{x}_t - \boldsymbol{\mu}_{t|1:t-1}) = g(\boldsymbol{\mu}_{t|1:t-1}) + \boldsymbol{G}_t(\mathbf{x}_t - \boldsymbol{\mu}_{t|1:t-1})$$

where $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are the corresponding Jacobian matrices. This process results in algorithm 9.

Algorithm 9: Extended Kalman filter algorithm

Input: $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{1|1}, \boldsymbol{\Sigma}_{1|1})$, f, g, $\boldsymbol{Q}$, $\boldsymbol{R}$

for $t = 2, 3, \ldots$ **do**

  Prediction step:

  $\boldsymbol{\mu}_{t|t-1} = f(\boldsymbol{\mu}_{t-1|1:t-1}, \mathbf{u}_{t-1})$

  $\boldsymbol{\Sigma}_{t|t-1} = \boldsymbol{F}_t\boldsymbol{\Sigma}_{t-1|1:t-1}\boldsymbol{F}_t^T + \boldsymbol{Q}$

  Measurement step:

  Receive $\mathbf{z}_t$

  $\boldsymbol{K}_t = \boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{G}_t^T(\boldsymbol{G}_t\boldsymbol{\Sigma}_{t|1:t-1}\boldsymbol{G}^T + \boldsymbol{R})^{-1}$

  $\boldsymbol{\mu}_{t|1:t} = \boldsymbol{\mu}_{t|1:t-1} + \boldsymbol{K}_t(\mathbf{z}_t - g(\boldsymbol{\mu}_{t|1:t-1}))$

  $\boldsymbol{\Sigma}_{t|1:t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{G}_t)\boldsymbol{\Sigma}_{t|1:t-1}$

Bibliography

Abbeel, Pieter (2011). *Lecture notes in Advanced Robotics* (cit. on p. 147).

Abbeel, Pieter and Andrew Y. Ng (2004). "Apprenticeship learning via inverse reinforcement learning". In: *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004* (cit. on pp. 10, 32, 33, 63, 81).

Abbeel, Pieter, Dmitri Dolgov, Andrew Y. Ng, and Sebastian Thrun (2008). "Apprenticeship learning for motion planning with application to parking lot navigation". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*, pp. 1083–1090 (cit. on pp. 39, 61).

Agamennoni, G., J. I. Nieto, and E. M. Nebot (Aug. 2012). "Estimation of Multivehicle Dynamics by Considering Contextual Information". In: *IEEE Transactions on Robotics* 28.4, pp. 855–870 (cit. on pp. 7, 10, 48, 49, 85, 89, 90, 108, 138).

Althoff, M. (2010). *Reachability Analysis and Its Application to the Safety Assessment of Autonomous Cars* (cit. on p. 140).

Althoff, M. and R. Lösch (2016). "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 485–491 (cit. on p. 140).

Ammoun, S. and F. Nashashibi (2009). "Real time trajectory prediction for collision risk estimation between vehicles". In: *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, pp. 417–422 (cit. on pp. 41, 45).

Aoude, G., J. Joseph, N. Roy, and J. How (2011). "Mobile Agent Trajectory Prediction using Bayesian Nonparametric Reachability Trees". In: *American Institute of Aeronautics and Astronautics Infotech at Aerospace Conference* (cit. on p. 45).

Ardelt, Michael, Constantin Coester, and Nico Kaempchen (2012). "Highly Automated Driving on Freeways in Real Traffic Using a Probabilistic Framework". In: *IEEE Trans. Intelligent Transportation Systems* 13.4, pp. 1576–1585 (cit. on pp. 39, 51).

Argall, Brenna, Sonia Chernova, Manuela M. Veloso, and Brett Browning (2009). "A survey of robot learning from demonstration". In: *Robotics and Autonomous Systems* 57.5, pp. 469–483 (cit. on p. 32).

Atev, S., G. Miller, and N. P. Papanikolopoulos (2010). "Clustering of Vehicle Trajectories". In: *IEEE Transactions on Intelligent Transportation Systems* 11.3, pp. 647–657 (cit. on p. 44).

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (2002). "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47.2, pp. 235–256 (cit. on p. 30).

Bahram, M., C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr (2016). "A Combined Model- and Learning-Based Framework for Interaction-Aware Maneuver Prediction". In: *IEEE Transactions on Intelligent Transportation Systems* 17.6, pp. 1538–1550 (cit. on pp. 39, 47).

Bai, Haoyu, David Hsu, Wee Sun Lee, and Vien A. Ngo (2011). "Monte Carlo Value Iteration for Continuous-State POMDPs". In: *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*. Ed. by David Hsu, Volkan Isler, Jean-Claude Latombe, and Ming C. Lin. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 175–191 (cit. on pp. 31, 32).

Bandyopadhyay, Tirthankar, Kok Sung Won, Emilio Frazzoli, et al. (2013). "Intention-Aware Motion Planning". In: *Algorithmic Foundations of Robotics X*. Ed. by Emilio Frazzoli, Tomas Lozano-Perez, Nicholas Roy, and Daniela Rus. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 475–491 (cit. on pp. 55, 56).

Bar-Shalom, Yaakov and Xiao-Rong Li (2001). *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc. (cit. on p. 95).

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press (cit. on pp. 10, 17, 22, 53).

Barber, David (2006). "Expectation Correction for Smoothed Inference in Switching Linear Dynamical Systems". In: *Journal of Machine Learning Research* 7, pp. 2515–2540 (cit. on pp. 85, 90, 93, 146).

Bellman, Richard (Nov. 1954). "The theory of dynamic programming". In: *Bull. Amer. Math. Soc.* 60.6, pp. 503–515 (cit. on p. 25).

– (1957). "A Markovian Decision Process". In: *Journal of Mathematical Mechanics* 6 (cit. on pp. 24, 25).

Bertsekas, Dimitri P. (2000). *Dynamic Programming and Optimal Control*. 2nd. Athena Scientific (cit. on p. 25).

Bertsekas, D.P. and J.N. Tsitsiklis (2002). *Introduction to Probability*. Athena Scientific books. Athena Scientific (cit. on p. 17).

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Springer (cit. on pp. 17, 22, 145, 147).

Bliss, James P and Sarah A Acton (2003). "Alarm mistrust in automobiles: how collision alarm reliability affects driving". In: *Applied Ergonomics* 34.6, pp. 499 –509 (cit. on p. 4).

Boularias, Abdeslam, Jens Kober, and Jan Peters (2011). "Relative Entropy Inverse Reinforcement Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pp. 182–189 (cit. on p. 35).

Bouton, M., A. Cosgun, and M. J. Kochenderfer (June 2017). "Belief state planning for autonomously navigating urban intersections". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 825–830 (cit. on pp. 9, 55, 56, 104–106, 132).

Brechtel, S., T. Gindele, and R. Dillmann (2011). "Probabilistic MDP-behavior planning for cars". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1537–1542 (cit. on pp. 53, 56, 133).

– (2014). "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 392–399 (cit. on pp. 54, 56).

Brechtel, Sebastian (2015). "Dynamic Decision-making in Continuous Partially Observable Domains: A Novel Method and its Application for Autonomous Driving". PhD thesis (cit. on pp. 32, 51).

Brechtel, Sebastian, Tobias Gindele, and Rüdiger Dillmann (2013). "Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pp. 370–378 (cit. on pp. 32, 54).

Broadhurst, A., S. Baker, and T. Kanade (2005). "Monte Carlo road safety reasoning". In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* Pp. 319–324 (cit. on p. 42).

Brooks, Alex, Alexei Makarenko, Stefan B. Williams, and Hugh F. Durrant-Whyte (2006). "Parametric POMDPs for planning in continuous state spaces". In: *Robotics and Autonomous Systems* 54.11, pp. 887–897 (cit. on p. 31).

Browne, C. B., E. Powley, D. Whitehouse, et al. (2012). "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1, pp. 1–43 (cit. on pp. 103, 117).

Buehler, Martin, Karl Iagnemma, and Sanjiv Singh (2007). *The 2005 DARPA Grand Challenge: The Great Robot Race*. 1st. Springer Publishing Company, Incorporated (cit. on p. 50).

– (2009). *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. 1st. Springer Publishing Company, Incorporated (cit. on p. 51).

Byravan, Arunkumar, Mathew Monfort, Brian D. Ziebart, Byron Boots, and Dieter Fox (2015). "Graph-Based Inverse Optimal Control for Robot Manipulation". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 1874–1880 (cit. on pp. 36, 61).

California Department of Motor Vehicles (Feb. 2016). *Report of traffic accident involving an autonomous vehicle. Signed by Chris Urmson, program director.* Tech. rep. (cit. on p. 3).

Cassandra, Anthony, Michael L. Littman, and Nevin L. Zhang (1997). "Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes". In: *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. UAI'97. Providence, Rhode Island, pp. 54–61 (cit. on p. 27).

Couëtoux, Adrien, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard (2011). "Continuous Upper Confidence Trees". In: *Learning and Intelligent Optimization*. Ed. by Carlos A. Coello Coello. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 433–445 (cit. on p. 117).

Dagli, I. and D. Reichardt (2002). "Motivation-based approach to behavior prediction". In: *Intelligent Vehicle Symposium, 2002. IEEE.* Vol. 1, 227–233 vol.1 (cit. on p. 139).

Dagli, Ismail, Michael Brost, and Gabi Breuel (2003). "Action Recognition and Prediction for Driver Assistance Systems Using Dynamic Belief Networks". In: *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*. Ed. by Jaime G. Carbonell, Jörg Siekmann, Ryszard Kowalczyk, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 179–194 (cit. on p. 139).

Dickmanns, E. D., R. Behringer, D. Dickmanns, et al. (1994). "The seeing passenger car 'VaMoRs-P'". In: *Proceedings of the Intelligent Vehicles '94 Symposium*, pp. 68–73 (cit. on pp. 49, 50).

Erdmann, Jakob (2015). "SUMO's Lane-Changing Model". In: *Modeling Mobility with Open Data*. Ed. by Michael Behrisch and Melanie Weber. Cham: Springer International Publishing, pp. 105–123 (cit. on p. 120).

Finn, Chelsea, Sergey Levine, and Pieter Abbeel (2016). "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *Proceedings of the 33nd International Conference on Machine Learning, (ICML 2016)*, pp. 49–58 (cit. on pp. 35, 36).

Fletcher, Luke, Seth Teller, Edwin Olson, et al. (2008). "The MIT—Cornell collision and why it happened". In: *Journal of Field Robotics* 25.10, pp. 775–807 (cit. on pp. 52, 53).

Fraichard, Thierry and Hajime Asama (2004). "Inevitable collision states — a step towards safer robots?" In: *Advanced Robotics* 18.10, pp. 1001–1024 (cit. on p. 140).

Fraichard, Thierry and Thomas M. Howard (2012). "Iterative Motion Planning and Safety Issue". In: *Handbook of Intelligent Vehicles*. Ed. by Azim Eskandarian. London: Springer London, pp. 1433–1458 (cit. on p. 36).

Fulgenzi, C., C. Tay, A. Spalanzani, and C. Laugier (2008). "Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian processes". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1056–1062 (cit. on p. 45).

Galceran, Enric, Alexander G. Cunningham, Ryan M. Eustice, and Edwin Olson (Aug. 2017). "Multipolicy Decision-making for Autonomous Driving via Changepoint-based Behavior Prediction: Theory and Experiment". In: *Auton. Robots* 41.6, pp. 1367–1382 (cit. on p. 139).

Garzón, Mario and Anne Spalanzani (Nov. 2018). "An hybrid simulation tool for autonomous cars in very high traffic scenarios". In: *ICARCV 2018 - 15th International Conference on Control, Automation, Robotics and Vision*. Singapore, Singapore, pp. 1–6 (cit. on p. 118).

Gill, A. (1962). *Introduction to the Theory of Finite-state Machines*. McGraw-Hill Electronic Sciences series. McGraw-Hill (cit. on p. 51).

Gindele, T., S. Brechtel, and R. Dillmann (2010). "A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments". In: *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 1625–1631 (cit. on pp. 48, 49, 54, 55).

– (Oct. 2013). "Learning context sensitive behavior models from observations for predicting traffic situations". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 1764–1771 (cit. on pp. 7, 48).

GM Cruise (2017). *GM Cruise 2017 Disengagement Report*. Tech. rep. (cit. on p. 2).

Goldhoorn, A., A. Garrell, R. Alquézar, and A. Sanfeliu (2014). "Continuous real time POMCP to find-and-follow people by a humanoid service robot". In: *2014 IEEE-RAS International Conference on Humanoid Robots*, pp. 741–747 (cit. on pp. 104, 112).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press (cit. on pp. 17, 18).

Grover, C, I Knight, F Okoro, et al. (2008). *Automated Emergency Braking Systems: Technical requirements, costs and benefits*. Tech. rep. (cit. on p. 2).

Guestrin, Carlos and Dirk Ormoneit (2001). "Robust Combination of Local Controllers". In: *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pp. 178–185 (cit. on p. 64).

Hansen, Eric A. (1998a). "An Improved Policy Iteration Algorithm for Partially Observable MDPs". In: *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*. NIPS '97. Denver, Colorado, USA, pp. 1015–1021 (cit. on p. 29).

– (1998b). "Solving POMDPs by Searching in Policy Space". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI'98. Madison, Wisconsin, pp. 211–219 (cit. on p. 29).

Helbing, Dirk, Lubos Buzna, Anders Johansson, and Torsten Werner (2005). "Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions". In: *Transportation Science* 39.1, pp. 1–24 (cit. on p. 55).

Henry, Peter, Christian Vollmer, Brian Ferris, and Dieter Fox (2010). "Learning to navigate through crowded environments". In: *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 981–986 (cit. on pp. 40, 70).

Howard, Thomas M., Colin J. Green, Alonzo Kelly, and Dave Ferguson (2008). "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments". In: *J. Field Robotics* 25.6-7, pp. 325–345 (cit. on pp. 61, 64).

Hubmann, C., M. Aeberhard, and C. Stiller (Nov. 2016). "A generic driving strategy for urban environments". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1010–1016 (cit. on p. 7).

Ji, Shihao, Ronald Parr, Hui Li, Xuejun Liao, and Lawrence Carin (2007). "Point-based Policy Iteration". In: *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*. AAAI'07. Vancouver, British Columbia, Canada, pp. 1243–1249 (cit. on p. 29).

Jo, K., M. Lee, J. Kim, and M. Sunwoo (2016). "Tracking and Behavior Reasoning of Moving Vehicles Based on Roadway Geometry Constraints". In: *IEEE Transactions on Intelligent Transportation Systems* PP.99, pp. 1–17 (cit. on p. 42).

Joseph, Joshua, Finale Doshi-Velez, Albert S. Huang, and Nicholas Roy (2011). "A Bayesian nonparametric approach to modeling motion patterns". In: *Autonomous Robots* 31.4, p. 383 (cit. on pp. 44, 45).

Joseph, Joshua Mason, Finale Doshi-Velez, and Nicholas Roy (2010). "A Bayesian Nonparametric Approach to Modeling Mobility Patterns". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010* (cit. on pp. 44, 45).

Kaelbling, Leslie Pack, Michael L. Littman, and Anthony R. Cassandra (1998). "Planning and acting in partially observable stochastic domains". In: *Artificial Intelligence* 101.1, pp. 99 –134 (cit. on pp. 17, 27, 29).

Kaempchen, N., K. Weiss, M. Schaefer, and K. C. J. Dietmayer (2004). "IMM object tracking for high dynamic driving maneuvers". In: *IEEE Intelligent Vehicles Symposium, 2004*, pp. 825–830 (cit. on pp. 42, 85, 95).

Kaempchen, N., B. Schiele, and K. Dietmayer (2009). "Situation Assessment of an Autonomous Emergency Brake for Arbitrary Vehicle-to-Vehicle Collision Scenarios". In: *IEEE Transactions on Intelligent Transportation Systems* 10.4, pp. 678–687 (cit. on p. 42).

Kalman, R. E. (1960). "A New Approach to Linear Filtering And Prediction Problems". In: *ASME Journal of Basic Engineering* (cit. on p. 22).

Katrakazas, Christos, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka (2015). "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". In: *Transportation Research Part C: Emerging Technologies* 60, pp. 416 –442 (cit. on p. 36).

Kelly, Alonzo and Bryan Nagy (2003). "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control". In: *I. J. Robotics Res.* 22.7-8, pp. 583–602 (cit. on p. 64).

Kesting, Arne, Martin Treiber, and Dirk Helbing (2010). “Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity”. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 368.1928, pp. 4585–4605 (cit. on p. 54).

Kitani, Kris M., Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert (2012). “Activity Forecasting”. In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part IV*, pp. 201–214 (cit. on pp. 34, 69, 75).

Klingelschmitt, S., M. Platho, H. Groß, V. Willert, and J. Eggert (2014). “Combining behavior and situation information for reliably estimating multiple intentions”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 388–393 (cit. on pp. 43, 44, 46).

Knuth, Donald E. (Apr. 1976). “Big Omicron and Big Omega and Big Theta”. In: *SIGACT News* 8.2, pp. 18–24 (cit. on p. 143).

Kober, J., J. Andrew (Drew) Bagnell, and J. Peters (2013). “Reinforcement Learning in Robotics: A Survey”. In: (cit. on p. 32).

Kocsis, Levente and Csaba Szepesvári (2006). “Bandit Based Monte-Carlo Planning”. In: *Machine Learning: ECML 2006*. Ed. by Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 282–293 (cit. on p. 116).

Koenig, N. and A. Howard (2004). “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3, 2149–2154 vol.3 (cit. on p. 119).

Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press (cit. on p. 17).

Krajzewicz, Daniel, Jakob Erdmann, Michael Behrisch, and Laura Bieker (2012). “Recent Development and Applications of SUMO - Simulation of Urban MObility”. In: *International Journal On Advances in Systems and Measurements* 5.3&4, pp. 128–138 (cit. on p. 119).

Krauß, Stefan (1998). “Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics”. PhD thesis. Mathematisches Institut, Universität zu Köln (cit. on pp. 120, 122).

Kretzschmar, Henrik, Markus Spies, Christoph Sprunk, and Wolfram Burgard (2016). “Socially compliant mobile robot navigation via inverse reinforcement learning”. In: *The International Journal of Robotics Research* 35.11, pp. 1289–1307 (cit. on p. 36).

Kuderer, Markus, Shilpa Gulati, and Wolfram Burgard (2015). “Learning driving styles for autonomous vehicles from demonstration”. In: *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pp. 2641–2646 (cit. on pp. 36, 40, 41, 61, 64, 68, 72, 73).

Kumar, P., M. Perrollaz, S. Lefèvre, and C. Laugier (June 2013). “Learning-based approach for online lane change intention prediction”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 797–802 (cit. on pp. 7, 9, 43).

Kurniawati, Hanna and Vinay Yadav (2016). “An Online POMDP Solver for Uncertainty Planning in Dynamic Environment”. In: *Robotics Research: The 16th International Symposium ISRR*. Ed. by Masayuki Inaba and Peter Corke. Springer International Publishing, pp. 611–629 (cit. on p. 31).

Kurniawati, Hanna, David Hsu, and Wee Sun Lee (2008). “SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces”. In: *Robotics: Science and Systems IV, Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland, June 25-28, 2008* (cit. on p. 55).

Lawitzky, Andreas, Daniel Althoff, Christoph F. Passenberg, et al. (2013). "Interactive scene prediction for automotive applications". In: *2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast City, Australia, June 23-26, 2013*, pp. 1028–1033 (cit. on pp. 7, 46, 83, 85).

Lee, S. H. and S. W. Seo (May 2017). "A learning-based framework for handling dilemmas in urban automated driving". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1436–1442 (cit. on pp. 36, 40, 41, 61, 64, 72).

Lefevre, Stéphanie, Christian Laugier, and Javier Ibanez Guzman (2012). "Evaluating risk at road intersections by detecting conflicting intentions". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 4841–4846 (cit. on p. 48).

Lefèvre, Stéphanie, Dizan Vasquez, and Christian Laugier (2014). "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH Journal* 1.1, pp. 1–14 (cit. on pp. 9, 41).

Lerner, Uri N. (2012). "Hybrid Bayesian Networks for Reasoning about Complex Systems". PhD Thesis. Stanford University (cit. on p. 10).

Levine, Sergey and Pieter Abbeel (2014). "Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics". In: *Neural Information Processing Systems (NIPS)* (cit. on p. 36).

Levine, Sergey and Vladlen Koltun (2012). "Continuous Inverse Optimal Control with Locally Optimal Examples". In: *ICML '12: Proceedings of the 29th International Conference on Machine Learning* (cit. on pp. 36, 61).

Levine, Sergey, Zoran Popovic, and Vladlen Koltun (2011). "Nonlinear Inverse Reinforcement Learning with Gaussian Processes". In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.* Pp. 19–27 (cit. on pp. 10, 34, 35).

Liu, W., S. W. Kim, S. Pendleton, and M. H. Ang (2015). "Situation-aware decision making for autonomous driving on urban road using online POMDP". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1126–1133 (cit. on pp. 55, 56).

Lovejoy, William S. (1991). "Computationally Feasible Bounds for Partially Observed Markov Decision Processes". In: *Operations Research* 39.1, pp. 162–175 (cit. on p. 28).

McNaughton, Matthew, Chris Urmson, John M. Dolan, and Jin-Woo Lee (2011). "Motion planning for autonomous driving with a conformal spatiotemporal lattice". In: *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pp. 4889–4895 (cit. on pp. 37, 39, 61, 64, 65, 69).

Meyer-Delius, D., C. Plagemann, and W. Burgard (2009). "Probabilistic situation recognition for vehicular traffic scenarios". In: *2009 IEEE International Conference on Robotics and Automation*, pp. 459–464 (cit. on p. 45).

Miller, Isaac, Mark Campbell, Dan Huttenlocher, et al. (2008). "Team Cornell's Skynet: Robust perception and planning in an urban environment". In: *Journal of Field Robotics* 25.8, pp. 493–527 (cit. on p. 52).

Montemerlo, Michael, Jan Becker, Suhrid Bhat, et al. (2009). "Junior: The Stanford Entry in the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, pp. 91–123 (cit. on pp. 9, 39, 51, 61).

Morton, J. and M. J. Kochenderfer (2017). "Simultaneous policy learning and latent state inference for imitating driver behavior". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6 (cit. on p. 139).

Murphy, Kevin Patrick (2002). "Dynamic Bayesian Networks: Representation, Inference and Learning". AAI3082340. PhD thesis (cit. on pp. 22, 48).

Naser, F., D. Dorhout, S. Proulx, et al. (2017). "A parallel autonomy research platform". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 933–940 (cit. on p. 2).

National Highway Traffic Safety Administration (2013). *Analysis of Lane Change Crashes*. Tech. rep. (cit. on p. 4).

– (2015). *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Tech. rep. (cit. on p. 1).

National Transportation Safety Board (2017). *Accident Report: Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida, May 7, 2016*. Tech. rep. (cit. on p. 3).

– (2018). *Preliminary report: highway HWY18MH010. Accident in Tempe, Maricopa County, Arizona, on March 18, 2018.* Tech. rep. (cit. on p. 3).

Nègre, Amaury, Lukas Rummelhard, and Christian Laugier (June 2014). "Hybrid Sampling Bayesian Occupancy Filter". In: *IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, United States (cit. on p. 41).

Neumann, Gerhard, Michael Pfeiffer, and Wolfgang Maass (2007). "Efficient Continuous-Time Reinforcement Learning with Adaptive State Graphs". In: *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 250–261 (cit. on p. 64).

Ng, Andrew Y. and Stuart J. Russell (2000). "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pp. 663–670 (cit. on pp. 9, 32, 33, 62).

Ng, Andrew Y., Daishi Harada, and Stuart J. Russell (1999). "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping". In: *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999)*, pp. 278–287 (cit. on p. 32).

Nilsson, J., J. Fredriksson, and E. Coelingh (Sept. 2015). "Rule-Based Highway Maneuver Intention Recognition". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 950–955 (cit. on p. 4).

Okal, Billy and Kai Oliver Arras (2016). "Learning socially normative robot navigation behaviors with Bayesian inverse reinforcement learning". In: *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, pp. 2889–2895 (cit. on pp. 36, 61, 64).

Oliver, N. and A. P. Pentland (2000). "Graphical models for driver behavior recognition in a SmartCar". In: *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*, pp. 7–12 (cit. on pp. 44, 46).

Ong, Sylvie C. W., Shao Wei Png, David Hsu, and Wee Sun Lee (2010). "Planning under Uncertainty for Robotic Tasks with Mixed Observability". In: *I. J. Robotics Res.* 29.8, pp. 1053–1068 (cit. on p. 55).

Paden, B., M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli (2016). "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1.1, pp. 33–55 (cit. on p. 36).

Paquet, Sébastien, Ludovic Tobin, and Brahim Chaib-draa (2005). "An Online POMDP Algorithm for Complex Multiagent Environments". In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '05. The Netherlands, pp. 970–977 (cit. on pp. 30, 54).

Paquet, Sébastien, Brahim Chaib-draa, and Stéphane Ross (2006). "Hybrid POMDP algorithms". In: *In Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM-2006* (cit. on p. 30).

Perez-Higueras, N., Fernando Caballero, and Luis Merino (May 2018). "Learning Human-Aware Path Planning with Fully Convolutional Networks". In: *ICRA 2018 - Proceedings of the 2018 IEEE International Conference on Robotics and Automation*. Brisbane, Australia (cit. on pp. 64, 72).

Pérez-Higueras, Noé, Fernando Caballero, and Luis Merino (Apr. 2018). "Teaching Robot Navigation Behaviors to Optimal RRT Planners". In: *International Journal of Social Robotics* 10.2, pp. 235–249 (cit. on pp. 36, 61, 68).

Pineau, Joelle, Geoff Gordon, and Sebastian Thrun (2003). "Point-based Value Iteration: An Anytime Algorithm for POMDPs". In: *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico, pp. 1025–1030 (cit. on p. 28).

Pineau, Joelle, Geoffrey J. Gordon, and Sebastian Thrun (2006). "Anytime Point-Based Approximations for Large POMDPs". In: *Journal of Artificial Intelligence Research* 27, pp. 335–380 (cit. on p. 27).

Pivtoraiko, Mikhail and Alonzo Kelly (2005). "Efficient constrained path planning via search in state lattices". In: *The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space* (cit. on p. 64).

Pomerleau, D. A. (1991). "Efficient Training of Artificial Neural Networks for Autonomous Navigation". In: *Neural Computation* 3.1, pp. 88–97 (cit. on pp. 32, 49, 50).

Porta, Josep M., Nikos A. Vlassis, Matthijs T. J. Spaan, and Pascal Poupart (2006). "Point-Based Value Iteration for Continuous POMDPs". In: *Journal of Machine Learning Research* 7, pp. 2329–2367 (cit. on pp. 31, 133).

Poupart, Pascal and Craig Boutilier (2004). "Bounded Finite State Controllers". In: *Advances in Neural Information Processing Systems 16*. Ed. by S. Thrun, L. K. Saul, and B. Schölkopf. MIT Press, pp. 823–830 (cit. on p. 29).

Powell, Warren B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. New York, NY, USA: Wiley-Interscience (cit. on p. 25).

Prince, S.J.D. (2012). *Computer Vision: Models Learning and Inference*. Cambridge University Press (cit. on pp. 17, 145).

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA: John Wiley & Sons, Inc. (cit. on p. 25).

Randløv, Jette and Preben Alstrøm (1998). "Learning to Drive a Bicycle Using Reinforcement Learning and Shaping". In: *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pp. 463–471 (cit. on p. 32).

Rasmussen, CE. and CKI. Williams (Jan. 2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, p. 248 (cit. on p. 44).

Ratliff, Nathan, J. Andrew (Drew) Bagnell, and Martin Zinkevich (2006a). "Maximum Margin Planning". In: *International Conference on Machine Learning*. Pittsburgh, PA (cit. on pp. 34, 61).

Ratliff, Nathan D., David M. Bradley, J. Andrew Bagnell, and Joel E. Chestnutt (2006b). "Boosting Structured Prediction for Imitation Learning". In: *Advances in Neural Information Processing Systems 19, Vancouver, British Columbia, Canada, December 4-7, 2006*, pp. 1153–1160 (cit. on pp. 34, 35).

Riedmiller, M. and H. Braun (1993). "A direct adaptive method for faster backpropagation learning: the RPROP algorithm". In: *IEEE International Conference on Neural Networks*, 586–591 vol.1 (cit. on p. 40).

Rong Li, X. and Vesselin P. Jilkov (2000). "Survey of maneuvering target tracking: dynamic models". In: vol. 4048, pp. 4048 –4048 –24 (cit. on p. 41).

Ross, Stéphane and Brahim Chaib-draa (2007). "AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs". In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 2592–2598 (cit. on p. 30).

Ross, Stéphane, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa (2008). "Online Planning Algorithms for POMDPs". In: *J. Artif. Intell. Res.* 32, pp. 663–704 (cit. on pp. 30, 31).

Roughgarden, T. (2017). *Algorithms Illuminated, Part 1: The Basics*. Soundlikeyourself Publishing, LLC (cit. on p. 143).

Rummelhard, L., A. Nègre, and C. Laugier (Sept. 2015). "Conditional Monte Carlo Dense Occupancy Tracker". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 2485–2490 (cit. on pp. 69, 83, 95).

Rummelhard, Lukas, Amaury Nègre, Mathias Perrollaz, and Christian Laugier (June 2014). "Probabilistic Grid-based Collision Risk Prediction for Driving Application". In: *ISER*. Marrakech/Essaouira, Morocco (cit. on p. 41).

Russell, Stuart J. (1998). "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT 1998, Madison, Wisconsin, USA, July 24-26, 1998*. Pp. 101–103 (cit. on pp. 9, 32).

Sadigh, Dorsa, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan (2016). "Planning for Autonomous Cars that Leverages Effects on Human Actions". In: *Proceedings of the Robotics: Science and Systems Conference (RSS)* (cit. on p. 61).

SAE International (2016). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Tech. rep. (cit. on pp. 2, 3).

Satia, Jay K. and Roy E. Lave. (1973). "Markovian Decision Processes with Uncertain Transition Probabilities". In: *Operations Research* 21.3, pp. 728–740 (cit. on p. 30).

Schubert, R. (2012). "Evaluating the Utility of Driving: Toward Automated Decision Making Under Uncertainty". In: *IEEE Transactions on Intelligent Transportation Systems* 13.1, pp. 354–364 (cit. on pp. 53, 56).

Schubert, R., E. Richter, and G. Wanielik (June 2008). "Comparison and evaluation of advanced motion models for vehicle tracking". In: *2008 11th International Conference on Information Fusion*, pp. 1–6 (cit. on p. 41).

Schwarting, W. and P. Pascheka (Oct. 2014). "Recursive conflict resolution for cooperative motion planning in dynamic highway traffic". In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 1039–1044 (cit. on pp. 7, 47, 85).

Schwarting, W., J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus (2017). "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928–1935 (cit. on p. 2).

Shani, Guy, Joelle Pineau, and Robert Kaplow (2013). "A survey of point-based POMDP solvers". In: *Autonomous Agents and Multi-Agent Systems* 27.1, pp. 1–51 (cit. on p. 29).

Shariff, Azim, Jean-François Bonnefon, and Iyad Rahwan (Sept. 2017). "Psychological roadblocks to the adoption of self-driving vehicles". In: *Nature Human Behaviour* 1 (10), pp. 694–696 (cit. on p. 12).

Shiarlis, K., J. Messias, and S. Whiteson (May 2017). "Rapidly exploring learning trees". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1541–1548 (cit. on p. 61).

Shimosaka, M., T. Kaneko, and K. Nishi (2014). "Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning". In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1694–1700 (cit. on pp. 40, 47, 61).

Shimosaka, Masamichi, Junichi Sato, Kazuhito Takenaka, and Kentarou Hitomi (2017). "Fast Inverse Reinforcement Learning with Interval Consistent Graph for Driving Behavior Prediction". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017*, pp. 1532–1538 (cit. on pp. 61, 69).

Silver, David and Gerald Tesauro (2009). "Monte-Carlo simulation balancing". In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pp. 945–952 (cit. on p. 104).

Silver, David and Joel Veness (2010). "Monte-Carlo Planning in Large POMDPs". In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.* Pp. 2164–2172 (cit. on pp. 11, 30, 31, 113, 116, 132).

Silver, David, James A. Bagnell, and Anthony Stentz (2008). "High Performance Outdoor Navigation from Overhead Data using Imitation Learning". In: *Robotics: Science and Systems IV, Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland, June 25-28, 2008* (cit. on pp. 61, 103, 113).

Smallwood, Richard D. and Edward J. Sondik (1973). "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon". In: *Operations Research* 21.5, pp. 1071–1088 (cit. on p. 27).

Smith, Trey and Reid G. Simmons (2004). "Heuristic Search Value Iteration for POMDPs". In: *UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, Banff, Canada, July 7-11, 2004*, pp. 520–527 (cit. on p. 29).

– (2005). "Point-Based POMDP Algorithms: Improved Analysis and Implementation". In: *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pp. 542–547 (cit. on p. 29).

Somani, Adhiraj, Nan Ye, David Hsu, and Wee Sun Lee (2013). "DESPOT: Online POMDP Planning with Regularization". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, pp. 1772–1780 (cit. on p. 31).

Sondik, Edward J. (1971). "The Optimal Control of Partially Observable Markov Processes". PhD thesis. Stanford University (cit. on p. 27).

– (1978). "The Optimal Control of Partially Observable Markov Processes Over the Infinite Horizon: Discounted Costs". In: *Operations Research* 26.2, pp. 282–304 (cit. on p. 27).

Sonu, Ekhlas, Zachary Sunberg, and Mykel J. Kochenderfer (2018). “Exploiting Hierarchy for Scalable Decision Making in Autonomous Driving”. In: *Intelligent Vehicles Symposium*. Changshu (cit. on pp. 54, 56).

Sorstedt, J., L. Svensson, F. Sandblom, and L. Hammarstrand (2011). “A New Vehicle Motion Model for Improved Predictions and Situation Assessment”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4, pp. 1209–1219 (cit. on pp. 39, 46).

Spaan, Matthijs T. J. and Nikos A. Vlassis (2005). “Perseus: Randomized Point-based Value Iteration for POMDPs”. In: *J. Artif. Intell. Res.* 24, pp. 195–220 (cit. on p. 28).

Sunberg, Z. N., C. J. Ho, and M. J. Kochenderfer (2017). “The value of inferring the internal state of traffic participants for autonomous freeway driving”. In: *2017 American Control Conference (ACC)*, pp. 3004–3010 (cit. on pp. 55, 56, 104, 105, 108, 132).

Sunberg, Zachary N. and Mykel J. Kochenderfer (2018). “Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces”. In: *International Conference on Automated Planning and Scheduling (ICAPS)*. Delft (cit. on pp. 32, 105, 113, 117).

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement learning: An introduction*. Adaptive computation and machine learning. MIT Press (cit. on pp. 17, 24).

Tay, Christopher (Sept. 2009). “Analysis of Dynamic Scenes: Application to Driving Assistance”. Theses. Institut National Polytechnique de Grenoble - INPG (cit. on pp. 7, 9, 44–46).

Tesla (2016). *Tesla 2016 Disengagement Report*. Tech. rep. (cit. on p. 2).

Tesla Autopilot. `https://www.tesla.com/autopilot`. Accessed: 26-06-2018 (cit. on p. 2).

Thrun, Sebastian (1999). “Monte Carlo POMDPs”. In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, 1999]*, pp. 1064–1070 (cit. on p. 31).

Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2005). *Probabilistic Robotics*. The MIT Press (cit. on pp. 17, 22, 24, 147).

Thrun, Sebastian, Michael Montemerlo, Hendrik Dahlkamp, et al. (2006). “Stanley: The robot that won the DARPA Grand Challenge”. In: *J. Field Robotics* 23.9, pp. 661–692 (cit. on pp. 39, 50).

Toledo-Moreo, R. and M. A. Zamora-Izquierdo (Mar. 2009). “IMM-Based Lane-Change Prediction in Highways With Low-Cost GPS/INS”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.1, pp. 180–185 (cit. on pp. 43, 85, 89, 95, 96).

Treiber, Martin and Arne Kesting (2009). “Modeling Lane-Changing Decisions with MOBIL”. In: *Traffic and Granular Flow ’07*. Ed. by Cécile Appert-Rolland, François Chevoir, Philippe Gondret, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 211–221 (cit. on p. 54).

Treiber, Martin, Ansgar Hennecke, and Dirk Helbing (2000). “Congested traffic states in empirical observations and microscopic simulations”. In: *Phys. Rev. E* 62 (2), pp. 1805–1824 (cit. on pp. 88, 95).

Ulbrich, S. and M. Maurer (2013). “Probabilistic online POMDP decision making for lane changes in fully automated driving”. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 2063–2067 (cit. on pp. 5, 9, 54, 56).

United Nations Economic Commission for Europe (2015). *The 2015 bulletin on statistics of road traffic accidents in Europe and North America*. Tech. rep. (cit. on p. 1).

Urmson, Chris, Joshua Anhalt, Drew Bagnell, et al. (2009). "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic, George Air Force Base, Victorville, California, USA*, pp. 1–59 (cit. on pp. 2, 9).

U.S. Department of Transportation (2006). *NGSIM: Next Generation Simulation*. https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm. Accessed: 02-07-2018 (cit. on p. 4).

Vasquez, D. (May 2016). "Novel planning-based algorithms for human motion prediction". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3317–3322 (cit. on p. 75).

Vasquez, D., T. Fraichard, and C. Laugier (2009). "Incremental Learning of Statistical Motion Patterns With Growing Hidden Markov Models". In: *IEEE Transactions on Intelligent Transportation Systems* 10.3, pp. 403–416 (cit. on p. 44).

Vasquez, D., B. Okal, and K. O. Arras (Sept. 2014). "Inverse Reinforcement Learning algorithms and features for robot navigation in crowds: An experimental comparison". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1341–1346 (cit. on pp. 40, 70).

Verband der Automobilindustrie (2015). *From Driver Assistance Systems to Automated Driving*. Tech. rep. (cit. on p. 1).

Vogel, Katja (2003). "A comparison of headway and time to collision as safety indicators". In: *Accident Analysis & Prevention* 35.3, pp. 427 –433 (cit. on p. 78).

Washington, Richard (1997). "BI-POMDP: Bounded, Incremental, Partially-Observable Markov-Model Planning". In: *Recent Advances in AI Planning, 4th European Conference on Planning, ECP'97, Toulouse, France, September 24-26, 1997, Proceedings*, pp. 440–451 (cit. on p. 30).

Waymo (2017). *Waymo 2017 Disengagement Report*. Tech. rep. (cit. on pp. 2, 3).

Wei, J. and J. M. Dolan (June 2009). "A robust autonomous freeway driving algorithm". In: *2009 IEEE Intelligent Vehicles Symposium*, pp. 1015–1020 (cit. on pp. 39, 61).

Weiss, K., N. Kaempchen, and A. Kirchner (2004). "Multiple-model tracking for the detection of lane change maneuvers". In: *IEEE Intelligent Vehicles Symposium, 2004*, pp. 937–942 (cit. on pp. 43, 85).

Wolf, M. T. and J. W. Burdick (2008). "Artificial potential functions for highway driving with collision avoidance". In: *2008 IEEE International Conference on Robotics and Automation*, pp. 3731–3736 (cit. on pp. 39, 40, 61).

World Health Organization (2015). *Global Status Report on Road Safety 2015*. Tech. rep. (cit. on p. 1).

Wulfmeier, M., D. Z. Wang, and I. Posner (Oct. 2016). "Watch this: Scalable cost-function learning for path planning in urban environments". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2089–2095 (cit. on p. 69).

Wulfmeier, Markus, Peter Ondruska, and Ingmar Posner (2015). "Maximum Entropy Deep Inverse Reinforcement Learning". In: *Neural Information Processing Systems Conference, Deep Reinforcement Learning Workshop* (cit. on pp. 34, 35).

Wulfmeier, Markus, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner (2017). "Large-scale cost function learning for path planning using deep inverse reinforcement learning". In: *The International Journal of Robotics Research* 36.10, pp. 1073–1087 (cit. on pp. 35, 40, 139).

Ziebart, Brian D., Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey (2008). "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pp. 1433–1438 (cit. on pp. 33–35, 63, 76).

Ziebart, Brian D., Nathan D. Ratliff, Garratt Gallagher, et al. (2009). "Planning-based prediction for pedestrians". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, pp. 3931–3936 (cit. on p. 75).

Ziegler, J., P. Bender, H. Lategahn, et al. (2014a). "Kartengestütztes automatisiertes Fahren auf der Bertha-Benz-Route von Mannheim nach Pforzheim". In: *9. Workshop Fahrerassistenzsysteme (FAS2014), Walting, Deutschland*, pp. 79–94 (cit. on p. 9).

Ziegler, J., P. Bender, M. Schreiber, et al. (2014b). "Making Bertha Drive: An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2, pp. 8–20 (cit. on pp. 9, 51, 52).

Zyner, A., S. Worrall, and E. Nebot (2018). "A Recurrent Neural Network Solution for Predicting Driver Intention at Unsignalized Intersections". In: *IEEE Robotics and Automation Letters* 3.3, pp. 1759–1764 (cit. on p. 43).

Åström, K.J (1965). "Optimal control of Markov processes with incomplete state information". In: *Journal of Mathematical Analysis and Applications* 10.1, pp. 174 –205 (cit. on pp. 24, 26).